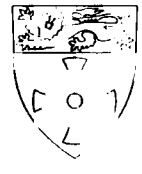
An Algebra of High Level Petri Nets

Jon G. Hall



December 1996

Being a thesis submitted for the degree of Doctor of Philosophy in the University of Newcastle upon Tyne

To Lucia—my wife, love and life

Abstract

Petri nets were introduced by C.A. Petri as a theoretical model of concurrency in which the causal relationship between actions, rather than just their temporal ordering, can be represented. As a theoretical model of concurrency, Petri nets have been widely successful. Moreover, Petri nets are popular with practitioners, providing practical tools for the designer and developer of real concurrent and distributed systems.

However, it is from this second context that perhaps the most widely voiced criticism of Petri nets comes. It is that Petri nets lack any algebraic structure or modularity, and this results in large, unstructured models of real systems, which are consequently often intractable. Although this is not a criticism of Petri nets *per se*, but rather of the uses to which Petri nets are put, the criticism is well taken.

We attempt to answer this criticism in this work. To do this we return to the view of Petri nets as a model of concurrency and consider how other models of concurrency counter this objection. The foremost examples are then the *synchronisation trees* of Milner, and the *traces* of Hoare, (against which such criticism is rarely, if ever, levelled). The difference between the models is clear, and is to be found in the richness of the algebraic characterisations which have been made for synchronisation trees in Milner's Calculus of Communicating Systems (CCS), and for traces in Hoare's Communicating Sequential Processes (CSP).

With this in mind we define, in this thesis, a class of high level Petri nets, *High Level Petri Boxes*, and provide for them a very general algebraic description language, the *High Level Petri Box Algebra*, with novel ideas for synchronisation, and including both refinement and recursion among its operators. We also begin on the (probably open-ended task of the) algebraic characterisation of High Level Petri Boxes.

The major contribution of this thesis is a full behavioural characterisation of the High Level Petri Boxes which form the semantic domain of the algebra. Other contributions are: a very general method of describing communication protocols which extend the synchronisation algebras of Winskel; a recursive operator that preserves finiteness of state (the best possible, given the generality of the algebra); a refinement operator that is syntactic in nature, and for which the recursive construct is a behavioural fix-point; and a notion of behavioural equivalence which is a congruence with respect to a major part of the High Level Petri Box Algebra.

Contents

1 Introduction 1

1.1 High Level Petri Box Algebra . . . 2

1.2 High Level Petri Boxes . . . 3

1.3 Structure of the Thesis . . . 5

2 The High Level Petri Box Model 7

2.1 Labels and Label Algebras . . . 7

2.1.1 Labels . . . 7

2.1.2 Label Algebras . . . 8

2.1.3 Expressivity of Label Algebras . . . 9

2.1.3.1 Taubner's Label Algebra . . . 10

2.1.4 Label Algebras for CCS, TCSP, and PBC . . . 11

2.1.4.1 CCS Label Algebra . . . 11

2.1.4.2 TCSP Label Algebra . . . 11

2.1.4.3 PBC Label Algebra . . . 12

2.2 Behaviour of High Level Petri Boxes . . . 13

2.2.1 Distributed Transition Systems . . . 13

2.2.2 H-synchrony . . . 15

2.2.2.1 Relabelling . . . 15

2.2.2.2 Asynchrony . . . 16

2.2.2.3 2-synchrony . . . 16

2.2.2.4 Expressivity of H-synchrony . . . 16

2.3 Contexts in High Level Petri Boxes . . . 16

2.3.1 Signed Contexts . . . 18

2.4 The Flow Predicate . . . 20

2.4.1 Label Produced by a Transition . . . 22

2.4.2 Symmetries of the Flow Predicate . . . 23

2.5 High Level Petri Boxes . . . 24

2.5.1 Places and Place Labels . . . 24

2.5.1.1 Ensuring Disjointness of the Name Space 24
2.5.1.2 Place Labels 26
2.5.2 Local Transitions 26
2.5.3 Abstract Transitions 27
2.5.4 Pre-High Level Petri Boxes 28
2.5.4.1 Operators on Pre-High Level Petri Boxes 29
2.5.4.2 A Partial Order on pre-High Level Petri Boxes 30
2.5.4.3 Sociability of pre-High Level Petri Boxes 30
2.5.5 High Level Petri Boxes 31
2.5.5.1 Markings of High Level Petri Boxes 33
2.5.5.2 Standard Initial and Terminal Markings 34
2.5.6 Why pre-High Level Petri Boxes? 35
2.5.6.1 On the Representation of High Level Petri Boxes 35
2.5.7 Isomorphism of High Level Petri Boxes 35
2.6 Summary 38

3 The Meaning of a High Level Petri Box 39
3.1 Predicate/Transition Nets 39
3.2 Predicate/Transition Net Semantics of High Level Petri Boxes 41
3.2.1 The Language and Logic of High Level Petri Boxes 42
3.2.2 The Denotation of a High Level Petri Box 43
3.2.2.1 Examples of the Denotation 44
3.2.2.2 Predicate/Transition Net Denotations of Markings 46
3.3 The Behaviour of a High Level Petri Box 47
3.3.1 Indices and Substitutions 47
3.3.2 Local-Strictness 47
3.3.3 Label Produced by a Feasible Substitution 48
3.3.4 A Symbolic Firing Rule for High Level Petri Box 49
3.3.4.1 A Symbolic Firing Rule for Predicate/Transition Nets 49
3.3.4.2 Notions of Behaviour for High Level Petri Boxes 50
3.3.4.3 Interpretation on High Level Petri Boxes 51
3.3.4.4 Examples 51
3.3.4.5 Distinguishability of Substitutions 52
3.4 The Most Permissive Label Algebra 53
3.4.1 Positive Monotonic Properties 54
3.4.2 Completeness and Decoupling in the Most Permissive Label Algebra . . . 55
3.4.3 Initial Marking Independence 56

3.5 A P/T Net Unfolding of High Level Petri Boxes . . . 58
3.5.1 A Transition Centred Place/Transition Net Model . . . 58
3.5.2 α-Instances of Transitions . . . 59
3.5.3 The Construction . . . 59
3.6 Relations on High Level Petri Boxes . . . 61
3.6.1 Showing Reachable Equivalence from a Standard Initial Marking . . . 62
3.7 Discussion . . . 63
3.7.1 A Comparison with the Denotation of Taubner . . . 63
3.8 Summary . . . 64

4 Algebra and Semantics I **65**
4.1 Termination and Deadlock . . . 65
4.1.1 Sequential Composition . . . 66
4.1.2 The Terminal State of Parallel and Choice Compositions . . . 67
4.2 The Pragmatic Choice is Sequence over Prefix . . . 68
4.3 Algebra, Semantics, Compositionality and Modularity . . . 69
4.3.1 Modularity . . . 70
4.4 Notation and Conventions . . . 70
4.4.1 Functions and Relations . . . 70
4.4.2 Sane and Separable Relations . . . 71
4.5 Auxiliary Operators . . . 73
4.5.1 High Level Petri Box Union . . . 73
4.5.2 Combinational Closure . . . 77
4.5.3 Context Manipulation . . . 79
4.5.4 Relational Lifting . . . 80
4.5.5 Auxiliary Operator Precedence . . . 82
4.6 Top Level Operators . . . 82
4.6.1 Concurrent Composition . . . 83
4.6.2 Causal Composition . . . 85
4.6.3 Choice composition . . . 86
4.6.4 Relabelling . . . 89
4.7 Basic High Level Petri Boxes . . . 90
4.8 The High Level Petri Box Algebra . . . 90
4.8.1 Syntax . . . 91
4.8.2 Semantics . . . 91
4.8.2.1 Well-definedness of the Semantic Function . . . 92
4.8.3 Class-wide Properties of High Level Petri Boxes . . . 93

4.8.3.1 Finitary Reasoning on High Level Petri Boxes 93
4.8.3.2 Completeness 94
4.8.3.3 Local Strictness of Syntactically Generated High Level Petri Boxes 94
4.8.3.4 Safeness and Memorylessness of Syntactically Generated High Level Petri Boxes 96
4.8.4 Isomorphism and Syntactically Generated High Level Petri Boxes 98
4.8.5 Extended High Level Petri Boxes 100
4.9 Discussion 101
4.9.1 An Alternative Representation of stop 101
4.9.2 The Unit Pre-High Level Petri Box ϵ 101
4.9.2.1 Regaining Closure 102
4.9.2.2 A Causal Unit 103
4.9.2.3 Safeness and Memorylessness 103
4.10 Summary 104

5 Normal Form **105**
5.1 Normal Forms 105
5.1.1 Rewrite Systems, Termination and Confluence 106
5.1.1.1 Using an Equivalence as a Rewrite 107
5.1.2 Auxiliary Syntax 108
5.2 A Normal Form for High Level Petri Boxes 108
5.2.1 An Anatomy of the Interrelation of Auxiliaries 109
5.2.2 Derived Rewrite Rules 114
5.2.3 Flat Terms and High Level Petri Boxes 118
5.2.3.1 Behaviours of Flat High Level Petri Boxes 119
5.2.3.2 A Low Level Box Model based on Flat High Level Petri Boxes? 119
5.3 Characterising the Structural Relationship induced by β 121
5.4 A Partial Order on the Structure of a High Level Petri Box 123
5.4.1 Partial Orders 123
5.4.2 A Partial Order on Local Transitions 124
5.4.2.1 Maximal Lines of $\Phi(B)$ 127
5.4.3 Further Properties of Local Transitions 128
5.4.3.1 η and α 130
5.5 Applications 135
5.5.1 stop as a Behavioural Unit of Choice 135
5.5.2 Commutativity and Associativity of Concurrent and Choice Composition 135
5.5.2.1 Extending the Rewrite System 138

5.6 Summary 139

6 Algebra and Semantics II **140**

6.1 Refinement 140
6.2 Order Sorted Algebra 141
6.3 Order Sorted Algebra Encoding of the High Level Petri Box Algebra 144
6.3.1 Process Variables 145
6.3.2 Characterisation of Refinement for High Level Petri Boxes 147
6.4 Semantics of Refinement 149
6.4.1 Local Transition Refinement 149
6.4.2 Properties of Local Transition Refinement 154
6.4.2.1 Basic Properties 154
6.4.2.2 Places 154
6.4.2.3 Local Transition Refinement modulo Isomorphism 155
6.4.2.4 Local Transitions 155
6.4.2.5 Commutativity of Local Transition refinement with Itself 155
6.4.2.6 Commuting Local Transition Refinement with the Top Level Operators 156
6.4.3 Full Refinement 158
6.4.3.1 Commutativity of Refinement with the High Level Petri Box Operators 159
6.5 Recursion 160
6.5.1 Semantics of Recursion 162
6.5.2 Pre- and Post-Guarding and Winding 162
6.5.2.1 Why Post-Guarding? 162
6.5.3 Winding 163
6.5.4 Silent Labels 164
6.5.5 The Guard Auxiliary Operator 166
6.5.5.1 Y-guarded High Level Petri Boxes 168
6.5.6 The Winding Auxiliary Operator 170
6.5.7 The Recursive Construct 173
6.6 The High Level Petri Box Syntax 175
6.6.1 Ground Terms and Free variables 176
6.6.2 Non-ground Terms and Structural Induction 176
6.7 Examples of the Recursive Construct 177
6.7.1 Tail-end Recursion 177
6.7.2 Unbounded Parallelism 178
6.7.3 Unbounded Parallelism and Synchronisation 178

6.7.4 Interwoven alignment preambles are not allowed 178
6.7.5 Turing Expressivity 181
6.8 Discussion 183
6.8.1 Other Net Theoretic Approaches to Variables and their Denotation 183
6.8.2 The Refinement Operators of the Low Level Petri Box Model 184
6.8.3 Tail-End and Non-Tail-End Recursion 187
6.8.3.1 Refinement and β-Reduction 187
6.9 Summary 188

7 Algebra and Behaviour **190**
7.1 Safeness and Memorylessness of Syntactically Generated High Level Petri Boxes 191
7.2 Strong Bisimulation on High Level Petri Boxes 208
7.2.1 Lifting Interface Respecting Relations to High Level Petri Boxes 209
7.2.2 Strong Bisimulation 211
7.2.2.1 Strong Bisimulation Preserves Concurrency 212
7.2.3 Congruence Nature of Strong Bisimulation 213
7.3 The Fix-Point Relationship between Refinement and Recursion 218
7.3.1 The Nature of the Behavioural Relationship Between $\Omega_\gamma(B)$ and $\mu Y.B$. 221
7.3.1.1 Formalising the Naming Convention 222
7.4 Discussion 228
7.4.1 Strong Bisimulation of High Level Petri Boxes 228
7.4.2 Strong Bisimulation as a Full Congruence 229
7.4.3 Weaker Bisimulations 230
7.4.4 Finite Approximants to Recursive Behaviours 232
7.5 Summary 234

8 Conclusions **235**
8.1 Summary 235
8.1.1 Label Algebras 235
8.1.1.1 H-Synchrony 235
8.1.2 The High Level Petri Box Model 236
8.1.3 The High Level Petri Box Algebra 236
8.1.4 Structural and Behavioural Characterisation of the Semantic Domain . . 238
8.2 Future Work 240

A Index of Notation **244**

List of Figures

1.1 The Conceptual Model of a High Level Petri Box 4

2.1 An example of a distributed transition system transition 13
2.2 An example of a pre-High Level Petri Box 29
2.3 Showing that the defining properties of High Level Petri Boxes are independent . 32
2.4 A High Level Petri Box 33
2.5 The definition of High Level Petri Box isomorphism 36

3.1 A High Level Petri Box and its PrT net denotation 44
3.2 A High Level Petri Box and its PrT net denotation 45
3.3 A High Level Petri Box at a non-locally-strict marking together with its PrT net denotation 49
3.4 In which two feasible substitutions between the same markings can be distinguished 53
3.5 A High Level Petri Box, B, in which the push/pop behaviour is not always well-defined 57
3.6 Illustrating the relationship between a PrT net and its unfolding 60

4.1 Parallel composition according to Taubner and Olderog 68
4.2 Illustrating the High Level Petri Box union of two High Level Petri Boxes 74
4.3 Illustrating the combinational closure of a High Level Petri Box 77
4.4 Illustrating the context manipulation of a High Level Petri Box 80
4.5 Illustrating the relational lifting of a sane relation to a High Level Petri Box . . . 81
4.6 Illustrating the concurrent composition of two High Level Petri Boxes 83
4.7 In which the behavioural commutativity of concurrent composition is compromised 84
4.8 Illustrating the causal composition of two High Level Petri Boxes 86
4.9 Illustrating the choice composition of two simple High Level Petri Boxes 88
4.10 Illustrating the choice composition of two High Level Petri Boxes 88
4.11 Illustrating the relabelling of a High Level Petri Box 89
4.12 The Basic High Level Petri Boxes 91
4.13 An unsafe High Level Petri Box 97
4.14 High Level Petri Boxes with memory 97

4.15 In which the unit pre-High Level Petri Box destroys representability by local transitions 103

4.16 Commuting diagram enabled by the safeness and memorylessness of High Level Petri Boxes 104

5.1 Demonstrating the necessity of a behavioural equivalence for a postulated rewrite rule 111

5.2 In which rewrite β does not produce a behavioural equivalence 112

5.3 'Rewritten' High Level Petri Boxes which are both structurally and behaviourally distinguishable even from a standard initial marking 113

5.4 A flat High Level Petri Box 119

5.5 The deduction of the normal form of a High Level Petri Box term. 120

5.6 Illustrating the application of the function α to a maximal line of a High Level Petri Box 131

5.7 In which an iterative concurrent composition rewrite is shown impractical 139

6.1 Characterising the free construction on an OSA 143

6.2 The extended semantic function. 146

6.3 The Basic High Level Petri Box for a process variable 148

6.4 Illustrating the local transition refinement of a High Level Petri Box 151

6.5 Illustrating the local transition refinement of a High Level Petri Box 152

6.6 Illustrating the local transition refinement of a High Level Petri Box 153

6.7 Illustrating the proof of Proposition 6.4.9 when $l^\bullet \subseteq B_1{}^\bullet$ 157

6.8 Regular and irregular denotations of High Level Petri Boxes with the same (observable) behaviour 163

6.9 Generating the language $\{a^n c b^n d \mid n \in \mathbb{N}\}$ with Petri nets 165

6.10 Guarding a High Level Petri Box 167

6.11 Y-guarding a High Level Petri Box 169

6.12 Winding a High Level Petri Box 172

6.13 The recursive construct applied to the conceptual model (finessed) 174

6.14 The denotation of the tail-end recursive term $\mu Y.((a;Y)+b)$ and its construction 179

6.15 A firing sequence for the recursive term $\mu Y.((a;Y)+b)$ 180

6.16 Unbounded parallelism in a finite High Level Petri Box: the denotation of the term $\mu Y.a \| Y$ 181

6.17 Synchronisation and unbounded parallelism in a finite High Level Petri Box . . . 182

6.18 A firing sequence for the recursive term $\mu Y.a \| (Y[r])[ccs]$ 183

6.19 The denotation of the inner loop of a High Level Petri Box term 184

6.20 Illustrating the behaviour of a High Level Petri Box term to which the recursive operator has been applied 185

6.21 A register 186

7.1 Illustrating the construction of the tree used in the proof of Lemma 7.1.2 200
7.2 Illustrating the proof that $B \approx B'$ implies $B[f] \approx B'[f]$. 215
7.3 Illustrating the structural relationship between $\mu Y.B$ and $\Omega_Y(B)$ 220
7.4 In which strong bisimilarity is shown not to be a congruence for non-syntactically generated High Level Petri Boxes 230
7.5 Weaker notions of strong bisimilarity are not necessarily algebra-wide congruences 233

List of Tables

3.1 Interrelation between High Level Petri Box Equivalences 62

5.1 The commutativity of the auxiliary operators 109
5.2 Proof of Conflation of Critical Pairs 116

8.1 Equivalences of High Level Petri Box Terms 239
8.2 Partial characterisation of transformation rules for a restricted combinational closure operator in combination with a renaming f 241

Acknowledgments

The research described herein was undertaken while the author was employed on the ESPRIT Basic Research Action DEMON (#3148) and Working Group CALIBAN (#6067) in the University of Newcastle upon Tyne. It was continued and completed while the author was employed on the EPSRC (formerly SERC) SCHEMA project (GR/G49531) in the University of York.

To Richard Hopkins (my original supervisor) goes my thanks for getting me started and his comments on earlier versions of the thesis. To Maciej Koutny (my final supervisor) goes my thanks for his careful reading of, and comments and suggestions on later versions of the thesis. Lucia Rapanotti has carefully read and made many useful comments on all versions of this thesis. Eike Best, Javier Esparza, Elisabeth Pelz and Oliver Botti have provided invaluable discussions on all things nets.

More recently, John McDermid employed me, and has had the patience of Job with me in his three year wait on the 'T' word. Jonathan Moffett has both encouraged me when down and made many useful comments on the presentation of the thesis. To both go my gratitude.

Shirley Craig has handled all things bibliographic with her usual wizardry. The staff of the Department of Computing Science of the University of Newcastle upon Tyne and of the Department of Computer Science of the University of York have provided a pleasant environment for work.

Donald Knuth deserves eternal recognition for his TEX[1], and Leslie Lamport and co. for their LATEX.

To my parents and brother go my thanks for their love and support over the years.

Lastly, but by no means least, Lucia, my wife, has waited, loved and cared for me even through her own thesis, without complaint, impatience or (as it must have seemed at times) end.

Lucia, I look forward to all the things we have planned to do together...

[1] Asymptotic to TEXπ.

Chapter 1

Introduction

Petri nets were introduced by C.A. Petri [Pet62] as a model of concurrency in which the causal relationship between actions, rather than just their temporal ordering, can be represented. With the ability to represent causal relationships comes a semantics of concurrency that does not depend upon the reduction of its components to the non-determinism that characterises so-called interleaving approaches (for instance, [Mil80, Hoa85]). Petri nets therefore provide a theoretical approach that distinguishes concurrency from non-determinism.

It appears always to have been the intention of Petri [Pet62] that his nets should provide practical tools for the designer and developer of real concurrent and distributed systems, and, indeed, much of the literature is taken with the investigation of the benefits and problems in the use of nets in this wider setting. In this context, the positive ability of nets to represent the basic situations of concurrent and distributed systems is far outweighed by the lack of structuring mechanisms, with the corollary that attempting the methodological synthesis of realistically sized systems produces nets that are large, unwieldy and opaque.

A partial solution to this problem is provided by *higher level* Petri nets. These have generally developed from the notion that the tokens that populate them may be imbued with an algebraic structure, suitable for the immediate representation of data items, abstract data types, even whole algebraic structures. From this enhancement comes much of their descriptive power; with the thus orthogonality of 'control flow' and 'data flow', we have a separation of concerns that well matches the situation in (imperative) programming languages, and leads to powerful models including Predicate Transition nets [GL81], Coloured Petri nets [Jen87], SEGRAS [Kra85, Kra87], Algebraic nets [Vau87, RV87, Rei91], Multi-sorted High Level Nets [Bil89], Abstract Data nets [BCC+86], OBJSA nets [BdCM88], CO-OPN [BG91a] and A-nets [KP95].

The expressive power of these higher level models is usually greatly increased over the original (low level) model of Petri, allowing a more compact representation, often finite, of situations that in the low level net model would be necessarily infinite. Many of them provide powerful structural and behavioural analysis techniques (for instance, [Gen89, Jen92]).

One widely voiced criticism of Petri nets is that they have, *per se*, no algebraic structure, and such criticism is not *a priori* addressed by the high level models. Although the criticism is certainly potent, it misses the mark in the sense that the status of Petri nets is as a model of concurrent and distributed systems, and so should be placed, theoretically, alongside, for instance, Hoare's traces [Hoa85] or Milner's synchronisation trees [Mil80], and other models on which an algebra may be built. A partial but immediate response to the criticism is to ask whether the algebraic structures of those two (and other) approaches have a meaning when interpreted on Petri nets. The literature now includes a full discussion of this question, including [dCdMPS83, DdNM85, DdNM87, DdNM88, DdNM90, Bes87, Gol88, GM84, Old87, Old91, HHB92, HH91], with a rather complete presentation in [Tau89].

The criticism will only be fully answered, however, when we find an approach that fully develops any inherent algebraic nature of Petri nets, in the same sense that CCS has developed as the algebraic characterisation of synchronisation trees or TCSP as the algebraic characterisation of traces. To do this we should first step back and see Petri nets not as complete but as requiring investigation into the expression they give to the semantic characterisations of situations and their interrelation and to express them within a suitable syntax.

It was with this intention that we began this work.

1.1 High Level Petri Box Algebra

Beyond this original direction, the motivation for the work presented here was to extend and complement the original Petri Box model [BH92, BDH92], which was based on the fundamental net class of (weighted) *Place/Transition* (P/T) nets, with one based on high level nets. The model we have chosen for this is the *Predicate Transition* (PrT) nets of Genrich and Lautenbach ([GL81, Gen87]). Through a more expressive class of high level nets it was originally hoped that the unbounded nature of computation that necessitates infinite structure in low level models could be encoded in the individuated tokens that may appear therein, allowing finite representations of process terms. However this is too demanding, even for high level nets as has been shown by Taubner [Tau89, Pg. 154], following Goltz [Gol88]: the process algebra TCSP [BHR84]

contains the possibility for synchronisations in which no *a priori* finite limit on the number of processes which partake can be given. Taubner argues that a high level net semantics as expressive as PrT nets must reflect this with infinite structure. Although we concur with Taubner that the underlying structure must be infinite, we show that the encoding of state in such nets may remain finite; only infinite numbers of transitions are required, places may remain finite with judicious choice of operators. Moreover, the notion of label algebra and its use in the High Level Petri Box model means that much useful reasoning can be performed on an associated, wholly finite graph.

Besides that mentioned above, another possibility for unbounded behaviour occurs in the modelling of data. However, the process algebra of nets we describe does not include 'data building' operations as basic. Such an explicit representation of (algebraic) data structures within high level models proceeds independently of this work, in very general terms and in ways that are orthogonal to the representation of this presentation; see, for instance, [BdCM88, BG91b, Rei91, Pet91, KP95, DK95].

Perhaps the most obvious possibility for unbounded behaviour occurs in control flow, through iteration and recursion. In the light of the comments above, we provide a 'cardinality preserving' semantics for recursion (from which operators equivalent to iteration are derivable). By 'cardinality preserving' we mean that there are High Level Petri Boxes to which recursion has been applied but for which the set of places is not of finite cardinality. However, it is not the modelling of the recursive construct that causes the infinity.

1.2 High Level Petri Boxes

A High Level Petri Box is a structured high level Petri net, the structure being indicated through a net annotation. The tokens of a High Level Petri Box are individuated, and carry information that affects the behaviour of the High Level Petri Box. The structure of a High Level Petri Box allows its presentation as a regular object with interfaces for *control* and *communication*, which may thus be used to abstract away from implementation details.

The conceptual model for a High Level Petri Box appears in Figure 1.1.

The low level net underlying a High Level Petri Box is a P/T net ([RT86]). called the *skeleton*. In notation adapted from [Gol88], the skeleton is 'transition centred', by which we mean that a transition is extensionally equal to any other with the same pre- and post-set (and annota-

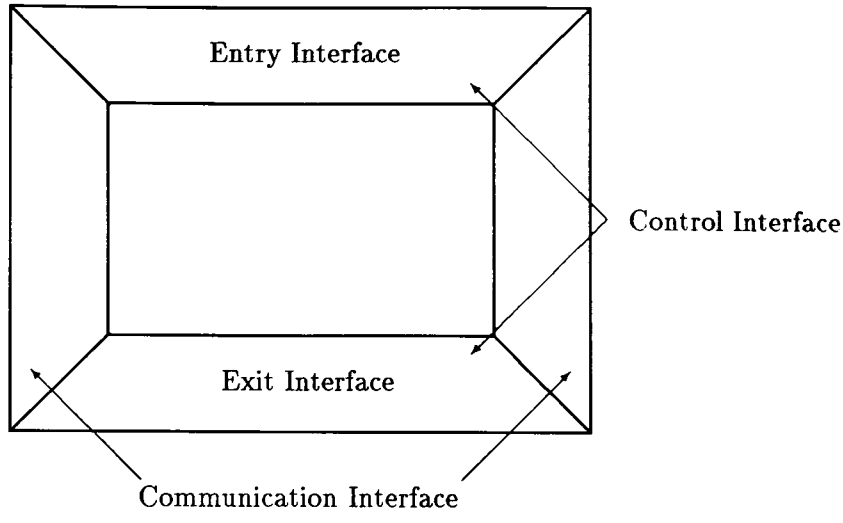


Figure 1.1: The Conceptual Model of a High Level Petri Box, showing the control and communication interfaces.

tion). Transition centredness has disadvantages, as we shall see, but is less disadvantaged than traditional notation for our purposes.

On top of the skeleton is built the annotation. Places, transitions and arcs each are annotated and identify components of a High Level Petri Box that can be manipulated through the constructions defined herein. This leads to a highly modular structure: we encourage the view of a High Level Petri Box as a single block whose discernible features are control and communication interfaces, and whose internal detail is hidden. This view creates a natural block structure on High Level Petri Boxes that leads to a highly modular model.

To synthesise High Level Petri Boxes the user must decide the details of the process to be modelled, for which we provide a *process algebra*, the *High Level Petri Box Algebra*, which is parametrised by a description of how a High Level Petri Box communicates with other High Level Petri Boxes, for which we provide *label algebras*. The coupling between a High Level Petri Box and the model it uses for communication is loose in that we may alter the communication protocol independently of the structure of the High Level Petri Box. One benefit of this is that the process algebra (described in Chapters 4 and 6) is compact, with the possibilities for increasing expressivity being through the manipulation of the communication model.

1.3 Structure of the Thesis

The thesis is structured as follows. Because of the wide range of topics touched on in the thesis, each chapter provides its own short survey of the relevant literature.

Chapter 2 gives the definitions upon which the structure of a High Level Petri Box is built. It begins with the introduction of a very general method for the description of communication protocols, so-called *label algebras*. Expressions within label algebras form the 'atomic actions' of the High Level Petri Box model. A characterisation of these atomic actions is given through a single Structured Operational Semantics-style (or SOS style, [Plo81]) rule, called *H-synchrony*, which not only includes and extends the behaviour of the traditional Asynchrony and Synchrony SOS rules for the description of concurrency but also that of Morphism (see, for instance, [Old91]).

The language from which are taken the individuated tokens of the high level model is described in terms of label algebras, and this allows us to investigate properties of the H-synchrony rule that are the basis for the properties of the behaviour of a well-behaved sub-class of High Level Petri Boxes.

The chapter continues with the definitions of High Level Petri Boxes and certain useful auxiliary operators, and safe and unsafe markings of High Level Petri Boxes.

The chapter concludes with the description of a partial order on High Level Petri Boxes based on the label algebra upon which the High Level Petri Box is defined.

Chapter 3 introduces the standard denotation of High Level Petri Boxes, in terms of a well-known and much studied class of high level Petri net, the PrT nets of Genrich and Lautenbach. The translation to PrT nets gives technical soundness of our approach, and leads to the definition of the formal notion of the behaviour of a High Level Petri Box, from which we see how the properties of the H-synchrony rule determine the behaviour of a sub-class of High Level Petri Boxes.

Also defined is a low level Petri net unfolding, which together with the PrT net semantics provides for the definition of a rich collection of structural and behavioural equivalences on High Level Petri Boxes.

Chapters 2 & 3 introduce the High Level Petri Box model. With Chapter 4 we commence the description and characterisation of the class of High Level Petri Boxes with the introduction of the first portion of the algebra, that including concurrent, causal and choice compositions, and

relabelling. After the definitions come a number of technical results that justify our claim to have established an algebraic structure upon High Level Petri Boxes, called the High Level Petri Box Algebra[1].

We conclude the technical contribution of the chapter with the demonstration of certain properties of the sub-class of High Level Petri Boxes defined in the chapter that witness their modular properties.

Chapter 5 shows that the sub-class of High Level Petri Boxes defined in Chapter 4 are well-behaved in the sense that their behaviour is to some extent independent of their initial marking. This result is used to transfer certain properties of the H-synchrony rule to the behaviours of this class of High Level Petri Boxes.

Chapter 6 completes the description of the class of High Level Petri Boxes with the introduction of the operators of refinement and recursion. Refinement is proved to have properties similar to those of the operation of β-reduction rule of the λ-calculus [Bar84], or equivalently, the substitution operators of Order Sorted Algebra [Gog78] (in terms of which it is defined).

The chapter continues with the definition of the last of our operators, recursion. With recursion comes full Turing expressibility of the High Level Petri Box Algebra, which is demonstrated by the modelling of a *register*.

Chapter 7 provides the major technical contribution of this work: a full behavioural characterisation of the High Level Petri Boxes which form the semantic domain of the High Level Petri Box Algebra. We also develop a concurrency preserving strong bisimulation on such High Level Petri Boxes.

We use this behavioural characterisation to show three other important results of such High Level Petri Boxes: that they are safe and memoryless; that concurrency preserving strong bisimulation is a congruence with respect to a large part of the algebra—concurrent, choice and causal compositions, and relabelling—and a partial congruence with respect to refinement and recursion; and that the recursive construct is a behavioural fix-point of the refinement operator (the being the strongest result generally possible given the finite nature of the construction).

Chapter 8 collects the main results of the thesis to provide an algebraic characterisation of High Level Petri Boxes as a model for the High Level Petri Box Algebra. We conclude with some pointers to where further effort might be gainfully employed.

[1] Extended in Chapter 6.

Chapter 2

The High Level Petri Box Model

In this chapter we introduce the semantic domain of the Algebra, High Level Petri Boxes. We begin with the description of the model for the very general communication protocols we allow, called *label algebras*.

2.1 Labels and Label Algebras

A *label algebra* describes how *labels* of a process algebra interact through synchronisation to allow multi-way synchronous communication between processes. The model for multi-way synchronisation is very general and is related to, but is strictly more powerful than, the *synchronisation algebras* of Winskel ([Win85]).

We first introduce our language for the description of multi-way synchronisations, through Label Algebras. We then show how more traditional models may be expressed as instances.

2.1.1 Labels

A *label* is the effect of an action, i.e., a label is that entity by which the occurrence of an action is recorded. We will assume that the collection of all labels forms a countable set, $\mathbb{L}$, called the *label universe*.

We assume that processes of a particular process algebra produce labels from a single *alphabet*: a non-empty subset of $\mathbb{L}$ (usually denoted A, decorated as necessary). Examples of alphabets are those of CCS ([Mil80]): $A_{CCS} = \{\tau, a, \overline{a}, b, \overline{b}, \ldots\}$; TCSP ([BHR84]): $A_{TCSP} = \{\tau, a, b, \ldots\}$;

and the PBC ([BDH92, BH92]): $A_{PBC} = \mathcal{M}_f(\{a, \overline{a}, b, \overline{b}, \ldots\})$[1].

Concurrent processes wishing to synchronise contribute their individual labels to a (commutative, associative) *formal sum*. The traditional model of parallel composition is as a (behaviourally) commutative operator, which is the reason for a formal sum and not an ordered structure, such as a tuple. Moreover, a set is not necessarily sufficient, as many processes may wish to contribute the same label. For example, the formal sum $a_1 \dot{+} \cdots \dot{+} a_n$ represents the attempted synchronisation of n processes, one of which contributes a_1, one of which contributes a_2, *etc.* The 'wish' to synchronise is not sufficient. Processes must be allowed to synchronise by the communication model. For instance, a process contributing a together with a process contributing a would not be permitted to synchronise in the CCS communication model, but would in the TCSP model.

To produce the synchronisations of a particular communication model, then, we must be able to manipulate formal sums of labels. To this end we introduce *label algebras*.

2.1.2 Label Algebras

A *label algebra* provides a representation for the interaction of labels. A label algebra is a 5-tuple, $\mathcal{L} = \langle A, \oplus, \mathbf{0}, \iota, R \rangle$ where A is an alphabet, $\langle A^{\oplus}, \oplus, \mathbf{0} \rangle$ is a monoid (commutative operation $\dot{+}$, zero $\mathbf{0}$), called the *label monoid* (providing the formal sums mentioned above); R is a countable set of *relabellings*; ι is the (natural) inclusion of A in $A^{\oplus}$.

$\mathbf{0}$, the zero of the label monoid, represents the dead action, i.e., an action which may never occur. A relabelling $r \in R$ is a mapping on the label monoid $r: A^{\oplus} \to A^{\dot{+}}$ which although not necessarily being a monoid homomorphism, is such that $r(\mathbf{0}) = \mathbf{0}$, meaning that the dead action is never 'reincarnated'. The general form of a relabelling will, in addition to disallowing a synchronisation (by mapping to $\mathbf{0}$), and allowing a synchronisation (by mapping to an element of A), allow a third possibility of *don't know* (by mapping it to a member of $A^{\dot{+}} \setminus (\iota(A) \cup \{\mathbf{0}\})$). Such a *dormant* synchronisation, as it were, is disallowed from occurring as it stands, but may contribute its label to a further attempted synchronisation, as we shall see.

We will usually omit the monoid operation, zero and inclusion components of a label algebra, writing $\langle A, \oplus, \mathbf{0}, \iota, R \rangle$ as $\langle A, R \rangle$.

EXAMPLE 2.1.1 Let $\mathcal{L}_0 = \langle A_0, R_0 \rangle$ be such that $A_0 = \{\tau, a, \overline{a}\}$ and $R_0 = \{ccs, r\}$ with

[1] Where $\mathcal{M}_f(S)$ is all finite multisets on the set S.

$ccs(a \oplus \overline{a}) = ccs(\overline{a} \oplus a) = \tau$, $ccs(a) = a$, $ccs(\overline{a}) = \overline{a}$, $ccs(\tau) = \tau$ and $ccs(k) = 0$ otherwise; $r(a) = \overline{a}$, $r(\overline{a}) = a$, $r(\tau) = \tau$, $r(0) = 0$ and $r(l_1 \oplus \cdots \oplus l_n) = r(l_1) \oplus \cdots \oplus r(l_n)$ otherwise.

The actions of the relabellings may be described as:

1. the action of all relabellings on **0** is prescribed,
2. processes offering a and $\overline{a}$, and wishing to synchronise will offer $a \oplus \overline{a}$. Consider the label $\overline{a}$ as the conjugate of label a in the CCS sense; this is allowed by the CCS-type synchronisation, ccs, which maps it to τ,
3. as above, processes offering a and a, and wishing to synchronise will offer $a \oplus a$. Extending the analogy with CCS, this should not contribute to behaviour: the action of ccs is to disallow the synchronisation by mapping $a \oplus a$ to **0** which, irreversibly, prevents it from contributing to behaviour,
4. ccs's action on τ is consistent with the CCS analogy.
5. as well as being a relabelling, r is also a monoid homomorphism and such relabellings play no part in forming synchronisations. ■ 2.1.1

2.1.3 Expressivity of Label Algebras

Our multi-way synchronisations are not limited to binary synchronisations as in CCS, nor to synchronisations of TCSP and the Low Level Petri Box model in which synchronisation is no longer necessarily binary but which may always be decomposed into iterated binary synchronisation. That this forms a strict extension to the expressivity of the synchronisation algebras of Winskel [Win85] is shown in the following example:

EXAMPLE 2.1.2 Let $\mathcal{L}_a = \langle A_a, R_a \rangle$ be the label algebra with $A_a = \{a\}$ and $R_a = \{f_3\}$ and such that:

$$f_3(k) = \begin{cases} \tau & k = a \oplus a \oplus a \\ \mathbf{0} & \text{otherwise} \end{cases}$$

Then $\mathcal{L}_a$ disallows all forms of synchronisation other than tertiary. Moreover, there is no binary decomposition of such synchronisations in this label algebra. ■ 2.1.2

2.1.3.1 Taubner's Label Algebra

In [Tau89], Taubner introduces a relatively general method for the expression of (binary) synchronisation, renaming, restriction and hiding operators based on the union of the features of those operators in process algebras such as CCS, TCSP and ACP ([BW90]). Each synchronisation operator of these generating languages is binary, and Taubner assumes the same binary nature for his, proposing unordered pairs of basic actions to represent synchronisation.

In more detail, Taubner assumes the existence of a countably infinite alphabet *Alph* of names; a disjoint copy of *Alph*, $\overline{Alph}$, of *conames*, together with a bijection $\bar{\cdot}: Alph \to \overline{Alph}$ (with inverse also written $\bar{\cdot}$ so that $\bar{\bar{a}} = a$, for instance). His *visible actions*, *Vis*, are taken from $Alph \cup \overline{Alph}$. Unordered pairs of visible actions are *extended visible actions*, *EVis*. His action alphabet definition is completed through the inclusion of Milner's silent action τ:

$$Act = \{\tau\} \cup Vis \cup EVis$$

Taubner does not explicitly state that *Vis* and *EVis* are disjoint. However, it is clearly a simplifying assumption that they are disjoint as this implies that a synchronisation is composed only of (completed) binary synchronisations; and otherwise Taubner's parallel composition operator ($\between$) is not behaviourally associative: $(a \between b) \between c$ being able to perform the action $[[a, b], c]$ but not $[a, [b, c]]$. As associativity of parallel composition is to be desired we assume that Taubner requires *Vis* and *EVis* to be disjoint, in which case $[a, b]$ does not appear in *Vis*, whence it is not available for $(a \between b)$ to perform.

In the representation of Taubner's communication model as a label algebra, we need not explicitly include the unordered pairs of *EVis* in the alphabet, as these may be represented by sums of length two. Moreover, Taubner's *undefined* symbol, $\perp$, may be represented by our dead label **0**. We may thus define the alphabet of Taubner's algebra, **A**, in our setting as $A_{\mathbf{A}} = \{\tau\} \cup Alph \cup \overline{Alph}$, with $Alph = \{a, b, c, \ldots\}$.

The equivalent of a relabelling in Taubner's algebra is called an *action manipulation operator*. Any function $f: Act \cup \{\mathbf{0}\} \to Act \cup \{\mathbf{0}\}$ which preserves **0** and τ is allowed as an action manipulation operator. Taubner does not explicitly state whether the range of an action manipulation operator may have non-empty intersection with *EVis*. While allowing this appears to have no undesirable ramifications, Taubner appears never to use this property in his exposition. For completeness we do allow it, defining

$$R_{\mathbf{A}} = \{f \in Act^{\oplus} \to Act^{\oplus} \mid f(\mathbf{0}) = \mathbf{0} \wedge f(\tau) = \tau \wedge f(k) = \mathbf{0} \text{ when } |k| > 2\}$$

It is a simple matter to define the label algebra of Taubner's algebra A as:

DEFINITION 2.1.3 *[Taubner's Label Algebra]* $\mathcal{L}_{\mathbf{A}} = (A_{\mathbf{A}}, R_{\mathbf{A}})$ is the label algebra for A.

■ 2.1.3

2.1.4 Label Algebras for CCS, TCSP, and PBC

For comparison with Taubner's descriptions of the communication models of CCS ([Mil80]) and TCSP ([BHR84]) we give the label algebra encoding. Also included is the label algebra for the communication model of the (Low Level) Petri Box Calculus ([BDH92]).

In the sequel, let A be the action alphabet $\{a, b, c, \ldots\}$, $\overline{A}$ the set $\{\overline{a}, \overline{b}, \overline{c}, \ldots\}$, disjoint from A, and suppose $\tau \notin A \cup \overline{A}$. We assume that the label universe, $\mathbb{L}$, contains $A \cup \overline{A} \cup \{\tau\}$.

2.1.4.1 CCS Label Algebra

For CCS we define:

DEFINITION 2.1.4 *[CCS Label Algebra]* Let $A_{CCS} = \{\tau\} \cup A \cup \overline{A}$ be the alphabet for CCS.

Define relabellings for *CCS synchronisation*, Ψ_{CCS} such that

$$\Psi_{CCS}(k) = \begin{cases} k & k \in A_{CCS} \\ \tau & k = l \oplus \overline{l} \text{ and } l \in A \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

For $B \subseteq A_{CCS} \setminus \{\tau\}$, define relabellings for *CCS restriction* Υ^B_{CCS} such that

$$\Upsilon^B_{CCS}(k) = \begin{cases} \mathbf{0} & k \in B \\ k & k \in A_{CCS} \setminus B \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

A relabelling f is a *CCS relabelling* whenever $f(A_{CCS}) \subseteq A_{CCS}$, $f(\overline{a}) = \overline{f(a)}$, $f(l \div k) = f(l) \div f(k)$ for $l, k \in A_{CCS}$, and $f(k) = \mathbf{0}$ otherwise. Let $\Theta_{CCS} = \{f \mid f \text{ is a CCS relabelling}\}$.

Define $\mathcal{L}_{CCS} = (A_{CCS}, \{\Psi_{CCS}\} \cup \{\Upsilon^B_{CCS}\}_{B \subseteq A_{CCS} \setminus \{\tau\}} \cup \Theta_{CCS})$ as the label algebra for CCS.

■ 2.1.4

2.1.4.2 TCSP Label Algebra

For TCSP we define:

DEFINITION 2.1.5 *[TCSP Label Algebra]* Let $A_{TCSP} = \{\tau\} \cup A$ be the alphabet for TCSP. For $B \subseteq A_{TCSP} \setminus \{\tau\}$ define relabellings for synchronisation Ψ^B_{TCSP}, for restriction Υ^B_{TCSP} and for hiding Θ^B_{TCSP} such that

$$\Psi^B_{TCSP}(k) = \begin{cases} k & k \in A_{TCSP} \setminus B \\ l & k = l \oplus l \text{ and } l \in B \\ \mathbf{0} & \text{otherwise} \end{cases}$$

$$\Upsilon^B_{TCSP}(k) = \begin{cases} \mathbf{0} & k \in B \\ k & k \in A_{TCSP} \setminus B \\ \mathbf{0} & \text{otherwise} \end{cases}$$

$$\Theta^B_{TCSP}(k) = \begin{cases} \tau & \text{if } k \in B \\ k & \text{if } k \in A_{TCSP} \setminus B \\ \mathbf{0} & \text{otherwise} \end{cases}$$

Define $\mathcal{L}_{TCSP} = (A_{TCSP}, \{\Psi^B_{TCSP}, \Upsilon^B_{TCSP}, \Theta^B_{TCSP}\}_{B \subseteq (A_{TCSP} \setminus \{\tau\})})$ as the label algebra for TCSP. ■ 2.1.5

2.1.4.3 PBC Label Algebra

For the PBC we define:

DEFINITION 2.1.6 *[PBC Label Algebra]* Let $A_{PBC} = \mathcal{M}_f(A \cup \overline{A})$, the finite multisets on $A \cup \overline{A}$. Synchronisation in the PBC is parametrised by a label, so let $a \in A$. Let $k \in A^{\dagger}_{PBC}$. Then $k = l_1 \oplus \ldots \oplus l_s \oplus l_{s+1} \oplus \ldots \oplus l_{s+n}$ where $\{\!|a, \overline{a}|\!\} \cap l_i \neq \emptyset$[2] if and only if $i \leq s$ (always possible as $\oplus$ is commutative). Define

$$\mathbf{sy}\, a(k) = \begin{cases} k & k \in A_{PBC} \wedge \{\!|a, \overline{a}|\!\} \cap k = \emptyset \\ (\sum_{i=1}^{s} l_i) \setminus (s-1)\{\!|a, \overline{a}|\!\} & n = 0 \wedge \min(\sum_{i=1}^{s} l_i(a), \sum_{i=1}^{s} l_i(\overline{a})) \geq s - 1 \\ \mathbf{0} & \text{otherwise} \end{cases}$$

$$\mathbf{rs}\, a(k) = \begin{cases} k & k \in A_{PBC} \wedge \{\!|a, \overline{a}|\!\} \cap k = \emptyset \\ \mathbf{0} & \text{otherwise} \end{cases}$$

Define $\mathcal{L}_{PBC} = (A_{PBC}, \{\mathbf{sy}\, a, \mathbf{rs}\, a\}_{a \in A_{PBC}})$ as the label algebra for the PBC. ■ 2.1.6

[2] We use $\{\!|s_1, \ldots, s_n|\!\}$ to represent the multiset consisting of the elements $s_1, \ldots, s_n$. $\{\!|\,|\!\}$ is the empty multiset. Set operators such as $\cup$, $\cap$, and relations $\in$, $\subseteq$, $\supseteq$, *etc.* will not, in general, be distinguished from their multiset counterparts; which to use being distinguished by the type of the operands. Definitionally, we use the following: for a multiset S, $S(s)$ is the multiplicity of s in S and $s \in S$ if and only if $S(s) > 0$; for multisets S and T, $(S \cup T)(s) = S(s) + T(s)$, $(S \cap T)(s) = min(S(s), T(s))$ and $(S \setminus T)(s) = max(S(s) - T(s), 0)$; $S \subseteq T$ if and only if $S(s) \leq T(t)$ for all $s \in S$; for a positive integer n, $(n.S)(s) = n.s$.

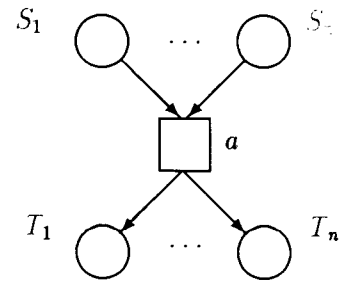


Figure 2.1: The distributed transition system transition $\{S_1, \ldots, S_n\} \xrightarrow{a} \{T_1, \ldots, T_n\}$.

2.2 Behaviour of High Level Petri Boxes

Although it may appear a little premature to begin a discussion of the behaviour of a High Level Petri Box, the motivation for many aspects of our approach may be found in the derivation of the behaviour of a single transition. In this section we introduce a generalised synchrony rule for the description of multi-way synchronisations in distributed transition systems as the basis of this behaviour.

In being a Petri net, the behaviour of a High Level Petri Box will follow from this in combination with the interrelationship between transitions defined by the graph structure of the skeleton.

2.2.1 Distributed Transition Systems

Essentially, the relationship between the *Distributed Transition Systems* (DTS) of Degano *et al.* [DdNM87, DdNM88] and Petri nets is that the *sequential components of a term* in their algebra may be identified with the places of a net and that a (DTS) transition between two sets of sequential components, $\{S_1, \ldots, S_n\} \xrightarrow{a} \{T_1, \ldots, T_n\}$, may be identified with a transition. Such a transition is illustrated in Figure 2.1.

The details of the decomposition of a term into its sequential components is not of relevance to our presentation, other than the fact that sequential components are recombined into (full) processes using the $\cup$ operator; we also use distributed union in this role below. The essential detail that we assume of a single sequential component, i.e., when $n = 1$, is that it performs its *first action*[3] with label a, as long as $a \neq 0$[4]. The transitions of High Level Petri Boxes are motivated by DTS transitions in the following way.

[3]Expressed through prefix in [DdNM88], but we prefer sequential composition (see Chapter 4 for a discussion).

[4]The expression of deadlock as a *dead* action is supported in the notation of label algebras, and is discussed with reference to the expression of causal dependence through sequential composition through prefix in Chapter 4.

Binary asynchronous parallel composition in a distributed transition system is modelled by the disjoint unions of its operands. Degano introduces *contexts* $\vdash$ and $\dashv$[5] with which he annotates the operands of the union, so ensuring disjointness. Contexts do not, by themselves, alter the behaviour of a process as the transition rule for asynchrony (adapted from the elegant presentation of distributed transition systems given by Olderog in [Old91]) shows:

$$\text{Asynchrony} \quad \frac{P \xrightarrow{a} P'}{P[\vdash] \xrightarrow{a} P'[\vdash]} \qquad \frac{P \xrightarrow{a} P'}{P[\dashv] \xrightarrow{a} P'[\dashv]}$$

Moreover, the appearance of contexts within a concurrent process indicates which sub-processes should be considered concurrent as the synchrony rule shows. The synchrony rule presented below is adapted from Olderog, but is more general in that it allows the label of the synchronisation to be different from those of its constituent actions (Olderog uses the COSY synchronisation model ([Bes87])). We have expressed this in terms of a relabelling Ψ, taken from some label algebra. However, in this case, the rule finds equally natural expression through the relabellings of the synchronisation algebras of Winskel:

$$\text{2-synchrony} \quad \frac{P \xrightarrow{a} P' \qquad Q \xrightarrow{b} Q'}{P[\vdash] \cup Q[\dashv] \xrightarrow{\Psi(a \oplus b)} P'[\vdash] \cup Q'[\dashv]} \; \Psi(a \oplus b) \neq 0$$

where Ψ is a binary synchronisation in the sense of Winskel and $\cup$ is the a 'disjoint union of processes'.

The commutativity of the formal sum $\oplus$ implies that we may interchange P and Q in the 2-synchrony rule and not be able to distinguish the result by its behaviour. This is, of course, the behavioural commutativity of parallel composition.

As we may continue to distinguish operands using $\vdash$ and $\dashv$ we may derive higher order synchronisations from 2-synchrony. For instance, two applications give:

$$\text{3-synchrony} \quad \frac{\dfrac{P \xrightarrow{a} P' \qquad Q \xrightarrow{b} Q'}{P[\vdash] \cup Q[\dashv] \xrightarrow{\Psi(a \oplus b)} P'[\vdash] \cup Q'[\dashv]} \; \Psi(a \oplus b) \neq 0 \qquad R \xrightarrow{c} R'}{P[\vdash\vdash] \cup Q[\dashv\vdash] \cup R[\dashv] \xrightarrow{\Psi(\Psi(a \oplus b) \oplus c)} P'[\vdash\vdash] \cup Q'[\dashv\vdash] \cup R'[\dashv]} \; \Psi(\Psi(a \oplus b) \oplus c) \neq 0$$

However, such explicit rules for n-synchrony, for each $n \in \mathbb{N}$, are not required as they may be derived.

Within the High Level Petri Box model we will also model asynchronous parallel composition by disjoint union (of nets, called concurrent composition, and defined in Chapter 4) and use

[5] Degano actually uses the symbol $|_{id}$ written postfix and $_{id}|$ written prefix to distinguish left and right operands, but the effect is equivalent.

contexts as Degano does, to ensure disjointness. Moreover, through a unification of contexts and relabellings, so that contexts are considered as *nominal relabellings*[6], we are able to combine morphism, asynchrony, and synchrony rules into a single rule, which we will call *H-synchrony*[7], that allows the expression of the more general communication models which are expressible through label algebras (and because of this, is not derivable from any simple rule it replaces).

2.2.2 H-synchrony

The H-synchrony rule is described by the following (distributed) transition:

$$\text{H-synchrony}\ \frac{P_i \xrightarrow{a_i} P_i'}{\bigcup_i P_i[R_i] \xrightarrow{\text{Label}(a_1,\ldots,a_n,R_1,\ldots,R_n)} \bigcup_i P_i'[R_i]}\ \text{Flow}(a_1,\ldots,a_n,R_1,\ldots,R_n)$$

Where the P_i are processes, R_i are strings of relabelling names (which may now include $\vdash$ and $\dashv$) so that $P_i[R_i]$ is process P_i in the scope of relabellings R_i. The definitions of $\text{Label}(a_1,\ldots,a_n,R_1,\ldots,R_n)$ and of $\text{Flow}(a_1,\ldots,a_n,R_1,\ldots,R_n)$ are rather complex, as we shall see; so as to be able to fully motivate their form, we approach their definition through a comparison with the rules replaced.

2.2.2.1 Relabelling

The relabelling rule of Olderog [Old91][8] is parametrised by a relabelling[9] r and is given by:

$$\text{Relabelling}\ \frac{P \xrightarrow{a} P'}{P[r] \xrightarrow{r(a)} P'[r]}\ r(a)\neq 0$$

i.e., given a process which produces action a, the process P in the scope of relabelling r may perform action $r(a)$, as long as $r(a)$ is not the dead label. Now, for a single process, the H-synchrony rule is

$$\text{H-synchrony}\ \frac{P \xrightarrow{a} P'}{P[r] \xrightarrow{\text{Label}(a,r)} P'[r]}\ \text{Flow}(a,r)$$

Comparing the two we see that, if $\text{Label}(a,r)$ equals $r(a)$ and $\text{Flow}(a,r) \Rightarrow r(a) \neq \mathbf{0}$, H-synchrony includes the behaviour of relabelling. When r is a relabelling we define $\text{Label}(a.r)$

[6] Cf. Taubner [Tau89].

[7] The 'H' in H-synchrony is derived from the juxtaposition of the nominal relabellings $\vdash$ and $\dashv$, so: $\vdash\!\dashv$, pronounced H.

[8] Actually called morphism, but we use Winskel's language.

[9] A unary relabelling in [Old91].

to be $r(a)$. The full definition of Flow is complex and is given in Section 2.4. It will, however, be seen to have the property that $\text{Flow}(a, r) \Rightarrow r(a) \neq \mathbf{0}$.

2.2.2.2 Asynchrony

The two rules for asynchrony (given above) may be regarded as particular instances of relabelling, given that we have allowed $\vdash$ and $\dashv$ to be considered as nominal relabellings. Moreover, we see that, when interpreted as relabellings we should regard them as being the identity relabelling.

2.2.2.3 2-synchrony

Using juxtaposition to represent the composition of relabellings (so that $(x)\vdash\Psi = \Psi(\vdash(x))$, for instance) we may derive the 2-synchrony rule from H-synchrony with two processes, thus:

$$\frac{P \xrightarrow{a} P' \qquad Q \xrightarrow{b} Q'}{P[\vdash\Psi] \cup Q[\dashv\Psi] \xrightarrow{a[\vdash\Psi]\oplus b[\dashv\Psi]} P'[\vdash\Psi] \cup Q'[\dashv\Psi]}\ \text{Flow}(a,b,\vdash\Psi,\dashv\Psi)$$

when $a[\vdash\Psi] \oplus b[\dashv\Psi] = \Psi(a \oplus b)$ and $\text{Flow}(a, b, \vdash\Psi, \dashv\Psi) \Rightarrow \Psi(a \oplus b) \neq \mathbf{0}$. The behavioural commutativity of parallel composition implies that $\Psi(a \oplus b) = \Psi(b \oplus a)$, reflecting our assumption that $\oplus$ is commutative, and implying that $a[\vdash\Psi] \oplus b[\dashv\Psi] = a[\dashv\Psi] \oplus b[\vdash\Psi]$.

2.2.2.4 Expressivity of H-synchrony

In the context of the expressive power of label algebras, it is interesting to note that H-synchrony with three processes is not necessarily 3-synchrony as described above. Consider:

$$\frac{P \xrightarrow{a} P' \qquad Q \xrightarrow{b} Q' \qquad R \xrightarrow{c} R'}{P[\vdash\vdash\Psi] \cup Q[\dashv\vdash\Psi] \cup R[\dashv\Psi] \xrightarrow{\Psi(a\oplus b\oplus c)} P'[\vdash\vdash\Psi] \cup Q'[\dashv\vdash\Psi] \cup R'[\dashv\Psi]}\ \Psi(a\oplus b\oplus c)\in A$$

in which a three way communication has been formed without the requirement of being decomposable into binary synchronisations. We will therefore be able to use the full expressive power of label algebras through the H-synchrony rule and hence, as High Level Petri Box transitions are derived from it, in High Level Petri Boxes as well.

2.3 Contexts in High Level Petri Boxes

In High Level Petri Boxes, rather than a *context* being only a $\vdash$ or a $\dashv$, it will be a string of relabelling names and nominal relabellings, as in [Tau89].

Like the DTS approach, we have a natural decomposition of a process into its sequential components, through $\vdash$s and $\dashv$s. Unlike the DTS approach, however, the block structure of High Level Petri Boxes, expressed through the conceptual model, naturally introduces the notion of the *scope* of a relabelling. There are, thus, two uses of contexts in the structure of High Level Petri Boxes:

1. as annotation of input and output arcs of a High Level Petri Box, defining the beginning and ending of scope for relabellings inscribed thereon, and
2. as a token holding a *sequential component* of the process, together with the relabellings that are currently in scope.

Intuitively, a token 'passing along an arc from a place to a transition' has the context annotating that arc 'pushed' onto it. Similarly, a token 'passing along an arc from a transition to a place' has the context annotating that arc 'popped' from it. The popping of a context from a token in this way is not always well defined. However, for the class of *syntactically generated* High Level Petri Boxes defined in Chapter 4 we show that the enabling of a transition never depends on the well-definedness of this popping.

Unlike Taubner, and due to the special requirements of the choice composition in the presence of refinement (discussed in Chapter 4) and of recursion in the presence of sequential composition (discussed in Chapter 6), we introduce three other nominal relabellings, $\lfloor$, $\rfloor$ and $\perp$, each of which will be interpreted, like $\vdash$ and $\dashv$, as the identity relabelling on labels. We thus require, for any label algebra $\mathcal{L} = \langle A, R \rangle$, that $R \cap \{\vdash, \dashv, \lfloor, \rfloor, \perp\} = \emptyset$.

DEFINITION 2.3.1 *$\mathcal{L}$-context* Given a label algebra $\mathcal{L} = \langle A, R \rangle$, an $\mathcal{L}$-context is a term in the language defined through the following syntax

$$
\begin{array}{lll}
C ::= & . & \text{the trivial context} \\
| & Cf & f \in R \\
| & CN & N \in \{\vdash, \dashv, \lfloor, \rfloor, \perp\}
\end{array}
$$

A *simple context* is a context in which no nominal relabellings appear. ■ 2.3.1

We will denote the language of contexts on the label algebra $\mathcal{L}$ by $\mathcal{C}_{\mathcal{L}}$ although we will often omit the label algebra decoration when it is clear by context. Parentheses will often be used to delimit contexts especially in the case of the trivial context, when we will write (.) instead of just . .

The operation of pushing is actually only context concatenation:

DEFINITION 2.3.2 Define *context concatenation*, $\_^\frown\_ : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ such that[10] $C^\frown D = D[C/.]$.

■ 2.3.2

$C^\frown D$ will often be abbreviated to CD.

2.3.1 Signed Contexts

Although the action of popping contexts from a sub-class of High Level Petri Boxes is always well-defined, we may not assume this for all High Level Petri Boxes. To catch the possibility of 'underflow' from a pop we will define *signed contexts*:

DEFINITION 2.3.3 A *signed context* is a term of the following syntax:

$$D ::= +C \mid -C$$

where $C \in \mathcal{C}$. The collection of signed contexts will be denoted $\mathcal{SC}$. ■ 2.3.3

We will make the usual correspondence between $\mathcal{C}$ and $\mathcal{SC}$ so that $+C$ will usually be written C and $\mathcal{C}$ will be regarded as a subset of $\mathcal{SC}$. We will identify the representations of the empty stack, it being either (.) or $-(.)$. We will also define a unary minus operator to convert contexts to signed contexts and back again:

DEFINITION 2.3.4 Define a unary operator $-: \mathcal{SC} \to \mathcal{SC}$ such that for a signed context C,

$$-(C) = \begin{cases} -C & C \in \mathcal{C} \\ C' & C = -C' \wedge C' \in \mathcal{C} \end{cases}$$

■ 2.3.4

$\mathcal{C}$ has two useful partial orderings: *is-a-prefix-of* and *is-a-postfix-of*:

[10] Where $A[C/B]$ is the simultaneous textual replacement of B by C throughout A.

DEFINITION 2.3.5 Define the relations $\leq$, $\sqsubseteq$ on $\mathcal{C}$ such that:

1. $C \leq D$ (C *is a prefix of* D) if and only if there is a context E such $D = C \frown E$.

2. $C \sqsubseteq D$ (C *is a postfix of* D) if and only if there is a context E such $D = E \frown C$. ■2.3.5

As usual, we will use the symbols $<$ and $\sqsubset$ when the ordering is strict, i.e., when, in addition, $E \neq (.)$.

The binary operation of context minus finds, for a pair of contexts comparable under $\leq$, the longest suffix by which they differ. It uses signed contexts to record the information as to which of the pair was the longer context:

DEFINITION 2.3.6 Define a partial binary operator, $\_ - \_ : \mathcal{C} \times \mathcal{C} \rightarrow \mathcal{SC}$, by

$$C - D = \begin{cases} E & C = D \frown E, \text{ i.e., } D \leq C \\ -E & D = C \frown E, \text{ i.e., } C \leq D \end{cases}$$

where $E \in \mathcal{C}$. ■ 2.3.6

That $\_ - \_$ is partial is a corollary of the fact that not all contexts are comparable under $\leq$. Note that, when $C = D$, $C - D = (.) = D - C$.

The relationship between contexts and manipulations of labels may now be defined:

DEFINITION 2.3.7 Let $\mathcal{L} = \langle A, R \rangle$ be a label algebra, and let $k \in A^{\oplus}$. For $C \in \mathcal{C}_{\mathcal{L}}$ we define

$$k[C] = \begin{cases} k & C = (.) \\ f(k)[C'] & C = (.f) \frown C' \text{ with } f \in R \\ k[C'] & C = (.f) \frown C' \text{ with } f \in \{\vdash, \dashv, \lfloor, \rfloor, \perp\} \end{cases}$$

■ 2.3.7

EXAMPLE 2.3.8 Examples of the applications of the operators and relations defined above are:

1. $(.fh) \frown (.g) = (.fh)(.g) = (.g)[.fh/.] = (.fhg)$,

2. $(.) = -(.)$, by definition,

3. $(.f) \leq (.fg)$ and $(.f) < (.fg)$.

4. $(.g) \sqsubseteq (.fg)$ and $(.g) \sqsubset (.fg)$,

5. $(.fg) - (.f) = (.g)$,

6. $(.f) - (.fg) = -(.g)$,

7. $(a \oplus \overline{a})[.\Psi_{CCS}] = (\Psi_{CCS}(a \oplus \overline{a}))[(.)] = \tau[(.)] = \tau$. ■ 2.3.8

2.4 The Flow Predicate

The conditions for the derivation of 2-synchrony from H-synchrony require that $a[\vdash\Psi] \oplus b[\dashv\Psi] = \Psi(a \oplus b)$ or, more generally, that $a[\phi_1\vdash\Psi] \oplus b[\phi_2\dashv\Psi] = \Psi(\phi_1(a) \oplus \phi_2(b))$. which will be seen to be a corollary of the definition of the Flow predicate. This extends to the 3-synchrony rule as $(a[\vdash\vdash\Psi] \oplus b[\dashv\vdash\Psi]) \oplus c[\dashv\Psi] = \Psi(a \oplus b \oplus c)$, so that repeated synchronisations (without intermediate binary synchronisations) are independent of the order of the compositions. The properties of higher order synchronisations follow from this by a natural extension.

Note that, in general, $a[\phi_1\Psi] \oplus b[\phi_2\Psi] \neq a[\phi_1\vdash\Psi] \oplus b[\phi_2\dashv\Psi] = \Psi(\phi_1(a) \oplus \phi_2(b))$, which explains Taubner's observation of the same, [Tau89, page 124]. We also note that the equality holds when Ψ is a monoid homomorphism.

Given this, the Flow predicate must combine the following observations of the synchronisation of the $P_i[R_i]$ of the H-synchrony rule:

1. that they be executing concurrently, which is expressed through the structure of the $\vdash$s and $\dashv$s which appear in the (contexts) R_i,

2. that no 'sub-synchronisation' is a deadlock; for example, $a[\phi_1\vdash\Psi] \oplus b[\phi_2\dashv\Psi]$ should not be allowed when either $\phi_1(a) = \mathbf{0}$ or $\phi_2(b) = \mathbf{0}$,

3. that the action produced by the whole synchronisation is from the alphabet of the process. Of course, we might allow a 'sub-synchronisation' to be an arbitrary non-zero formal sum,

4. that the order of the asynchronous parallel compositions which produced the concurrency between the P_i should not be important to the generated label. For instance, and as we have seen, we will require that $a[\vdash\vdash\Psi] \oplus b[\dashv\vdash\Psi] \oplus c[\dashv\Psi] = a[\vdash\Psi] \oplus b[\vdash\dashv\Psi] \oplus c[\dashv\dashv\Psi] = \Psi(a \oplus b \oplus c)$.

Intuitively, the definition of Flow may be approached inductively on the number of its arguments, as follows:

1. we should have that $\text{Flow}(a_1, R_1) \Leftrightarrow a_1[R_1] \in A$.

2. given that we have $\text{Flow}(a_1, \ldots, a_n, R_1, \ldots, R_n)$ and $\text{Flow}(b_1, \ldots, b_m, Q_1, \ldots, Q_m)$, then we should check that $R_i = R_i'(\vdash C)$, $i = 1, \ldots n$ and $Q_i = Q_i'(\dashv C)$, $i = 1, \ldots, m$, whence if $\text{Label}(a_1, \ldots, a_n, R_1, \ldots, R_n) = D_\vdash$ and $\text{Label}(b_1, \ldots, b_m, Q_1, \ldots, Q_m) = D_\dashv$ then $\text{Flow}(a_1, \ldots, a_n, b_1, \ldots, b_m, R_1, \ldots, R_n, Q_1, \ldots, Q_m)$ should require that $D_\vdash \neq \mathbf{0}$ and $D_\dashv \neq \mathbf{0}$ and $\lambda = (D_\vdash \oplus D_\dashv)[C] \in A$. The label of the transition will then be λ.

To extend this to the general case we must only make sure that the synchronisation is independent of the order in which the a_i and R_i, and the b_i and Q_i are presented. As was the case for D above, in the following we will often distinguish 'left' and 'right' using $\vdash$ and $\dashv$ written subscripted.

DEFINITION 2.4.1 Given a label algebra $\mathcal{L} = (A, R)$, define $\text{Flow}_0 \subseteq \mathcal{M}_f(A \times C_\mathcal{L}) \times A$ such that

1. $\text{Flow}_0(\{\!|(a, R)|\!\}, a[R]) \Leftrightarrow a[R] \neq \mathbf{0}$

2. $\text{Flow}_0(P_\vdash \cup P_\dashv, (D_\vdash \oplus D_\dashv)[C]) \Leftrightarrow \text{Flow}_0(P_\vdash', D_\vdash) \wedge \text{Flow}_0(P_\dashv', D_\dashv) \wedge (D_\vdash \oplus D_\dashv)[C] \neq \mathbf{0}$ when

$$P_\vdash = \{\!|(a_1, R_1(\vdash)C), \ldots, (a_n, R_n(\vdash)C)|\!\} \neq \{\!|\,|\!\}$$
$$P_\dashv = \{\!|(b_1, Q_1(\dashv)C), \ldots, (b_m, Q_m(\dashv)C)|\!\} \neq \{\!|\,|\!\}$$

and

$$P_\vdash' = \{\!|(a_1, R_1), \ldots, (a_n, R_n)|\!\}$$
$$P_\dashv' = \{\!|(b_1, Q_1), \ldots, (b_m, Q_m)|\!\}$$

3. $\text{Flow}_0(P, D) \Leftrightarrow$ *false* otherwise. ■ 2.4.1

We note that the partitioning given by $P_\vdash$ and $P_\dashv$ is unique, as $\vdash \neq \dashv$.

For Flow_0 to be satisfied, its first argument must be a set:

LEMMA 2.4.2 Given labels $a_1, \ldots, a_n$ and contexts $R_1, \ldots, R_n$ of some label algebra then $\text{Flow}_0(\{\!|(a_1, R_1), \ldots, (a_n, R_n)|\!\}, D)$ implies $\forall i \neq j: R_i \neq R_j$. □ 2.4.2

Proof: Assume $\text{Flow}_0(\{\!|(a_1, R_1), \ldots, (a_n, R_n)|\!\}, D)$ holds. The proof proceeds by induction over n.

Base Case: $n = 1$ the result is immediate.

Inductive Step: Suppose $n > 1$ and that the result holds for all $i < n$.

From the definition, we may partition $\{\!|(a_1, R_1), \ldots, (a_n, R_n)|\!\}$ into $P_\vdash$ and $P_\dashv$, such that $(a, Q) \in P_\vdash$ implies $Q = Q'(.\vdash)C$ and $(s, Q) \in P_\dashv$ implies $Q = Q'(.\dashv)C$. If $R_i \in P_\vdash$ and $R_j \in P_\dashv$ the result follows immediately.

Otherwise R_i, $R_j \in P_\vdash$ or R_i, $R_j \in P_\dashv$. We will assume, without loss of generality, the former. Suppose $R_i = R_i'(.\vdash)C$ and $R_j = R_j'(.\vdash)C$. Then, from the definition, $D = (D_\vdash \oplus D_\dashv)[C]$ and $\text{Flow}_0(P_\vdash', D_\vdash)$ holds, where $P_\vdash' = \{\!|(a, Q') \mid (a, Q) \in P_\vdash \wedge Q = Q'(.\vdash)C|\!\}$.

As $P_\dashv' \neq \{\!|\,|\!\}$ we have that $|P_\vdash| < |P|$, so that $R_i' \neq R_j'$ from the induction hypothesis, and the result follows. ■ 2.4.2

Given Lemma 2.4.2, when Flow_0 is satisfied for particular P and D, we have that P is a set, whence we will often write P using set notation rather than multi-set notation.

We are now in a position to define the Flow predicate:

DEFINITION 2.4.3 Given a label algebra $\mathcal{L} = \langle A, R \rangle$ and $n \in \mathbb{N}$, define a relation $\text{Flow}^n : A^n \times \mathcal{C}_\mathcal{L}^n$ such that

$$\text{Flow}^n(a_1, \ldots, a_n, R_1, \ldots, R_n) \quad \Leftrightarrow \quad \exists D \in A : \text{Flow}_0(\{\!|(a_1, R_1), \ldots, (a_n, R_n)|\!\}, D)$$

Define $\text{Flow} = \bigcup \text{Flow}^n$. ■ 2.4.3

From Lemma 2.4.2:

PROPOSITION 2.4.4 $\text{Flow}(a_1, \ldots, a_n, R_1, \ldots, R_n)$ implies $\forall i \neq j,\ R_i \neq R_j$. □ 2.4.4

2.4.1 Label Produced by a Transition

The derivation of the Flow predicate was motivated by the H-synchrony rule. We have so far, however, defined only the enabling conditions for application of the rule through Flow; the transition also generates a label. This label appears within the Flow predicate—it is the D over which Flow_0 is quantified in the definition of Flow:

DEFINITION 2.4.5 Suppose $\text{Flow}(a_1, \ldots, a_n, R_1, \ldots, R_n)$ holds. Define the *transition label* $\text{Label}(a_1, \ldots, a_n, R_1, \ldots, R_n)$ to be the D such that $\text{Flow}_0(\{\!|(a_1, R_1), \ldots, (a_n, R_n)|\!\}, D)$. ■ 2.4.5

We have that:

PROPOSITION 2.4.6 The transition label is well-defined. □ 2.4.6

Proof: That D exists is a corollary of the fact that Flow$(a_1, \ldots, a_n, R_1, \ldots, R_n)$ holds. That it is defined uniquely follows from the fact that the partitioning of P in Flow$_0(P, D)$ is unique and that $\oplus$ is commutative. ■ 2.4.6

2.4.2 Symmetries of the Flow Predicate

The commutativity of the $\oplus$ operator is motivated by the commutativity of the H-synchrony rule, which in turn motivated by the commutativity of the synchrony and asynchrony rules of the literature. For instance

$$\frac{P \xrightarrow{a} P' \quad Q \xrightarrow{b} Q'}{P[\vdash r] \cup Q[\dashv r] \xrightarrow{(a \oplus b)[r]} P'[\vdash r] \cup Q'[\dashv r]} \; (a \oplus b)[r] \neq 0$$

and

$$\frac{Q \xrightarrow{b} Q' \quad P \xrightarrow{a} P'}{Q[\vdash r] \cup P[\dashv r] \xrightarrow{(b \oplus a)[r]} Q'[\vdash r] \cup P'[\dashv r]} \; (b \oplus a)[r] \neq 0$$

are indistinguishable by the label they produce.

This is reflected in the definition of Flow$_0$ (and hence the Flow predicate) in that, given that Flow$_0(P, D)$ is satisfied for a particular choice of P and D, the partitioning of P is unique and symmetric in the $\vdash$ and $\dashv$ allowing the polarities to be reversed whilst retaining the truth of Flow$_0$. For instance, if $P = P_\vdash \cup P_\dashv$ with

$$\begin{aligned} P_\vdash &= \{(a_1, R'_1(.\vdash)C), \ldots, (a_n, R'_n(.\vdash)C)\} \\ P_\dashv &= \{(b_1, Q'_1(.\dashv)C), \ldots, (b_m, Q'_m(.\dashv)C)\} \end{aligned}$$

then Flow$_0(P_\vdash \cup P_\dashv, D) \Leftrightarrow$ Flow$_0(Q_\vdash \cup Q_\dashv, D)$ where

$$\begin{aligned} Q_\vdash &= \{(a_1, R'_1(.\dashv)C), \ldots, (a_n, R'_n(.\dashv)C)\} \\ Q_\dashv &= \{(b_1, Q'_1(.\vdash)C), \ldots, (b_m, Q'_m(.\vdash)C)\} \end{aligned}$$

with the label, D, an invariant of the transformation.

More generally, we may see the recursive partitioning of the first argument of the Flow$_0$ as defining a *strictly binary tree* [TA86], with arcs labelled by $(.\vdash)/(.\dashv)$ pairs. Such transformations as described above may be applied to any non-leaf node of the tree.

Moreover, this observation leads to other useful symmetries which, instead of working on single $(.\vdash)/(.\dashv)$ pairs, apply to n-tuples, and corresponding to the (generalised) associative nature of the synchrony and asynchrony rules. For instance, consider labels and contexts satisfying $\text{Flow}_0(\{\!|(a_1,d_1),(a_2,d_2),(a_3,d_3)|\!\},D)$ such that the contexts have the following form:

$$\begin{array}{lllll} d_1 & = & C_1(.\vdash) & & (.\vdash)C_3 \\ d_2 & = & C_1'(.\dashv) & & (.\vdash)C_3 \\ d_3 & = & & C_2' & (.\dashv)C_3 \end{array}$$

in which the context which would be in 'position' C_2 is missing. Using arguments similar to those above we may transform this to

$$\begin{array}{lllll} d_1' & = & & C_1 & (.\vdash)C_3 \\ d_2' & = & C_1'(.\vdash) & & (.\dashv)C_3 \\ d_3' & = & C_2'(.\dashv) & & (.\dashv)C_3 \end{array}$$

while preserving the truth of $\text{Flow}_0(\{\!|(a_1,d_1'),(a_2,d_2'),(a_3,d_3')|\!\},D)$.

We will see, in Chapter 5, how these symmetries of the Flow predicate translate to the commutative and associative nature of certain operators on High Level Petri Boxes; namely concurrent and choice compositions, defined in Chapter 4.

2.5 High Level Petri Boxes

The structure of a High Level Petri Box is an annotated Petri net with the following components.

2.5.1 Places and Place Labels

Let $\mathcal{P}$ be a countably infinite set. We will choose the places which form our High Level Petri Boxes from $\mathcal{P}$. Places will usually be written in italic script s, with decoration and subscripting as necessary.

2.5.1.1 Ensuring Disjointness of the Name Space

For technical reasons it will be necessary to assume that $\mathcal{P}$ is closed under taking arbitrary binary (commutative, associative) Cartesian products with itself[11]. This condition will ensure

[11] With definition $A \otimes B = \{a \otimes b \mid a \in A \wedge b \in B\}$.

that the *name space* from which we choose the set of places underlying a High Level Petri Box is sufficiently structured to allow a form of *place multiplication* (often used for Petri net composition and used, for instance, in [Win84]) as the basis of our structural composition operators. The reason for the commutative associative binary nature of the $\otimes$-product is that Cartesian product is too concrete in distinguishing (s, s') and (s', s), and $((s, s'), s'')$ and $(s, (s', s''))$. This has the corollary that the commutativity and associativity of certain operators is structural equality, rather than just isomorphism. We see this as a refinement of the usual place multiplication scheme.

Examples of members of $\mathcal{P}$ are, then, $s_1 \otimes s_2 \otimes s_3 \otimes s_4$ and $s \otimes s' \otimes s''$.

For the PBC [BDH92] a very general scheme for the generation of place names must be given to be able to describe the arbitrarily complex refinements underlying the fix-point semantics of recursion; see [Dev95, Dev93]. This is the best possible for a low level model, due to the restricted modelling power of P/T nets. However, for the class of High Level Petri Boxes generated through the High Level Petri Box Algebra, place manipulations are finite. This explains our simpler 'calculus' of place names.

Each collection of places may be assigned an 'identity', i.e., something which distinguishes it 'sufficiently' from other sets of places, the use of which will permit arbitrary compositions on pairs of High Level Petri Boxes. The flavour of the place multiplication we use generates place names 'on the fly' and so it is worth being a little careful, initially, as to which pairs of sets of places we allow to be composed. Examples illustrating the problems which may occur are given in Chapter 4.

DEFINITION 2.5.1 The *identity* of the place $s = s_1 \otimes \cdots \otimes s_n$ is $\mathrm{id}(s) = \{s_1, \ldots, s_n\}$. Given a set of places $S \subseteq \mathcal{P}$ define the *identity* of S, $\mathrm{id}(S) = \bigcup_{s \in S} \mathrm{id}(s)$. ■ 2.5.1

For instance, the set $\{s_1 \otimes s_2 \otimes s_3, s_4 \otimes s_1\}$ has identity $\{s_1, s_2, s_3, s_4\}$. Sets of places may be different, whilst not having distinct identities. Such sets will be called *anti-social*. It is the pairs of *sociable* sets of places with which we prefer to deal. They have certain desirable properties which are preserved across place multiplication. Formally:

DEFINITION 2.5.2 Let S_1 and S_2 be sets of places. We say that S_1 and S_2 are *sociable* if and only if $\mathrm{id}(S_1) \cap \mathrm{id}(S_2) = \emptyset$. Otherwise S_1 and S_2 are *anti-social*. ■ 2.5.2

A non-empty set of places is always anti-social with itself. Trivially:

PROPOSITION 2.5.3 For S_1 and S_2 sociable sets of places, $S_1 \cap S_2 = \emptyset$. □ 2.5.3

We will later transfer the property of sociability to High Level Petri Boxes.

2.5.1.2 Place Labels

Let $\mathcal{A}_\mathcal{P}$, *the alphabet of place labels*, be the set[12] $\mathbb{P}\{E, X\}$. The annotation of places by members of $\mathcal{A}_\mathcal{P}$ will identify the *control interface* of our High Level Petri Boxes: the labels of *Entry places* will contain E; those of *eXit places* will contain X; *internal places* will have the label $\emptyset$. It follows that places with the label $\{E, X\}$ are both entry and exit places.

2.5.2 Local Transitions

A transition of a sequential component of a distributed transition system, at least when not in the context of other processes, might be thought of as local. A *local transition* within a High Level Petri Box represents this. Thus, the information a local transition must carry should include pre- and post-state, the action label (taken from some label algebra alphabet, but which may also be **0**, to represent inaction), together with the contribution it makes to an enclosing 'block' (contexts of the same label algebra annotating its arcs):

DEFINITION 2.5.4 *[Local Transition]* Given a label algebra $\mathcal{L} = \langle A, R \rangle$ define an *$\mathcal{L}$-local transition* as a five-tuple $\langle S, C, k, C', S' \rangle$ where $S, S' \subseteq \mathcal{P}$ are finite, non-empty sets of places, $C, C' \in \mathcal{C}_\mathcal{L}$ are $\mathcal{L}$-contexts, and $k \in A \cup \{\mathbf{0}\}$ is an $\mathcal{L}$-label.

The collection of $\mathcal{L}$-local transitions will be denoted $loc_\mathcal{L}$. ■ 2.5.4

As usual, we will tend to omit the label algebra decorations when no ambiguity arises.

DEFINITION 2.5.5 Let $l = \langle S, C, k, C', S' \rangle \in loc$ be a local transition. Define

$$
\begin{aligned}
{}^\bullet l &= S && \text{the } \textit{pre-set} \text{ of } l,\\
l^\bullet &= S' && \text{the } \textit{post-set} \text{ of } l,\\
{}^C l &= C && \text{the } \textit{pre-contexts} \text{ of } l,\\
l^C &= C' && \text{the } \textit{post-contexts} \text{ of } l,\\
\overline{l} &= k && \text{the } \textit{label} \text{ of } l.
\end{aligned}
$$

■ 2.5.5

[12] Where $\mathbb{P}S$ is the power set of the set S.

We will also use the abbreviation $^\bullet l^\bullet$ for the set $^\bullet l \cup l^\bullet$.

2.5.3 Abstract Transitions

An *abstract transition* is the High Level Petri Box equivalent of an application of the H-synchrony rule.

The model of a distributed transition system transition as a Petri net transition collects together finitely many transitions of sequential components to produce a synchronisation. Abstract transitions are, thus, finite multisets of local transitions. A multiset, rather than a set, will allow the contribution towards a synchronisation by two separate processes to come from the same local transition (this being a requirement for the finite representation of recursion, in which unbounded numbers of processes sharing the same High Level Petri Box structure may be generated through dynamic process creation). That we prevent infinite numbers of processes from synchronising together implies that the number of transitions contained in a (syntactically generated) High Level Petri Box will be at most countably infinite. Moreover, this will *not* prevent an observable synchronisation from occurring for such High Level Petri Boxes as an 'infinite synchronisation' may then only occur through the 'creation of an infinite number of processes'. The only way of achieving this for High Level Petri Boxes will be through applications of the recursive operator (introduced in Chapter 6). From the definition of that operator, all recursive processes are guarded so that any such synchronisation will be possible only through an infinitely long execution which, by definition, will therefore not be observable.

DEFINITION 2.5.6 *[Abstract Transition]* Given a label algebra $\mathcal{L}$, an *$\mathcal{L}$-abstract transition*, t, is a finite multiset of $\mathcal{L}$-local transitions, i.e., $t \in \mathcal{M}_{\mathcal{F}}(loc_{\mathcal{L}})$.

The *order of an abstract transition* t, $|t|$, is the sum of the multiplicities of its constituent local transitions. The collection of abstract transitions on the label algebra $\mathcal{L}$ will be denoted $\mathcal{T}_{\mathcal{L}}$.

■ 2.5.6

As usual, we will omit the decoration when the label algebra is clear by context.

An abstract transition of order n represents the attempted synchronisation of actions from n local transitions. In being a finite multiset of local transitions an abstract transition t can be regarded as having an element of $A^\oplus$ as label. Examples of abstract transitions appear in Figures 2.2 and 2.4.

The components of an abstract transition are given by the following operators:

DEFINITION 2.5.7 Let $t = \{\!|l_1, \ldots, l_n|\!\} \in \mathcal{T}_{\mathcal{L}}$ for some label algebra $\mathcal{L}$. Define

$$\begin{array}{lcll} {}^\bullet t & = & \bigcup_i {}^\bullet l_i & \text{the } \textit{pre-set} \text{ of } t, \\ t^\bullet & = & \bigcup_i l_i^\bullet & \text{the } \textit{post-set} \text{ of } t, \\ \overline{t} & = & \overline{l_1} \oplus \cdots \oplus \overline{l_n} & \text{the } \textit{label} \text{ of } t \end{array}$$

■ 2.5.7

As for local transitions, we will use the abbreviation ${}^\bullet t^\bullet$ for the set ${}^\bullet t \cup t^\bullet$.

2.5.4 Pre-High Level Petri Boxes

A High Level Petri Box is a restricted form of *pre-High Level Petri Box.*

DEFINITION 2.5.8 Given a label algebra $\mathcal{L}$, a pre-High Level Petri Box over $\mathcal{L}$ is a triple, $\langle S, T, \lambda \rangle$, such that

$$\begin{array}{lcll} S & \subseteq & \mathcal{P} & \\ T & \subseteq & \mathcal{T}_{\mathcal{L}} & \text{such that } \forall t \in T: {}^\bullet t^\bullet \subseteq S, \\ \lambda & : & S \to \mathcal{A}_{\mathcal{P}} & \end{array}$$

The class of pre-High Level Petri Boxes on a label algebra $\mathcal{L}$ will be denoted $\mathcal{PHLPB}_{\mathcal{L}}$.

■ 2.5.8

The pre-High Level Petri Box $B = \langle S, T, \lambda \rangle$ over the label algebra $\mathcal{L}_0$ of Example 2.1.1 (page 8) is illustrated in Figure 2.2, where

$$\begin{array}{lcl} S & = & \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\} \\ T & = & \{t_0, t_1\} \\ \lambda & = & \{s_0 \to \{E, X\}, s_1 \to \emptyset, s_2 \to \{E\}, s_3 \to \{X\}, s_4 \to \{E, X\}, s_5 \to \{X\}, s_6 \to \{E\}\} \\ t_0 & = & \{\!|\langle \{s_0\}, .g, a, .g, \{s_2\}\rangle, \langle \{s_1\}, .f, a, .f, \{s_3\}\rangle|\!\} \\ t_1 & = & \{\!|\langle \{s_5\}, .g, \mathbf{0}, .\vdash, \{s_5, s_6\}\rangle|\!\} \end{array}$$

We use the usual graphical representation, i.e., places are denoted as circles, transitions as squares, with the flow relation being indicated as directed arcs between them. As a notational convenience, we will often refer to the structure of a pre-High Level Petri Box by name without explicitly listing its components; for instance, given a pre-High Level Petri Box with the name B, the components of B will be distinguished by B written subscripted: for example, a pre-High

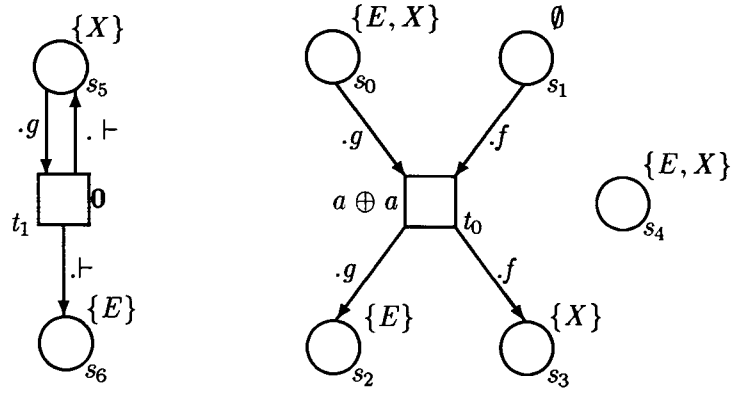


Figure 2.2: The pre-High Level Petri Box B.

Level Petri Box of name B_1 comprises $\langle S_{B_1}, T_{B_1}, \lambda_{B_1} \rangle$; or by annotating the components with the structure's decoration; for instance, $\langle S_1, T_1, \lambda_1 \rangle$ in the previous case. Also, in an attempt to clarify figures containing pre-High Level Petri Boxes, we will often omit the names of places and transitions, the annotation of internal places, and the trivial context when used as an arc annotation.

2.5.4.1 Operators on Pre-High Level Petri Boxes

As for local and abstract transitions, we define the operators which extract the components of a pre-High Level Petri Box:

DEFINITION 2.5.9 Given a pre-High Level Petri Box B, define

$$
\begin{array}{rcll}
{}^\bullet B & = & \{s \in S_B \mid E \in \lambda(s)\} & \text{the } \textit{entry interface} \text{ of } B \\
B^\bullet & = & \{s \in S_B \mid X \in \lambda(s)\} & \text{the } \textit{exit interface} \text{ of } B \\
\dot{B} & = & S_B \setminus ({}^\bullet B \cup B^\bullet) & \text{the } \textit{internal places} \text{ (or } \textit{midset}) \text{ of } B \\
LT(B) & = & \{l \in t \mid t \in T_B\} & \text{the } \textit{local transitions} \text{ of } B \\
{}^C B & = & \{{}^C l \mid l \in LT(B) \wedge {}^\bullet l \subseteq {}^\bullet B\} & \text{the } \textit{entry contexts} \text{ of } B \\
B^C & = & \{l^C \mid l \in LT(B) \wedge l^\bullet \subseteq B^\bullet\} & \text{the } \textit{exit contexts} \text{ of } B
\end{array}
$$

■ 2.5.9

As usual, ${}^\bullet B^\bullet$ will abbreviate ${}^\bullet B \cup B^\bullet$. Moreover, we will often talk of the *input (output) arcs* of a pre-High Level Petri Box, by which we will mean the arcs which are incident on and flow from, the pre-set (respectively, incident on and flow to, the post-set).

A pre-High Level Petri Box with a single local transition may be constructed directly from that local transition using the *augment* operation:

DEFINITION 2.5.10 Define a partial mapping $[\_]: loc_{\mathcal{L}} \rightharpoonup \mathcal{PHLCPB}_{\mathcal{L}}$ such that $[l] = \langle {}^\bullet l^\bullet, \{\{\!|l|\!\}\}, \{{}^\bullet l \times \{E\}, l^\bullet \times \{X\}\}\rangle$ when ${}^\bullet l \cap l^\bullet = \emptyset$, and undefined otherwise. When defined, $[l]$ is called the *augment* of l. ■ 2.5.10

Although it would be simple enough to remove the partiality of the augment operator, we will never be required to augment a local transition such that ${}^\bullet l \cap l^\bullet \neq \emptyset$.

2.5.4.2 A Partial Order on pre-High Level Petri Boxes

We may define a simple containment relation on pre-High Level Petri Boxes:

DEFINITION 2.5.11 For pre-High Level Petri Boxes B_1 and B_2, define $B_1 \subseteq B_2$ whenever $S_1 = S_2$, $\lambda_1 = \lambda_2$ and $T_1 \subseteq T_2$. ■ 2.5.11

Clearly, $\subseteq$ is a partial order.

2.5.4.3 Sociability of pre-High Level Petri Boxes

We have assigned to each set of places an identity (Definition 2.5.1). As pre-High Level Petri Boxes comprise sets of places we may lift the concept of identity to pre-High Level Petri Boxes:

DEFINITION 2.5.12 Given a pre-High Level Petri Box, $B = \langle S, T, \lambda\rangle$, define the *identity* of B as $\mathrm{id}(B) = \mathrm{id}(S)$. ■ 2.5.12

Pairs of pre-High Level Petri Boxes may thus be regarded as sociable:

DEFINITION 2.5.13 Let B_1 and B_2 be pre-High Level Petri Boxes. We say that B_1 and B_2 are *sociable* when $\mathrm{id}(B_1) \cap \mathrm{id}(B_2) = \emptyset$. ■ 2.5.13

The following properties of sociable pre-High Level Petri Boxes follow easily from Proposition 2.5.3:

PROPOSITION 2.5.14 Let B_1 and B_2 be sociable pre-High Level Petri Boxes. Then:

1. $S_{B_1} \cap S_{B_2} = \emptyset$,
2. $LT(B_1) \cap LT(B_2) = \emptyset$,
3. $T_{B_1} \cap T_{B_2} = \emptyset$.

□ 2.5.14

2.5.5 High Level Petri Boxes

The following properties characterise High Level Petri Boxes from pre-High Level Petri Boxes.

DEFINITION 2.5.15 Let $B = \langle S, T, \lambda \rangle$ be a pre-High Level Petri Box. Then B is a High Level Petri Box if, in addition:

1. ${}^\bullet B \neq \emptyset \neq B^\bullet$,
2. ${}^\bullet B = {}^\bullet LT(B) \setminus LT(B)^\bullet$ and $B^\bullet = LT(B)^\bullet \setminus {}^\bullet LT(B)$,
3. for each local transition $l \in LT(B)$, either ${}^\bullet l \subseteq {}^\bullet B$ or ${}^\bullet l \cap {}^\bullet B = \emptyset$ and either $l^\bullet \subseteq B^\bullet$ or $l^\bullet \cap B^\bullet = \emptyset$.

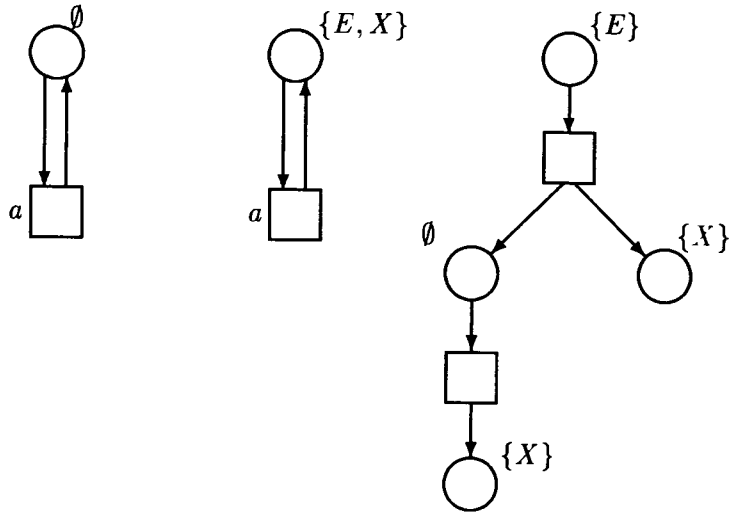
For a label algebra $\mathcal{L}$, the class of $\mathcal{L}$-High Level Petri Boxes will be denoted[13] $\mathcal{HLPB}^*_{\mathcal{L}}$.

■ 2.5.15

Properties 1-3 ensure of a High Level Petri Box the following properties:

1. that each High Level Petri Box has both an entry and exit interface, giving a well-defined interface for compositions involving control flow. Property 1 also implies that the markings expressing initialisation and termination of a High Level Petri Box (so-called standard initial and standard terminal markings) are non-empty,
2. that the entry places of B should be those places which have no preceding local (and hence abstract) transitions. (The motivation for this will be found in Chapter 5.) One half of the equality, $\subseteq$, prevents entry places being re-marked after the firing of an initial abstract transition (with a symmetric statement holding for exit-places). (The reverse inclusion which completes) Property 2 will be shown to be necessary for preserving the structure of

[13] The undecorated name, $\mathcal{HLPB}_{\mathcal{L}}$, is used for the subclass of *syntactically generated* High Level Petri Boxes in Chapter 4.



a. b. c.

Figure 2.3: Illustrating the independence of the properties of Definition 2.5.15. In a. only Property 1 fails, in b. only Property 2 fails, in c. only Property 3 fails.

a High Level Petri Box through compositions—without this condition it is possible to move outside of the class of High Level Petri Boxes (and indeed pre-High Level Petri Boxes) on a given label algebra using the operators we define; see Section 4.9.2 for a discussion. That this is a deficiency of pre-High Level Petri Boxes and not of the operators we have chosen will also be argued,

3. that the pre-set of a local transition should either be wholly contained in the entry places of a High Level Petri Box or have an empty intersection is again made to ensure, together with Property 2, that High Level Petri Boxes are closed under application of the operations we have defined.

Properties 1-3 although interrelated, are independent, as the pre-High Level Petri Boxes of Figure 2.3 show. (It is to be assumed that all transitions of the pre-High Level Petri Boxes have order 1). Pre-High Level Petri Box a. satisfies all conditions except 1; b. satisfies all conditions except 2; c. satisfies all conditions except 3. We also note that the definition of pre- and post-contexts of a local transition, together with Property 1 above, implies that the entry contexts (resp. exit contexts) of a High Level Petri Box are precisely those contexts which annotate the input (resp. output) arcs of a High Level Petri Box.

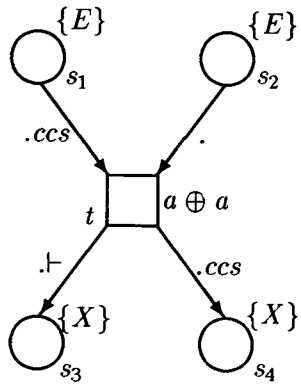


Figure 2.4: The High Level Petri Box B'.

Figure 2.4 gives the example of the High Level Petri Box $B' = \langle S', T', \lambda' \rangle$ over the label algebra $\mathcal{L}_0$ of Example 2.1.1, where:

$$\begin{aligned} S' &= \{s_1, s_2, s_3, s_4\} \\ T' &= \{t\} \\ \lambda' &= \{s_1 \to \{E\}, s_2 \to \{E\}, s_3 \to \{X\}, s_4 \to \{X\}\} \end{aligned}$$

and t is the abstract transition consisting of the local transitions $\langle \{s_1\}, .ccs, a, .\vdash, \{s_3\} \rangle$ and $\langle \{s_2\}, ., a, .ccs, \{s_4\} \rangle$.

2.5.5.1 Markings of High Level Petri Boxes

Given a label algebra $\mathcal{L}$, the tokens which populate a High Level Petri Box *over* $\mathcal{L}$ are the contexts of $\mathcal{C}_\mathcal{L}$. As we allow multiple copies of the same token on a place we may define a marking as:

DEFINITION 2.5.16 Let B be a High Level Petri Box. A *marking* of B is a mapping $M: S_B \to \mathcal{M}_\mathcal{F}(\mathcal{C}_\mathcal{L})$.

If, in addition to being a function in[14] $S_B \to \mathcal{M}_\mathcal{F}(\mathcal{C}_\mathcal{L})$, a marking M is also a function in $S_B \to \mathbb{P}\mathcal{C}_\mathcal{L}$, we will say that M is a *safe marking.* ■ 2.5.16

At a safe marking the tokens which form the marking of any particular place are distinguishable. This is equivalent to the property of *strictness* for PrT nets. However, a safe marking of a High

[14]Where, if S is a set, we use the natural injection $\mathbb{P}S \to \mathcal{M}_\mathcal{F}(S)$.

Level Petri Box corresponds to a safe marking in the basic net model semantics defined in Chapter 3, which is why we use the term safe rather than strict.

To represent a marking as a function we will often omit mention of elements of the domain of the function whose image is the empty set. For instance, for a High Level Petri Box B with $S_B = \{s_1, s_2, s_3\}$, the marking $\{s_1 \mapsto \{\!|\}\!|, s_2 \mapsto \{\!|., .f|\}\!|, s_3 \mapsto \{\!|\}\!|\}$ will be represented as, simply, $\{s_2 \mapsto \{\!|., .f|\}\!|\}$.

2.5.5.2 Standard Initial and Terminal Markings

There are two classes of markings in which we are particularly interested, the *standard initial* and *standard terminal markings.* In general, the behaviours we will consider of High Level Petri Boxes will be from standard initial markings, and termination of a behaviour will be achieved only when a standard terminal marking is attained.

DEFINITION 2.5.17 Let B be a High Level Petri Box and let $d \in C_{\mathcal{L}}$. The *standard initial marking of B on d* is the marking $M_d^{I,B}$ such that

$$M_d^{I,B}(s) = \begin{cases} \{\!|d|\}\!| & E \in \lambda(s) \\ \{\!|\}\!| & \text{otherwise} \end{cases}$$

The *standard terminal marking of B on d* is the marking $M_d^{T,B}$ such that

$$M_d^{T,B}(s) = \begin{cases} \{\!|d|\}\!| & X \in \lambda(s) \\ \{\!|\}\!| & \text{otherwise} \end{cases}$$

When d is the trivial context $(.)$, we will refer to the constructed marking as *the* standard initial marking (i.e., without mentioning d), and similarly for *the* standard terminal marking.

■ 2.5.17

We will often abbreviate $M_d^{I,B}$ to M_d^I and $M_d^{T,B}$ to M_d^T when this can be done without causing ambiguity.

We note that the standard initial and terminal markings are safe as defined in Definition 2.5.16. In the next chapter we define a symbolic firing rule for High Level Petri Boxes, i.e., the allowed transformation of markings into markings, from which comes notions of behaviour, reachability, *etc.*

2.5.6 Why pre-High Level Petri Boxes?

In general, and until Chapter 4, we will not use pre-High Level Petri Boxes in the description of the High Level Petri Box Algebra or its semantics. However, with the increased requirements of our recursive construct (given in Chapter 6) certain sub-classes of pre-High Level Petri Boxes will be used to facilitate the definition. Their definition is left to that chapter.

The composition of pre- and High Level Petri Boxes is, however, potentially 'explosive' in that we are able to move outside the class of High Level Petri Boxes through the seemingly innocuous combination of the two. A discussion of this phenomenon follows the description of the initial portion of the High Level Petri Box Operators, in Chapter 4.

2.5.6.1 On the Representation of High Level Petri Boxes

An immediate corollary of Definition 2.5.15(2) is that entry and exit places of a High Level Petri Boxes are disjoint:

LEMMA 2.5.18 Given a High Level Petri Box B, ${}^\bullet B \cap B^\bullet = \emptyset$. □ 2.5.18

This means that a place label in a High Level Petri Box may never be $\{E, X\}$, only one of $\emptyset$, $\{E\}$ or $\{X\}$. For clarity, we will usually omit the set brackets in figures.

2.5.7 Isomorphism of High Level Petri Boxes

For High Level Petri Boxes to be isomorphic means we may relate the components with bijections which preserve the interface and relative structure of the High Level Petri Box:

DEFINITION 2.5.19 Let $\mathcal{L} = \langle A, R \rangle$ be a label algebra. An isomorphism, ϕ, between $\mathcal{L}$-High Level Petri Boxes, B_1 and B_2, is a triple of bijections, $\langle \sigma, \kappa, \tau \rangle$, such that:

$$\begin{aligned} \sigma &: S_{B_1} \rightarrow S_{B_2} \\ \kappa &: LT(B_1) \rightarrow LT(B_2) \\ \tau &: T_{B_1} \rightarrow T_{B_2} \end{aligned}$$

and such that diagram in Figure 2.5 commutes. ■ 2.5.19

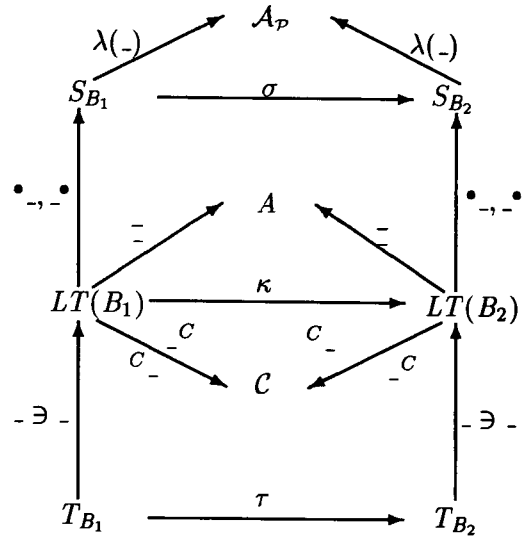


Figure 2.5: A commutative diagram for the isomorphism ϕ: $B_1 \equiv B_2$.

That ϕ is an isomorphism between B_1 and B_2 we will denote[15] ϕ: $B_1 \equiv B_2$, or just $B_1 \equiv B_2$, when the isomorphism is clear from context or unimportant. When the components of the isomorphism are to be made explicit we will write, for instance, $\langle \sigma, \kappa, \tau \rangle$: $B_1 \equiv B_2$.

From bijection composition and a little 'diagram chasing' we have that:

PROPOSITION 2.5.20 $\equiv$ is an equivalence relation. □ 2.5.20

Definition 2.5.19 includes a bijection on the local transitions of a High Level Petri Box, a component which is not explicit in the definition of High Level Petri Boxes. However, this device will often help us in technical details of the sequel for, as the following proposition demonstrates, there is a strong relationship between the components of an isomorphism of High Level Petri Boxes (as might be expected), with the local transitions 'midway' between places and abstract transitions.

PROPOSITION 2.5.21 Given $\langle \sigma, \kappa, \tau \rangle$: $B_1 \equiv B_2$ then

1. $\kappa = \hat{\sigma}$
2. $\tau = \hat{\hat{\sigma}}$

where $\hat{\sigma}(l) = \langle \sigma(^\bullet l), {}^C l, \overline{l}, l^C, \sigma(l^\bullet) \rangle$ and $\hat{\hat{\sigma}}(t) = \{\!|\hat{\sigma}(l) \mid l \in t|\!\}$. □ 2.5.21

[15] We will also use the symbol $\equiv$ to represent isomorphism between *arbitrary* structures.

Proof: 1. From the commutativity of Figure 2.5 we know that

$$\begin{aligned} \sigma({}^\bullet l) &= {}^\bullet\kappa(l) \\ \sigma(l^\bullet) &= \kappa(l)^\bullet \\ {}^C l &= {}^C\kappa(l) \\ l^C &= \kappa(l)^C \\ \overline{l} &= \overline{\kappa(l)} \end{aligned}$$

whence

$$\begin{aligned} \kappa(l) &= \langle \sigma({}^\bullet l), {}^C l, \overline{l}, l^C, \sigma(l^\bullet) \rangle \\ &= \hat{\sigma}(l) \end{aligned}$$

Hence the result.

2. Similar. ■ 2.5.21

A corollary of Proposition 2.5.21 is that, to show isomorphism, we must only find a bijection between the constituent places which 'lifts' to an isomorphism of local and abstract transitions. That isomorphism implies a bijection on places is a very strong condition for the isomorphism of High Level Petri Boxes and, in particular, is much stronger than the duplication equivalence of [BDH92]. However, the development of High Level Petri Boxes will proceed, in this text, through the provision of a model for a particular algebra, and so we will have a much 'tighter' control on the High Level Petri Boxes that are composed than was assumed in [BDH92].

This tight control appears to imply that we generate the 'natural' equivalence class representative under duplication equivalence.

Isomorphism of High Level Petri Boxes, is the *renaming equivalence* of [BDH92], given that we are using a place centred representation for our High Level Petri Boxes. We will subsequently show that such a renaming is a congruence with respect to the operators we define (under certain 'sociability conditions'). This indicates that to consider individual place names as distinguishing High Level Petri Boxes is too concrete in some senses. In fact, it is the relationship between local and abstract transitions that is carried by places in which we are most interested, rather than being able to name any particular state. The abstraction from named states is formalised in Chapter 4.

However, the operators we propose in Chapters 4 and 6 find their natural expression through the place names of a High Level Petri Box so that we prefer to work with this 'too concrete' representation, for the moment.

2.6 Summary

In this chapter we have introduced label algebras, which allow multi-way synchronisation, together with the semantic domain of High Level Petri Boxes. For High Level Petri Boxes we have provided an 'intuitive' characterisation of their behaviour in terms of the H-synchrony rule. In the next chapter we define the formal denotation of a High Level Petri Box, from which follows a formal behavioural characterisation.

Chapter 3

The Meaning of a High Level Petri Box

In this chapter we present a denotational semantics of High Level Petri Boxes. The semantics is given in terms of PrT nets [GL81, Gen87]. Through this denotation we establish the relationship between High Level Petri Boxes and a well-known net class. From (a modified version of) the standard semantics of PrT nets we may give a low level net denotation of a High Level Petri Box, from which notions of behaviour and behavioural equivalence for High Level Petri Boxes stem.

It is for the *syntactically generated* High Level Petri Boxes, i.e., those which arise as the denotations of terms of the High Level Petri Box Algebra that the semantics is intended. The definition of the semantics is, therefore, partial: it is from the definitions and properties of the algebra and its semantics, in Chapters 4 and 6, that the totality of the semantics on the syntactically generated High Level Petri Boxes follows.

3.1 Predicate/Transition Nets

PrT nets are a class of high level Petri nets. As such, along with control flow, PrT nets allow tokens to carry other information which may be used to influence behaviour. Structurally, PrT nets build an algebraic structure upon a P/T net (called the *skeleton*). This algebraic structure may be characterised as a signature (i.e., constants, operators, relational predicates) on top of which is built a *first order predicate logic*. The link between the algebra and net parts is forged through the net annotation: specifically, *dynamic (relational) predicates*, which annotate places,

act as variable *types* for variables of the first order language to range over. At a given marking of a PrT net, the members of the dynamic types are the tokens which form the marking of the place which the dynamic predicate annotates.

The standard denotation (or *unfolding*) of PrT nets is in terms of *Condition/Event (C/E)* systems [Pet76, Thi87]. C/E systems were introduced by Petri as a 'common reference model for net theory'. They have strict structural requirements for their definition: there may be no side-loops, i.e., no place may be a member of both the pre- and post-set of a single transition; there may be neither isolated places nor transitions; nor may there be *dead* transitions[1]. The behaviour of a C/E system is defined through a 'look-ahead' firing rule which prevents unsafe markings from being produced. Practically, this means that not all syntactically well-formed PrT nets have denotations as C/E systems.

Our original requirement of a basic net semantics for High Level Petri Boxes was to allow the derivation of a symbolic firing rule for High Level Petri Boxes which was consistent with that of the low level Petri Box model, i.e., P/T nets. To this end we use a P/T net denotation of PrT nets rather than the standard C/E system semantics. As P/T nets are definitionally less complex than C/E systems, this has the benefit of allowing a simplified presentation of the unfolding.

The unfolding we define is derived from and has much in common with that of Genrich [Gen87]. As in that approach, we will assume that any PrT net to be unfolded is safe[2]. This establishes a proof obligation that the PrT nets derived from High Level Petri Boxes are safe (and explains the partiality of the semantics). We will later show that all markings reachable from standard initial markings of syntactically generated High Level Petri Boxes are safe, and this extends trivially to the safeness of their denotations as PrT nets.

A complete introduction to PrT nets is unnecessary for the presentation of this thesis. The PrT net model was introduced in [GL81]. A technically complete description appears in [Gen87], to which we refer the reader.

Each marking of a PrT net defines an interpretation for a first order predicate logic. As the marking of a PrT net changes through the firing of transitions, the interpretation changes. The parts of the interpretation which are permitted to change in this way are restricted to a pre-defined set of so-called *dynamic* predicates, Π_v; all other predicates are called *static* to make

[1] I.e., no reachable marking enables them.

[2] Genrich uses the term *strict*, with meaning *multiple occurrences of [the same] token [...] on places are disallowed.* This concurs with the definition of being safe given in Definition 2.5.16.

the distinction clear, and form the set Π_s. Dynamic predicates are not directly permitted to appear in formulae: reference is made to them only by arc annotations through which they form generalised quantifiers for the logical variables appearing in formulae.

The following definition of PrT nets is adapted from [Gen87].

DEFINITION 3.1.1 *[cf., Definition 2.10, [Gen87, Pg. 216]]* Let **L** be a first order language and let $\mathbf{L}_s$ designate the sub-language using only Π_s, the predicates denoting static relations. A Predicate/Transition net, $P = \langle N, W, M \rangle$, is an unmarked annotated P/T net N, together with its annotation W in **L**, and its marking M, i.e,

1. N is an (unweighted) P/T net, $N = \langle S, T, F \rangle$,
2. W is the annotation of N, $W = \langle W_N, W_S, W_T, W_F \rangle$ where:
   (a) W_N is a first order structure for $\mathbf{L}_s$ called the *support* of M;
   (b) W_S is a bijection between the set of places, S, and the set of dynamic predicates, Π_v;
   (c) W_T is a mapping of the set of transitions, T, into the set of formulae (called *transition selectors*) that use only operators and static predicates (i.e., are in $\mathbf{L}_s$);
   (d) W_F is a mapping of the set of arcs, F, into the set of symbolic sums of tuples of terms of **L**, LC, such that for an arc $(x, y) \in F$ leading into or out of a place s (i.e., $x = s$ or $y = s$) and n being the index of the predicate annotating s, $W_F(x, y) \in LC^n$.
3. M is a marking of the places: it is a mapping that assigns to each place $s \in S$ a set of constants of the first order language. If, for each place s, $M(s) = \emptyset$, we will call P unmarked, and permit M to be omitted from the triple, writing instead $P = \langle N, W \rangle$.

■ 3.1.1

3.2 Predicate/Transition Net Semantics of High Level Petri Boxes

In this section, given a High Level Petri Box over a particular label algebra we will define a PrT net as its formal denotation. The translation which underlies the denotation is relatively transparent, the only non-trivial part being the translation of the stack-like behaviour of a local transition.

3.2.1 The Language and Logic of High Level Petri Boxes

The following definitions give the algebraic signature of the first order language and the term and formula building operations of the logic underlying the denotation of a High Level Petri Box (equivalently, defining the PrT net sub-class). We then define a *structure* (the model or interpretation for the logic) in terms of label algebras, contexts and places.

Notationally we follow Genrich and use $o^{(n)}$ to denote that the formal symbol o is an operator or predicate of arity n. $\Omega^{(n)}$ (respectively, $\Pi^{(n)}$) is the collection of all operators (respectively, predicates) of arity n. Operators of arity zero are constants[3]. The formal symbols introduced here will be provided with interpretations in Definition 3.2.2.

DEFINITION 3.2.1 Let $\mathcal{L} = \langle A, R \rangle$ be a label algebra and let $S \subseteq \mathcal{P}$ be a set of places. Define

$$\begin{aligned}
\Omega^{(0)} &= \{C^{(0)} \mid C \in \mathcal{C}_{\mathcal{L}}\} \\
\Omega^{(2)} &= \{\frown^{(2)}\} \\
\Pi^{(1)} &= \{\hat{s}^{(1)} \mid s \in S\} \cup \{\mathrm{Flow}^{(1)}_{(a_1)} \mid a_1 \in A\} \\
\Pi^{(i)} &= \{\mathrm{Flow}^{(i)}_{(a_1,\ldots,a_i)} \mid a_1, \ldots, a_i \in A\} \qquad \text{for all } i > 1
\end{aligned}$$

and $\Pi^{(0)} = \emptyset$, $\Omega^{(i)} = \emptyset$ for all $i \notin \{0, 2\}$.

We define a first order language, $\mathbf{L}_{\mathcal{L}}$, with $\Omega = \bigcup \Omega^{(i)}$ and $\Pi = \bigcup \Pi^{(i)}$ as vocabulary. The vocabulary is augmented with a countably infinite collection of variables $\mathcal{V}$, disjoint from both Ω and Π.

The dynamic predicates are in bijective correspondence to the set of places, S: $\Pi_v = \{\hat{s}^{(1)} \mid s \in S\}$. All other predicates are static: $\Pi_s = \Pi \setminus \Pi_v$.

The rules for building *terms* and *formulae* over $\mathbf{L}_{\mathcal{L}}$ are:

1. Terms:
   (a) a variable is a term,
   (b) each constant is a term,
   (c) for terms u, v, $\frown(u, v)$ is a term (usually written infix, so: $u \frown v$),
   (d) no other expression is a term.

[3]Predicates of arity 0 are propositional variables, but these are not used in the semantics.

2. Formulae:

   (a) if u and v are terms then $u = v$ is a formula,

   (b) for labels $a_1, \ldots, a_n$ and terms $u_1, \ldots, u_n$, there is a formula $\text{Flow}_{(a_1,\ldots,a_n)}(u_1, \ldots, u_n)$.

   (c) for a formula p, $\neg p$ is a formula; for pairs of formulae p and q, $p \wedge q$ and $p \vee q$ are formulae,

   (d) for a variable $v \in \mathcal{V}$ and a formula p, $\exists v{:}\, p$ is a formula,

   (e) no other expression is a formula. ■ 3.2.1

By regarding $\text{Flow}(a_1, \ldots, a_n)$ as a predicate of arity[4] n we may define a structure for the static portion of the language $\mathbf{L}_{\mathcal{L}}$.

DEFINITION 3.2.2 *[Structure for $\mathbf{L}_{\mathcal{L}}$]* Let $\mathcal{L} = \langle A, R \rangle$ be a label algebra and let $S \subseteq \mathcal{P}$ be a set of places. Define

$$\mathcal{R}_{\mathcal{L}} = \langle \mathcal{C}_{\mathcal{L}}, \{\_ \frown \_\}, \{\text{Flow}(a_1, \ldots, a_i) \subseteq \mathcal{C}_{\mathcal{L}}^i \mid a_1, \ldots, a_i \in A\} \rangle$$

■ 3.2.2

Then:

LEMMA 3.2.3 Under the correspondence

$$\begin{array}{rcll} C^{(0)} & \rightarrow & C & C \in \mathcal{C}_{\mathcal{L}} \\ \frown^{(2)} & \rightarrow & \_ \frown \_ : \mathcal{C}_{\mathcal{L}} \times \mathcal{C}_{\mathcal{L}} \rightarrow \mathcal{C}_{\mathcal{L}} & \\ \text{Flow}^{(i)}_{(a_1,\ldots,a_i)} & \rightarrow & \text{Flow}(a_1, \ldots, a_i) \subseteq \mathcal{C}_{\mathcal{L}}^i & i \geq 1 \end{array}$$

$\mathcal{R}_{\mathcal{L}}$ forms a first order structure for the static portion of $\mathbf{L}_{\mathcal{L}}$. □ 3.2.3

3.2.2 The Denotation of a High Level Petri Box

We are now able to construct an unmarked PrT net from an unmarked High Level Petri Box.

[4] So that $\text{Flow}(a_1, \ldots, a_n) \subseteq \mathcal{C}_{\mathcal{L}}^n$.

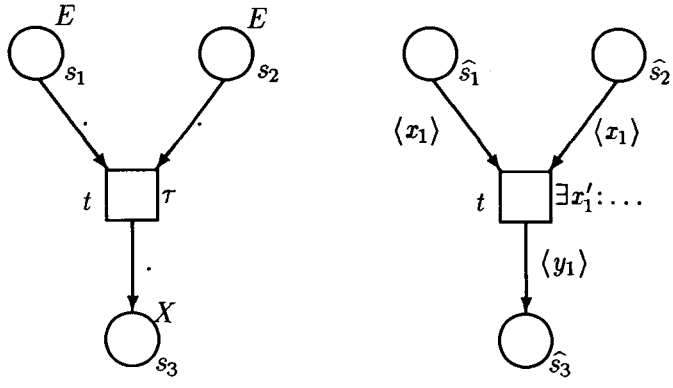


Figure 3.1: The High Level Petri Box of Example 3.2.5 and its PrT net denotation.

DEFINITION 3.2.4 Given a High Level Petri Box $B = \langle S, T, \lambda \rangle$ define an unmarked PrT net $PrT(B) = \langle N, W \rangle$ as follows:

1. $N = \langle S, T, F \rangle$ is a P/T net with flow relation $F = \{(s,t) \in S \times T \mid s \in {}^\bullet t\} \cup \{(t,s) \in T \times S \mid s \in t^\bullet\}$.

2. $W = \langle W_N, W_S, W_T, W_F \rangle$, is the annotation, where $W_N = \mathcal{R}_\mathcal{L}$ and[5] $W_S = \hat{\cdot} : S \to \hat{S}$.

3. Suppose $t = \{\!|l_1, \ldots, l_n|\!\} \in T$. For each l_i choose fresh variables[6] x_i and y_i. For $s \in {}^\bullet t$ define $W_F(s,t) = +_{\{i \mid s \in {}^\bullet l_i\}} \langle x_i \rangle$, and for $s \in t^\bullet$ define $W_F(t,s) = +_{\{i \mid s \in l_i{}^\bullet\}} \langle y_i \rangle$.

4. Let the variables which annotate the arcs into $t = \{\!|l_1, \ldots, l_n|\!\}$ be $\{x_1, \ldots, x_n\}$, those out of t be $\{y_1, \ldots, y_n\}$. Define

$$W_T(t) = \exists x_1', \ldots, x_n' : \bigwedge_{i=1}^n x_i' = {}^C l_i \frown x_i \wedge \bigwedge_{i=1}^n x_i' = l_i{}^C \frown y_i \wedge \mathrm{Flow}_{(\overline{l_1}, \ldots, \overline{l_n})}(x_1', \ldots, x_n')$$

■ 3.2.4

3.2.2.1 Examples of the Denotation

Throughout this section we will assume that all label algebra relative properties are defined with respect to the label algebra $\mathcal{L}_0$ of Example 2.1.1 (page 8).

[5] With $\hat{S} = \{\hat{s} \mid s \in S\}$.

[6] I.e., previously unused in the semantics. Always possible as $\mathcal{V}$ is infinite.

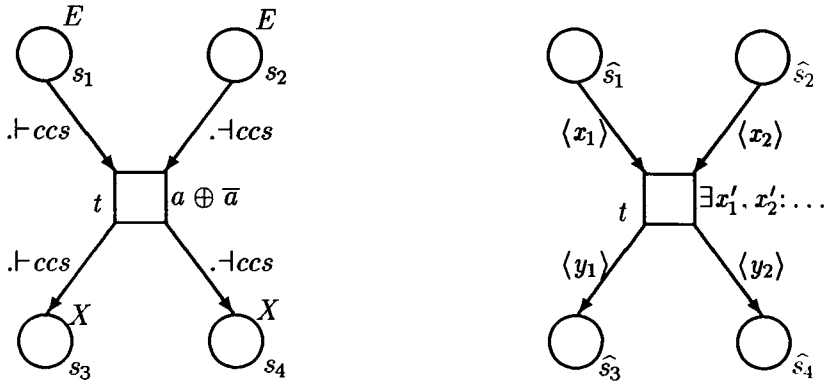


Figure 3.2: The High Level Petri Box of Example 3.2.6 and its PrT net denotation.

EXAMPLE 3.2.5 When an abstract transition consists of only a single local transition the translation is particularly simple. For, consider the High Level Petri Box $B = \langle S, T, \lambda \rangle$, shown in Figure 3.1, where

$$\begin{aligned} S &= \{s_1, s_2, s_3\} \\ T &= \{t\} \\ \lambda &= \{s_1 \to \{E\}, s_2 \to \{E\}, s_3 \to \{X\}\} \end{aligned}$$

and $t = \{\!|\langle \{s_1, s_2\}, (.), \tau, (.), \{s_3\} \rangle|\!\}$.

Then $PrT(B)$ is the labelled PrT net with components $\langle \langle S, T, F \rangle, \langle W_N, W_S, W_T, W_F \rangle \rangle$, where

$$\begin{aligned} F &= \{(s_1, t), (s_2, t), (t, s_3)\} \\ W_N &= \mathcal{R}_{\mathcal{L}_0} \\ W_S &= \hat{\_} : S \to \hat{S} \\ W_T(t) &= \exists x_1' : x_1' = (.) \frown x_1 \wedge x_1' = (.) \frown y_1 \wedge \mathrm{Flow}_{(\tau)}(x_1') \\ W_F(s_1, t) &= W_F(s_2, t) = \langle x_1 \rangle \\ W_F(t, s_3) &= \langle y_1 \rangle \end{aligned}$$

■ 3.2.5

EXAMPLE 3.2.6 When an abstract transition consists of more than a single local transition, synchronisation must be accommodated in the PrT net denotation. Consider the High Level

Petri Box $B = \langle S, T, \lambda \rangle$ of Figure 3.2, where

$$
\begin{aligned}
S &= \{s_1, s_2, s_3, s_4\} \\
T &= \{t\} \\
t &= \{\!|\langle \{s_1\}, (.\vdash ccs), a, (.\vdash ccs), \{s_3\}\rangle, \langle \{s_2\}, (.\dashv ccs), \overline{a}, (.\dashv ccs), \{s_4\}\rangle|\!\} \\
\lambda &= \{s_1 \to \{E\}, s_2 \to \{E\}, s_3 \to \{X\}, s_4 \to \{X\}\}
\end{aligned}
$$

$PrT(B)$ is the labelled PrT net with components $\langle \langle S, T, F \rangle, \langle W_N, W_S, W_T, W_F \rangle \rangle$, where

$$
\begin{aligned}
F &= \{(s_1, t), (s_2, t), (t, s_3), (t, s_4)\} \\
W_N &= \mathcal{R}_{\mathcal{L}_0} \\
W_S &= \hat{\cdot} : S \to \hat{S} \\
W_F(s_1, t) &= \langle x_1 \rangle \\
W_F(s_2, t) &= \langle x_2 \rangle \\
W_F(t, s_3) &= \langle y_1 \rangle \\
W_F(t, s_4) &= \langle y_2 \rangle
\end{aligned}
$$

$W_T(t)$, partially elided in the figure, is:

$$
\begin{aligned}
W_T(t) = \exists x_1', x_2' :\ & x_1' = (.\vdash ccs) \frown x_1 \wedge x_1' = (.\vdash ccs) \frown y_1 \\
\wedge\ & x_2' = (.\dashv ccs) \frown x_2 \wedge x_2' = (.\dashv ccs) \frown y_2 \\
\wedge\ & \mathrm{Flow}_{(a,\overline{a})}(x_1', x_2')
\end{aligned}
$$

■ 3.2.6

3.2.2.2 Predicate/Transition Net Denotations of Markings

We conclude this section and the definition of the PrT net semantics of High Level Petri Boxes with the denotation of a marking of a High Level Petri Box. The definition is simplified considerably as a High Level Petri Box and its PrT net denotation share the same places:

DEFINITION 3.2.7 Let M be a marking of a High Level Petri Box B. Define the *PrT net denotation of* M to be the marking $PrT(M)$ of $PrT(B)$ defined as $PrT(M) = M$. ■ 3.2.7

Because of their proximity, we will usually write M for both itself and its PrT denotation.

3.3 The Behaviour of a High Level Petri Box

The behaviour of a High Level Petri Box is defined as that of its denoting PrT net which is defined in this section.

In the following we will assume that $\mathcal{L} = \langle A, R \rangle$ is a label algebra, $B = \langle S, T, \lambda \rangle$ is a High Level Petri Box and that $P = PrT(B) = \langle N, W \rangle$ is its denoting PrT net with $N = \langle S, T, F \rangle$ and $W = \langle W_N, W_S, W_T, W_F \rangle$.

3.3.1 Indices and Substitutions

For a transition, t, of a PrT net, P, we will define the *index* of t, $index(t)$, to be the variables which annotate the arcs incident on t. From Definition 3.2.4, when $|t| = n$, $index(t) = \{x_1, \ldots, x_n, y_1, \ldots, y_n\}$. To be able to evaluate the transition selector of a transition, t, we must give values to the variables which appear in the index of t. The variables of the index of t will be bound to particular tokens which populate the places around t. References to the variables inside the transition selectors of t are thus instantiated with contexts. *Substitutions* are the mechanism through which this binding is achieved:

DEFINITION 3.3.1 A *substitution for a transition t over the label algebra $\mathcal{L}$* is a mapping $\alpha: index(t) \rightarrow C_{\mathcal{L}}$. ■ 3.3.1

An α-instance of an object $\mathcal{O}$, $\mathcal{O}{:}\alpha$, is defined as $\mathcal{O}[\alpha(x_1)/x_1, \ldots, \alpha(x_n)/x_n, \alpha(y_1)/y_1, \ldots, \alpha(y_n)/y_n]$. $\mathcal{O}$ may be any term, predicate, symbolic sum, *etc.*

3.3.2 Local-Strictness

Related to strictness is a property which we will term *local-strictness* for future reference. Local-strictness is used but not named by Genrich ([Gen87, Pg. 218, Definition 2.13(2)]). It ensures that substitutions evaluated on arc annotations produce sets. Practically, this implies that weighted nets are not required in the unfolding. As we will later define an unweighted P/T net unfolding of the High Level Petri Boxes which form the denotation of the algebra, we thus have established a proof obligation that High Level Petri Boxes produce locally-strict PrT nets. This is discharged for syntactically generated High Level Petri Boxes in Theorem 4.8.10.

DEFINITION 3.3.2 A substitution α is *locally-strict* if it creates a set on every arc incident on t, i.e., for every place s and arc annotation $a = W_F(s,t)$ or $a = W_F(t,s)$[7]. $0 \leq a{:}\alpha \leq +_{c \in C}\langle c \rangle$. ■ 3.3.2

When a transition selector is satisfied by a locally-strict substitution α, i.e, $W_T(t){:}\alpha$ holds, the transition may fire. In this case the substitution is said to be *feasible*. The feasible substitutions for a transition t are distinguished as:

DEFINITION 3.3.3 Let $t \in T$. Define $\text{subs}^P(t) = \{\alpha \mid \alpha \text{ locally-strict and } W_T(t){:}\alpha\}$. ■ 3.3.3

We may extend local-strictness to PrT nets:

DEFINITION 3.3.4 A PrT net, P, is *locally-strict* if for each transition t, each satisfying substitution for t is locally-strict. ■ 3.3.4

A High Level Petri Box B is locally-strict if $PrT(B)$ is locally-strict.

There are pathological cases of (marked) High Level Petri Boxes for which the marking defines a satisfying substitution of a transition of $PrT(B)$ which is not, however, locally-strict. An example is shown in Figure 3.3 in which local transitions l_1 and l_2 with the same pre- and post-set appear in the same abstract transition. The marking shown satisfies the transition selector of t:

$$\begin{aligned} W_T(t) \;=\; & \exists x_1', x_2' \colon \; x_1' = (.\vdash ccs) \frown x_1 \wedge x_1' = (.\vdash ccs) \frown y_1 \\ & \wedge \; x_2' = (.\dashv ccs) \frown x_2 \wedge x_2' = (.\dashv ccs) \frown y_2 \\ & \wedge \; \text{Flow}_{(a,\overline{a})}(x_1', x_2') \end{aligned}$$

(elided in the figure) in the label algebra $\mathcal{L}_0$ of Example 2.1.2 but the evaluation of $W_F(s_0,t)$ under the substitution $x_1 \mapsto (.)$, $x_2 \mapsto (.)$ is $\{\!|(.),(.)|\!\}$ which is not a set, meaning that the substitution is not locally-strict and so is not feasible. As mentioned above, such situations do not arise in syntactically generated High Level Petri Boxes.

3.3.3 Label Produced by a Feasible Substitution

The label produced by a transition has already been defined in terms of substitutions for the R_i in the Flow relation (Definition 2.4.5). As feasible substitutions satisfy the transition selectors, and hence their Flow conjunct, we may define the label of a transition under a feasible substitution.

[7] In which comparison is done component-wise.

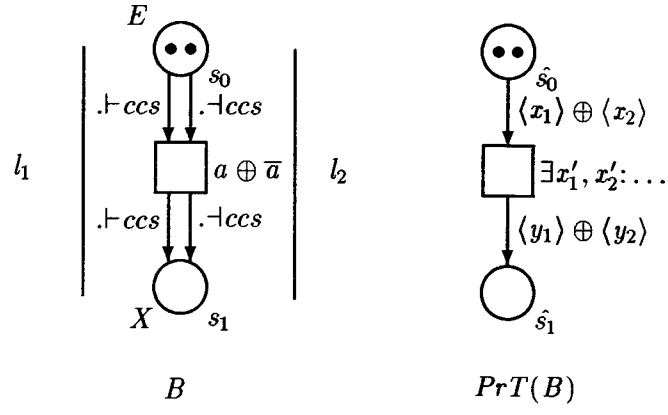


Figure 3.3: A High Level Petri Box at a marking whose PrT net denotation does not define a locally-strict substitution. The elided transition selector is given in the text.

DEFINITION 3.3.5 Let $t = \{\!|l_1, \ldots, l_n|\!\}$ and let $\alpha \in \text{subs}^P(t)$. Define *the label produced by the transition under* α to be $\overline{t{:}\alpha} = \text{Label}(\overline{l_1}, \ldots, \overline{l_n}, {}^C l_1\alpha(x_1), \ldots, {}^C l_n\alpha(x_n))$. ■ 3.3.5

3.3.4 A Symbolic Firing Rule for High Level Petri Box

To define the behaviour of a High Level Petri Box we must define what it means for an abstract transition to fire at a given marking, i.e., we must define the *symbolic firing rule.*

3.3.4.1 A Symbolic Firing Rule for Predicate/Transition Nets

Genrich defines his symbolic firing rule for PrT nets so that it commutes with his C/E system unfolding. Here we define a symbolic firing rule for PrT nets which commutes with a P/T net semantics, and which is subsequently used to define a symbolic firing rule for High Level Petri Boxes.

DEFINITION 3.3.6 *[cf., [Gen87, Pg. 219, Definition 2.16]]* Let $P = \langle N, A \rangle$ be a PrT net. Let M and M' be (not necessarily safe) markings of P, let t be a transition and α a substitution replacing the index of t by constants. Then the *α-occurrence* of t at M leading to M' is designated as $M[t{:}\alpha\rangle M'$ and defined by the following requirements:

1. α is a feasible substitution for t,
2. for all arcs (s, t) entering t, $W_F(s, t){:}\alpha \subseteq M(s)$,
3. for all places s, $M'(s) = (M(s) \setminus W_F(s, t){:}\alpha) \cup W_F(t, s){:}\alpha$. ■ 3.3.6

That there exists an α such that $M[t{:}\alpha\rangle M'$ we will denote $M[t\rangle M'$. That there exists a M' such that $M[t\rangle M'$ we will denote $M[t\rangle$. That there exists a t such that $M[t\rangle$, we denote $M[\rangle$.

Genrich's definition has a fourth requirement, omitted in the above, that $W_F(t,s){:}\alpha \cap M(s) = \emptyset$ which, together with his assumption of the purity of P, models the 'look ahead' firing rule for C/E systems.

3.3.4.2 Notions of Behaviour for High Level Petri Boxes

As B and $PrT(B)$ share the same places, transitions and flow relation, we may lift, directly, the notion of α-occurrence of a transition to a High Level Petri Box from which we may derive notions of behaviour for High Level Petri Boxes:

DEFINITION 3.3.7 Let M and M' be markings of a High Level Petri Box B. We say that M' is *directly reachable* from M, or that M' is an *immediate follower marking of* M, denoted $M[\rangle M'$, if there is a transition $t \in T_B$ and a feasible substitution α for t such that $M[t{:}\alpha\rangle M'$.

M' is *reachable from* M if[8] $M[\rangle^* M'$. Note that $M[\rangle^* M$.

The set of *reachable markings of* B *from the standard initial marking* d is defined as[9]

$$\mathcal{M}_d^B = \{M_d^I\} \lhd ([\rangle^*).$$

■ 3.3.7

We may combine an α-occurrence together with Definition 3.3.5 to produce a *labelled* α-*occurrence*:

DEFINITION 3.3.8 Let t be an abstract transition in some High Level Petri Box. Given the assumptions of Definition 3.3.6 define the *labelled* α-*occurrence of* t to be $M[\overline{t{:}\alpha}\rangle M'$, where $M[t{:}\alpha\rangle M'$ is the α-occurrence of t. ■ 3.3.8

We write $M \left[\begin{smallmatrix}\overline{t:\alpha}\\ t:\alpha\end{smallmatrix}\right\rangle M'$ for the combination of labelled and unlabelled (α-)occurrences.

Other standard notions of behaviour follow directly.

[8] Where $[\rangle^*$ is the reflexive, transitive closure of $[\rangle$.

[9] Where $\lhd$ is domain restriction, i.e., for a relation R and a set S, $S \lhd R = \{r \mid (s,r) \in R \wedge s \in S\}$.

3.3.4.3 Interpretation on High Level Petri Boxes

The intuition behind the symbolic firing rule for High Level Petri Boxes interprets a local transition as a stack-like object, in which pre-contexts are 'pushed' and post-contexts are 'popped'.

Consider a transition $t = \{\!|l_1, \ldots, l_n|\!\}$ in some High Level Petri Box B. That a substitution α forms an α-occurrence of t implies that we may find tokens $C_i \in C$ covering the pre-set of local transitions l_i, such that

SFR1. ${}^{C}l_i \frown \alpha(C_i)(= x_i') \in C$, i.e., we may 'push' the pre-contexts of the local transitions onto the tokens,

SFR2. ${}^{C}l_i\alpha(C_i) - l_i{}^{C} \in C$, i.e., we may 'pop' the post-contexts of the local transitions from the tokens,

SFR3. Flow$(\overline{l_1}, \ldots, \overline{l_n}, x_1', \ldots, x_n')$, i.e., the H-synchrony rule would permit the firing of the transition.

Thus the firing of a transition of a High Level Petri Box derives from the properties of the H-synchrony rule under the proviso that the push/pop behaviour of the local transitions of which it comprises is well-defined.

3.3.4.4 Examples

We return briefly to Examples 3.2.5 and 3.2.6 to illustrate the firing rule for High Level Petri Boxes.

EXAMPLE 3.3.9 *[Example 3.2.5 continued]* Let M be a marking of the PrT net such that $M(s_1) = \{\!|(.)|\!\} = M(s_2)$ and $M(s_3) = \{\!|\,|\!\}$. Transition t is enabled at M with feasible substitution α, such that $\alpha(x_1) = \alpha(y_1) = (.)$. H-synchrony requires only that $\tau(.) = \tau \in A$ which is, indeed, the case. The follower marking is M', defined by $M'(s_1) = \{\!|\,|\!\} = M'(s_2)$, $M'(s_3) = \{\!|(.)|\!\}$, i.e., $M[t{:}\,\alpha\rangle M'$. ■ 3.3.9

EXAMPLE 3.3.10 *[Example 3.2.6 continued]* Assume a standard initial marking M_d^I to be given, i.e., such that $M(s_1) = M(s_2) = \{\!|d|\!\}$, and that we wish to determine whether transition

t may fire at this marking. To this end we form the substitution α such that $\alpha(x_1) = d = \alpha(x_2)$. Performing the substitution $W_T(t)\colon\alpha$ gives

$$\begin{aligned} W_T(t)\colon\alpha \quad \Leftrightarrow \quad \exists x_1', x_2' \in C_{\mathcal{L}_0}\colon \quad & x_1' = (.\vdash ccs)d = (.\vdash ccs)d \\ \wedge \quad & x_2' = (.\dashv ccs)d = (.\dashv ccs)d \\ \wedge \quad & \mathrm{Flow}_{(a,\overline{a})}(x_1', x_2') \end{aligned}$$

We may choose the x_i' consistently if and only if $x_1' = (.\vdash ccs)d$ and $x_2' = (.\dashv ccs)d$.

Substituting for the x_i' in the definition of Flow when $n = 2$ gives:

$$\begin{aligned} & \mathrm{Flow}(a, \overline{a}, x_1', x_2') \\ & \quad \Leftrightarrow \exists D \in A_0\colon \mathrm{Flow}_0(\{\!|(a, C_1(.\vdash)C), (\overline{a}, C_2(.\dashv)C)|\!\}, D) \\ & \quad \Leftrightarrow \exists D \in A_0\colon \mathrm{Flow}_0(\{\!|(a, C_1)|\!\}, D_1) \wedge \mathrm{Flow}_0(\{\!|(\overline{a}, C_2)|\!\}, D_2) \wedge D = (D_1 \oplus D_2)[C] \\ & \quad \Leftrightarrow \exists D \in A_0\colon a[C_1] \neq \mathbf{0} \wedge \overline{a}[C_2] \neq \mathbf{0} \wedge D = (a[C_1] \oplus \overline{a}[C_2])[C] \end{aligned}$$

Substituting for C_1, C_2 and C we see that transition t may fire when

$$a \neq \mathbf{0} \wedge \overline{a} \neq \mathbf{0} \wedge (a \oplus \overline{a})[(.ccs)d] = \tau[d] \in A$$

which does not impose any further restriction on d, due to the definition of τ. The follower marking, M', is then $M'(s_1) = \{\!|\,|\!\} = M'(s_2)$ and $M'(s_3) = \{\!|d|\!\} = M'(s_4)$. The label produced by the transition is then $\tau[d] = \tau$. Hence

$$M \left[\begin{matrix} \tau \\ t\colon\alpha \end{matrix} \right\rangle M'.$$

■ 3.3.10

3.3.4.5 Distinguishability of Substitutions

$M[t\colon\alpha_1\rangle M'$ and $M[t\colon\alpha_2\rangle M'$ does not necessarily imply that $\overline{t\colon\alpha_1} = \overline{t\colon\alpha_2}$, so that distinct t-instances may be distinguishable through the labels they generate. An example of this is given in Figure 3.4.

However:

PROPOSITION 3.3.11 Let B be a High Level Petri Box with $t = \{\!|l|\!\} \in T_B$ and ${}^\bullet l \neq l^\bullet$. Then $M[t\colon\alpha_1\rangle M'$ and $M[t\colon\alpha_2\rangle M'$ implies $\alpha_1 = \alpha_2$. □ 3.3.11

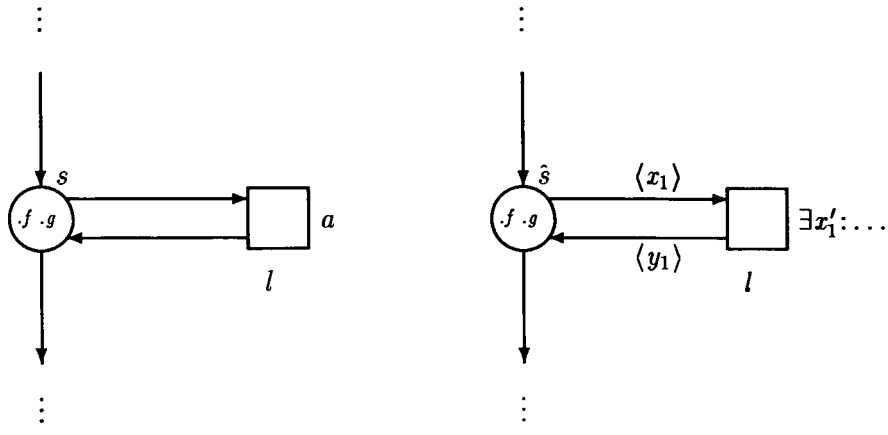


Figure 3.4: A (a) (portion of a) High Level Petri Box and (b) its PrT net denotation in which two occurrences between the same markings can be distinguished. At the marking $M = \{s \to \{\!|.f, .g|\!\}\}$: if $\alpha_1(x_1) = .f$, $\alpha_1(y_1) = .f$, $\alpha_2(x_1) = .g$, $\alpha_2(y_1) = .g$ then $M[\{\!|l|\!\}{:}\alpha_1\rangle M$ and $M[\{\!|l|\!\}{:}\alpha_2\rangle M$, but $\overline{\{\!|l|\!\}{:}\alpha_1} \neq \overline{\{\!|l|\!\}{:}\alpha_2}$ whenever $f(a) \neq g(a)$.

Proof: Suppose to the contrary, that $M[t{:}\alpha_1\rangle M'$ and $M[t{:}\alpha_2\rangle M'$ but $\alpha_1 \neq \alpha_2$. Suppose $index(t) = \{x, y\}$ (x annotating input arcs, y annotating output arcs) and let $\epsilon_i = \alpha_i(x)$ and $f_i = \alpha_i(y)$, $i = 1, 2$. As $\alpha_1 \neq \alpha_2$ and $\widehat{\ }$ is single valued, $\epsilon_1 \neq e_2$ and $f_1 \neq f_2$. Moreover, as $M[t{:}\alpha_1\rangle M'$ and $M[t{:}\alpha_2\rangle M'$, we have that

$$(M \setminus {}^\bullet l \times \{\!|e_1|\!\}) \cup l^\bullet \times \{\!|f_1|\!\} = M' = (M \setminus {}^\bullet l \times \{\!|e_2|\!\}) \cup l^\bullet \times \{\!|f_2|\!\}$$

Now, as ${}^\bullet l \neq l^\bullet$, there is a place $s \in ({}^\bullet l \setminus l^\bullet) \cup (l^\bullet \setminus {}^\bullet l)$. Assume that $s \in {}^\bullet l \setminus l^\bullet$ (the other case is similar) whence, if $M(s)(e_i) = n_i > 0$, $i = 1, 2$, then, under α_1, $M'(s)(e_1) = n_1 - 1$ and $M'(s)(e_2) = n_2$ whereas, under α_2, $M'(s)(e_1) = n_1$ and $M'(s)(e_2) = n_2 - 1$, a contradiction.

Hence the result. ■ 3.3.11

In particular, then, $\overline{t{:}\alpha_1} = \overline{t{:}\alpha_2}$ and $t = \{\!|l|\!\}$ and hence we may write $M[t\rangle M'$ without ambiguity.

3.4 The Most Permissive Label Algebra

Given two label algebras sharing the same signature, we will consider one *more permissive* than another if the former prevents less synchronisations from occurring than the latter. This defines a pre-order on label algebras with the same signature. Of particular interest is the maximal

label algebra with respect to this ordering, which we call the *most permissive label algebra*: it is the label algebra which does not prevent any synchronisation:

DEFINITION 3.4.1 Given an alphabet A, relabellings $R = \{f, \ldots\}$ forming a label algebra $\mathcal{L}$, and an element $a \in A$, a *most permissive label algebra* $\mathcal{L}_{\mathcal{MP}} = \langle A, R \rangle$ *(on A and R with respect to a)* is defined so that $f(\mathbf{0}) = \mathbf{0}$ and $f(k) = a$ for all other $k \in A^{\overline{=}}$. ■ 3.4.1

By definition, an alphabet A is non-empty, so that we may always choose $a \in A$. As the choice of a is unimportant we will generally speak of *the* most permissive label algebra.

We note that the references to labels within the Flow predicate test whether the label produced by the transition evaluates to an allowable action of the label algebra. Therefore if, for the label algebra $\mathcal{L} = \langle A, R \rangle$, we were to substitute the most permissive label algebra over $\langle A, R \rangle$ these components would only check whether the label produced is $\mathbf{0}$. As is clear, the finding of a partitioning is independent of the label algebra so that substituting the most permissive label algebra in this way does not otherwise affect the satisfaction of the predicate.

3.4.1 Positive Monotonic Properties

The importance of the most permissive label algebra can be seen when considering certain marking related properties of High Level Petri Boxes which are monotonic in the permissiveness of the label algebra over which the High Level Petri Box is defined. This means that, for a High Level Petri Box, B, and a property of markings $\mathcal{P}$, if $\mathcal{P}$ holds of B in label algebra $\mathcal{L}_1$ and $\mathcal{L}_2$ is less permissive that $\mathcal{L}_1$, then $\mathcal{P}$ holds of B in the label algebra $\mathcal{L}_2$. Therefore a proof for the most permissive label algebra constitutes a proof for all other label algebras for such properties.

Proving such properties in the most permissive label algebra is made easier as we do not have to consider the label produced by a transition in a firing, and we will thus be able to consider marking theoretic properties of High Level Petri Boxes even for which we have no notion of 'labelled' behaviours. This will be particularly important when we extend label sets with *process variables*, and introduce constructs on High Level Petri Boxes which model refinement and recursion.

DEFINITION 3.4.2 Let $\mathcal{P}$ be a property of High Level Petri Boxes. Then $\mathcal{P}$ is said to be *positive monotonic on label algebras* if, when $\mathcal{P}$ holds of B over label algebra $\mathcal{L}_1$ and $\mathcal{L}_2$ is less permissive than $\mathcal{L}_1$, then $\mathcal{P}$ holds of B over the label algebra $\mathcal{L}_2$. ■ 3.4.2

3.4.2 Completeness and Decoupling in the Most Permissive Label Algebra

In a *complete* High Level Petri Box all local transitions appear as abstract transitions:

DEFINITION 3.4.3 Let B be a High Level Petri Box. B is said to be *complete* if $l \in t \in T_B$ implies $\{\!|l|\!\} \in T_B$. ■ 3.4.3

However, even in a complete High Level Petri Box we cannot be sure that if t is enabled at a marking then any local transition $l \in t$, when regarded as an abstract transition, is enabled at that marking. In the most permissive label algebra, however, no non-dead transition is disabled. Hence:

LEMMA 3.4.4 *[Decoupling Lemma]* Suppose $M[t\rangle M'$ in a complete High Level Petri Box B over the most permissive label algebra. Then $t = \{\!|l_1, \ldots, l_n|\!\}$ implies there are markings $M_0 = M, M_1, \ldots, M_n = M'$ such that $M_0[\{\!|l_1|\!\}\rangle \ldots [\{\!|l_n|\!\}\rangle M_n$. □ 3.4.4

Proof: As B is complete, $l_i \in t \in T_B$ implies $\{\!|l_i|\!\} \in T_B$. $M[t\rangle M'$ implies there is a feasible substitution α such that $M[t{:}\alpha\rangle M'$. As we are working with the most permissive label algebra, that α is a feasible substitution for t at M implies[10] for each i, $\alpha\lfloor_{\{x_i, y_i\}}$ forms a feasible substitution for $\{\!|l_i|\!\}$ at M, where x_i and y_i are the variables corresponding to the local transition l_i.

The result follows. ■ 3.4.4

We will call a transition sequence which contains only local transitions considered as abstract transitions a *local transition sequence.* For a local transition sequence we will usually write $M[l\rangle N$ for $M[\{\!|l|\!\}\rangle N$. We will also write $\alpha^{(i)}$ for $\alpha\lfloor_{\{x_i, y_i\}}$ as identified in the proof Lemma 3.4.4.

The particular value of the decoupling lemma is that it allows the proof of positive monotonic properties to be conducted in the most permissive label algebra, on local transition sequences, but with full generality.

LEMMA 3.4.5 Let $\mathcal{P}$ be a property of reachable markings of High Level Petri Boxes which is positive monotonic in the permissiveness of the label algebra. Then for all complete High Level Petri Boxes B over a label algebra $\mathcal{L}$ if $\mathcal{P}$ holds of markings reachable through local transition sequences of B over the most permissive label algebra $\mathcal{L}_{\mathcal{MP}}$ then $\mathcal{P}$ holds of all reachable markings of B over $\mathcal{L}$. □ 3.4.5

[10] Where, for a function $f{:}\ U \to V$, $f\lfloor_W{:}(U \cap W) \to V$ is such that $f\lfloor_W(w) = f(w)$, the *restriction of* f *to* W.

Proof: In a complete High Level Petri Box over a label algebra $\mathcal{L}$, using Lemma 3.4.4, we may generate all reachable markings of B by considering the local transition sequences of B. The result follows. ■ 3.4.5

Hence, to establish a positive monotonic marking theoretic property of a complete High Level Petri Box B over a label algebra $\mathcal{L}$, we need establish it only for local transition sequences of B in the most permissive label algebra.

Examples of positive monotonic properties of High Level Petri Boxes are safeness and memorylessness (which are defined in Section 4.8.3.4).

Working with local transition sequences provides us with much freedom as to the order in which we consider the firings of the local transitions:

LEMMA 3.4.6 Let l_1, $l_2 \in LT(B)$ for some High Level Petri Box B, and suppose $M[l_1{:}\alpha_1\rangle M'[l_2{:}\alpha_2\rangle N$. Then ${l_1}^\bullet \cap {}^\bullet l_2 = \emptyset$ implies there is a marking M'' such that $M[l_2{:}\alpha_2\rangle M''[l_1{:}\alpha_1\rangle N$. □ 3.4.6

Proof: ${l_1}^\bullet \cap {}^\bullet l_2 = \emptyset$ implies α_2 is a feasible substitution for l_2 at M. The result follows. ■ 3.4.6

3.4.3 Initial Marking Independence

For a local transition l to be enabled at a marking M there must be a token e such that ${}^\bullet l \times \{\!|e|\!\} \subseteq M$ and ${}^C l^\frown e - l^C \in C$. But then, for any $d \in C$, ${}^C l^\frown (e^\frown d) - l^C \in C$ also. Whence

THEOREM 3.4.7 Let B be a complete High Level Petri Box over the most permissive label algebra. Then $M \in \mathcal{M}^B_{(.)}$ implies[11] $M(d) \in \mathcal{M}^B_d$. □ 3.4.7

The reverse result is not so general. Consider the High Level Petri Box B illustrated in Figure 3.5. Then, even though B's single transition may fire at a standard initial marking $M^B_{(.f)}$, it is disabled at standard initial marking $M^B_{(.)}$. The best we can achieve is:

THEOREM 3.4.8 Let d be a context. Then for B a complete High Level Petri Box over a most permissive label algebra such that $\forall M \in \mathcal{M}^B_d$, $\forall e \in \mathrm{ran}(M)$, $d \sqsubseteq e$, $M \in \mathcal{M}^B_d$ implies that the unique marking M' such that $M = M'(d)$ satisfies $M' \in \mathcal{M}^B_{(.)}$. □ 3.4.8

[11] Where $M(d) = \{\!|(s, e^\frown d) \mid (s,e) \in M|\!\}$.

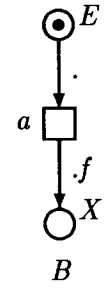


Figure 3.5: A High Level Petri Box, B, in which the push/pop behaviour is not always well-defined.

Proof: As we work in the most permissive label algebra, the Flow component of the symbolic firing rule will not cause a transition to be disabled. Let $M \in \mathcal{M}_d^B$. The proof will be by induction over the length of the transition sequence which led to M. From Lemma 3.4.4, we need only consider local transition sequences for full generality. So let

$$M_d^I = M_0[l_1\rangle \ldots M_{n-1}[l_n\rangle M_n = M$$

be a local transition sequence which led to M.

If $n = 0$ then $M = M_d^I$ and $M' = M_{(.)}^I$ satisfies the property.

Otherwise $n > 0$. Suppose the property holds for $n-1$, i.e., $\exists M'_{n-1} \in \mathcal{M}_{(.)}^B$ such that $M_{n-1} = M'_{n-1}(d)$. As l_n is enabled at M_{n-1}, there are contexts ϵ and f such that ${}^\bullet l \times \{\!|e|\!\} \subseteq M_{n-1}$, $f = {}^C l_n e - l_n{}^C$ and

$$M_n = (M_{n-1} \setminus {}^\bullet l_n \times \{\!|e|\!\}) \cup l^\bullet \times \{\!|f|\!\}$$

As $M_n \in \mathcal{M}_d^B$ we have, from the hypotheses, that $d \sqsubseteq e$ so that ${}^\bullet l_n \times \{\!|e|\!\} \subseteq M_{n-1} = M'_{n-1}(d)$ implies ${}^\bullet l_n \times \{\!|e-d|\!\} \subseteq M'_{n-1}$. Moreover, as $d \sqsubseteq f, f-d = {}^C l_n(e-d) - l_n{}^C \in C$ and there is a marking $M' \in \mathcal{M}_{(.)}^B$ such that

$$M' = (M'_{n-1} \setminus {}^\bullet l_n \times \{\!|e-d|\!\}) \cup l_n{}^\bullet \times \{\!|f-d|\!\}$$

Moreover, $M'(d) = M_n$. Set $M_n = M'$ for the result. ■ 3.4.8

As, in the proof, we relate markings using the same local transition, we have actually proven much more with Theorems 3.4.7 and 3.4.8; namely that, under any unlabelled notion of behaviour, the behaviours of B from standard initial marking (.) and from standard initial marking d are equal (modulo the d postfix).

Establishing the hypotheses of Theorem 3.4.8 for certain classes of High Level Petri Boxes will occupy much of the remainder of this thesis. For the High Level Petri Boxes of Chapter 4, the result is obtained by consideration only of the *structure* of the High Level Petri Box (in the sense as explained in that chapter). For the High Level Petri Boxes of Chapter 6, the result is obtained for a wider class of High Level Petri Boxes (those involving our recursive construct) although, for this wider class, we are required to consider the behaviours of the High Level Petri Box, rather than just structure.

3.5 A P/T Net Unfolding of High Level Petri Boxes

In Chapter 5 we will be required to show that two structurally different High Level Petri Boxes have the same behaviour. This equivalence is defined as the isomorphism of reachable subnets of the following P/T net unfolding of a High Level Petri Box. The unfolding derives from the C/E system unfolding of PrT nets which appears in [Gen87]. The differences are that:

1. we do not insist on the *purity* of the source High Level Petri Box, B, i.e., there may be transitions $t \in T_B$ such that ${}^\bullet t \cap t^\bullet \neq \emptyset$, and
2. we do not remove isolated transitions and places from the denotation of the High Level Petri Box.

Each allows us to simplify the presentation a little.

3.5.1 A Transition Centred Place/Transition Net Model

The target P/T net model used in the unfolding is, like High Level Petri Boxes, transition centred:

DEFINITION 3.5.1 Given a set A, $N = \langle S, T \rangle$ is an *A-labelled P/T net* where S is a set of places, and $T \subseteq \mathbb{P}S \times A \times \mathbb{P}S$ is a set of transitions. $N = \langle S, T, M \rangle$ is a *marked A-labelled P/T net* if $\langle S, T \rangle$ is an A-labelled net and $M: S \to \mathbb{N}$.

For $t = \langle S_1, a, S_2 \rangle \in T$, we write ${}^\bullet t = S_1$, $\overline{t} = a$ and $t^\bullet = S_2$. ■ 3.5.1

The following are standard: a transition t is *enabled at a marking* M if ${}^\bullet t \subseteq M$. If a transition t is enabled at a marking M of a marked net N, then t may *fire*, written $M \left[\begin{smallmatrix} a \\ t \end{smallmatrix} \right\rangle M'$, when

$M' = (M \setminus {}^\bullet t) \cup t^\bullet$ and $a = \overline{t}$. We write $M[a\rangle M'$ if there is a transition t such that $M \left[\begin{smallmatrix} a \\ t \end{smallmatrix}\right\rangle M'$. and $M[\rangle M'$ if there is an a such that $M[a\rangle M'$. M' *is reachable from* M if $M([\rangle^*)M'$. The *reachable markings of a marked net* $N = \langle S, T, M\rangle$ is the set $[M\rangle = M \triangleleft ([\rangle^*)$. A marked net $N = \langle S, T, M\rangle$ is *safe* if for all $M' \in [M\rangle$ and for all $s \in S$, $M'(s) \leq 1$ (whence we may treat each marking as a set rather than a multiset). The *reachable subnet of a safe marked net* $N = \langle S, T, M\rangle$ is the (safe) marked net $\mathrm{Reach}(N) = \langle \bigcup [M\rangle, \{t \in T \mid \exists M' \in [M\rangle : {}^\bullet t \subseteq M'\}, M\rangle$.

For details of the properties of such transition centred representations of P/T nets we refer the reader to, for instance, [Gol88].

3.5.2 α-Instances of Transitions

Due to the special form of the dynamic predicates of our relational structure (i.e., they are unary) and as we require a labelled P/T net denotation, we define the following specialisation of the *α-instances* of transitions of Genrich:

The reader will recall that we are assuming $\mathcal{L} = \langle A, R\rangle$ is a label algebra, B is a High Level Petri Box over $\mathcal{L}$ and that $P = PrT(B)$ is its denoting PrT net.

DEFINITION 3.5.2 Let $\alpha \in \mathrm{subs}^P(t)$. Define an *$\alpha$-instance of* t, $t{:}\alpha$, to be the triple $\langle {}^\bullet t{:}\alpha, \overline{t{:}\alpha}, t{:}\alpha^\bullet\rangle$ where

$$
\begin{aligned}
{}^\bullet t{:}\alpha &= \{W_S(s)d \mid s \in {}^\bullet t \wedge d \in W_F(s,t){:}\alpha\} \\
t{:}\alpha^\bullet &= \{W_S(s)d \mid s \in t^\bullet \wedge d \in W_F(t,s){:}\alpha\}
\end{aligned}
$$

■ 3.5.2

α-instances of a transition are combined in the construction that follows to form a P/T net from a PrT net. Each corresponds to a transition of a PrT net enabled at a marking (which determines a feasible substitution), together with the label that would be produced by the transition and the follower marking if fired. Note that, as α is feasible and hence locally-strict, sets (and not multi-sets) are sufficient to define ${}^\bullet t{:}\alpha$ and $t{:}\alpha^\bullet$.

3.5.3 The Construction

We may now define the construction of a P/T net unfolding of a High Level Petri Box:

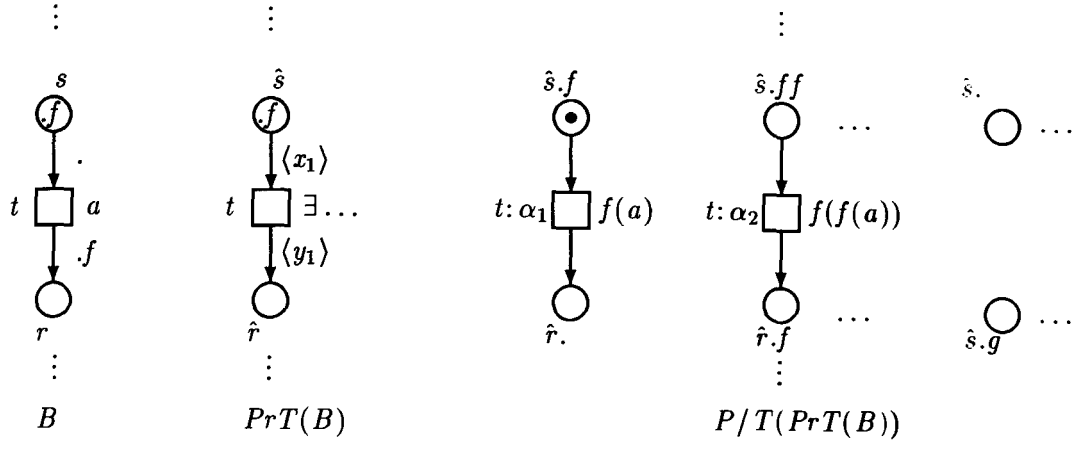


Figure 3.6: Illustrating the relationship between a PrT net and its unfolding. $\alpha_1 = \{x_1 \to (.f), y_1 \to (.)\}$ and $\alpha_2 = \{x_1 \to (.ff), y_1 \to (.f)\}$ are the illustrated feasible substitutions. The markings of the various nets shown are equivalent.

DEFINITION 3.5.3 Given the unmarked PrT net denotation of the High Level Petri Box B, $P = \langle N, W\rangle$ where $N = \langle S, T, F\rangle$ and $W = \langle W_N, W_S, W_T, W_F\rangle$, define an unmarked A-labelled P/T net, $P/T(P) = \langle S', T'\rangle$ with

$$\begin{aligned} S' &= \{W_S(s)d \mid s \in S \wedge d \in C\} \\ T' &= \{t{:}\alpha \mid t \in T \wedge \alpha \in \mathrm{subs}^P(t)\} \end{aligned}$$

Define the *P/T net unfolding* of B, $P/T(B) = P/T(P)$. ■ 3.5.3

The places of the P/T unfolding of B consist of place/token pairs: a marking of a place $\hat{s}d$ in the P/T net corresponds to the marking of the place s by d in the PrT net. This correspondence is illustrated in Figure 3.6 in which is shown a single transition of some High Level Petri Box B (over a label algebra with a single label a and two relabellings f and g such that $f(a) = a$ and $g(a) = \mathbf{0}$), its PrT net denotation, $PrT(B)$, and its P/T net unfolding, $P/T(B)$. To determine the transitions of the P/T net unfolding corresponding to the transition t we must calculate the feasible substitutions for t. Given the interpretation of the transition selector for t (elided in the figure) is $W_T(t) = \exists x_1' \in C{:}\, x_1' = (.)^\frown x_1 \wedge x_1' = (.f)^\frown y_1 \wedge a[x_1'] \in A$, these consist of substitutions of the form $\{\alpha_n \mid \alpha_n = \{x_1 \mapsto .f^n.\, y_1 \mapsto .f^{(n-1)}\} \wedge n \geq 1\}$. Two such feasible substitutions, α_1 and α_2, are illustrated in the figure together with other places which do not contribute α-instances of t.

Using this correspondence between markings, we may define the marking of the P/T net unfolding of a High Level Petri Box B corresponding to a standard initial marking M_d^I. It consists

of the set $S^d = \{ W_S(s)d \mid E \in \lambda(s)\} \subseteq S'$.

If $P/T(B) = \langle S, T\rangle$ and $\langle S, T, S^d\rangle$ is a safe marked net, we define Reach(B, d) = Reach$(\langle S, T, S^d\rangle)$, i.e., that subnet of $P/T(B)$ which is reachable from the marking S^d.

3.6 Relations on High Level Petri Boxes

In addition to equality, $=$, and isomorphism, $\equiv$, of High Level Petri Boxes, the rich semantics defined in this chapter provides an equally rich collection of relations which may be defined between High Level Petri Boxes. We briefly define and describe their interrelationship in this section.

DEFINITION 3.6.1 Define relations $=_{PrT}$, $\equiv_{PrT}$, $=_{P/T}$, and $\equiv_{P/T}$ on High Level Petri Boxes such that, for High Level Petri Boxes B_1 and B_2:

1. $B_1 =_{PrT} B_2 \Leftrightarrow PrT(B_1) = PrT(B_2)$ (modulo α-congruence of variables of arc annotations),
2. $B_1 \equiv_{PrT} B_2 \Leftrightarrow PrT(B_1) \equiv PrT(B_2)$ as unmarked PrT nets (modulo α-congruence of variables of arc annotations and logical equivalence of transition selectors),
3. $B_1 =_{P/T} B_2 \Leftrightarrow P/T(B_1) = P/T(B_2)$,
4. $B_1 \equiv_{P/T} B_2 \Leftrightarrow P/T(B_1) \equiv P/T(B_2)$ as labelled P/T nets.

Additionally, for d a context, define

5. $B_1 \equiv^d_{Reach} B_2 \Leftrightarrow$ Reach$(B_1, d) \equiv$ Reach(B_2, d), i.e., the subnets of the labelled P/T denotations of B_1 and B_2, reachable from a standard initial marking d. are isomorphic.

We will write $B_1 \equiv_{Reach} B_2$ when $B_1 \equiv^d_{Reach} B_2$ for all contexts d. ■ 3.6.1

Suitable definitions of the class isomorphisms are available in the literature. Their interrelation is given in Table 3.1. Of some note is that $=_{PrT} \subseteq =$, i.e., High Level Petri Boxes are in bijective correspondence with a subclass of PrT nets.

From its definition the reader will note that $\equiv^d_{Reach}$ characterises behavioural equivalence between High Level Petri Boxes in terms of the structure of their respective P/T net unfoldings. The relation is used in Chapter 5 where we will wish to show that structurally unequal High Level Petri Boxes have the same behaviours.

	$=$	$\equiv$	$=_{PrT}$	$\equiv_{PrT}$	$=_{P/T}$	$\equiv_{P/T}$	$\equiv^{d}_{Reach}$
$=$		$\subseteq$	$\subseteq$	$\subseteq$	$\subseteq$	$\subseteq$	$\subseteq$
$\equiv$				$\subseteq$		$\subseteq$	$\subseteq$
$=_{PrT}$	$\subseteq$	$\subseteq$		$\subseteq$	$\subseteq$	$\subseteq$	$\subseteq$
$\equiv_{PrT}$		$\subseteq$				$\subseteq$	$\subseteq$
$=_{P/T}$						$\subseteq$	$\subseteq$
$\equiv_{P/T}$							$\subseteq$
$\equiv^{d}_{Reach}$							

Table 3.1: Interrelation between High Level Petri Box Equivalences.

3.6.1 Showing Reachable Equivalence from a Standard Initial Marking

We have mentioned that the symbolic firing rule for High Level Petri Boxes commutes with that of its P/T net unfolding. Essentially, what this means is that $M \left[\begin{smallmatrix} a \\ t:\alpha \end{smallmatrix} \right\rangle M'$ in $P/T(B)$ if and only if $M_{PrT} \left[\begin{smallmatrix} a \\ t:\alpha \end{smallmatrix} \right\rangle M'_{PrT}$, with $M_{PrT} = \{s \rightarrow e \mid \hat{s}e \in M\}$ and $M'_{PrT} = \{s \rightarrow e \mid \hat{s}e \in M'\}$. Note that, when defined in this way, $(S^d)_{PrT} = M^I_d$.

The establishment of reachable equivalence at the level of High Level Petri Boxes then reduces to:

PROPOSITION 3.6.2 Let B_1, B_2 be High Level Petri Boxes over a label algebra $\mathcal{L} = \langle A, R \rangle$ with $d \in \mathcal{C}_{\mathcal{L}}$, such that:

1. $S_1 = S_2$,
2. there is a bijective correspondence $\Psi: LT(B_1) \equiv LT(B_2)$, such that ${}^\bullet\Psi(l) = {}^\bullet l$, $\overline{\Psi(l)} = \bar{l}$, $\Psi(l)^\bullet = l^\bullet$, and such that $\Psi: T_1 \equiv T_2$, with $\Psi(\{\!|l_1, \ldots, l_n|\!\}) = \{\!|\Psi(l_1), \ldots, \Psi(l_n)|\!\}$;
3. there is a bijection $\mathcal{X}: \mathcal{M}_d^{B_1} \rightarrow \mathcal{M}_d^{B_2}$ such that:
   (a) $\mathcal{X}(M^I_d) = M^I_d$,
   (b) for all M, $M' \in \mathcal{M}_d^{B_1}$, $t \in T_1$, $\alpha \in \mathrm{subs}^{B_1}(t)$ such that $M \left[\begin{smallmatrix} a \\ t:\alpha \end{smallmatrix} \right\rangle M'$ in B_1 there is a unique feasible substitution $\gamma \in \mathrm{subs}^{B_2}(\Psi(t))$ such that
   $$\mathcal{X}(M) \left[\begin{smallmatrix} a \\ \Psi(t):\gamma \end{smallmatrix} \right\rangle \mathcal{X}(M').$$

(c) for all M, $M' \in \mathcal{M}_d^{B_2}$, $t \in T_2$, $\alpha \in \text{subs}^{B_2}(t)$ such that $M \left[\begin{smallmatrix} a \\ t:\alpha \end{smallmatrix} \right\rangle M'$ in B_2 there is a unique feasible substitution $\gamma \in \text{subs}^{B_1}(\Psi^{-1}(t))$ such that

$$\mathcal{X}^{-1}(M) \left[\begin{smallmatrix} a \\ \Psi^{-1}(t):\gamma \end{smallmatrix} \right\rangle \mathcal{X}^{-1}(M').$$

Then $B_1 \equiv^d_{Reach} B_2$. □ 3.6.2

Proof: (Outline) Follows by an inductive argument over the reachable markings, using the commutativity of symbolic firing rules for High Level Petri Boxes and P/T nets to extend Hypotheses 3(a)–3(c) to the reachable subnets of the P/T net unfolding. ■ 3.6.2

There are more general statements of Proposition 3.6.2—for instance, we may relax the requirement that $S_1 = S_2$ to be a bijective correspondence, and allow the relationship between feasible substitutions to be one-many. However, such generality is unnecessary in the following treatment.

It is useful to note that if $\mathcal{X}: \mathcal{M}_d^{B_1} \to \mathcal{M}_d^{B_2}$ is defined such that $\mathcal{X}(M) = \{\!|s \to \chi_s(\epsilon) \mid s \to \epsilon \in M|\!\}$ where, for each place $s \in S_1$, $\chi_s: \mathcal{C}_\mathcal{L} \leftrightarrow \mathcal{C}_\mathcal{L}$ with $\chi_s \cap (\text{dom}(\chi_s) \times \text{ran}(\chi_s))$ a bijection, then Items 3(a)—3(c) are sufficient to establish the bijective nature of $\mathcal{X}: \mathcal{M}_d^{B_1} \equiv \mathcal{M}_d^{B_2}$.

3.7 Discussion

3.7.1 A Comparison with the Denotation of Taubner

The *shorthand notation* of Taubner gives PrT net denotations of certain terms[12] of his algebra A. Whereas we have discussed his algebra with regard to expressivity and label algebras, there are interesting parallels between the structure of his and our denotations.

Taubner's synchronisations, as we have seen, are inherently binary in nature. Although Taubner does not explicitly refer to the *order of a transition* (as defined in Definition 2.5.6), the notion may be usefully applied to his transitions: his 'singular' transitions have order 1, his 'synchronisation' transition have order 2. No transition has order greater than 2.

For High Level Petri Boxes, in the case that $|t| = 1$, i.e., $t = \{\!|l|\!\}$, that t does not contribute to a synchronisation is reflected in the simplification of the transition selector $W_T(t)$ at a substitution α to:

$$W_T(t):\alpha \quad \Leftrightarrow \quad \exists x_1' \in \mathcal{C}: x_1' = {}^{\mathcal{C}}l\alpha(x_1) \wedge x_1' = l^{\mathcal{C}}\alpha(y_1) \wedge \bar{l}[x_1'] \in A$$

[12] Those without initial parallelism in a choice composition, and with guarded recursion.

This is, essentially, the transition selector of Taubner, the differences being syntactic. Indeed in this case, we have a very simple firing rule which does not depend upon any synchronisation considerations—the transition is enabled if the push/pop behaviour is well-defined and if the relabelled label is in A.

The case for $|t| = 2$ is somewhat different: Taubner generates two transitions for each synchronisation transition, one for each possible ordering of the 'synchro-ands'. Because of the nominal relabellings distinguishing the operands of a parallel composition, one of these transitions is redundant: one of the transitions is always dead at any (safe/reachable) marking. Each of these transitions may thus be considered to hold 'half' of the transition selector of our unique transition: Taubner's scheme for his transition selectors is equivalent to the distinguishing of the partitioning in the definition of Flow_0. A corollary is that, whereas our scheme always produces only a single transition, Taubner's scheme (in the case of $n = 2$, and were it to be generalised to synchronisations of orders $n > 2$) would give 2^{n-1} transitions for a synchronisation of order n, at least all but one of which are dead at any reachable marking.

3.8 Summary

In this chapter we have related the semantic domain of High Level Petri Boxes to the well-known high level Petri net model PrT nets. With this relation we have been able to ascribe to High Level Petri Boxes, through the PrT net annotation and its subsequent unwinding, a symbolic firing rule which derives its properties directly from the H-synchrony rule introduced in the previous chapter. Indeed we will see, in Chapter 5, that for the syntactically generated High Level Petri Boxes many of their behavioural properties follow from those of the H-synchrony rule.

As the behaviour of a High Level Petri Box is defined through that of its PrT net denotation and, hence, through the unfolding of that denotation into a P/T net, we will often be required to consider these more concrete models in the proofs of behavioural properties in the sequel.

We have, in giving a PrT net denotation, shown that the structure of the *class* of High Level Petri Boxes is well-defined. We next identify a particular property rich subclass of High Level Petri Boxes which are those forming the semantic domain of our High Level Petri Box Algebra.

Chapter 4

Algebra and Semantics I

In this chapter we introduce the initial portion of the High Level Petri Box Algebra, i.e., constants and operators for the expression of causal dependency, conflict, concurrency and relabelling, and their expression on the semantic domain of High Level Petri Boxes.

As the algebraic characterisation of processes is well developed we have, in general, followed the literature to guide the development of our algebra: for instance, we provide operators for the expression of the important process concepts of causal dependence and independence, conflict, synchronisation and recursion[1].

However, process algebras appear not to have reached the point in their development at which all *non-essential characteristics* of their model based development[2] have been lost: in particular, the (traditional) trace denotation of processes brings with it the operationally appealing expression of causal dependency as the countable infinity of *prefix operators*, one for each atomic action α, with signature $\alpha\_\_$: *Process* $\rightarrow$ *Process*. As prefix is not just a 'Curried' application of some single binary 'causal' operator on processes (which would, we feel, be a more traditional algebraic expression) we might see it as emphasising the essential difference in *type* between an *atomic action* and a *process*, which is that between the items in a trace and the trace itself.

4.1 Termination and Deadlock

The action bias within interleaving models of concurrency allows the notion of state to be almost entirely abstracted away. This has benefits for the tractability of, for instance, trace models: it

[1] See Chapter 6 for the definition of the recursive construct, as well as that of refinement.
[2] By which we mean, essentially, the interleaving model based development.

is detrimental to the expression of concepts naturally expressed in terms of state.

The expression of termination in CCS, for instance, is in terms of the constant *nil*. *nil* is not an action; rather, it forms a terminal symbol of the CCS syntax and has as semantics the non-deterministic choice between zero processes, a construct which is always deadlocked. The model of termination using *nil* is then the deadlocking of a process on reaching its terminating *nil* and is, thus, indistinguishable from other deadlock situations. With the expression of causal dependence through prefix this is unremarkable as there can be no extension of a process after termination.

However, this 'confusion' is unfortunate for a *compositional* expression of the causal dependence of one process on another. Milner's derivation of sequential composition from prefix [Mil89] is rather complex. It involves being able to distinguish the deadlock before *nil* from other deadlocks. This is done by prefixing all instances of *nil* by the distinguished action $\surd$, termination then being a process which has performed the action $\surd$ and is deadlocked. To derive sequential composition, two other distinguished actions, $\surd_1$ and $\surd_2$, are required; the sequential composition of terms P and Q (each with $\surd$-prefixed *nil*s) being[3] $P;Q = (P[\surd_1/\surd] \mid \overline{\surd_1}.Q) - \{\surd_1\}$. Causal dependence is, then, given through the synchronisation of the two operand processes on the silent action derived from the renamed $\surd$.

Although this is sufficient when process P terminates sequentially, it is insufficient when distributed termination is involved as there may then be many terminating $\surd$s, one from each concurrent process. In resolving this, Milner [Mil85] defines a new parallel composition operator which forces (the equivalent of) a sequential termination from a distributed one and is defined, again using the notation of CCS, as:

$$P \mid_{\surd} Q = (P[\surd_1/\surd] \mid Q[\surd_2/\surd]) \mid (\overline{\surd_1}.\overline{\surd_2}.\surd.nil + \overline{\surd_2}.\overline{\surd_1}.\surd.nil) - \{\surd_1, \surd_2\}$$

which, in effect, produces a single $\surd$ from those of the individual processes.

4.1.1 Sequential Composition

In Petri net theory an action bias is missing and we are able to refer to states explicitly: (although this is not done in many net models) in the case of a High Level Petri Box we identify an explicit exit interface whose purpose is to indicate the terminal state of a process. For the expression of causal dependence of a *causally superior* High Level Petri Box over a *causally inferior* High

[3] Using the symbols of CCS: | being CCS parallel composition, α._ prefix by α, _ − _ hiding, and [_/_] renaming

Level Petri Box we force the provision of an initial state for the causally inferior High Level Petri Box to be dependent on the terminal state of the causally superior High Level Petri Boxes.

An explicitly deadlocked process (the High Level Petri Box equivalent of *nil*) still has its uses: as in the low level Petri Box Calculus we will define **stop** to be the already deadlocked process; the essential detail of its semantics, as we shall see, is that it is not able to reach its terminal marking. Its semantics is the simplest possible (given the structural restrictions on High Level Petri Boxes): a transition labelled with the dead label, **0**, present in every label algebra.

4.1.2 The Terminal State of Parallel and Choice Compositions

Within the distributed transition system of Section 2.2, the behaviour of the asynchronous parallel composition of two processes is modelled by the 'disjoint union' of the behaviour of the operand processes. The corresponding operator for High Level Petri Boxes is also disjoint union of its operands, although we compose structure rather than behaviour. One of the main requirements of the asynchrony rule in the distributed transition system is that it should indicate which actions are able to synchronise. This is also a requirement of our concurrent composition operator defined below; the method of distinguishing the operands of the composition is similar, as will be seen.

Because of the apparent acceptance of the 'prefix approach' to process algebras and the confusion of terminal state and deadlock, there appears no agreement on the provision of the final state to parallel compositions in the literature: for Olderog [Old91] the final state of a parallel composition is the disjoint union of those of its operands, which are, thus, held separate; for Taubner [Tau89] (motivated by Goltz [Gol88]) the final state of the operand processes are identified. These two possibilities are illustrated in Figure 4.1.

While such properties are incidental in combination with prefix, the issue as to which is 'correct' for High Level Petri Box is forced by our inclusion of sequential composition. This issue is simple to resolve: for sequential composition with, as causally superior process, a parallel composition, Olderog's expression (distributed termination) is preferable as we do not generate 'unsafe' markings through it.

However, for the expression of conflict, Olderog's scheme does not, in general, determine whether its terminal state should be distributed or not. Again, with sequential composition, it becomes an issue. For us, that the terminal state of a choice should not be a 'distributed termination' of the operands is clear, given that choice does not introduce concurrency. We will thus identify ter-

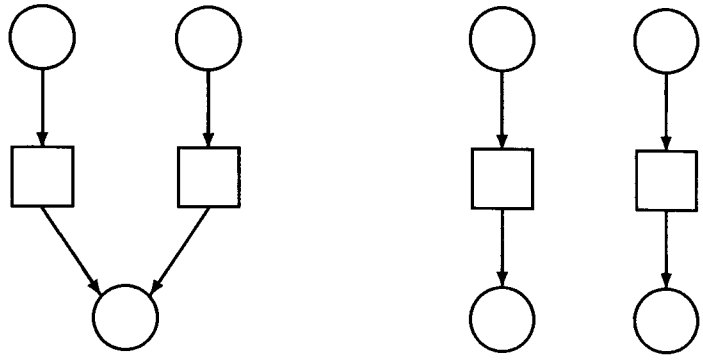


Figure 4.1: The Parallel compositions of (a.) Taubner and (b.) Olderog.

minal states. This gives a pleasing (and useful) structural symmetry for the choice composition operator.

As an aside, it is worth noting that in Milner's derivation of sequential composition from prefix with, as the causally superior process, a choice composition the identification of terminal states is achieved: it is a corollary of the replication of actions which occurs through the parallel composition of CCS.

4.2 The Pragmatic Choice is Sequence over Prefix

That prefix is an inadequate expression of causal dependence is certainly not proven by the difficulties described above. In fact, we see the beauty and simplicity of Milner's characterisation of communicating systems as those with no right distributive law of prefix over choice as powerful arguments in favour of prefix; the same statement in terms of sequential composition and choice is, although possible, certainly not as clean. Moreover, the derivation of sequential composition from prefix is complex, leading us to believe that sequential composition is an inherently more complex operation. However, pragmatically, sequential composition is the construct of choice[4] for, for instance, the expression of local scope and concepts derived from it, i.e., modularity, scalability, manageability, *etc.* It is for this reason that we have included it here.

[4]If you will forgive the unfortunate choice (or sequence) of words.

4.3 Algebra, Semantics, Compositionality and Modularity

An *algebra* consists of a syntax for forming terms. A *process algebra* is an algebra in which there is a sort corresponding to *process*. This is, traditionally, the only sort.

The major requirement for the High Level Petri Box Algebra was that we should be able to provide a *compositional semantics*. By *semantics*[5] we mean a functional relationship between two collections of (mathematical) objects (the domain will be the High Level Petri Box terms, the range High Level Petri Boxes).

DEFINITION 4.3.1 *[Semantics]* Let $\mathcal{C}$ be the collection of terms of the language generated by a syntax $\mathcal{T}$. Given a collection of objects forming a *semantic domain*, $\mathcal{D}$, a *$\mathcal{D}$-semantics* for the language $\mathcal{C}$ is a mapping $sem: \mathcal{C} \rightarrow \mathcal{D}$. *sem* will be called the *semantic function.* ■ 4.3.1

A *compositional semantics* is a semantics in which the semantic function is a homomorphism between syntactic and semantic domains:

DEFINITION 4.3.2 *[Compositional Semantics]* A $\mathcal{D}$-semantics is *compositional* if the semantic function, *sem*, extends to a homomorphism between syntactic and semantic domains, i.e., for each operation symbol of the syntax *op* of arity n there is an operation on $\mathcal{D}$, $\widehat{op}$ on the semantic domain, also of arity n, such that for terms $t_1, \ldots, t_n$ the *homomorphism condition* holds:

$$sem(op(t_1, \ldots, t_n)) = \widehat{op}(sem(t_1), \ldots, sem(t_n)).$$

■ 4.3.2

By reading the homomorphism condition as a definition rather than a constraint, and given interpretations of the constants of a syntax together with interpretations of the operators, we may often be able to recursively define a compositional semantics for a syntax:

DEFINITION 4.3.3 Let $C = \{c_1 \ldots, c_m\}$ be the collection of the constants of a syntax $\mathcal{T}$, and $O = \{o_1, \ldots, o_n\}$ be the collection of the operators, where o_i has arity h_i. Given a semantic domain $\mathcal{D}$, interpretations of the constants, $\{d_1, \ldots, d_m\}$, and interpretations of the operators, $\{p_1, \ldots, p_n\}$, we may define a compositional $\mathcal{D}$-semantics of $\mathcal{T}$, *cs*, where for each constant c_i, $cs(c_i) = d_i$, and for a term $t = o_i(t_1, \ldots, t_{h_i})$, $cs(o_i(t_1, \ldots, t_{h_i})) = p_i(cs(t_1), \ldots, cs(t_{h_i}))$.

■ 4.3.3

[5] We would like to use the term *denotational semantics* in the sense that we provide an element from the semantic domain as the denotation of a term. However, denotational in its traditional sense also implies that fix-point techniques are used for the description of recursion, and this is not a property of our semantics.

cs will then satisfy the conditions to be considered as a compositional denotational semantics.

4.3.1 Modularity

Compositionality is an absolute property, depending only on the properties of the semantic function. Modularity, when applied to a semantics, appears to mean 'serving to hide structure', which may be true to a greater or lesser extent.

We claim, for the High Level Petri Box, high modularity of both structure and behaviour. Our justification for this is twofold:

1. (at the structural level) that knowledge of much of the 'implementation' of a process through a High Level Petri Box is not needed to be able to perform the compositions on pairs of High Level Petri Boxes,

2. (at the behavioural level) that, given two simple conditions, High Level Petri Boxes with the same behaviour compose in the same way, and are relatively independent of the structure which produced the behaviour. (This will be shown in Chapter 7.)

4.4 Notation and Conventions

Before beginning the description of the semantic domain of the High Level Petri Box Algebra, it will be useful to review the relationship between functions and relations.

4.4.1 Functions and Relations

By regarding the Boolean algebra 2 (the set $\{false, true\}$ together with the usual *meets*, *joins* and *inverse*) as a set, and given sets A and B, there is a well-known bijection between $A \to \mathbb{P}B$ and $A \times B \to 2$. As we may interpret $A \times B \to 2$ as the collection of binary relations between A and B, $A \leftrightarrow B$, we thus have a bijection between $A \to \mathbb{P}B$ and $A \leftarrow B$, the details of the correspondence being that the function $f: A \to \mathbb{P}B$ maps to the relation $F: A \leftarrow B$ where $(a, \{b_1, \ldots, b_{n_a}\}) \in f$ if and only if for all $j \in \{1, \ldots, n_a\}$, $(a, b_j) \in F$.

Clearly, a function from A to $\mathbb{P}B$ will not, in general, remain a function when interpreted under the bijection as a relation between A and B. However, we use the bijection to justify the free interchange of *function from A to* $\mathbb{P}B$ and *relation between A and B* in the sequel, using the symbols $f, g, \ldots$ for each.

Other observations will also be useful: even though in general $\mathbf{P}A \cup \mathbf{P}B \neq \mathbf{P}(A \cup B)$ we have that $\mathbf{P}A \cup \mathbf{P}B \subseteq \mathbf{P}(A \cup B)$; with the natural injection from set A into $\mathbf{P}A$ (in which elements map to singleton subsets), we may interpret the set of partial functions $A \nrightarrow B$ as a subset of $A \nrightarrow \mathbf{P}B$, and hence also as a relation. As usual, for $f: A \to \mathbb{P}B$ a function, we will extend $f: \mathbf{P}A \nrightarrow \mathbf{P}B$ (and to $\mathcal{M}_{\mathcal{F}}(A) \to \mathcal{M}_{\mathcal{F}}(A)$) by interpreting $f(S) = \bigcup_{s \in S} f(s)$, for $S \subseteq A$.

4.4.2 Sane and Separable Relations

We next introduce *sane* and *separable* relations. That a relation is sane will ensure that it can be *lifted* to a High Level Petri Box. It is a property of a class of relations that includes those used in the definitions of causal and choice compositions, later in this chapter, and that of refinement in Chapter 6.

DEFINITION 4.4.1 A relation $f: A \to \mathbb{P}B$ is *sane* if $\forall s, s' \in A: s \neq s' \Leftrightarrow f(s) \cap f(s') = \emptyset$.

■ 4.4.1

We note that sanity is, in particular, stronger than injectivity which would require only that $s \neq s' \Rightarrow f(s) \neq f(s')$.

A consequence of the reverse implication is that $f(s) \neq \emptyset$, for all $s \in \mathrm{dom}(f)$.

That a relation is separable will ensure that it may be used for place multiplication (in the sense given by Proposition 4.4.3).

DEFINITION 4.4.2 A relation $f: S \to \mathbb{P}S \cup \mathbb{P}(S \otimes S')$ is *separable (on S and S')* if there is a function $P_f: S \to \mathbb{P}S'$ such that

$$f(s) = \begin{cases} \{s\} & P_f(s) = \emptyset \\ \{s\} \otimes P_f(s) & \text{otherwise} \end{cases}$$

■ 4.4.2

The relationship between sociability, sanity and separability is partially characterised by the following:

PROPOSITION 4.4.3 Let $S, S' \subseteq \mathcal{P}$ be sociable sets of places. Then f a separable relation on S and S' implies f is sane. □ 4.4.3

Proof: For f to be sane we must show that $s \neq s' \Leftrightarrow f(s) \cap f(s') = \emptyset$.

$\Rightarrow$: Suppose not, i.e., $\exists s \neq s' \in S: f(s) \cap f(s') \neq \emptyset$. Let $s'' \in f(s) \cap f(s')$. Then

$$s'' = \begin{cases} s & P_f(s) = \emptyset \quad (1) \\ s \otimes p & \text{some } p \in P_f(s) \quad (2) \end{cases}$$

$$= \begin{cases} s' & P_f(s') = \emptyset \quad (3) \\ s' \otimes p' & \text{some } p' \in P_f(s') \quad (4) \end{cases}$$

The proof follows by comparison of cases:

1&3 imply $s = s'$, contradicting the assumption that $s \neq s'$.

1&4 imply $s = s' \otimes p'$ so that $p' \in \mathrm{id}(S) \cap \mathrm{id}(S')$, contradicting the sociability of S and S'.

2&3 is similar to the previous case.

2&4 imply $s \otimes p = s' \otimes p'$. As S and S' are sociable it follows from Proposition 2.5.3 (page 26) that $s = s'$ and $p = p'$, contradicting the assumption that $s \neq s'$.

$\Leftarrow$: Clear, as for all s, $f(s) \neq \emptyset$. ■ 4.4.3

Sanity is preserved through composition:

PROPOSITION 4.4.4 f and g sane implies $f \circ g$ sane. □ 4.4.4

Proof: Suppose not, i.e., $\exists t \in f \circ g(s) \cap f \circ g(s')$ with $s \neq s'$. Then, for such a t, $\exists t_1 \in g(s)$. $t_2 \in g(s')$ such that $t \in f(t_1) \cap f(t_2)$, and hence $t_1 = t_2$, as f is sane, so that $t_1 \in g(s) \cap g(s')$. But then $s = s'$ from the assumption on g, a contradiction. ■ 4.4.4

Given a High Level Petri Box B and $S' \subseteq \mathcal{P}$ sociable with S_B, a sane relation $f: S_B \multimap S'$ will be called a *sane relation on* B; a separable relation $f: S_B \multimap \mathbb{P}S_B \cup \mathbb{P}(S_B \otimes S')$ will be called *separable relation on* B.

EXAMPLE 4.4.5 Given sociable High Level Petri Boxes B and B' such that $S_B = \{s_0, s_1\}$, $S_{B'} = \{s_2, s_3, s_4, s_5\}$, then relation f with $P_f(s_0) = \emptyset$, $P_f(s_1) = \{s_2, s_3\}$ (so that $f(s_0) = \{s_0\}$. $f(s_1) = \{s_1 \otimes s_2, s_1 \otimes s_3\}$) is sane and separable. ■ 4.4.5

(In Example 4.4.5 we have, surreptitiously, specified the *place multiplication* of the exit places of B (when $B^\bullet = \{s_1\}$) with the entry places of B' (when ${}^\bullet B' = \{s_2, s_3\}$). Separable relations are sufficient to describe place multiplications. Moreover, as long as they are defined on sociable High Level Petri Boxes, then the relational descriptions of place multiplications they give rise to

will be sane. Although there are many ways of expressing causal dependence, possibly the most conceptually attractive scheme for Petri nets and the one adopted in this exposition, is that of place multiplication for the expression of the structural dependence. Place multiplication was first suggested as a method for composing nets in [GM84]. It has the advantage of being relatively straightforward, entails little technical overhead to express and, unlike the CCS derivation of sequential composition, introduces no silent transition between the processes.

4.5 Auxiliary Operators

The observation is not new ([Kot78, Win84, BDH92], for example) that there are 'basic' operations on nets from which may be derived net implementations of traditional process algebra operations. In this section we identify a number of *auxiliary* operations on High Level Petri Boxes from which we derive the semantics of the operators of the High Level Petri Box Algebra. These correspond to component-wise manipulations of their operand High Level Petri Boxes.

4.5.1 High Level Petri Box Union

The first auxiliary operator introduced is that of High Level Petri Box union (shortly union) which forms a single High Level Petri Box from its two operands. This it achieves by identifying parts of their control interfaces. Other than their control interfaces, we will assume that the operands to a union are disjoint. Moreover, we will assume that the operands share the same label algebra (which is promoted to be that of the union). Whereas these assumptions make the definition of High Level Petri Box union partial, they are sufficiently permissive to cover all cases that arise through the semantics.

DEFINITION 4.5.1 *[High Level Petri Box Union]* Let B_1 and B_2 High Level Petri Boxes such that:

1. if $\mathcal{A} \in \{{}^\bullet B_1, B_1{}^\bullet\}$, $\mathcal{B} \in \{{}^\bullet B_2, B_2{}^\bullet\}$, then $\mathcal{A} \cap \mathcal{B} \neq \emptyset$ implies $\mathcal{A} = \mathcal{B}$,
2. $\overset{\bullet}{B}_1 \cap S_2 = \emptyset$ and $\overset{\bullet}{B}_2 \cap S_1 = \emptyset$,
3. ${}^\bullet B_1 \neq B_2{}^\bullet$ or ${}^\bullet B_2 \neq B_1{}^\bullet$.

Define the *High Level Petri Box union* of B_1 and B_2 as

$$B_1 \cup B_2 = \langle S_1 \cup S_2, T_1 \cup T_2, \lambda \rangle$$

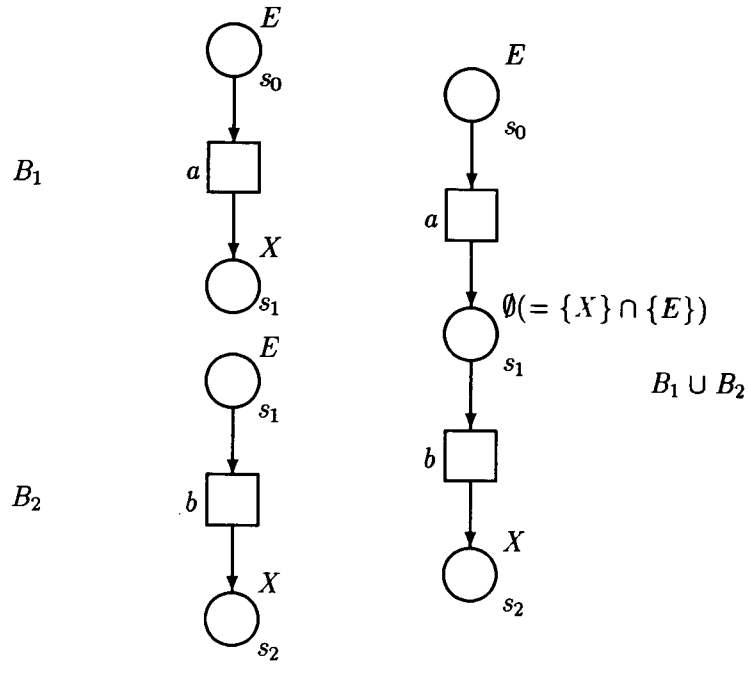


Figure 4.2: An example of the High Level Petri Box union of two High Level Petri Boxes. s_1 is a place shared by both B_1 and B_2. It is internalised in their union.

where

$$\lambda(s) = \begin{cases} \lambda_1(s) \cap \lambda_2(s) & \text{if } s \in S_1 \cap S_2 \\ \lambda_1(s) & \text{if } s \in S_1 \setminus S_2 \\ \lambda_2(s) & \text{if } s \in S_2 \setminus S_1. \end{cases}$$

■ 4.5.1

Pairs of High Level Petri Boxes satisfying Conditions 1, 2 and 3 of Definition 4.5.1 will be termed *unionable.*

Figure 4.2 illustrates the definition of High Level Petri Box union.

Conditions 1-3 ensure the following properties of the operand High Level Petri Boxes of a union and will be essential in the derivation of its properties:

a that their respective entry and exit interfaces should be either disjoint or exactly coincide. On coincidence of respective entry and exit places, their union should be internalised, and otherwise left as interface places. The form of place labels and the definition of the new place labelling provides an agreeable way of achieving this (as is illustrated in Figure 4.2).

b that they should have no internal places in common, and

c that the composed net should retain both an entry and exit interface.

We have the following:

PROPOSITION 4.5.2 For B_1 and B_2 unionable High Level Petri Boxes, $B_1 \cup B_2$ is a High Level Petri Box. □ 4.5.2

Proof: The proof depends on the following easily checkable lemma which allows us to relate the interface of a High Level Petri Box union to those of its operands:

LEMMA 4.5.3 For A, B, C, D sets such that $A \cap C \cap D = \emptyset = D \cap A \cap B$, then

$$((A \setminus B) \setminus (C \setminus D)) \cup ((D \setminus C) \setminus (B \setminus A)) = (A \cup D) \setminus (B \cup C).$$

□ 4.5.3

That $B_1 \cup B_2$ is a pre-High Level Petri Box is clear from the definition. We are required to show that $B_1 \cup B_2$ has Properties 1-3 of Definition 2.5.15. We will find it convenient to demonstrate Property 2 first:

Property 2 We show that ${}^\bullet(B_1 \cup B_2) = {}^\bullet LT(B_1 \cup B_2) \setminus LT(B_1 \cup B_2)^\bullet$, the result for $(B_1 \cup B_2)^\bullet$ following by symmetry. Note that $S_{B_1 \cup B_2} = S_1 \cup S_2$.

$$\begin{aligned}
&{}^\bullet(B_1 \cup B_2) \\
&= \{s \in S_1 \cup S_2 \mid E \in \lambda_{B_1 \cup B_2}(s)\} \\
&= ({}^\bullet B_1 \setminus S_2) \cup ({}^\bullet B_2 \setminus S_1) \cup ({}^\bullet B_1 \cap {}^\bullet B_2) \\
&= [\overset{\bullet}{B_2} \cap S_1 = \emptyset = \overset{\bullet}{B_1} \cap S_2] \\
&\quad {}^\bullet B_1 \setminus ({}^\bullet B_2 \cup B_2{}^\bullet) \cup {}^\bullet B_2 \setminus ({}^\bullet B_1 \cup B_1{}^\bullet) \cup ({}^\bullet B_1 \cap {}^\bullet B_2) \\
&= [\text{Lemma 2.5.18}] \\
&\quad {}^\bullet B_1 \setminus B_2{}^\bullet \cup {}^\bullet B_2 \setminus B_1{}^\bullet \\
&= [\text{Lemma 4.5.3}] \\
&\quad ({}^\bullet LT(B_1) \cup {}^\bullet LT(B_2)) \setminus (LT(B_1)^\bullet \cup LT(B_2)^\bullet) \\
&= {}^\bullet LT(B_1 \cup B_2) \setminus LT(B_1 \cup B_2)^\bullet
\end{aligned}$$

Property 1 we show that ${}^\bullet(B_1 \cup B_2) \neq \emptyset$. The result for $(B_1 \cup B_2)^\bullet$ following by symmetry. But from the proof of Property 2 we see that ${}^\bullet(B_1 \cup B_2) = {}^\bullet B_1 \setminus B_2{}^\bullet \cup {}^\bullet B_2 \setminus B_1{}^\bullet$ which is non-empty, as B_1 and B_2 satisfy Property 3 of Definition 4.5.1 by assumption.

Property 3 we will show that: $\forall l \in LT(B_1 \cup B_2)$: ${}^\bullet l \cap {}^\bullet(B_1 \cup B_2) = \emptyset \vee {}^\bullet l \subseteq {}^\bullet(B_1 \cup B_2)$.

the case for the post-set following by symmetry.

From the definition, $LT(B_1 \cup B_2) = LT(B_1) \cup LT(B_2)$ so that $l \in LT(B_1 \cup B_2) \Rightarrow l \in LT(B_1) \cup LT(B_2)$. Assume without loss of generality that $l \in LT(B_1)$. Suppose that Property 2 does not hold of the union, i.e., that ${}^\bullet l \cap {}^\bullet(B_1 \cup B_2) \neq \emptyset$ and ${}^\bullet l \not\subseteq {}^\bullet(B_1 \cup B_2)$. Let $s \in {}^\bullet l \cap {}^\bullet(B_1 \cup B_2)$, $s' \in {}^\bullet l \setminus {}^\bullet(B_1 \cup B_2)$. Then

$$
\begin{aligned}
& s \in {}^\bullet l \cap {}^\bullet(B_1 \cup B_2) \\
& \quad \Rightarrow \quad E \in \lambda_{B_1 \cup B_2}(s) \\
& \quad \Rightarrow \quad [\lambda_{B_1 \cup B_2}(s) \subseteq \lambda_1(s) \text{ as } s \in {}^\bullet l \subseteq S_1] \\
& \qquad\quad E \in \lambda_1(s) \\
& \quad \Rightarrow \quad s \in {}^\bullet B_1.
\end{aligned}
$$

Also

$$
\begin{aligned}
& s \in S_{B_2} \\
& \quad \Rightarrow \quad [\lambda_{B_1 \cup B_2}(s) \subseteq \lambda_{B_2}(s) \text{ as } s \in S_{B_2}] \\
& \qquad\quad s \in {}^\bullet B_2
\end{aligned}
$$

Assume $s' \notin {}^\bullet B_1$. Then ${}^\bullet l \cap {}^\bullet B_1 \neq \emptyset$ and ${}^\bullet l \not\subseteq {}^\bullet B_1$ contradicting Property 3 of Definition 4.5.1 for B_1. Hence $s' \in {}^\bullet B_1$ so that as $s' \in {}^\bullet l \setminus {}^\bullet(B_1 \cup B_2)$, by Property 2 of Definition 2.5.15 $s' \in B_2{}^\bullet$.

Hence we may assume that $s' \in {}^\bullet B_1 \cap B_2{}^\bullet \wedge s \in {}^\bullet B_1 \wedge (s \in S_{B_2} \Rightarrow s \in {}^\bullet B_2)$. Distinguishing the cases when $s \in S_{B_2}$ and $s \notin S_{B_2}$:

$s \in S_{B_2}$: whence:

$$
\begin{aligned}
& s, s' \in {}^\bullet B_1 \wedge s' \in B_2{}^\bullet \wedge s \in {}^\bullet B_2 \\
& \quad \Rightarrow \quad {}^\bullet B_1 \cap B_2{}^\bullet \neq \emptyset \wedge {}^\bullet B_1 \cap {}^\bullet B_2 \neq \emptyset \\
& \quad \Rightarrow \quad {}^\bullet B_1 = {}^\bullet B_2 \wedge {}^\bullet B_1 = B_2{}^\bullet \\
& \quad \Rightarrow \quad \{{}^\bullet B_1 \neq \emptyset\} \\
& \qquad\quad {}^\bullet B_2 = B_2{}^\bullet \neq \emptyset
\end{aligned}
$$

contradicting Lemma 2.5.18.

$s \notin S_{B_2}$: whence $s, s' \in {}^\bullet B_1$ and $s' \in B_2{}^\bullet$ and $s \notin S_{B_2}$ implies ${}^\bullet B_1 \cap B_2{}^\bullet \neq \emptyset$ and ${}^\bullet B_1 \neq B_2{}^\bullet$ contradicting Property 1 of Definition 4.5.1.

Hence the result. ■ 4.5.2

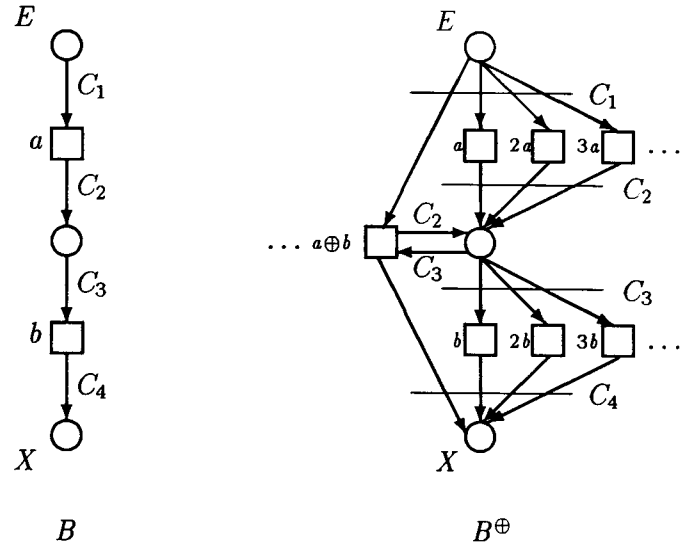


Figure 4.3: An example of the combinational closure of a High Level Petri Box.

4.5.2 Combinational Closure

The H-synchrony rule relaxes the restriction on the formation of synchronisations that they be necessarily binary in nature. The combinational closure of a High Level Petri Box has the same places, place labelling and local transitions as its operand, but has as its set of abstract transitions all combinations of those of its operand, i.e., one corresponding to each possible multi-way synchronisation. As there are a countable infinity of such combinations we will then have a countable infinity of abstract transitions in the High Level Petri Box to which combinational closure has been applied. However, we note that the cardinality of the collection of local transitions underlying the High Level Petri Box does not alter.

DEFINITION 4.5.4 *[Combinational Closure]* Given a High Level Petri Box B, define the *combinational closure* of B, $B^{\oplus} = \langle S, T^{\oplus}, \lambda \rangle$[6]. ■ 4.5.4

Note:

1. as there is no restriction on the multisets formed it is possible that (directly or indirectly) causally independent local transitions are part of the same abstract transition. Figure 4.3 illustrates this situation. The 'cardinality preserving' semantics of recursion, developed in Chapter 6, requires this possibility, as through it, two disjoint processes may share the same net structure.

[6]Where, for T a set, $T^{\oplus} = \{t_1 \cup \ldots \cup t_n \mid t_i \in T \wedge n \geq 1\}$ with $\cup$ being multiset union.

2. diagrammatically, combinational closure introduces four problems in the representation of High Level Petri Boxes:

   (a) as we have no natural ordering on the members of a multiset with which to form the correspondence between local and abstract transitions in diagrams we will use the following scheme for describing this relationship. A multiset appearing in a diagram will be written as a (formal) sum in which summands of an additive partitioning of the multiplicity of a member are written prefixed (or omitted when unity) to copies of the label: for instance, the multiset $\{\!|a, a, a, b|\!\}$ may be written as $3a \oplus b$, $2a \oplus b \oplus a$, $b \oplus a \oplus 2a$, *etc.* The particular partitioning of the multiplicity of a member may then be related to the ordering of the local transitions whose labels contribute to that multiset: for instance, for local transitions k_0, k_1 and k_2 with labels a, b and a respectively, we might write the multiset of labels corresponding to two appearances of k_0, one of k_1 and one of k_2, in that order, as $2a \oplus b \oplus a$.

   (b) related is the problem of arc annotations: for local transitions repeated in the same abstract transition (such as that labelled $2a$ in the figure) we will draw a single arc and, singly, the context annotating it. This will allow us to clarify the figures as follows:

   (c) for clarity in the figures, a line (———) through any number of arcs together with a context indicate that that context forms the annotation of each of those arcs.

   (d) the representation of an infinite number of abstract transitions causes problems. However, because of the regular nature of the construction, the use of ellipsis (...) will generally be sufficient.

We have the following:

PROPOSITION 4.5.5 Let B be a High Level Petri Box. Then $B^{\oplus}$ is a High Level Petri Box. □ 4.5.5

Proof: That $B^{\oplus}$ is a pre-High Level Petri Box is clear from the definition, the only non-trivial part being that $T^{\oplus} \subseteq \mathcal{T}$, which follows from $t \in T^{\oplus}$ implying $\exists r_1^t, \ldots, r_{n_t}^t \in T$ not necessarily distinct, such that $t = r_1^t \cup \ldots \cup r_{n_t}^t$. But then $t \in \mathcal{M}_{\mathcal{F}}(loc)$.

We also note that it follows from this that $LT(B^{\oplus}) = LT(B)$.

Properties 1-3 of Definition 2.5.15 for $B^{\oplus}$ follow from the corresponding properties for B.

Hence the result. ■ 4.5.5

An important structural property of combinational closure is the following:

PROPOSITION 4.5.6 For High Level Petri Boxes B_1 and B_2, $B_1{}^{\oplus} = B_2{}^{\oplus}$ implies that there exists a High Level Petri Box C such that $C \subseteq B_1, B_2 \subseteq C^{\oplus}$. □ 4.5.6

Proof: Suppose $B_i = \langle S_i, T_i, \lambda_i \rangle$. As $B_1{}^{\oplus} = B_2{}^{\oplus}$ we know that $S_1 = S_2$, $\lambda_1 = \lambda_2$ and $LT(B_1) = LT(B_2)$. Define $C = \langle S_1, T', \lambda_1 \rangle$ such that $T' = \{\{\!|l|\!\} \mid l \in LT(B_1)\}$. C clearly has the desired properties. ■ 4.5.6

4.5.3 Context Manipulation

Context manipulation appends its context operands onto the annotating contexts of the input and output arcs of its operand. It corresponds (in all uses except for that in refinement) to the definition of the scope of a relabelling.

DEFINITION 4.5.7 *[Context Manipulation]* Let B be a High Level Petri Box and $C, C' \in \mathcal{C}$. Define the *context manipulation of* B, $B \odot (C, C') = \langle S, T', \lambda \rangle$ where $T' = \{t(C, C')_B | t \in T\}$ and

$$\begin{aligned} t(C, C')_B &= \{l(C, C')_B \mid l \in t\} \\ l(C, C')_B &= \langle {}^\bullet l, {}^C l \frown D, \overline{l}, l^C \frown D', l^\bullet \rangle \end{aligned}$$

and

$$D = \begin{cases} C & {}^\bullet l \subseteq {}^\bullet B \\ (.) & \text{otherwise} \end{cases}$$

$$D' = \begin{cases} C' & l^\bullet \subseteq B^\bullet \\ (.) & \text{otherwise} \end{cases}$$

■ 4.5.7

Figure 4.4 illustrates the definition of context manipulation.

Given the many uses of context manipulation in the sequel, we will often write $B : C$ for $B \odot (C, C)$, $t(C)_B$ for $t(C, C)_B$, $l(C)_B$ for $l(C, C)_B$ and, when l is a local transition, $t(l)_B$ for $t({}^C l, l^C)_B$ and $B(l)$ for $B \odot ({}^C l, l^C)$.

From the properties of B, the following is immediate

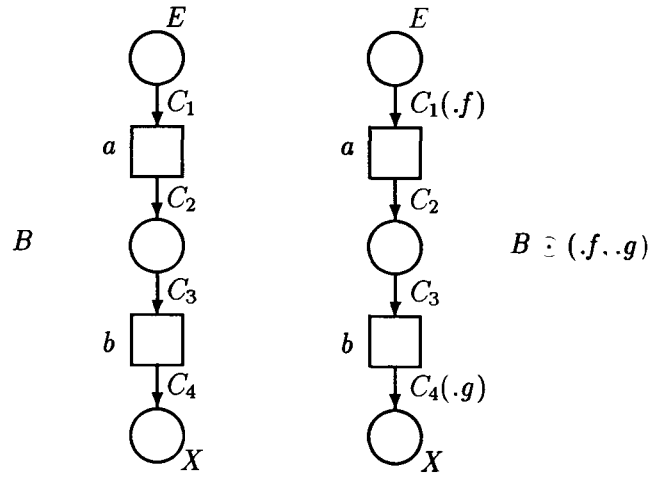


Figure 4.4: The context manipulation $B \textcircled{?} (.f, .g)$.

PROPOSITION 4.5.8 Let B be a High Level Petri Box. Then for C, $C' \in \mathcal{C}$, $B \textcircled{?} (C, C')$ is a High Level Petri Box. □ 4.5.8

4.5.4 Relational Lifting

Relational lifting extends the effect of a relation from the places of its operand High Level Petri Box to the structure of that High Level Petri Box.

DEFINITION 4.5.9 *[Relational Lifting]* For B a High Level Petri Box, $S' \subseteq \mathcal{P}$ and $f: S_B \rightarrow S'$ a relation define the *relational lifting of B under f*, $\hat{f}(B) = \langle f(S_B), f(T_B), f(\lambda_B) \rangle$ where $f(l) = \langle f({}^\bullet l), {}^C l, \overline{l}, l^C, f(l^\bullet) \rangle$, $f(\{\!|l_1, \ldots, l_n|\!\}) = \{\!|f(l_1), \ldots, f(l_n)|\!\}$ and $f(\lambda_B)(r) = \bigcup_{r \in f(p)} \lambda_B(p)$.

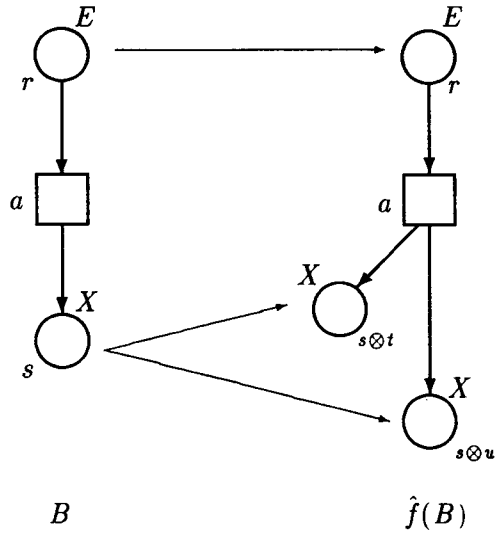
■ 4.5.9

Note:

1. if the relation f is sane the control interface of the resulting High Level Petri Box is that of the operand as, in this case, $r \in f(p) \cap f(p')$ implies $p = p'$ and $\lambda_B(p) = \lambda_B(p')$,

2. the distributed union of the definition of the label of related places ensures that the relational lifting is well-defined when the intersection of images of places are not disjoint. It will also be required for the definition of the recursive construct in Chapter 6,

3. for notational convenience we will omit the $\hat{}$ decoration, writing f for both f and its lifting.



with f defined as
$f(r) = \{r\}$
$f(s) = \{s\} \otimes \{t, u\}$
as shown.

Figure 4.5: An example of the relational lifting of the sane relation f to a High Level Petri Box.

4. in passing we note that, in the notationally sparse situation of f being applied to the empty multiset of local transitions we have that $f(\{\!|\}\!|) = \{\!|\}\!|$. This will be of use in the definition of refinement in Chapter 6.

Figure 4.5 illustrates the definition of the relational lifting of a sane relation to a High Level Petri Box.

PROPOSITION 4.5.10 For B a High Level Petri Box and f a sane relation on B, then $f(B)$ is a High Level Petri Box. □ 4.5.10

The proof depends on the following lemma which allows us to relate the components of an High Level Petri Box and its lifting:

LEMMA 4.5.11 For B a High Level Petri Box and f a sane relation, the following hold:

1. $\forall l \in LT(B), {}^\bullet f(l) = f({}^\bullet l)$ and $f(l)^\bullet = f(l^\bullet)$,
2. $LT(f(B)) = f(LT(B))$,
3. $f({}^\bullet LT(B)) = {}^\bullet f(LT(B))$ and $f(LT(B)^\bullet) = f(LT(B))^\bullet$,
4. $f({}^\bullet LT(B) \setminus LT(B)^\bullet) = f({}^\bullet LT(B)) \setminus f(LT(B)^\bullet)$ and $f(LT(B)^\bullet \setminus {}^\bullet LT(B)) = f(LT(B)^\bullet) \setminus f({}^\bullet LT(B))$,

5. ${}^\bullet f(B) = f({}^\bullet B)$ and $f(B)^\bullet = f(B^\bullet)$,

6. ${}^\bullet f(B) = {}^\bullet LT(f(B)) \setminus LT(f(B))^\bullet$ and $f(B)^\bullet = LT(f(B))^\bullet \setminus {}^\bullet LT(f(B))$. □ 4.5.11

Proof: All follow from the sanity of f. ■ 4.5.11

Proof: That $f(B)$ is a pre-High Level Petri Box is clear from the definitions.

It remains to show that $f(B)$ has Properties 1-3 of Definition 2.5.15.

1. From Lemma 4.5.11, ${}^\bullet f(B) = f({}^\bullet B) \neq \emptyset$, as B is a High Level Petri Box and f is sane. The case for $f(B)^\bullet$ is symmetric.
2. Follows from Lemma 4.5.11.
3. Suppose $l' \in LT(f(B)) = f(LT(B))$, by Lemma 4.5.11(3), i.e., there is an $l \in LT(B)$ such that $l' = f(l)$.

   Suppose ${}^\bullet l' \cap {}^\bullet f(B) \neq \emptyset$. Then

   $$\begin{aligned} & {}^\bullet l' \cap {}^\bullet f(B) \neq \emptyset \\ \Rightarrow\ & f({}^\bullet l) \cap f({}^\bullet B) \neq \emptyset \\ \Rightarrow\ & [f \text{ is sane}] \\ & {}^\bullet l \cap {}^\bullet B \neq \emptyset \\ \Rightarrow\ & [\text{Hypothesis}] \\ & {}^\bullet l \subseteq {}^\bullet B \\ \Rightarrow\ & {}^\bullet l' = f({}^\bullet l) \subseteq f({}^\bullet B) = {}^\bullet f(B). \end{aligned}$$

Hence the result. ■ 4.5.10

4.5.5 Auxiliary Operator Precedence

For the parsing of expressions involving the auxiliary operators we define relational lifting to have highest precedence, then combinational closure, context manipulation and, with lowest precedence, union. We will also use parentheses liberally to disambiguate.

4.6 Top Level Operators

We now define the top level operators of the High Level Petri Box Algebra.

For notational convenience throughout the remainder of this section, we will assume that a particular label algebra $\mathcal{L} = \langle A, R \rangle$ has been chosen, with respect to which all label algebra

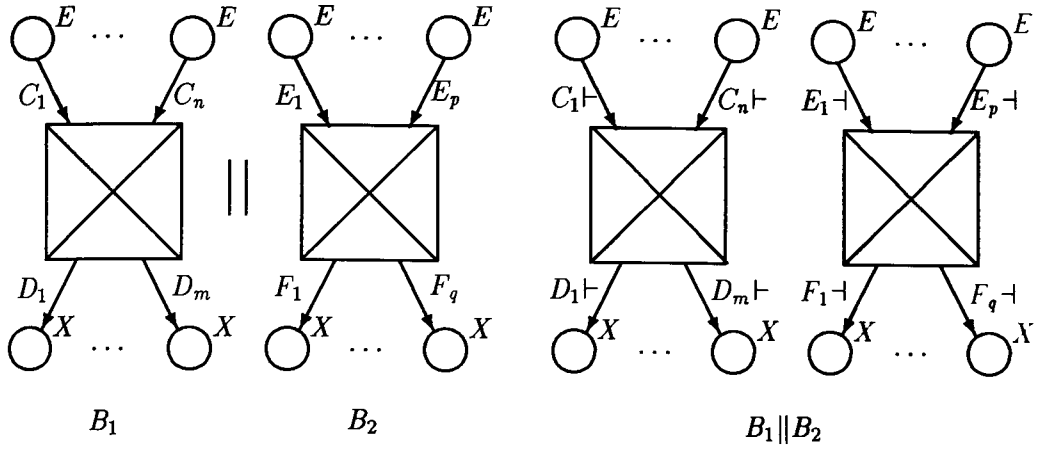


Figure 4.6: The concurrent composition of High Level Petri Boxes B_1 and B_2 in terms of the conceptual model.

relative concepts are defined. In addition, we will assume that B_1, B_2 and B are High Level Petri Boxes with B_1 and B_2 sociable.

4.6.1 Concurrent Composition

Concurrent composition is the High Level Petri Box implementation of asynchronous parallel composition. As in distributed transition systems, we achieve this through taking the 'disjoint union' of the operand High Level Petri Boxes. Disjointness is ensured by the annotation of the operand High Level Petri Boxes by the nominal relabellings $\vdash$ and $\dashv$. The interface of the composed High Level Petri Box is inherited from those of the operands.

DEFINITION 4.6.1 *[Concurrent Composition]* Define the *concurrent composition of* B_1 *and* B_2, $B_1 \| B_2 = (B_1 \odot .\vdash) \cup (B_2 \odot .\dashv)$. ■ 4.6.1

Figure 4.6 shows an example of the concurrent composition of two High Level Petri Boxes.

Although the decoration of the operands clearly destroys the structural commutativity and associativity of the concurrent composition, it does not destroy their behavioural equivalents, which follows from the symmetries present in the feasible substitutions of the PrT Denotation of a High Level Petri Box given in Chapter 2. The only complication in the argument is that the symmetries derive from the properties of the Flow predicate and not the transition selectors of the PrT semantics. The relationship between the two is obviously very close: we have described the action of the conjuncts of the transition selectors other than the Flow predicate as modelling

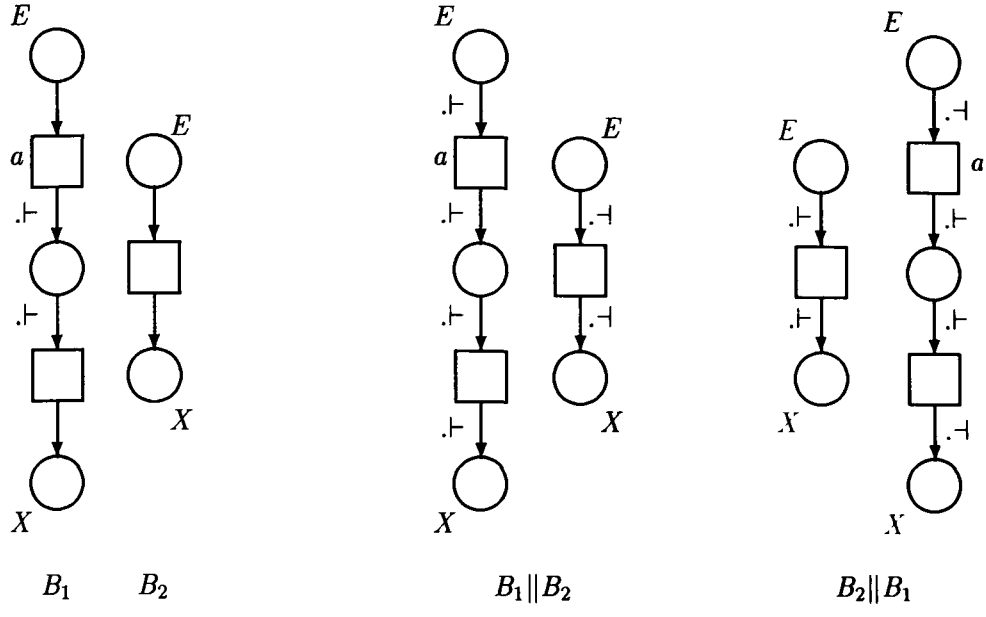


Figure 4.7: In which the behavioural commutativity of concurrent composition is compromised by poor scoping: from the standard initial marking, $B_1 \| B_2$ may fire transition labelled a, whereas $B_2 \| B_1$ may not.

the 'scope' of the contexts which annotate the input and output arcs of a transition and, as such, the symmetries apply to already 'scoped' processes.

It is possible, then, that if this scoping operation is not well-defined with respect to the structure of the High Level Petri Box it may affect the commutativity of the behaviour of the High Level Petri Box. This situation has already been discussed in Section 3.3.4 and is further illustrated in Figure 4.7. To see that this does not happen when a High Level Petri Box is syntactically generated we must wait until Chapter 5.

Similar arguments apply for behavioural associativity.

We have the following:

PROPOSITION 4.6.2 Given sociable High Level Petri Boxes B_1 and B_2, $B_1 \| B_2$ is a High Level Petri Box. □ 4.6.2

Proof: As B_1 and B_2 are sociable, Proposition 2.5.3 implies that $S_1 \cap S_2 = \emptyset$. From Definition 4.5.7, $S_{B \odot .f} = S_B$, so that $B_1 \odot .\vdash$ and $B_2 \odot .\dashv$ are also sociable and so satisfy the preconditions of Definition 4.5.1. The result then follows from Propositions 4.5.8 and 4.5.2. ■ 4.6.2

4.6.2 Causal Composition

Causal composition makes each initial transition of its second operand dependent for an enabling marking on the firing of each of the terminal transitions of its first. This is achieved through place multiplication of the exit places of the first operand with the entry places of the second, and is implemented through relational lifting in the manner of Example 4.4.5. The entry interface of the composition is that of the first operand, the exit interface is that of the second, with the multiplied places becoming internal.

DEFINITION 4.6.3 *[Causal Composition]* Define separable relations $;_1 : S_1 \nrightarrow (S_1 \cup (S_1 \otimes S_2))$ and $;_2 : S_2 \nrightarrow (S_2 \cup (S_1 \otimes S_2))$ such that

$$;_1(s) = \begin{cases} \{s\} \otimes {}^\bullet B_2, & \text{if } s \in B_1{}^\bullet \\ \{s\}, & \text{otherwise} \end{cases}$$

$$;_2(s) = \begin{cases} B_1{}^\bullet \otimes \{s\}, & \text{if } s \in {}^\bullet B_2 \\ \{s\}, & \text{otherwise} \end{cases}$$

Define the *causal composition of* B_1 *and* B_2, $B_1;B_2 = ;_1(B_1) \cup ;_2(B_2)$. ■ 4.6.3

Figure 4.8 illustrates the causal composition of two High Level Petri Boxes.

PROPOSITION 4.6.4 Given sociable High Level Petri Boxes B_1 and B_2, $B_1;B_2$ is a High Level Petri Box. □ 4.6.4

Proof: $;_1$ and $;_2$ are separable relations on sociable sets and so are sane. Also, as B_1 and B_2 are sociable then $;_1(B_1)$ and $;_2(B_2)$ are unionable. Hence the result. ■ 4.6.4

Causal composition is not, of course, commutative. However, and unlike concurrent and (to follow) choice composition, for sociable High Level Petri Boxes causal composition is *structurally* (and hence behaviourally) associative: distinguishing of the operands is achieved through place multiplication.

PROPOSITION 4.6.5 For B_1, B_2 and B_3 pairwise sociable High Level Petri Boxes, $(B_1;B_2);B_3 = B_1;(B_2;B_3)$. □ 4.6.5

Proof: That B_2 is a High Level Petri Box implies ${}^\bullet B_2 \cap B_2{}^\bullet = \emptyset$, so that the domains of definition of the auxiliary relations, $;_1$ and $;_2$ are disjoint. The result follows easily. ■ 4.6.5

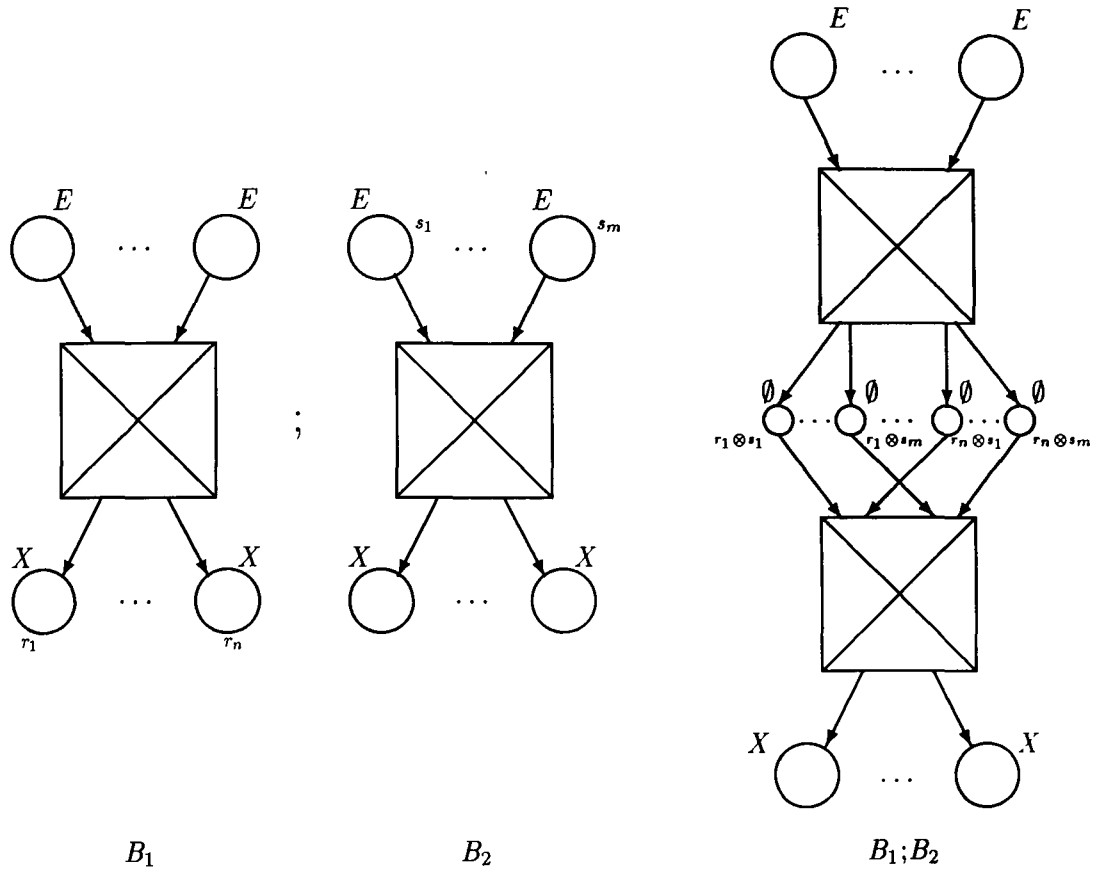


Figure 4.8: The causal composition of High Level Petri Boxes B_1 and B_2 in terms of the conceptual model.

4.6.3 Choice composition

Choice composition places each of the initial transitions of its operand High Level Petri Boxes in conflict. It also produces a local termination of the composition through the identification of their exit places. This is achieved through union and place multiplication. The entry and exit interface places are inherited, although in a place multiplied form, from the operands.

DEFINITION 4.6.6 *[Choice Composition]* Define separable relations $+_1: S_1 \rightharpoonup (S_1 \cup (S_1 \otimes S_2))$ and $+_2: S_2 \leftrightarrow (S_2 \cup (S_1 \otimes S_2))$ such that

$$+_1(s) = \begin{cases} \{s\} \otimes {}^\bullet B_2, & \text{if } s \in {}^\bullet B_1 \\ \{s\} \otimes B_2{}^\bullet, & \text{if } s \in B_1{}^\bullet \\ \{s\}, & \text{otherwise} \end{cases}$$

$$+_2(s) = \begin{cases} {}^\bullet B_1 \otimes \{s\}, & \text{if } s \in {}^\bullet B_2 \\ B_1{}^\bullet \otimes \{s\}, & \text{if } s \in B_2{}^\bullet \\ \{s\}, & \text{otherwise} \end{cases}$$

Define the *choice composition of* B_1 *and* B_2, $B_1 + B_2 = (+_1(B_1) \odot .\lfloor) \cup (+_2(B_2) \odot .\rfloor)$. ■4.6.6

As with concurrent composition, choice composition distinguishes its operands, labelling its left operand with $\lfloor$, and its right operand with $\rfloor$ (which is the source of the embedded context manipulation of the definition). To distinguish the operands of a choice composition may, at first, appear strange; however, in Chapter 6 we introduce the operator of refinement for High Level Petri Boxes which we will require to be 'syntactic', i.e., its application commutes with that of the other operators of the algebra. For this property to extend to choice composition requires, essentially, that we should force the semantic High Level Petri Boxes of a and of $a+a$ to be distinct, which is achieved though the annotation. (This is the greatest disadvantage of our transition centred approach.) As for concurrent composition, this annotation does not destroy behavioural commutativity and associativity of the operator, the proof of which follows the same lines as that for concurrent composition, given in the next chapter.

Figures 4.9 and 4.10 illustrate the choice composition of two High Level Petri Boxes. Figure 4.9 gives the simpler situation in which the operand High Level Petri Boxes have singleton pre- and post-sets so that the effect of the context manipulation by the auxiliary contexts $\lfloor$ and $\rfloor$ can be seen clearly. Figure 4.9 shows, essentially, how arc annotations are manipulated through the operation. Figure 4.10 shows how the interface places are manipulated in the more complex situation when entry and exit interfaces are non-trivial.

PROPOSITION 4.6.7 Given sociable High Level Petri Boxes B_1 and B_2, $B_1 + B_2$ is a High Level Petri Box. □4.6.7

Proof: Follows from a combination of the arguments for concurrent and causal composition. ■4.6.7

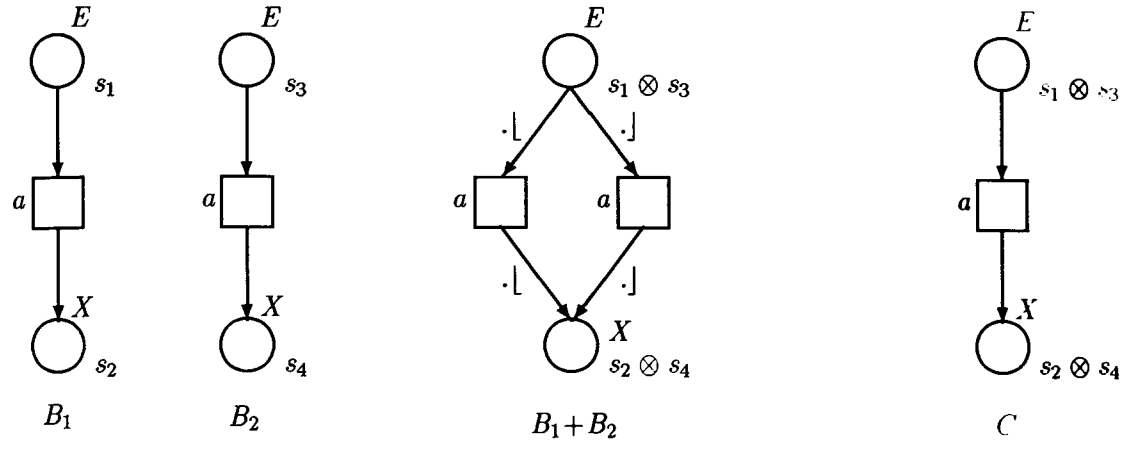


Figure 4.9: The choice composition of High Level Petri Boxes B_1 and B_2, both with interfaces consisting of single places. High Level Petri Box C shows the result of the choice composition if we were to omit the annotation of the arcs with nominal relabellings, a situation we wish to avoid.

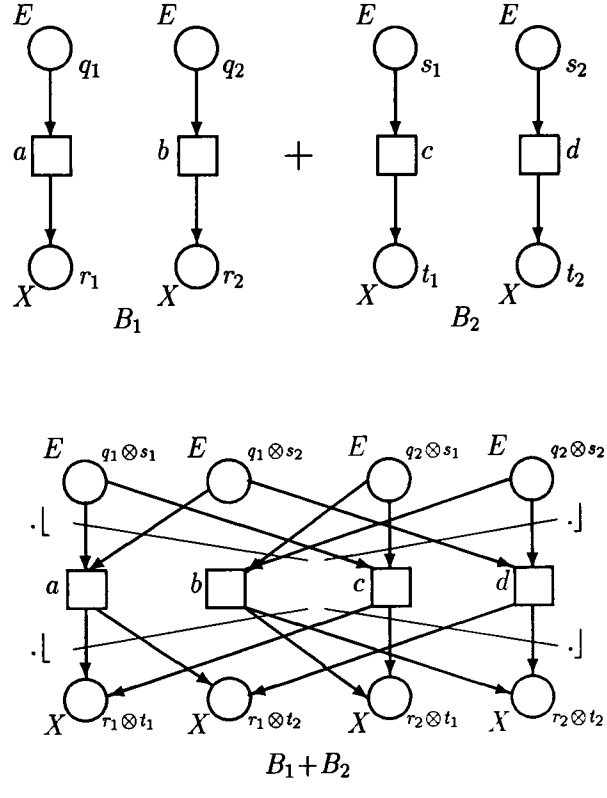


Figure 4.10: The choice composition of High Level Petri Boxes B_1 and B_2, both with interfaces consisting of two places. Notice that firing any initial transition in the image of B_1 disables each initial transition in the image of B_2 and *vice versa*.

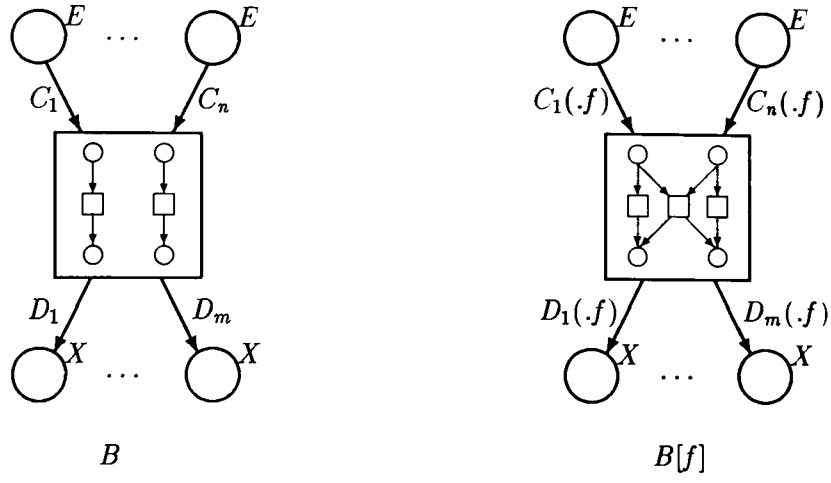


Figure 4.11: The relabelling of the High Level Petri Box in terms of the conceptual model.

4.6.4 Relabelling

Relabelling implements the relabelled multi-way synchronisation of its operand High Level Petri Box. The embedded combinational closure forms all possible 'don't care actions', whereas the context manipulation creates the scope of the relabelling.

DEFINITION 4.6.8 *[Relabelling]* Given a relabelling f, define the *relabelling of B under f*, $B[f] = B^{\oplus} \odot .f$. ■ 4.6.8

The reader will note the use of the combinational closure of B in the above definition: it provides the higher order abstract transitions which will be considered for synchronisation under the relabelling $.f$.

Figure 4.11 shows an example of the relabelling of a High Level Petri Box.

PROPOSITION 4.6.9 Given a High Level Petri Box B and a relabelling f then $B[f]$ is a High Level Petri Box. □ 4.6.9

Proof: Follows from the definition, noting that its operand satisfies Property 3 of Definition 2.5.15. ■ 4.6.9

The following is immediate from the properties of combinational closure:

PROPOSITION 4.6.10 Given a High Level Petri Box B and relabellings f and g then $B[f][g] =_{PrT} B[fg]$. □ 4.6.10

4.7 Basic High Level Petri Boxes

The synthesis of the algebra of High Level Petri Boxes over a label algebra $\mathcal{L}$ is through the repeated application of the operators defined in the previous section. The starting point of the synthesis are Basic High Level Petri Boxes, which correspond to the interpretation of the labels of the label algebra.

Apart from the observation that it is a High Level Petri Box, the denotation of a label is unremarkable. The 'place-transition-place' structure appears to be the natural denotation of an 'atomic' action as a net, and may be found many times in the literature [Gol88, Tau89, Old91, BDH92].

DEFINITION 4.7.1 *[Basic High Level Petri Boxes]* Given the label algebra $\mathcal{L} = \langle A, R \rangle$ and label $u \in A \cup \{\mathbf{0}\}$ define the *Basic High Level Petri Box*

$$Box_{\mathcal{L}}(u) = [\langle \{s_1\}, (.), u, (.), \{s_2\} \rangle]$$

where $[\_]$ is the *local transition augment* (Definition 2.5.10) and s_1 and s_2 are *fresh* place names. ■ 4.7.1

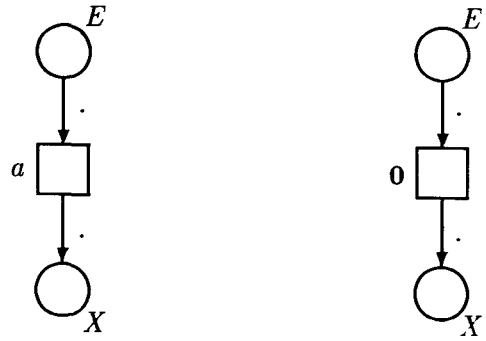
From the definition:

PROPOSITION 4.7.2 For $\mathcal{L} = \langle A, R \rangle$ a label algebra and $u \in A \cup \{\mathbf{0}\}$, $Box_{\mathcal{L}}(u)$ is a High Level Petri Box. □ 4.7.2

The denotation of the label a of the label algebra $\mathcal{L}_0$ of Example 2.1.1 and **stop** are illustrated in Figure 4.12.

4.8 The High Level Petri Box Algebra

We now define the initial portion of the High Level Petri Box Algebra (consisting of constants, and the operators of concurrent, causal and choice composition, and relabelling) and its compositional semantics in terms of High Level Petri Boxes. The definition of the algebra and its semantics is completed in Chapter 6.



$Box_{\mathcal{L}_0}(a)$ stop

Figure 4.12: The Basic High Level Petri Boxes $Box_{\mathcal{L}_0}(a)$ and **stop** = $Box_{\mathcal{L}_0}(\mathbf{0})$.

4.8.1 Syntax

The initial portion of the syntax of the High Level Petri Box Algebra is given by:

DEFINITION 4.8.1 *[High Level Petri Box Syntax]* Given a label algebra $\mathcal{L} = \langle A, R \rangle$, we define the *$\mathcal{L}$-Syntax of the High Level Petri Box Algebra* as follow:

t	::=	stop	
	|	u	$u \in A$
	|	$t \| \| t$	Concurrent composition
	|	$t;t$	Causal composition
	|	$t+t$	Choice composition
	|	$t[f]$	Relabelling, where $f \in R$

An *$\mathcal{L}$-term* is a term of the language generated through the $\mathcal{L}$-syntax. The collection of $\mathcal{L}$-terms will be denoted $Terms_{\mathcal{L}}$. ■ 4.8.1

As usual, we will omit label algebra decorations when they are clear by context.

4.8.2 Semantics

From Definition 4.3.2, to supply a compositional (denotational) semantics to the language generated from the High Level Petri Box Syntax we must identify the semantic domain and a semantic function which is a homomorphism. There are no[7] 'surprises' in the definition of the semantics.

[7] Almost; see Section 4.8.2.1.

The semantic interpretation of the constants and of the operators is that of the previous sections; compositionality is implied by an appeal to the homomorphism condition.

DEFINITION 4.8.2 Let $\mathcal{L} = \langle A, R \rangle$ be a label algebra and let t be an $\mathcal{L}$-term. Define

$$HLPB_{\mathcal{L}}(t) = \begin{cases} Box_{\mathcal{L}}(\mathbf{0}) & t = \mathtt{stop} \\ Box_{\mathcal{L}}(t) & t \in A \\ HLPB_{\mathcal{L}}(t_1) \| HLPB_{\mathcal{L}}(t_2) & t = t_1 \| t_2 \\ HLPB_{\mathcal{L}}(t_1) ; HLPB_{\mathcal{L}}(t_2) & t = t_1 ; t_2 \\ HLPB_{\mathcal{L}}(t_1) + HLPB_{\mathcal{L}}(t_2) & t = t_1 + t_2 \\ HLPB_{\mathcal{L}}(t_1)[f] & t = t_1[f], f \in R \end{cases}$$

■ 4.8.2

For the parsing of terms, we define relabelling to have highest precedence, followed by concurrent composition, causal composition and, with lowest precedence, choice composition. We will, however, use parentheses freely for disambiguation.

To end the definition of this portion of the semantics, we define the class of syntactically generated High Level Petri Boxes over the label algebra $\mathcal{L}$:

DEFINITION 4.8.3 $\mathcal{HLPB}_{\mathcal{L}} = \mathrm{ran}(HLPB_{\mathcal{L}})$. ■ 4.8.3

We return to $\mathcal{HLPB}_{\mathcal{L}}$ in Chapter 6.

4.8.2.1 Well-definedness of the Semantic Function

Actually, we have been a little premature in defining the class of syntactically generated High Level Petri Boxes, as Propositions 4.6.2, 4.6.4, 4.6.7 and 4.6.9 are not, in themselves, sufficient to show that the semantics of all terms of the High Level Petri Box Syntax are High Level Petri Boxes: it might be possible that through some previous composition, the operands of a concurrent, causal or choice composition may be unsociable. Theorem 4.8.4 shows that this is never the case. The result is, in fact, uncomplicated and follows from our (careful) restriction on Basic Boxes that their constituent places must be 'fresh'.

THEOREM 4.8.4 For every $\mathcal{L}$-term, t, $HLPB_{\mathcal{L}}(t)$ is a High Level Petri Box. □ 4.8.4

The proof of Theorem 4.8.4 follows from the properties of the auxiliary operators stated in the following lemmata, which:

1. characterise the relationship between the identities of the operands and the results of applications of the auxiliary operators, and

2. extend this to the structure of a High Level Petri Box term.

LEMMA 4.8.5 For B_1, B_2 High Level Petri Boxes:

1. $\mathrm{id}(B_1 \cup B_2) = \mathrm{id}(B_1) \cup \mathrm{id}(B_2)$,

2. $\mathrm{id}(B_1^{\oplus}) = \mathrm{id}(B_1)$,

3. for C, $C' \in \mathcal{C}$, $\mathrm{id}(B_1 \odot (C, C')) = \mathrm{id}(B_1)$,

4. for $f: S \to S'$ a sane relation, $\mathrm{id}(f(S)) \subseteq S'$. □ 4.8.5

Proof: Immediate ■ 4.8.5

LEMMA 4.8.6 Consider the parse tree[8] p_t of a term $t \in Terms_{\mathcal{L}}$. Then the High Level Petri Boxes which are the semantics of disjoint subtrees of p_t are sociable. □ 4.8.6

Proof: Follows from our requirement that the identities of the Basic High Level Petri Boxes which form the denotations of distinct leaf nodes of the parse tree are disjoint, and Lemma 4.8.5. ■ 4.8.6

Proof: Follows from Lemmata 4.8.5 and 4.8.6. ■ 4.8.4

4.8.3 Class-wide Properties of High Level Petri Boxes

4.8.3.1 Finitary Reasoning on High Level Petri Boxes

As the operator of relabelling is derived from that of combinational closure, there are syntactically generated High Level Petri Boxes whose set of constituent abstract transitions is (countably) infinite. The importance of the following result, then, is that it allows finitary reasoning to be used for properties of syntactically generated High Level Petri Box, as long as that reasoning may be stated in terms of the underlying local transitions or places.

[8] In the usual sense.

PROPOSITION 4.8.7 Let B a syntactically generated High Level Petri Box. Then $|S_B| \in \mathbf{N}$ and $|LT(B)| \in \mathbb{N}$. □4.8.7

Proof: We give the proof for S_B; that for $LT(B)$ is similar. Let t be such that $B = HLPB_{\mathcal{L}}(t)$ for some label algebra $\mathcal{L}$.

The proof proceeds by structural induction on t. For $t \in A \cup \{\mathbf{0}\}$ then $|S_B| = 2$, and the result is clear. Suppose $t = t_1 + t_2$. Consider the syntactically generated $B_i = HLPB_{\mathcal{L}}(t_i)$. Then from the induction hypothesis we have that $|S_{B_i}| = n_i \in \mathbb{N}$. But then, from the definition, $|S_B| = |{}^\bullet B_1 \otimes {}^\bullet B_2 \cup B_1{}^\bullet \otimes B_2{}^\bullet \cup \overset{\bullet}{B_1} \cup \overset{\bullet}{B_2}| \in \mathbb{N}$, and we have the result. The cases for $t = t_1;t_2$, $t = t_1 \| t_2$ and $t = t_1[f]$ are similar. ■4.8.7

4.8.3.2 Completeness

As we do not remove local transitions from a High Level Petri Box in the applications of High Level Petri Box operators another simple inductive argument gives the following:

PROPOSITION 4.8.8 A syntactically generated High Level Petri Box is complete. □4.8.8

Moreover:

PROPOSITION 4.8.9 Let B be a syntactically generated High Level Petri Box. Then for all $l \in LT(B)$, ${}^\bullet l \cap l^\bullet = \emptyset$. □4.8.9

Hence, we may apply the Decoupling Lemma (Lemma 3.4.4) and its associated machinery to syntactically generated High Level Petri Boxes.

4.8.3.3 Local Strictness of Syntactically Generated High Level Petri Boxes

We will use the notation $C[\_]$ to represent a 'term with a hole in it'[9]. For example, $C[\_] = a+(b\|\_);d$ is a term with a hole. Providing an argument for such a C fills in the hole: in the case of the example $C[d] = a+(b\|d);d$, whereas $C[t_1]$ for some term t_1, is the term $a+(b\|t_1);d$. We may view a term with a hole as defining manipulations on places, local transitions and abstract transitions: those which would be applied to the components of its operand in taking the semantics of its applied form. For instance, for $l \in LT(HLPB(t))$ and term with a hole

[9]Or *context*, [Bar84, page 29], which we do not use for obvious reasons.

$C[\_]$, we may define $C[l]$ as the l' in $LT(HLPB(C[t]))$ corresponding to l. No operator so far introduced removes local transitions so that these operations are well defined. Also, we will use the symbols $\pm$ and $\mp$ to represent the nominal relabellings $\vdash$ and $\dashv$ under the convention that a 'related pair' of $\pm$ and $\mp$ are 'opposite handed', i.e., $\pm = \dashv \Leftrightarrow \mp = \vdash$.

We may now show that syntactically generated High Level Petri Boxes are locally strict:

THEOREM 4.8.10 Let $B = HLPB(t)$ be a syntactically generated High Level Petri Box. Then B is locally-strict. □ 4.8.10

Proof: We require the following lemma:

LEMMA 4.8.11 Let t be a High Level Petri Box term and $l_1 \neq l_2 \in LT(HLPB(t))$. Then

1. ${}^\bullet l_1 \cap {}^\bullet l_2 \neq \emptyset$ implies there are terms t_1 and t_2 and a term with a hole $C[\_]$ such that $t = C[t_1 + t_2]$ (respectively, $t = C[t_2 + t_1]$) with $l_1 = C[[l_1'] + t_2]$ (respectively, $l_1 = C[t_2 + [l_1']]$), and $l_2 = C[t_1 + [l_2']]$ (respectively, $l_2 = C[[l_2'] + t_1]$), for $l_i' \in LT(HLPB(t_i))$, $i = 1, 2$.
2. for such l_i, there are contexts D_1, D_2 and D such that ${}^C l_1 = D_1(.\lfloor)D$ and ${}^C l_2 = D_2(.\rfloor)D$, or ${}^C l_1 = D_1(.\rfloor)D$ and ${}^C l_2 = D_2(.\lfloor)D$. □ 4.8.11

Proof: For a syntactically generated High Level Petri Box, the non-empty intersection of the pre-sets of local transitions is produced only through a choice composition. The result follows. ■ 4.8.11

Returning to the proof of Theorem 4.8.10, we will suppose that the result does not hold, i.e., $B = HLPB(t)$ but B is not locally-strict. Then there is a transition $r = \{\!|l_1, \ldots, l_n|\!\} \in B$ with $s \in {}^\bullet r^\bullet$ and a feasible substitution $\alpha \in \mathrm{subs}^B(r)$ with $W_F(s,r)\!:\alpha$ or $W_F(r,s)\!:\alpha$ not a set. We will assume that $W_F(s,r)\!:\alpha$ is not a set, the other case being similar. Then there are $i \neq j$ such that $\alpha(x_i) = \alpha(x_j) = E$, say.

Suppose x_i and x_j correspond to local transitions l_i and $l_j \in r$, respectively. If $l_i = l_j$ then ${}^C l_i = {}^C l_j$ and hence ${}^C l_i \alpha(x_i) = {}^C l_j \alpha(x_j)$ in contradiction of Proposition 2.4.4. Hence $l_i \neq l_j$. As $s \in {}^\bullet l_i \cap {}^\bullet l_j$, we may appeal to Lemma 4.8.11 to give contexts D_1, D_2 and D such that ${}^C l_i = D_1(.\lfloor)D$ and ${}^C l_j = D_2(.\rfloor)D$ or ${}^C l_i = D_1(.\rfloor)D$ and ${}^C l_j = D_2(.\lfloor)D$. We assume, without loss of generality, that the first case applies so that ${}^C l_i \alpha(x_i) = D_1(.\lfloor)DE$ and ${}^C l_j \alpha(x_j) = D_2(.\rfloor)DE$.

However, from the definition of Flow, there are also contexts C_1. C_2 and C such that ${}^{C}l_i E = C_1(.\pm)C$ and ${}^{C}l_j E = C_2(.\mp)C$. This is a contradiction as $\{\vdash, \dashv\} \cap \{\lfloor, \rfloor\} = \emptyset$.

Hence, B is locally-strict. ■ 4.8.10

4.8.3.4 Safeness and Memorylessness of Syntactically Generated High Level Petri Boxes

A marking of a P/T net is safe [BF86] if defines a set rather than a multiset. A P/T net is safe from a given marking if all markings reachable therefrom are safe. A P/T net is *empty* [BDKP91, Dev92] or *clean* [Hac76] if, when it has terminated it leaves no other tokens in the P/T net. In this section we extend these properties to High Level Petri Boxes. Later in this chapter and in Chapter 7 we show that syntactically generated High Level Petri Boxes, when marked with and from standard initial markings, share them both.

DEFINITION 4.8.12 Let B be a High Level Petri Box and d a context. B is *d-safe* when for all markings $M \in \mathcal{M}_d^B$, M is safe.

B is *safe* if it is d-safe for all standard initial markings $\mathcal{M}_d^I$. ■ 4.8.12

For the application of memorylessness to High Level Petri Boxes we must consider the individuality of tokens:

DEFINITION 4.8.13 Let B be a High Level Petri Box and d a context. B is *d-memoryless* when for all markings $M \in \mathcal{M}_d^B$, $B^\bullet \subseteq \mathrm{dom}(M)$ implies $M = M_d^T$.

B is *memoryless* if it is d-memoryless for all standard initial markings d. ■ 4.8.13

That a High Level Petri Box is memoryless implies that, other than for the passage of control, no information may be passed in tokens. Interaction with other High Level Petri Boxes is therefore solely in terms of the labels a memoryless High Level Petri Box generates. Moreover, a memoryless High Level Petri Box has no 'state', in the sense that no information is held between 'invocations'.

Not all High Level Petri Boxes are safe or memoryless; an unsafe[10] High Level Petri Box 'with memory' is illustrated in Figure 4.13; other High Level Petri Boxes 'with memory' are illustrated in Figure 4.14.

[10]Which is, nevertheless, locally-strict.

Figure 4.13: Example of an unsafe High Level Petri Box. From the given standard initial marking, no finite bound can be put on the number of (.) tokens which may be present in the exit-place.

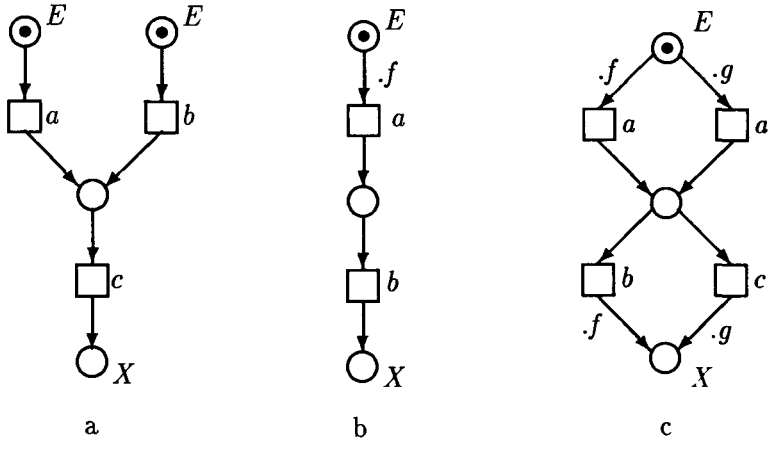


Figure 4.14: Example of High Level Petri Boxes 'with memory'. From the standard initial marking, the High Level Petri Box illustrated in a. is not memoryless as covering its exit places does not imply all other places are empty. The High Level Petri Box illustrated in b. is also not memoryless as it leaves the token $.fd$ in its exit place on termination rather than the token d which formed its standard initial marking. In c. an undesirable property of non-memoryless High Level Petri Boxes is shown. When $f(a) = g(a)$, but $f(b) \neq g(c)$, we are able to distinguish which alternative of the upper choice was chosen by subsequent synchronisation on $f(b)$ or $g(c)$, even though the labels they generate are equal.

In being concerned only with the cardinality of markings, safeness and memorylessness are properties monotonic in the permissiveness of a label algebra. As the High Level Petri Box operators preserve the completeness of their operands, using the techniques of Section 3.4 we may reduce the complexity of proofs of these properties by considering only markings reachable through local (rather than abstract) transition sequences.

The properties of safeness and memorylessness are true not only of the syntactically generated High Level Petri Boxes of this chapter, but also of the High Level Petri Boxes which form the denotations of terms of the second portion of the algebra defined in Chapter 6. As the proofs are essentially the same, although we make the statement of the theorem here, we postpone its proof to be presented together with that for Chapter 6.

THEOREM 4.8.14 A syntactically generated High Level Petri Box is both safe and memoryless. □ 4.8.14

We discuss the importance of safeness and memorylessness in the discussion which ends this chapter.

4.8.4 Isomorphism and Syntactically Generated High Level Petri Boxes

We have defined a compositional High Level Petri Box semantics for terms from the portion of the High Level Petri Box Algebra defined in this chapter. However, the proof that compositions always gave High Level Petri Boxes relied upon the fact that all distinct Basic High Level Petri Boxes are pairwise sociable. This technical consideration has the logical conclusion that, in modelling systems with the High Level Petri Box Algebra we should be aware of all places which have *ever* been used through previous applications of the semantic function.

We may mitigate this situation if we are able to show that isomorphism of High Level Petri Boxes is a congruence with respect to the operators; then, given two unsociable High Level Petri Boxes, we can always rename the places of one to arrive at sociable High Level Petri Boxes, which we may then compose without problem.

We note that this does not only apply to High Level Petri Boxes and it is not unrelated to the problems associated with accidental capture of bound variables which occurs through, for instance, *substitution* in the λ-calculus, as well as many other theories. In fact, the solution we propose is only a form of *α-conversion* for place names of High Level Petri Boxes.

In content, then, this section is similar to [Bar84, Pg. 25] and we adopt a similar 'moral' [Bar84, Pg. 27, 2.1.14]: that, using the results of this section, we may work with High Level Petri Boxes in a naive way.

PROPOSITION 4.8.15 Let B_1, B_1', B_2 and B_2' be pairwise sociable High Level Petri Boxes with $B_1 \equiv B_1'$ and $B_2 \equiv B_2'$. Then:

1. $B_1 \| B_2 \equiv B_1' \| B_2'$
2. $B_1 ; B_2 \equiv B_1' ; B_2'$
3. $B_1 + B_2 \equiv B_1' + B_2'$
4. $B_1[f] \equiv B_2[f]$, for relabelling f. □ 4.8.15

Proof: We give the proof for choice composition, which is the most complex of the three binary cases, and for relabelling.

3. The only difficulty in the proof is the definition of the 'correct' bijection between the composed nets, this is mitigated by the binary product operator.

   From Definition 4.5.7 we have that $S_{B \odot C} = S_B$, for any context C. Hence $\sigma_1 : B_1 \cdot (.\rfloor) \equiv B_1' \odot (.\rfloor)$ and $\sigma_2 : B_2 \odot (.\rfloor) \equiv B_2' \odot (.\rfloor)$. Moreover, the context manipulated High Level Petri Boxes are also sociable.

   From Definition 4.6.6 we have that $B + B' = +_1(B \odot .\rfloor) \cup +_2(B' \odot .\rfloor)$.

   Define a mapping $\sigma : S_{B_1+B_2} \to S_{B_1'+B_2'}$ by

$$\sigma(s) = \begin{cases} \sigma_1(s) & s \in S_1 \\ \sigma_2(s) & s \in S_2 \\ \sigma_1(s_1) \otimes \sigma_2(s_2) & s = s_1 \otimes s_2 \in {}^\bullet(B_1+B_2)^\bullet \end{cases}$$

   That σ is an isomorphism follows by a purely technical manipulation of the definitions.

4. The result for relabelling is derived from the following properties of the auxiliary operators:

   LEMMA 4.8.16 Given an isomorphism $\sigma : B_1 \equiv B_1'$ then:

   (a) $B_1{}^{\oplus} \equiv B_1'{}^{\oplus}$

   (b) $B_1 \odot (C_1, C_2) \equiv B_1' \odot (C_1, C_2)$. □ 4.8.16

   Proof: (a) Follows easily from the observation that

$$\begin{aligned} & \sigma(T_{B_1^{\oplus}}) \\ &= \sigma\{n_1 t_1 \oplus \ldots \oplus n_m t_m \mid t_i \in T_1\} \\ &= \{n_1 \sigma(t_1) \oplus \ldots \oplus n_m \sigma(t_m) \mid \sigma(t_i) \in T_2\} \\ &= \sigma(T_{B_2^{\oplus}}) \end{aligned}$$

   as $\sigma : T_1 \equiv T_2$ by assumption.

(b) Follows easily from the observation that it is the relation of places to the entry and exit places of the High Level Petri Box which defines the manipulation and this relation is not altered by the isomorphism.

■ 4.8.16

Hence the result. ■ 4.8.15

The other technicality in the result is to show that all Basic High Level Petri Boxes for the same label are isomorphic; the result follows immediately from the definitions:

LEMMA 4.8.17 Given a label algebra $\mathcal{L} = \langle A, R \rangle$ and $u \in A \cup \{\mathbf{0}\}$ then (using separate applications of the semantic function) for $B_1 = HLPB_{\mathcal{L}}(u)$ and $B_2 = HLPB_{\mathcal{L}}(u)$, $B_1 \equiv B_2$.

□ 4.8.17

With Lemma 4.8.17 forming the base case and a simple inductive argument, we may extend the result to all syntactically generated High Level Petri Boxes:

LEMMA 4.8.18 Let t_1, t_2 be $\mathcal{L}$-terms for some label algebra $\mathcal{L}$, and $f \in R_{\mathcal{L}}$. Through separate applications of the semantic function define $B_1 = HLPB_{\mathcal{L}}(t_1)$, $B_2 = HLPB_{\mathcal{L}}(t_2)$, $B_1' = HLPB_{\mathcal{L}}(t_1)$ and $B_2' = HLPB_{\mathcal{L}}(t_2)$. Then:

1. $B_1 \| B_2 \equiv B_1' \| B_2'$
2. $B_1 ; B_2 \equiv B_1' ; B_2'$
3. $B_1 + B_2 \equiv B_1' + B_2'$
4. $B_1[f] \equiv B_1'[f]$. □ 4.8.18

We have arrived at our goal of this section, that:

THEOREM 4.8.19 *[Congruence Theorem]* Isomorphism of syntactically generated High Level Petri Boxes is a congruence relation with respect to the top-level operators. □ 4.8.19

4.8.5 Extended High Level Petri Boxes

That isomorphism is a congruence with respect to the top level operators we have defined in this chapter means that we may partition High Level Petri Boxes according to equivalence classes in the usual way. As usual let $\mathcal{L} = \langle A, R \rangle$ be a label algebra.

DEFINITION 4.8.20 For $B \in \mathcal{HLPB}_{\mathcal{L}}$ define the *Extended High Level Petri Box*, $[B] = \{B' \in \mathcal{HLPB}_{\mathcal{L}} \mid B' \equiv B\}$. Define the class of Extended High Level Petri Boxes $\mathcal{EHLPB}_{\mathcal{L}} = \mathcal{HLPB}_{\mathcal{L}}/\equiv$. ■ 4.8.20

From the results of the previous section, using the Congruence Theorem, we may define versions of the top level operators for Extended High Level Petri Boxes:

DEFINITION 4.8.21 Let B, B_1, B_1', B_2 and $B_2' \in \mathcal{HLPB}_{\mathcal{L}}$ such that B_1' and B_2' are sociable, $B_1 \equiv B_1'$ and $B_2 \equiv B_2'$. From Definition 4.8.20, $[B_1] = [B_1']$ and $[B_2] = [B_2']$. Define

1. $[B_1]\hat{\|}[B_2] = [B_1' \| B_2']$,
2. $[B_1]\hat{;}[B_2] = [B_1'; B_2']$,
3. $[B_1]\hat{+}[B_2] = [B_1' + B_2']$,
4. $[B]\widehat{[f]} = [B[f]]$, for f a relabelling. ■ 4.8.21

Whereas, in general, definitions which follow will be in terms of unextended High Level Petri Boxes so that the identity of places (and their manipulation) is evident, we will return to the class of extended High Level Petri Boxes when we construct a refinement operator on High Level Petri Boxes.

4.9 Discussion

4.9.1 An Alternative Representation of stop

An alternative definition of stop to that given in Definition 4.7.1 is that of the low level Petri Box Calculus: as single entry and exit places without an intervening transition. Although intuitively attractive, this representation removes the property of High Level Petri Boxes, used in Chapter 5, that entry and exit places of a High Level Petri Box are necessarily connected, a property that facilitates the definition of the normal form for High Level Petri Boxes. We note, however, that the P/T net unfolding of stop is, essentially, the low level Petri Box representation.

4.9.2 The Unit Pre-High Level Petri Box ϵ

The conditions which characterise High Level Petri Boxes from pre-High Level Petri Boxes provide for the representation of abstract transitions as collections of local transitions. We

were careful to disallow the 'unit pre-High Level Petri Box', ϵ, as a High Level Petri Box as it compromises closure of the class of High Level Petri Boxes under composition by auxiliary and top-level operators, as we shall see.

DEFINITION 4.9.1 Define the *unit pre-High Level Petri Box*

$$\epsilon = \langle \{s\}, \emptyset, \{s \rightarrow \{E, X\}\} \rangle$$

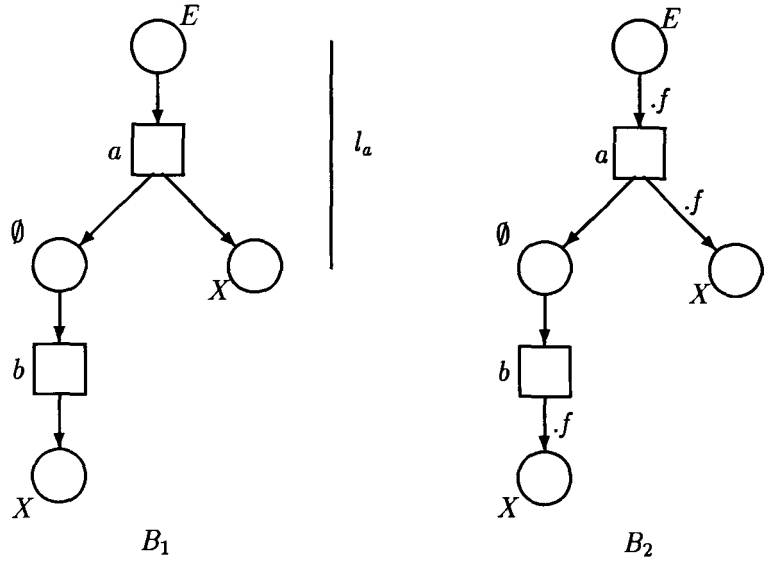
(where, as usual, $s \in \mathcal{P}$ is a fresh place). ■ 4.9.1

As ϵ has no local transitions, it follows that it cannot satisfy the conditions to be regarded as a High Level Petri Box. However, if we wished to extend the operators defined in this chapter to allow ϵ to be included within their range it would not be difficult to do so.

However, to include ϵ as a High Level Petri Box causes representational problems. To see this, consider the label algebra of Example 2.1.1 (page 8) (or any which has actions a and b) and the term $E = a;(b||\epsilon)$ in which ϵ is concurrently composed with action b, the resulting expression being causally inferior to the action a. The High Level Petri Box semantics of this expression is shown in Figure 4.15. As is clear from the figure, this particular combination of concurrent and causal composition in the presence of ϵ has the effect of 'unbalancing' the post-places of the local transition underlying action a (shown as l_a in the figure) so that Property 3 of Definition 2.5.15 (page 31) does not hold for l_a. This is unfortunate for a subsequent and immediate context manipulation as the two output arcs of l_a would become differently annotated, preventing l_a from being a (single) local transition.

4.9.2.1 Regaining Closure

Therefore, whereas, we must disallow ϵ as a High Level Petri Box if we wish to retain our local transition decomposition in the general case, if we are able to prevent an immediate context manipulation on such High Level Petri Boxes as that deriving from the term E, we will prevent the unbalancing of the local transition and, hence, retain representability. One method is to use the pre-High Level Petri Box ϵ only as part of a derived operation which does not allow an immediately following context manipulation. This is adopted in the definition of recursion given in Chapter 6. For more details we refer the reader to that chapter.



B_1 B_2

Figure 4.15: In which ϵ, the unit pre-High Level Petri Box, destroys the representability by local transitions. B_1 is the denotation of the term $a;(b\|\epsilon)$. B_2 is the denotation of the term $(a;(b\|\epsilon)) \odot .f$. Note that B_2 is not a High Level Petri Box as the local transition l_a does not have consistent post-contexts.

4.9.2.2 A Causal Unit

Though we must disallow ϵ from being a High Level Petri Box, it has one property that we feel is desirable in an algebra—it forms a unit of causal composition (up to isomorphism). With a unit of causal composition we conjecture that we have a non-commutative monoid on the class of High Level Petri Boxes with causal composition as the operation and ϵ as the unit.

It may be beneficial, at some later date, to investigate the properties of High Level Petri Boxes in the light of the work of Montanari and Meseguer ([MM89]) in which a monoid structure is built on top of Petri nets.

4.9.2.3 Safeness and Memorylessness

The transition-like nature of a High Level Petri Boxes endowed through safeness and memorylessness is important property for the behavioural characterisation of Chapter 7. The non-interference of safe and memoryless operands of the binary operators means that the diagram of Figure 4.16 commutes (where *Beh* is some behavioural semantics, such as the transition sequence semantics defined in Section 3.3.4). We discuss further the form of the behavioural equivalents of the High Level Petri Box operators (both of this chapter and of Chapter 6) in Section 7.1.

$$\begin{array}{ccccc} B_1 & \longrightarrow & B_1 \; op \; B_2 & \longleftarrow & B_2 \\ \downarrow & & \downarrow & & \downarrow \\ Beh(B_1) & \longrightarrow & Beh(B_1)\, Beh(op)\, Beh(B_2) & \longleftarrow & Beh(B_2) \end{array}$$

Figure 4.16: Commuting diagram enabled by the safeness and memorylessness of High Level Petri Boxes. *Beh* is some behavioural semantics; *op* is one of $\|$, ; or +. The details of $Beh(op)$ for each *op* are outlined in Section 7.1, and are, essentially, different disjoint unions.

4.10 Summary

In this chapter we have introduced the High Level Petri Box Algebra, consisting of a collection of syntaxes, one for each label algebra and, for each, a compositional semantics in terms of High Level Petri Boxes. We have shown that the operator interpretations are well-defined and preserve the characterising properties of High Level Petri Boxes. We have also shown that isomorphism, defined in Chapter 2, is a structural congruence with respect to the High Level Petri Box operators.

In the next chapter we show that the operators defined in this chapter allow the definition of a normal form for High Level Petri Boxes.

Chapter 5

Normal Form

In this chapter we show that every syntactically generated High Level Petri Box has a unique normal form. We show that the rewrite system underlying the normal form preserves certain characteristic behaviours of a High Level Petri Box, so that equality of normal forms is sufficient for behavioural equivalence. We also show that, in the general case, equivalence of behaviour is too dependent on the particulars of a label algebra over which a High Level Petri Box term is defined for equal normal forms to be considered necessary for equivalent behaviour.

After the definition and justification, we give a number of applications of the normal form:

1. we partially characterise the behaviour of a syntactically generated High Level Petri Box from a standard initial marking,
2. we show that `stop` is a behavioural unit of choice composition,
3. we show that concurrent and choice compositions are behaviourally commutative and associative (discharging our obligation established in the previous chapter).

5.1 Normal Forms

Normal forms provide powerful tools in the mathematics and meta-mathematics of a theory. Normal forms provide (partial) syntactic characterisations of semantic properties although, given their wide applications, the presentation of 'syntax' and 'semantics' is not always that of algebra. Of course, syntactic characterisations of semantic properties are not always possible, so that normal forms do not exist for arbitrary algebraic systems: for instance the solution of the general word problem, that is, whether or not two terms composed of variables and operators

can be proved equal as a consequence of a given set of identities satisfied by the operators is undecidable whereas it would be decidable given that suitable normal forms existed.

Normal forms have been used to derive deep properties of formal systems: for instance, the *Hauptsatz* of Gentzen [Kle52] provides a canonical representation of a proof in a sub-theory of number theory, which is then used as a basis for the proof of the consistency for that (sub-)theory. Well known normal forms used in logic are: disjunctive and conjunctive normal forms, Prenex normal form and Skolem normal form [Kle52]. Collectively, these form the basis of decision procedures for, amongst other things, a sub-theory of arithmetic (Presburger Arithmetic, [Sho77, Sho79]); the first three constitute the formal underpinning of the logic programming language Prolog [CM81]; and there are decision procedures for the propositional calculus based on the first two [Kle52]. On a more recreational level, lists of normal form representations of words and phrases are provided for crossword addicts, [Dai93], as a truly fast way of determining the anagrams of the particular collection of letters.

The reduction of a proof to its normal form in the *Hauptsatz* corresponds, as Gentzen described it, to:

> *[exploiting] the possibilities of the permutation of inferences within [the formal] system* [Kle52].

The inferences to which Gentzen refers comprise the formal justification, or proof, of a theorem within the formal system. The 'possibilities of permutation of inferences' are not unconstrained 'rewritings' of the proof but require that the validity of the expression should be preserved. If a proof is expressed through a syntax of inferences, with justifiability inherited through such inferences, then we might paraphrase Gentzen and say that a reduction to a normal form consists of syntactic manipulations that preserve semantics.

Our normal form consists of a rewrite system at the level of syntax and a semantic property, behaviour, preserved by the rewriting.

5.1.1 Rewrite Systems, Termination and Confluence

A *rewrite system* for a syntax is a set of pairs of terms from the language generated by that syntax. Usually such pairs of terms of a syntax will be organised into 'meta'-pairs, each representing a collection of pairs of similar forms which are manipulated by the rewrite system in regular ways.

By an *application* of a rewrite system, R, to a term, t, we mean that there is an element of the rewrite system with t as the first component of the pair. The *result of the application* is the second component of the pair. Notice that we do not insist that there is a *unique* pair for which this holds, so that there may be more than one possible rewrite of a given term. The *rewrite of a term t (under R)* is the iterated application of the rewrite system to that term. Given a term t and rewrites t' and t'', t' and t'' will be called *similar*. A term, t, is said to be *irreducible (with respect to R)* or *in normal form (with respect to R)* if there is no rewrite of t (under R).

A rewrite system is said to be *terminating* if its transitive closure has no infinite depth branches. The proof that a rewrite system terminates often proceeds through the identification of a *measure* for a term, i.e., a function, m, mapping terms to natural numbers, such that for terms t and t' if t' is a rewrite of t then $m(t) > m(t')$. Termination then follows by induction.

A rewrite system is said to be *confluent* if similar terms may always be rewritten to the same term.

A rewrite system is *Church-Rosser* (or *canonical*) if it is both confluent and terminating. A Church-Rosser rewrite system has the attractive property that it defines a *unique* normal form for each term of the syntax. Given that rewriting preserves semantics this means that there is a *partial semantic characterisation through syntax*, i.e., that if normal forms of terms t and t' are equal then they have the same semantics. A rewrite system is *complete* if the reverse implication holds, i.e., in which equal semantics implies equal normal forms. A complete rewrite system is particularly attractive as it gives a precise syntactic characterisation of the semantic property. (For a lucid discussion of this relation between syntax and semantics see [GLT89].)

The rewrite system we define in Section 5.2.2 is Church-Rosser but does not have the further property of completeness. In fact, this is the best we can hope for with this form of rewrite system, as is shown in Section 5.2.1.

5.1.1.1 Using an Equivalence as a Rewrite

An equivalence is, in particular, a symmetric relation. To derive a rewrite system based on an equivalence we might break the symmetry by choosing, for each instance of the equivalence, one side as being 'preferable' to the other, and agreeing always to rewrite a less preferable subterm with a more preferable subterm; for instance, if $t \equiv t'$ is an instance of an equivalence, by considering the right hand side as preferable to the left we will always replace $C[t]$ by $C[t']$, for $C[\_]$ some term with a hole, whenever we encounter it. The full 'Knuth-Bendix' procedure

is detailed in [KB70], where it runs to six sections: we do not reproduce it here. To use this technique for converting between our equivalences of commuting auxiliaries, derived below, and the rewrite rules of our normal form we will be careful to indicate the 'direction of preference'.

5.1.2 Auxiliary Syntax

The *rewrite system* through which the normal form is defined acts on the auxiliary operators. To this end we introduce the *auxiliary syntax* of the $\mathcal{HLPB}_{\mathcal{L}}$ which is that of Definition 4.8.1, but in which terms are expressed through their constituent auxiliary operators. For instance $t_1 \| t_2$ becomes $t_1 \odot .\vdash_{t_1} \cup\, t_2 \odot .\dashv_{t_2}$, $t_1 ; t_2$ becomes $;_1^{(t_1,t_2)}(t_1) \cup ;_2^{(t_1,t_2)}(t_2)$ and $t[f]$ becomes $t^{\oplus} \odot .f_{t^{\oplus}}$. Note that, unlike the top level syntax in which we are able to make the abstraction, we must state explicitly *all* components of a term, including the superscripts to relations and the subscripts to context manipulations, for otherwise they may be ambiguous.

5.2 A Normal Form for High Level Petri Boxes

For us, the syntax will be that of High Level Petri Boxes and the semantics a certain sub-class of High Level Petri Box behaviours. Our rewrite system will comprise the 'possibilities for permutation in the auxiliary syntax of High Level Petri Boxes'[1].

There are two observations of some note:

1. the rewrite system underlying our normal form derives from the possibilities for commuting the application of the auxiliary operators. However, to be able to apply rewrite rules of this form we must first convert a High Level Petri Box term into its auxiliary representation, as given in Section 5.1.2. t_{aux} will denote this auxiliary representation of a High Level Petri Box term t. In fact, with only a slight abuse of our definitions[2], we may consider the 'rewriting' of t as t_{aux} a rewrite of our system: it clearly preserves the behaviour of a High Level Petri Box and can have no effect on either the termination (as it may not apply more than once) or confluence (as this auxiliary representation is obviously unique) of the rewrite system so that this abuse is harmless. (For the sake of termination of the rewrite system, however, it is not profitable to define an 'inverse' rewrite of t_{aux}.)

[1] Where by 'permutation in the auxiliary syntax' we mean commutativity of application of pairs of auxiliary operators: to recap, if f and g are operators, the application of f and g commutes (with respect to $\equiv$) if and only if $f \circ g \equiv g \circ f$.

[2] Which is that we do not consider rewrite rules defined on top-level operators.

	$\cup$	$f(\_)$	$\odot$	$\oplus$
$\cup$	7			
$f(\_)$	1	8		
$\odot$	2	4	9	
$\oplus$	3	5	6	10

Table 5.1: The number of the item discussing the commutativity of auxiliary operator aux_1 and aux_2 is given as the entry in row aux_1, column aux_2

2. in defining the behaviour of High Level Petri Boxes in terms of PrT nets which, in turn, have their behaviours defined in terms of P/T nets, we may in the worst case, be required to show that the proposed commuting of auxiliary operators preserves the behaviour of P/T nets. Unfortunately, this complication does occur in the commuting of union and context manipulation. However, for all other pairs of auxiliaries the demonstration is trivial for, as we shall see, the commuting of pairs preserves the (static) structure of a High Level Petri Box and, thus, obviously the behaviour. The demonstration of the 'worst case', is not trivial, however, forming the main part of the technical content of this chapter.

5.2.1 An Anatomy of the Interrelation of Auxiliaries

Throughout the following, unless otherwise stated, High Level Petri Box will mean syntactically generated High Level Petri Box. We will also assume that we have been given a label algebra $\mathcal{L} = \langle A, R \rangle$ with respect to which all label algebra relative concepts are defined.

We refer the reader to Table 5.1 for an index to the following discussion of the commuting of auxiliaries.

Discussion:

1. The table entry represents the commuting of relation lifting and union, i.e., we are comparing $f(B_1 \cup B_2)$ and $f(B_1) \cup f(B_2)$ for f a sane relation and B_1 and B_2 unionable High Level Petri Boxes. If, by f applied to B_1 and B_2, we understand f with a suitably restricted domain then, from the definitions $f(B_1 \cup B_2) = f(B_1) \cup f(B_2)$. We thus have automatic equivalence of behaviour. To use this equivalence as a rewrite rule we will prefer to apply relational lifting before union, so that our rewrite is $f(B_1 \cup B_2) \rightarrow f(B_1) \cup f(B_2)$. This choice, however, is arbitrary.

2. The table entry represents the commuting of context manipulation and union, i.e., we are comparing $(B_1 \cup B_2) \odot C$ and $B_1 \odot C \cup B_2 \odot C$, for context C and unionable High Level Petri Boxes B_1 and B_2. We distinguish two cases:

   (a) when B_1 and B_2 are such that ${}^\bullet B_1 \cap B_2{}^\bullet = \emptyset = B_1{}^\bullet \cap {}^\bullet B_2$ then, from the definitions, $(B_1 \cup B_2) \odot C = B_1 \odot C \cup B_2 \odot C$ and, from Table 3.1 with it comes behavioural equivalence;

   (b) when B_1 and B_2 are such that ${}^\bullet B_1 = B_2{}^\bullet$ or $B_1{}^\bullet = {}^\bullet B_2$ (but not both as we assume them unionable) the result ceases to be structural of either the PrT or P/T net denotations as the example of Figure 5.1 shows. The figure illustrates High Level Petri Boxes B_1 and B_2 with $B_1{}^\bullet = {}^\bullet B_2$, and standard initially marked High Level Petri Boxes $(B_1 \cup B_2) \odot .f$ and (under the conjectured rewrite) $(B_1 \odot .f) \cup (B_2 \odot .f)$. Comparing the structure and behaviour of the two High Level Petri Boxes we see that:

      i. the contexts which annotate the arcs of local transitions incident on the respective internal places of the two High Level Petri Boxes differ in the two semantics,

      ii. if transition t_a is enabled at the standard initial marking, then tokens in $(B_1 \cup B_2) \odot .f$ reach the internal place containing the context $.f$,

      iii. if transition t'_a is enabled at the standard initial marking, then tokens in $(B_1 \odot .f) \cup (B_2 \odot .f)$ reach the internal place containing the trivial context $(.)$,

      iv. transitions t_a and t'_a, and t_b and t'_b are enabled and disabled together.

   For standard initially marked, syntactically generated High Level Petri Boxes this is, in fact, the general case, as is shown below.

   However, from other (non-standard) initial markings, we may distinguish t and t' by their behaviours. For, consider the marking in which the context $(.)$ occupies the internal place of both High Level Petri Boxes of Figure 5.1. Then (with a suitably permissive label algebra) abstract transition t'_b is enabled. However, abstract transition t_b is not enabled from this marking in any label algebra, as firing t_b requires being able to 'pop' from the token $(.)$ the context $(.f)$. It will be noted that such an intermediate marking is not reachable from any standard initial marking.

   Moreover, when B_1 and B_2 are not syntactically generated, the proposed rewrite does not necessarily produce equivalence of behaviours, even from standard initial markings. A situation in which this occurs is illustrated in Figure 5.2.

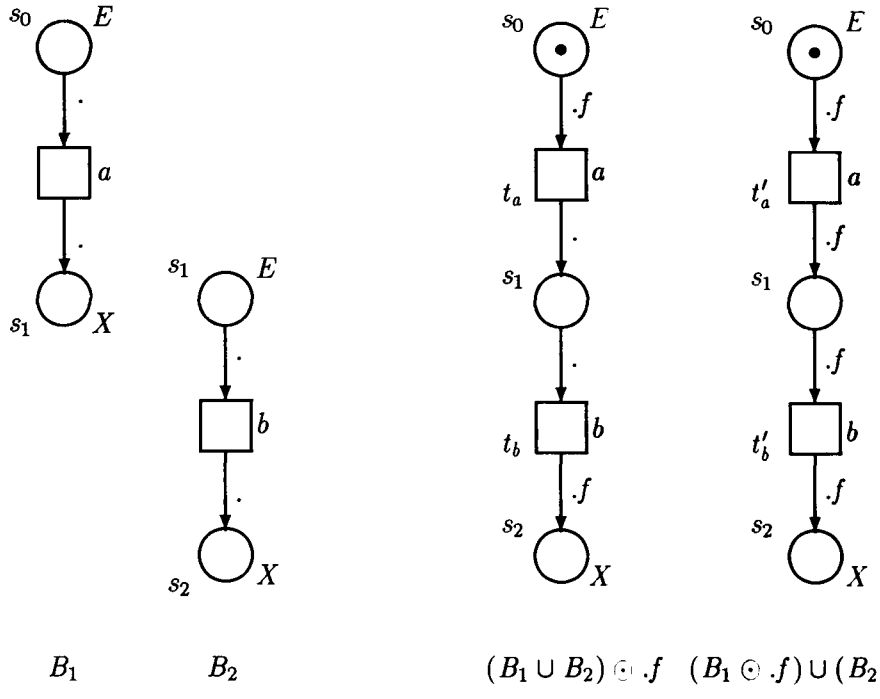


Figure 5.1: Demonstrating the necessity of a behavioural equivalence for the postulated rewrite rule β.

> The main result of this chapter is the development of properties of syntactically generated High Level Petri Boxes which leads to the formalisation and generalisation of the behavioural equivalence underlying this rewrite (subsequently called β).

It is notable that this leap in complexity arises only in the use of union for the expression of causal composition of two High Level Petri Boxes which is the sequential composition of two 'blocks'. Intuitively then, through the conjectured rewrite we are requiring that the block structure and sequential composition commute; in terms of the block structure of an *Algol 68*-type language this might be interpreted as forcing

begin		*begin*	
(1)		(1)	
end	and		to be equivalent.
begin			
(2)		(2)	
end		*end*	

As a rewrite rule we will prefer to apply context manipulations before union, so that we

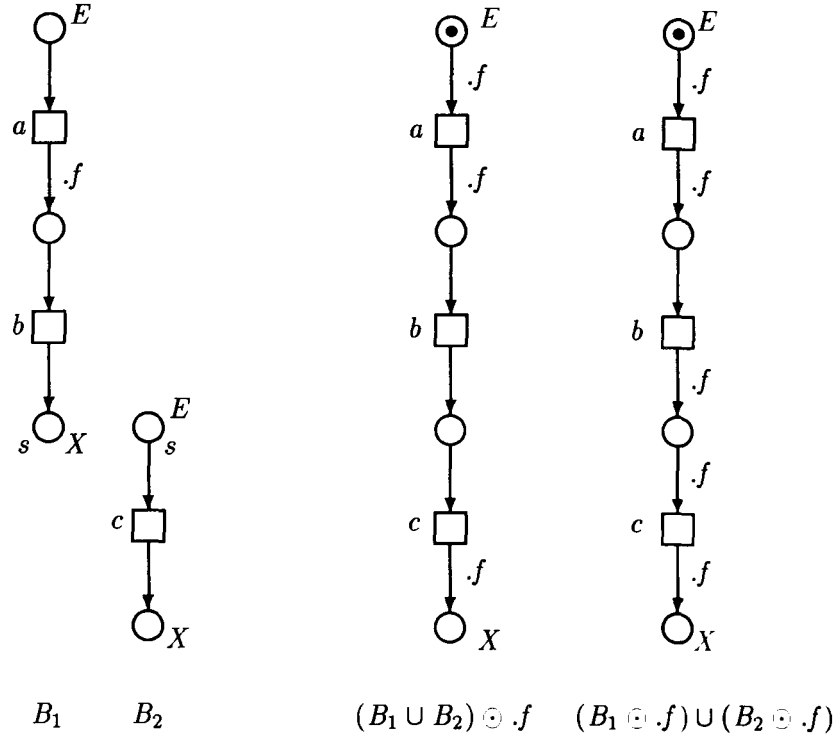


Figure 5.2: In which rewrite β does not produce a behavioural equivalence, $(B_1 \cup B_2) \odot .f \not\equiv_{Reach} (B_1 \odot .f) \cup (B_2 \odot .f)$ from the standard initial marking (in a suitable label algebra). The transition labelled b of $(B_1 \cup B_2) \odot .f$ may eventually fire, whereas that of $(B_1 \odot .f) \cup (B_2 \odot .f)$ will never be enabled.

read the equivalence as a rewrite rule in the left to right direction, so that our rewrite is $(t_1 \cup t_2) \odot .f \rightarrow (t_1 \odot .f) \cup (t_2 \odot .f)$.

3. The table entry represents the commuting of combinational closure and union, i.e., we are comparing $(B_1 \cup B_2)^{\oplus}$ and ${B_1}^{\oplus} \cup {B_2}^{\oplus}$, for B_1 and B_2 unionable High Level Petri Boxes. As the following example shows, we may distinguish the semantic High Level Petri Boxes of these terms both structurally and behaviourally, *even from standard initial markings*, so that we will not be considering this as a rewrite rule.

   For the High Level Petri Boxes B_1 and B_2 of Figure 5.3 (in the label algebra $\mathcal{L}_0$ of Example 2.1.1) we see the relative portions of $(B_1 \odot .\vdash \cup B_2 \odot .\dashv)^{\oplus}$ and its rewrite under the proposed rewrite $(B_1 \odot .\vdash)^{\oplus} \cup (B_2 \odot .\dashv)^{\oplus}$. We see that the left-hand side contains an abstract transition labelled $a \oplus \overline{a}$ which is missing from the right-hand side. Moreover, given a standard initial marking of $(.ccs)$ (also illustrated) this transition is enabled and

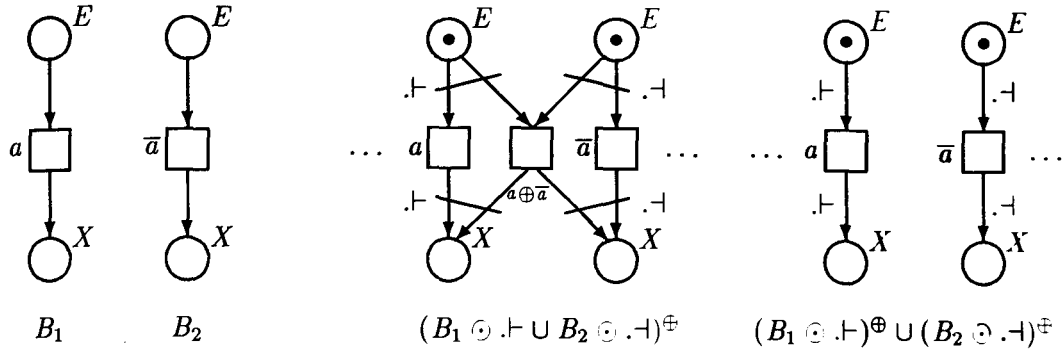


Figure 5.3: The High Level Petri Boxes $(B_1 \odot .\vdash \cup B_2 \odot .\dashv)^{\oplus}$ and $(B_1 \odot .\vdash^{\oplus}) \cup (B_2 \odot .\dashv^{\oplus})$ are both structurally and behaviourally distinguishable, even from a standard initial marking.

so may fire.

4. The table entry represents the commutativity of context manipulation and relational lifting, i.e., we are comparing $f(B) \odot C$ and $f(B \odot C)$, for f a sane relation and C a context. From the definitions, $f(B) \odot C = f(B \odot C)$, and behavioural equivalence follows immediately. We will use this rule as a rewrite in the left to right direction, so that context manipulations appear within relational liftings in rewritten terms.

5. The table entry represents the commuting of combinational closure and relational lifting, i.e., we are comparing $f(B)^{\oplus}$ and $f(B^{\oplus})$, for f a sane relation. From the definitions we have structural equality. We will use this rule as a rewrite in the right to left direction, so that relational liftings appear within combinational closures in rewritten terms.

6. The table entry represents the commuting of combinational closure and context manipulation, i.e., we are comparing $(B \odot C)^{\oplus}$ and $B^{\oplus} \odot C$, for C a context. From the definitions we again have structural equality. We will use this rule as a rewrite in the right to left direction, so that context manipulations appear within combinational closures in rewritten terms.

The form of the rules now changes, from showing the commutativity of the application of auxiliaries to that of the iterated application of an auxiliary operator:

7. The table entry represents the twice iterated application of union, which is more usually called associativity, i.e., $(B_1 \cup B_2) \cup B_3$ and $B_1 \cup (B_2 \cup B_3)$. It is not difficult to show (and follows from a purely technical manipulation of the definition of the place labelling of a

union) that when both sides are defined they are equal. Moreover, from Theorem 4.8.4 it is not difficult to see that for the syntactically generated High Level Petri Boxes considered in this chapter the two sides are always defined.

Associativity is not usually considered as a rewrite rule *per se*, rather it will justify the removal of parenthesis[3].

8. The table entry represents the twice iterated application of relational lifting. From the definitions it follows that $f_1(f_2(B)) = (f_1 \circ f_2)(B)$, where f_1 and f_2 are sane relations. We will use this rule as a rewrite in the left to right direction, preferring to perform the composition of relations before application.

9. The table entry represents the twice iterated application of context manipulation. Again $B \odot C \odot D = B \odot CD$ follows directly from the definitions. We will use this rule as a rewrite in the left to right direction, so that we prefer to concatenate contexts before a context manipulation.

10. The table entry represents the twice iterated application of combinational closure. That $B^{\oplus\oplus} = B^{\oplus}$ follows from the definitions, through the properties of multiset closure. We will use this rule as a rewrite in the left to right direction, so that consecutive applications of combinational closure may be removed.

5.2.2 Derived Rewrite Rules

By collecting together the rewrite rules derived from the above equivalences we may define the following rewrite system[4]:

DEFINITION 5.2.1 Given a High Level Petri Box term t, we say that t' is an *auxiliary rewrite* of t when there is a term t_{aux} such that t_{aux} is the auxiliary representation of t and t' derives from t_{aux} by application of the following rewrite rules:

(α)	$f(t_1 \cup t_2)$	$\rightarrow$	$f(t_1) \cup f(t_2)$	(β)	$(t_1 \cup t_2) \odot C$	$\rightarrow$	$t_1 \odot C \cup t_2 \odot C$
(γ)	$f(t) \odot C$	$\rightarrow$	$f(t \odot C)$	(δ)	$f(t^{\oplus})$	$\rightarrow$	$f(t)^{\oplus}$
(ϵ)	$t^{\oplus} \odot C$	$\rightarrow$	$(t \odot C)^{\oplus}$	(ζ)	$f_1(f_2(t))$	$\rightarrow$	$(f_1 \circ f_2)(t)$
(η)	$t \odot C \odot D$	$\rightarrow$	$t \odot CD$	(θ)	$t^{\oplus\oplus}$	$\rightarrow$	$t^{\oplus}$

[3] This is not sufficient to produce a unique normal form: we thus stipulate that union associates to the right, so that $\_ \cup \_ \cup \_ = \_ \cup (\_ \cup \_)$.

[4] Dependent on the justification of Item 2(b) of Section 5.2.1.

The rewriting of a High Level Petri Box term t to its auxiliary representation t_{aux} will be written $t \rightarrow_{aux} t_{aux}$; that term t' is the result of rewriting term t under some rule ϖ will be denoted $t \rightarrow_{\varpi} t'$ or $t \rightarrow t'$ when we do not wish to specify the rule name. That there are terms $t_1, \ldots, t_n$ such that $t = t_1 \rightarrow t_2 \rightarrow \cdots \rightarrow t_n = t'$ will be denoted $t \Rightarrow t'$. ■ 5.2.1

THEOREM 5.2.2 The rewrite system of Definition 5.2.1 is both terminating and confluent. □ 5.2.2

Proof: The proof consists of two parts: showing *local confluence* and showing *termination*. However, that the system terminates is clear: each of rules α, β, γ, δ and ϵ reduces the number of 'outer' operator applications over 'inner' ones, e.g., α reduces the number of relational liftings applied to unions by one. Rules ζ, η and θ reduce the number of relational liftings, context manipulations and combinational closure, respectively, by one. If we define the *complexity of a term* as the sum of these metrics, then the complexity is reduced by the application of each rewrite. Moreover, any finite term has a finite complexity. Hence, we can define a well-ordering on rewritten terms, and the rewrite system is terminating.

The proof of local confluence is standard. We list the *critical pairs* of the rewrite system—those terms to which apply two rewrites in two different ways—and show that, through the rewrite system, these pairs can be rewritten to the same term. For a full description of the technique we refer the reader to [KB70].

As an example of a critical pair for the rewrite system, rewrites α and γ apply to overlapping, non-trivial subterms of $f(t_1 \cup t_2) \odot C$. To show confluence, we must show that such critical pairs *conflate*, i.e., can be rewritten to the same term. Now:

Critical Pair: (α, γ)

$$
\begin{array}{rcccl}
(f(t_1) \cup f(t_2)) \odot C & \alpha\leftarrow & f(t_1 \cup t_2) \odot C & \rightarrow_{\gamma} & f((t_1 \cup t_2) \odot C) \\
f(t_1) \odot C \cup f(t_2) \odot C & \beta\leftarrow & & \rightarrow_{\beta} & f((t_1 \odot C) \cup (t_2 \odot C)) \\
f(t_1 \odot C) \cup f(t_2) \odot C & \gamma\leftarrow & & & \\
 & \rightarrow_{\gamma} & f(t_1 \odot C) \cup f(t_2 \odot C) & \alpha\leftarrow &
\end{array}
$$

and we have the result for this case. The demonstration of the confluence of all possible critical pairs is given in Table 5.2. As all critical pairs conflate, we have the result.

■ 5.2.3

Critical Pair: (α, γ)

$$
\begin{array}{rcccl}
(f(t_1) \cup f(t_2)) \odot C & \alpha\leftarrow & f(t_1 \cup t_2) \odot C & \rightarrow_\gamma & f((t_1 \cup t_2) \odot C) \\
f(t_1) \odot C \cup f(t_2) \odot C & \beta\leftarrow & & \rightarrow_\beta & f(t_1 \odot C) \cup (t_2 \odot C)) \\
 & \rightarrow_\gamma & f(t_1 \odot C) \cup f(t_2 \odot C) & \alpha\leftarrow &
\end{array}
$$

Critical Pair: (α, ζ)

$$
\begin{array}{rcccl}
f_1(f_2(t_1) \cup f_2(t_2)) & \alpha\leftarrow & f_1(f_2(t_1 \cup t_2)) & \rightarrow_\zeta & (f_1 \circ f_2)(t_1 \cup t_2) \\
f_1(f_2(t_1)) \cup f_1(f_2(t_2)) & \alpha\leftarrow & & & \\
(f_1 \circ f_2)(t_1) \cup f_1(f_2(t_2)) & \zeta\leftarrow & & & \\
 & \rightarrow_\zeta & (f_1 \circ f_2)(t_1) \cup (f_1 \circ f_2)(t_2) & \alpha\leftarrow &
\end{array}
$$

Critical Pair: (β, η)

$$
\begin{array}{rcccl}
((t_1 \odot C) \cup (t_2 \odot C)) \odot D & \beta\leftarrow & (t_1 \cup t_2) \odot C \odot D & \rightarrow_\eta & (t_1 \cup t_2) \odot CD \\
(t_1 \odot C \odot D) \cup (t_2 \odot C \odot D) & \beta\leftarrow & & & \\
(t_1 \odot CD) \cup (t_2 \odot C \odot D) & \eta\leftarrow & & & \\
 & \rightarrow_\eta & (t_1 \odot CD) \cup (t_2 \odot CD) & \beta\leftarrow &
\end{array}
$$

Critical Pair: (γ, δ)

$$
\begin{array}{rcccl}
f(t^\oplus \odot C) & \gamma\leftarrow & f(t^\oplus) \odot C & \rightarrow_\delta & f(t)^\oplus \odot C \\
f((t \odot C)^\oplus) & \epsilon\leftarrow & & \rightarrow_\epsilon & (f(t) \odot C)^\oplus \\
 & \rightarrow_\delta & f(t \odot C)^\oplus & \gamma\leftarrow &
\end{array}
$$

Critical Pair: (γ, η)

$$
\begin{array}{rcccl}
f(t) \odot CD & \eta\leftarrow & f(t) \odot C \odot D & \rightarrow_\gamma & f(t \odot C) \odot D \\
 & & & \rightarrow_\gamma & f(t \odot C \odot D) \\
 & \rightarrow_\gamma & f(t \odot CD) & \eta\leftarrow &
\end{array}
$$

Critical Pair: (γ, ζ)

$$
\begin{array}{rcccl}
f_1(f_2(t) \odot C) & \gamma\leftarrow & f_1(f_2(t)) \odot C & \rightarrow_\zeta & (f_1 \circ f_2)(t) \odot C \\
f_1(f_2(t \odot C)) & \gamma\leftarrow & & & \\
 & \rightarrow_\zeta & (f_1 \circ f_2)(t \odot C) & \gamma\leftarrow &
\end{array}
$$

Critical Pair: (δ, ζ)

$$
\begin{array}{rcccl}
f_1(f_2(t)^\oplus) & \delta\leftarrow & f_1(f_2(t^\oplus)) & \rightarrow_\zeta & (f_1 \circ f_2)(t^\oplus) \\
f_1(f_2(t))^\oplus & \delta\leftarrow & & & \\
 & \rightarrow_\zeta & (f_1 \circ f_2)(t)^\oplus & \delta\leftarrow &
\end{array}
$$

Critical Pair: (δ, θ)

$$
\begin{array}{rcccl}
f(t^\oplus)^\oplus & \delta\leftarrow & f(t^{\oplus\oplus}) & \rightarrow_\theta & f(t^\oplus) \\
f(t)^{\oplus\oplus} & \delta\leftarrow & & & \\
 & \rightarrow_\theta & f(t)^\oplus & \delta\leftarrow &
\end{array}
$$

Critical Pair: (ϵ, η)

$$
\begin{array}{rcccl}
(t \odot C)^\oplus \odot D & \epsilon\leftarrow & (t^\oplus) \odot C \odot D & \rightarrow_\eta & (t^\oplus) \odot CD \\
(t \odot C \odot D)^\oplus & \epsilon\leftarrow & & & \\
 & \rightarrow_\eta & (t \odot CD)^\oplus & \epsilon\leftarrow &
\end{array}
$$

Critical Pair: (ϵ, θ)

$$
\begin{array}{rcccl}
(t^\oplus \odot C)^\oplus & \epsilon\leftarrow & t^{\oplus\oplus} \odot C & \rightarrow_\theta & t^\oplus \odot C \\
(t \odot C)^{\oplus\oplus} & \epsilon\leftarrow & & & \\
 & \rightarrow_\theta & (t \odot C)^\oplus & \epsilon\leftarrow &
\end{array}
$$

Table 5.2: Proof of conflation of Critical Pairs of the rewrite system for High Level Petri Boxes

COROLLARY 5.2.3 For each High Level Petri Box term t there is a unique term t^{nf}, called the *normal form* for t, that is the result of an exhaustive application of the rewrite rules of Definition 5.2.1. □ 5.2.3

From an inspection of the case of the proofs for critical pairs (α, γ) and (δ, γ), it is interesting to note that:

COROLLARY 5.2.4 A rewrite system consisting only of the rules γ, ζ, η and θ is confluent and terminating. If

1. α is included, then β is required for conflation,
2. δ is included, then ϵ is required for conflation. □ 5.2.4

and

COROLLARY 5.2.5 No proper sub-system Definition 5.2.1 containing rules α, δ, γ, ζ, η and θ is confluent and terminating. □ 5.2.5

The consistency of rewrite β thus has increased importance for the normal form.

A confluent and terminating rewrite system induces a congruence, [KB70]:

DEFINITION 5.2.6 Define $t_1 \equiv_{\rightarrow} t_2$ when $t_1^{nf} = t_2^{nf}$. ■ 5.2.6

As usual, $[t] = \{t' \mid t \equiv_{\rightarrow} t'\}$.

Given the behavioural consistency of the rewrite β, we have the following important result:

THEOREM 5.2.7 $[t_1] = [t_2]$ implies $HLPB(t_1) \equiv_{Reach} HLPB(t_2)$. □ 5.2.7

Proof: The proof is immediate, given the following lemma:

LEMMA 5.2.8 $[t_1] = [t_2]$ implies that

1. $[t_1 \odot C] = [t_2 \odot C]$,
2. $[f(t_1)] = [f(t_2)]$,
3. $[t_1^{\oplus}] = [t_2^{\oplus}]$,

4. $[t_1 \cup t] = [t_2 \cup t]$. □ 5.2.8

Proof: From the confluence of the rewrite system, $op(t_1) \Rightarrow op(t_1^{nf})$ and $op(t_2) \Rightarrow op(t_2^{nf})$, where op is one of the operator applications above. $[t_1] = [t_2]$ implies $t_1^{nf} = t_2^{nf}$ so that $op(t_1)^{nf} = op(t_2)^{nf}$, and we have the result. ■ 5.2.8

■ 5.2.7

Unfortunately, although we have that t and t^{nf} have the same behaviour, many of the interesting behavioural equivalences can not be established through the rewrite system of Definition 5.2.1. One such instance is the behavioural commutativity of concurrent composition, as we do not have rewrite rules transforming $t_1 \odot .\vdash \cup\, t_2 \odot .\dashv$ into $t_1 \odot .\dashv \cup\, t_2 \odot .\vdash$. Such an extension is not trivial: we will discuss the extension further after a short discussion of the properties of terms and High Level Petri Boxes in normal form, and the completion of the justification of the current rewriting system.

5.2.3 Flat Terms and High Level Petri Boxes

The structure of terms in normal form is distinctive:

DEFINITION 5.2.9 A High Level Petri Box term in normal form will be called *flat*. If t^{nf} is the normal form of t then t^{nf} is said to be the *flattening* of t. ■ 5.2.9

Flat is chosen as descriptive of the normal form representation: all context manipulations, relational liftings and combinational closures are applied only once, and the binary tree structure of iterated unions has been flattened. From the (as yet not fully proven) behaviour preserving properties of rewriting, the denotation of t and its flattening will have the same behaviour from all standard initial markings. Figure 5.5 shows the 'flattening' of the High Level Petri Box term $(a;(b\|c))[f]$. Figure 5.4 illustrates the denotation of the same term and of its flattening.

From flat terms come *flat High Level Petri Boxes*:

DEFINITION 5.2.10 A *flat High Level Petri Box* is a High Level Petri Box which is the denotation of a flat High Level Petri Box term. ■ 5.2.10

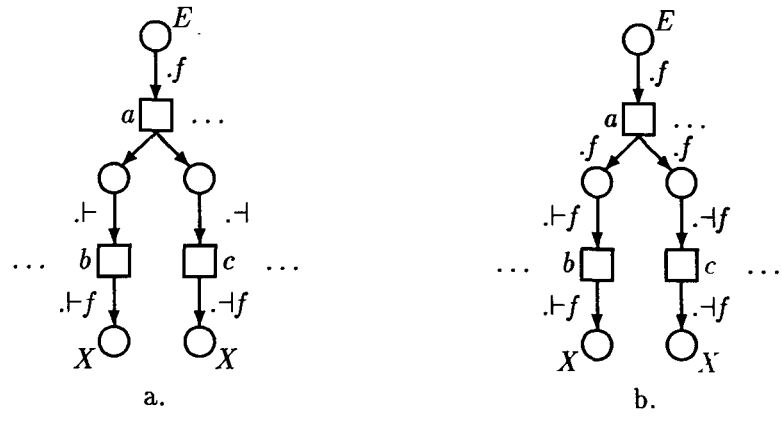


Figure 5.4: The High Level Petri Box of Figure 5.5: (a.) original and (b.) flattened.

5.2.3.1 Behaviours of Flat High Level Petri Boxes

The structure of a flat High Level Petri Box B, over some a algebra $\mathcal{L} = \langle A, R \rangle$, is markedly simpler than that of a general High Level Petri Box. In a flat High Level Petri Box all context manipulations have been applied directly to local transitions, which are thus *balanced*, i.e., have equal pre- and post-contexts. This balancing implies that the only token which populates a marking reachable from a standard initial marking d is d. In terms of the PrT net denotation this has the corollary that the feasible substitutions of a flat High Level Petri Box are a subset of the substitutions $\{index(t) \rightarrow d \mid d \in C\}$ and, in particular, those from the standard initial marking d are a subset of the singleton set $\{index(t) \rightarrow d\}$.

If $S^d = \{\hat{s}d \mid s \in S_B\}$ then, in terms of the A-labelled P/T nets[5], we may characterise the subnet of $P/T(B) = \langle S, T \rangle$ corresponding to the behaviour from the standard initial marking d as that subnet $B^d = \langle S^d, T^d \rangle$, where $T^d = T \cap (\mathbb{P}S^d \times A \times \mathbb{P}S^d)$.

5.2.3.2 A Low Level Box Model based on Flat High Level Petri Boxes?

The normal form also provides us with the possibility of creating a compositional semantics of (the current portion of) the High Level Petri Box Algebra based solely on flat High Level Petri Boxes. Replacing the current definition of context manipulation with the following 'flat' version:

[5]Where A is the label set of the label algebra.

$(a;(b\|\|c))[f]$

$\rightarrow_{aux}$ $(;_1^{(a,b\|\|c)}(a) \cup \underline{;_2^{(a,b\|\|c)}(b \odot .\vdash \cup c \odot .\dashv)})^{\oplus} \odot .f$

$\rightarrow_{\alpha}$ $\underline{(;_1^{(a,b\|\|c)}(a) \cup (;_2^{(a,b\|\|c)}(b \odot .\vdash) \cup ;_2^{(a,b\|\|c)}(c \odot .\dashv)))^{\oplus} \odot .f}$

$\rightarrow_{\epsilon}$ $\underline{(;_1^{(a,b\|\|c)}(a) \cup (;_2^{(a,b\|\|c)}(b \odot .\vdash) \cup ;_2^{(a,b\|\|c)}(c \odot .\dashv))) \odot .f}^{\oplus}$

$\rightarrow_{\beta}$ $(;_1^{(a,b\|\|c)}(a) \odot .f \cup \underline{(;_2^{(a,b\|\|c)}(b \odot .\vdash) \cup ;_2^{(a,b\|\|c)}(c \odot .\dashv)) \odot .f})^{\oplus}$

$\rightarrow_{\beta}$ $(\underline{;_1^{(a,b\|\|c)}(a) \odot .f} \cup (;_2^{(a,b\|\|c)}(b \odot .\vdash) \odot .f \cup ;_2^{(a,b\|\|c)}(c \odot .\dashv) \odot .f))^{\oplus}$

$\rightarrow_{\gamma}$ $(;_1^{(a,b\|\|c)}(a \odot .f) \cup (\underline{;_2^{(a,b\|\|c)}(b \odot .\vdash) \odot .f} \cup ;_2^{(a,b\|\|c)}(c \odot .\dashv) \odot .f))^{\oplus}$

$\rightarrow_{\gamma}$ $(;_1^{(a,b\|\|c)}(a \odot .f) \cup (;_2^{(a,b\|\|c)}(b \odot .\vdash \odot .f) \cup \underline{;_2^{(a,b\|\|c)}(c \odot .\dashv) \odot .f}))^{\oplus}$

$\rightarrow_{\gamma}$ $(;_1^{(a,b\|\|c)}(a \odot .f) \cup (;_2^{(a,b\|\|c)}(\underline{b \odot .\vdash \odot .f}) \cup ;_2^{(a,b\|\|c)}(c \odot .\dashv \odot .f)))^{\oplus}$

$\rightarrow_{\eta}$ $(;_1^{(a,b\|\|c)}(a \odot .f) \cup (;_2^{(a,b\|\|c)}(b \odot .\vdash f) \cup ;_2^{(a,b\|\|c)}(\underline{c \odot .\dashv \odot .f})))^{\oplus}$

$\rightarrow_{\eta}$ $(;_1^{(a,b\|\|c)}(a \odot .f) \cup (;_2^{(a,b\|\|c)}(b \odot .\vdash f) \cup ;_2^{(a,b\|\|c)}(c \odot .\dashv f)))^{\oplus}$

Figure 5.5: The flattening of the term $(a;(b\|\|c))[f]$. To aid the reader, in the application of the rewrite rules, we indicate the scope of rewrite by an underscore, with dots indicating the operators to which the rewrite applies.

DEFINITION 5.2.11 For B a High Level Petri Box and C, $C' \in \mathcal{C}$ define the *flat context manipulation*, $B \,\overline{\odot}\, (C, C') = \langle S, T', \lambda \rangle$ with $T' = \{t(C, C') | t \in T\}$ and

$$\begin{aligned} t(C, C') &= \{l(C, C') \mid l \in t\} \\ l(C, C') &= \langle {}^\bullet l, {}^C l \frown C, \overline{l}, l^C \frown C', l^\bullet \rangle \end{aligned}$$

■ 5.2.11

is all that would be required to implement this change.

Moreover, as arc annotations are balanced, we no longer need to allow two for each local transition, and so we might consider applying them directly to the label of a local transition. Without arc annotations, and with single tokens populating reachable markings, a low level net model would be possible[6].

However, contexts provide more than just the way of manipulating single labels—through the embedded nominal relabellings, they also provide important information as to how local transitions may be combined into synchronisations. Because of the 'run-time' nature of the enabling of a transition, the simple firing rule of low level net models appears to preclude a representation based on local transitions containing such information.

Why, then, do we base our presentation on the concept of a local transition? We will argue in Chapter 6 that the added value of local transitions is more than being just a convenient way of forming synchronisations: in Chapter 6 we show that, without complicating the High Level Petri Box model, local transitions can provide a basis for the *free construction* on the semantic domain of High Level Petri Boxes, facilitating the provision of process variables, a powerful refinement operation and, derived from it, a notion of recursion.

5.3 Characterising the Structural Relationship induced by β

Using the definitions of union and context manipulation, it is straightforward to formally characterise the structural relationship between the High Level Petri Boxes $(B_1 \odot .f) \cup (B_2 \odot .f)$ and $(B_1 \cup B_2) \odot .f$, as were illustrated in Figure 5.1 (page 111), and to lift this structural characterisation to all behaviours from standard initial markings.

Throughout the remainder of this section we will assume that B_1 and B_2 are syntactically generated High Level Petri Boxes such that $B_1^\bullet = {}^\bullet B_2$, and otherwise sociable. As the reader will recall, this was the only non-trivial case arising in the above discussion. For the remainder of this section we will abbreviate $(B_1 \odot .f) \cup (B_2 \odot .f)$ to B, and $(B_1 \cup B_2) \odot .f$ to $\widehat{B}$.

LEMMA 5.3.1 For all $M \in \mathcal{M}_d^{\widehat{B}}$, $s \in B_1^\bullet$, and for all $e \in M(s)$, $e = (.f)d$. □ 5.3.1

The proof of Lemma 5.3.1 is provided in the next section. It proceeds through the identification of a partial order on the local transitions of a syntactically generated High Level Petri Box. The relation upon which the partial order is defined is that of *local causality* which is, essentially,

[6] And would look very much like that of the low level Petri Box Calculus, although without recursion and iteration.

the interrelation between local transitions as given by the intersection of their pre- and post-sets. We go on to characterise the maximal lines of this partial order in terms of the syntactic structure of the High Level Petri Box and show that the contexts forming the annotations of local transitions may 'migrate' along maximal lines whilst preserving certain behaviours. Due to the careful choice of semantics[7] maximal lines of the defined partial order characterise the 'paths' of tokens through a (syntactically generated) High Level Petri Box from entry to exit places, and so many aspects of such a High Level Petri Box's behaviour.

The close relationship between B and $\widehat{B}$ is characterised by the following proposition:

PROPOSITION 5.3.2 $\quad \widehat{B} \equiv^{d}_{Reach} B.$ □ 5.3.2

Proof: We show that $\widehat{B}$ and B satisfy the hypotheses of Proposition 3.6.2 (page 62), whence the result.

That $S_{\widehat{B}} = S_B$ is clear from the definitions.

Define $\xi: LT(\widehat{B}) \to LT(B)$ such that

$$\xi(l) = \begin{cases} l & {}^\bullet l^\bullet \cap B_1{}^\bullet = \emptyset \\ \langle {}^\bullet l,\ {}^C l \quad ,\bar{l},\ l^C(.f)\ , l^\bullet \rangle & l^\bullet \subseteq B_1{}^\bullet (= {}^\bullet B_2) \\ \langle {}^\bullet l,\ {}^C l(.f)\ , \bar{l},\ l^C \quad , l^\bullet \rangle & {}^\bullet l \subseteq {}^\bullet B_2 \end{cases}$$

Then $\xi: LT(B_1) \equiv LT(B_2)$ such that ${}^\bullet\xi(l) = {}^\bullet l$, $\overline{\xi(l)} = \bar{l}$ and $\xi(l)^\bullet = l^\bullet$. Moreover, ξ lifts to a bijective correspondence between $T_{\widehat{B}}$ and T_B. As $\xi: T_{\widehat{B}} \equiv T_B$ for convenience we will assume that $index(t) = index(\xi(t))$, and that, for $l_i \in t_i$, x_i annotates arcs (s,t) and $(s,\xi(t))$ with $s \in {}^\bullet l_i = {}^\bullet\xi(l_i)$, and y_i annotates arcs (t,s) and $(\xi(t),s)$ with $s \in l_i{}^\bullet = \xi(l_i)^\bullet$.

For each $s \in S_{\widehat{B}}$, define $\xi_s: C_{\mathcal{L}} \to C_{\mathcal{L}}$ such that

$$\xi_s(e) = \begin{cases} e - (.f) & s \in B_1{}^\bullet (= {}^\bullet B_2) \\ e & \text{otherwise} \end{cases}$$

Note that $s \in S_B \setminus B_1{}^\bullet$ implies $\xi_s = \mathrm{id}$, whereas $s \in B_1{}^\bullet$ implies $\xi_s: C_{\mathcal{L}} \frown (.f) \equiv C_{\mathcal{L}}$.

Define $\Xi: \mathcal{M}_d^{\widehat{B}} \equiv \mathcal{M}_d^B$ such that $\Xi(M) = \{s \to \xi_s(e) \mid s - e \in M\}$.

We show that Ξ satisfies Properties 3(a)—3(c) of Proposition 3.6.2, whence the result:

[7] In particular, the denotation of stop given in Definition 4.7.1.

1. $\Xi(M_d^{I,B_1}) = M_d^{I,B_2}$:

$$\begin{aligned}\Xi(M_d^{I,B_1}) &= \{s \to \xi_s(d) \mid s \in {}^\bullet\widehat{B}\} \\ &= \{s \to d \mid s \in {}^\bullet B\} \\ &= M_d^{I,B_2}\end{aligned}$$

2. Suppose M, $M' \in \mathcal{M}_d^{\widehat{B}}$, $t \in T_{\widehat{B}}$, $\alpha \in \mathrm{subs}^{\widehat{B}}(t)$ such that $M \left[\begin{smallmatrix} a \\ t:\alpha \end{smallmatrix}\right\rangle M'$ in $\widehat{B}$.
Define $\gamma(x_i) = \xi_s(\alpha(x_i))$ for $s \in {}^\bullet l_i$, and $\gamma(y_i) = \xi_s(\alpha(y_i))$ for $s \in l_i^\bullet$. That γ as defined is unique is clear. Moreover, from the definitions, $\overline{\xi(t):\gamma} = \overline{t:\alpha}$. Let N be such that $\Xi(M) \left[\begin{smallmatrix} a \\ \Xi(t):\gamma \end{smallmatrix}\right\rangle N$. We claim $N = \Xi(M')$.
Now, for $s \in S_{\widehat{B}}$:

$$\begin{aligned}N(s) &= (\Xi(M)(s) \setminus W_F(s, \xi(t)):\gamma) \cup W_F(\xi(t), s):\gamma \\ &= (\xi_s(M(s)) \setminus \xi_s(W_F(s,t):\alpha)) \cup \xi_s(W_F(t,s):\alpha) \\ &= \xi_s((M(s) \setminus W_F(s,t):\alpha) \cup W_F(t,s):\alpha) \\ &= \Xi(M')(s)\end{aligned}$$

as required.

3. establishing Property 3(c) is similar.

Hence the result. ■ 5.3.2

5.4 A Partial Order on the Structure of a High Level Petri Box

Throughout this section we will assume that all label algebra relative properties are defined with respect to the label algebra $\mathcal{L} = \langle A, R \rangle$, and that B is a syntactically generated High Level Petri Box over $\mathcal{L}$.

We first recall the definition of a *partial order*.

5.4.1 Partial Orders

DEFINITION 5.4.1 A *partial order* is a pair $\langle S, \preceq \rangle$ consisting of a set S, the *carrier set*, and a transitive, reflexive, antisymmetric relation $\preceq$ on $S \times S$, the *partial order (relation)*. ■ 5.4.1

We will assume that the partial order $\langle S, \preceq \rangle$ is discrete, so that we may find a smallest contained relation, the *generator* $\prec$, such that $\preceq$ is the transitive reflexive closure of $\prec$. We will write $s \preceq s'$ to denote that s and s' are related under $\preceq$, i.e., $(s, s') \in \preceq$. As $\preceq$ is reflexive, for all $s \in S$,

$s \preceq s$. s and s' are said to be *comparable under* $\preceq$ (or just *comparable*) when either $s \preceq s'$ or $s' \preceq s$ (or $s = s'$). We use $s \# s'$ to denote that s and s' are not comparable.

A maximal element is 'greater than or equal to' every element to which it is comparable. A minimal element is 'less than or equal to' every element to which it is comparable:

DEFINITION 5.4.2 Let $\Phi = \langle S, \preceq \rangle$ be a partial order. We say that $m \in S$ is *maximal in* Φ when $\forall n \in S: n \# m \vee n \preceq m$. We say that $m \in S$ is *minimal in* Φ when $\forall n \in S: n \# m \vee m \preceq n$. ■ 5.4.2

A total order is a partial order in which each pair of elements is comparable:

DEFINITION 5.4.3 Let $\Phi = \langle S, \preceq \rangle$ be a partial order. Then Φ is a *total order* if $\forall s, s' \in S: \neg s \# s'$. In this case $\preceq$ may also be called a *total order*. ■ 5.4.3

Within a partial order there are distinguished subsets which form total orders:

DEFINITION 5.4.4 Let $\Phi = \langle S, \preceq \rangle$ be a partial order. A *line* is a set $M \subseteq S$ such that $\langle M, \preceq \lfloor_{M \times M} \rangle$ is a total order. A *maximal line* is a line, K, such that for all $m \in S \setminus K$, $K \cup \{m\}$ ceases to be a line, i.e., $\forall m \in S \setminus K: \exists n \in K: n \# m$. ■ 5.4.4

As an abuse of notation, we will often denote the partial order $\langle S, \preceq \rangle$ by $\preceq$ when S is clear by context and a line K of a partial order $\Phi = \langle S, \preceq \rangle$ by K when Φ is clear by context.

5.4.2 A Partial Order on Local Transitions

The dependence of one local transition on another given by a non-empty intersection of, respectively, the post- and pre-sets induces a relation on the set of local transitions which relates to the causal structure of the High Level Petri Box. In this section we define and develop the properties of this relation.

Throughout the remainder of this section we will assume that B, B_1 and B_2 are syntactically generated High Level Petri Boxes, with $l_1, l_2 \in LT(B)$.

DEFINITION 5.4.5 We say that l_1 *directly precedes* l_2 *in* $LT(B)$, written $l_1 \prec l_2$, when $l_1{}^\bullet \cap {}^\bullet l_2 \neq \emptyset$. We will refer to $\prec$ as the *local causality relation.* ■ 5.4.5

From the definition, for l a local transition ${}^\bullet l \neq \emptyset \neq l^\bullet$ whence $l_1 \prec l_2$ implies that the firing of an abstract transition in which l_1 is involved leaves tokens in the pre-set of l_2. Local causality, then, implies causality in the sense that if $l_i \in t_i$, for $i \in \{1,2\}$, then $l_1 \prec l_2$ only if t_2 is causally dependent on t_1.

We will prefer to work with the reflexive/transitive closure of local causality:

DEFINITION 5.4.6 We say that l_1 *precedes* l_2, denoted $l_1 \preceq l_2$, when $l_1(\prec)^* l_2$. ■ 5.4.6

DEFINITION 5.4.7 Define $\Phi(B) = \langle LT(B), \preceq \rangle$. ■ 5.4.7

The operators we allow in the construction of the High Level Petri Boxes of this chapter introduce no cyclic dependencies on local transitions. We might therefore expect the following:

THEOREM 5.4.8 $\Phi(B)$ is a finite partial order. □ 5.4.8

Proof: We develop a full characterisation of the partial order $\Phi(B)$ in terms of its decomposition into the application of auxiliary operations (with the proviso that any relation used in a relational lifting is sane).

Base Case: $HLPB(u)$, where $u \in A \cup \{\mathbf{0}\}$. Follows directly from the definition.

Inductive Step: Assume that $\Phi(B) = \langle LT(B), \preceq \rangle$, $\Phi(B_1) = \langle LT(B_1), \preceq_1 \rangle$ and $\Phi(B_2) = \langle LT(B_2), \preceq_2 \rangle$ are finite partial orders. We consider the application of each auxiliary operator to B, or, in the case of union, to B_1 and B_2:

$B^\oplus$: From the definition, $S_{B^\oplus} = S_B$ and $LT(B^\oplus) = LT(B)$. Suppose $\Phi(B^\oplus) = \langle LT(B), \prec_\oplus \rangle$. Then

$$\begin{aligned} l_1 \prec_\oplus l_2 &\Leftrightarrow l_1{}^\bullet \cap {}^\bullet l_2 \neq \emptyset \text{ in } B^\oplus \\ &\Leftrightarrow l_1{}^\bullet \cap {}^\bullet l_2 \neq \emptyset \text{ in } B \\ &\Leftrightarrow l_1 \prec l_2 \end{aligned}$$

But $\preceq_\oplus = (\prec_\oplus)^* = (\prec)^* = \preceq$, i.e., $\Phi(B^\oplus) = \Phi(B)$. The characterisation and result follow from the induction hypothesis.

$B \odot C$: From the definition, $S_{B \odot C} = S_B$ and $LT(B \odot C) = LT(B)(C)_B$. Suppose $\Phi(B \odot C) = \langle LT(B), \prec_{\odot} \rangle$. Then

$$\begin{aligned} l_1(C)_B \prec_{\odot} l_2(B)_C &\Leftrightarrow l_1(C)_B{}^{\bullet} \cap {}^{\bullet} l_2(C)_B \neq \emptyset \\ &\Leftrightarrow l_1{}^{\bullet} \cap {}^{\bullet} l_2 \neq \emptyset \\ &\Leftrightarrow l_1 \prec l_2 \end{aligned}$$

and the result follows as in the previous case.

$f(B)$ **for sane** f: as f is sane it preserves the intersection of pre- and post-sets, and hence local causality. From the definition, $S_{f(B)} = f(S_B)$ and $LT(f(B)) = f(LT(B))$. Suppose $\Phi(f(B)) = \langle LT(f(B)), \prec_f \rangle$. Then

$$\begin{aligned} f(l_1) \prec_f f(l_2) &\Leftrightarrow f(l_1)^{\bullet} \cap {}^{\bullet} f(l_2) \neq \emptyset \\ &\Leftrightarrow [f \text{ sane}] \\ &\quad l_1{}^{\bullet} \cap {}^{\bullet} l_2 \neq \emptyset \\ &\Leftrightarrow l_1 \prec l_2 \end{aligned}$$

and the result follows as in the previous cases.

(The reader will note that the sanity of f is required; otherwise, loops may be introduced into the High Level Petri Box destroying the finiteness of the partial order.)

$B_1 \cup B_2$: as B_1 and B_2 are unionable, either $B_1{}^{\bullet} \cap {}^{\bullet} B_2 = \emptyset$ or ${}^{\bullet} B_1 \cap B_2{}^{\bullet} = \emptyset$ (or both). Assume, without loss of generality, that ${}^{\bullet} B_1 \cap B_2{}^{\bullet} = \emptyset$. From the definition, $S_{B_1 \cup B_2} = S_{B_1} \cup S_{B_2}$, $LT(B_1 \cup B_2) = LT(B_1) \cup LT(B_2)$.

We first show that $\mathrm{ran}(\prec_2) \cap \mathrm{dom}(\prec_1) = \emptyset$. Suppose not, and let $l \in \mathrm{ran}(\prec_2) \cap \mathrm{dom}(\prec_1) \subseteq LT(B_1) \cap LT(B_2)$. Then we can find $l_1 \in LT(B_1)$ such that $l^{\bullet} \cap {}^{\bullet} l_1 \neq \emptyset$. But then $l^{\bullet} \subseteq B_1$ whence $l^{\bullet} \cap S_{B_2} = \emptyset$, from Property 2 of Definition 4.5.1 (page 73), contradicting that $l \in LT(B_1) \cap LT(B_2)$. Hence $\mathrm{ran}(\prec_2) \cap \mathrm{dom}(\prec_1) = \emptyset$, as required.

Suppose $\Phi(B_1 \cup B_2) = \langle LT(B_1 \cup B_2), \preceq_{\cup} \rangle$. We claim $\prec_{\cup} = \prec_1 \cup \prec_2 \cup \prec'$ where

$$\prec' = \{(l, l') \mid l \in LT(B_1) \wedge l' \in LT(B_2) \wedge \emptyset \neq l^{\bullet} \cap {}^{\bullet} l' \subseteq B_1{}^{\bullet} \cap {}^{\bullet} B_2\}$$

whence the result follows from the properties of $\prec_1$, $\prec_2$ and $\prec'$, and the fact that $\mathrm{ran}(\prec_2) \cap \mathrm{dom}(\prec_1) = \emptyset$.

For:

1. $\prec_{\cup} \supseteq \prec_1 \cup \prec_2 \cup \prec'$: immediate,
2. $\prec_{\cup} \subseteq \prec_1 \cup \prec_2 \cup \prec'$: Suppose l, $l' \in LT(B_1 \cup B_2)$ are such that $l \prec_{\cup} l'$. As $l^{\bullet} \cap {}^{\bullet} l' \neq \emptyset$, $l^{\bullet} \neq \emptyset \neq {}^{\bullet} l'$. We distinguish three cases (which are exhaustive

as B_1 and B_2 are High Level Petri Boxes) corresponding to the partitioning of $(B_1 \overset{\bullet}{\cup} B_2)$ by $\overset{\bullet}{B_1}$, $\overset{\bullet}{B_2}$, and $B_1{}^\bullet \cap {}^\bullet B_2$:

(a) $l^\bullet \cap {}^\bullet l' \subseteq \overset{\bullet}{B_1}$: then, as $LT(B_1) \cap LT(B_2) = \emptyset$, l, $l' \in LT(B_1) \setminus LT(B_2)$ so that $l \prec_1 l'$.

(b) $l^\bullet \cap {}^\bullet l' \subseteq \overset{\bullet}{B_2}$: by similar reasoning, we have that $l \prec_2 l'$

(c) $l^\bullet \cap {}^\bullet l' \subseteq B_1{}^\bullet \cap {}^\bullet B_2$: then $l_1 \in LT(B_1)$ and $l_2 \in LT(B_2)$ (for otherwise, if $l \in LT(B_2)$ as $l^\bullet \cap {}^\bullet B_2 \neq \emptyset$ contradicting Property 2 of Definition 2.5.15: $l' \in LT(B_1)$ is similar). But then $l \prec' l'$, and we have the result in this case.

Hence the claim.

The result follows as in the previous cases.

Returning to the case when B is syntactically generated; we may consider B as the repeated application of auxiliary operators (for which, if the auxiliary operator is relational lifting, then the relation is one of $;_1$, $;_2$, $+_1$ or $+_2$, and hence sane). Apply the above characterisation to give that $\Phi(B)$ is a partial order. ■ 5.4.8

5.4.2.1 Maximal Lines of $\Phi(B)$

We will characterise the maximal lines of the partial order on the local transitions of a syntactically generated High Level Petri Box in terms of the structure of the generating High Level Petri Box term.

DEFINITION 5.4.9 Define $\Psi(B) = \{K \mid K \text{ is a maximal line of } \Phi(B)\}$. ■ 5.4.9

We will use an abbreviated notation for the representation of maximal lines. For High Level Petri Boxes B_1 and B_2, with $K_1 = l_1 \prec_1 \ldots \prec_1 l_{n_1} \in \Psi(B_1)$, $K_2 = l'_1 \prec_2 \ldots \prec_2 l'_{n_2} \in \Psi(B_2)$, then $K_1 \cup K_2$ represents the maximal line of $B_1 \cup B_2$ consisting of local transitions $l_1 \prec_1 \cdots \prec_1 l_{n_1} \prec' l'_1 \prec_2 \cdots \prec_2 l'_{n_2}$ (where $\prec'$ is as defined in the proof of Theorem 5.4.8). $\Psi(B)(C)_B$ will represent the set $\{K(C)_B \mid K \in \Psi(B)\}$; $f(\Psi(B))$ will represent the set $\{f(K) \mid K \in \Psi(B)\}$ (which are, anyway, consequences of the definitional convention of Section 4.4.1).

From the proof of Theorem 5.4.8:

LEMMA 5.4.10

1. $\Psi(HLPB(u)) = \langle\{l\},\emptyset\rangle$ for $u \in A \cup \{\mathbf{0}\}$ and $HLPB(u) = [l]$,
2. $\Psi(B^{\oplus}) = \Psi(B)$,
3. $\Psi(B \odot C) = \Psi(B)(C)_B$ for a context C (where if $K = l_1 \prec \cdots \prec l_n$, then $K(C)_B = l_1(C)_B \prec \cdots \prec l_2(C)_B$),
4. $\Psi(f(B)) = f(\Psi(B))$ when f a sane relation (where if $K = l_1 \prec \cdots \prec l_n$, then $f(K) = f(l_1) \prec_f \cdots \prec_f f(l_n)$, for $\prec_f$ defined as in the proof of Theorem 5.4.8),
5. $\Psi(B_1 \cup B_2) = \begin{cases} \{K_1 \cup K_2 \mid K_i \in \Psi(B_i)\} & {}^\bullet B_1 = B_2{}^\bullet \vee B_1{}^\bullet = {}^\bullet B_2 \\ \Psi(B_1) \cup \Psi(B_2) & {}^\bullet B_1{}^\bullet \cap {}^\bullet B_2{}^\bullet = \emptyset \vee ({}^\bullet B_1 = {}^\bullet B_2 \wedge B_1{}^\bullet = B_2{}^\bullet) \end{cases}$

□ 5.4.10

Whence:

COROLLARY 5.4.11

1. $\Psi(HLPB(u)) = \langle\{l\},\emptyset\rangle$ for $u \in A \cup \{\mathbf{0}\}$ and $HLPB(u) = [l]$,
2. $\Psi(B_1 \| B_2) = (\Psi(B_1)(.\vdash)_{B_1}) \cup (\Psi(B_2)(.\dashv)_{B_2})$,
3. $\Psi(B_1 + B_2) = (+_1(\Psi(B_1)(.\lfloor)_{B_1})) \cup (+_2(\Psi(B_2)(.\rfloor)_{B_2}))$,
4. $\Psi(B_1 ; B_2) = \{;_1(K_1) \cup ;_2(K_2) \mid K_i \in \Psi(B_i)\}$, and
5. $\Psi(B[f]) = \Psi(B \odot .f)$.

□ 5.4.11

We note that only through Item 4 of Corollary 5.4.11 may two maximal lines of a syntactically generated High Level Petri Box share a local transition. Moreover, from Corollary 5.4.11:

COROLLARY 5.4.12 All local transitions of a High Level Petri Box lie on some maximal line.

□ 5.4.12

5.4.3 Further Properties of Local Transitions

The auxiliary operators manipulate the places and local transitions of a High Level Petri Box. The structure of the contexts which annotate local transitions is, then, closely related to the structure of $LT(B)$ induced by the sharing of places which is, in essence, the structure $\Phi(B)$. Our first result characterising this interrelationship relates the minimal and maximal local transitions and the entry and exit interfaces of a High Level Petri Box.

PROPOSITION 5.4.13 Let B be a High Level Petri Box. Then $l \in LT(B)$ implies

1. l is maximal in $\Phi(B) \Leftrightarrow l^\bullet \subseteq B^\bullet$,
2. l is minimal in $\Phi(B) \Leftrightarrow {}^\bullet l \subseteq {}^\bullet B$. □ 5.4.13

Proof: We prove Item 1 only, the other following by symmetry.

$\Rightarrow$: suppose l is maximal in $\Phi(B)$ but that $l^\bullet \not\subseteq B^\bullet$. Then, from Property 3 of Definition 2.5.15, $l^\bullet \cap B^\bullet = \emptyset$ and hence, from Property 2 of that definition, $l^\bullet \cap (LT(B)^\bullet \setminus {}^\bullet LT(B)) = \emptyset$. From the definition, ${}^\bullet l \neq \emptyset \neq l^\bullet$ so that, as $l^\bullet \subseteq LT(B)^\bullet$, $l^\bullet \subseteq {}^\bullet LT(B) \neq \emptyset$, a contradiction of the maximality of l.

$\Leftarrow$: suppose $l^\bullet \subseteq B^\bullet$ but that l is not maximal. Then there is a local transition, l' say, such that $l^\bullet \cap {}^\bullet l' \neq \emptyset$. But ${}^\bullet l' \cap B^\bullet = {}^\bullet l' \cap (LT(B)^\bullet \setminus {}^\bullet LT(B)) = \emptyset$ and $l^\bullet \not\subseteq B^\bullet$, a contradiction.

Hence the result. ■ 5.4.13

From this we see that context manipulations alter the pre-contexts of minimal local transitions (as, only for l a minimal local transition, ${}^\bullet l \in {}^\bullet B$), and the post-contexts of maximal local transitions (as then $l^\bullet \in B^\bullet$). As other operations either preserve local causality, or concatenate maximal lines, the contexts annotating local transitions are structured:

PROPOSITION 5.4.14 For each $l \in LT(B)$ either ${}^C l \leq l^C$ or $l^C \leq {}^C l$. Moreover:

1. l maximal in $\Phi(B)$ implies ${}^C l \leq l^C$,
2. l minimal in $\Phi(B)$ implies $l^C \leq {}^C l$. □ 5.4.14

Proof: As was the case in Theorem 5.4.8, we again prove a more general result, based on the application of auxiliary operators (with all lifted relations being sane).

Base Case: $HLPB(u)B$ with $u \in A \cup \{\mathbf{0}\}$ whence $B = [l]$ for some local transition l and ${}^C l = l^C$ for the result.

Inductive Step: Assume the result holds for B, B_1 and B_2. We consider the application of each auxiliary operator in turn:

$B^\oplus$: follows easily from the induction hypothesis and the characterisation of $\prec_\div$ given in the proof of Theorem 5.4.8,

$f(B)$ **when** f **sane:** f preserves local causality (see the characterisation of $\Phi(f(B))$ for sane f in the proof of Theorem 5.4.8) and the result follows from the induction hypothesis,

$B_1 \cup B_2$: follows from the characterisation of $\Phi(B_1 \cup B_2)$ of the proof of Theorem 5.4.8 and the induction hypothesis,

$B \odot C$: let $l' \in LT(B \odot C)$. From the definition $S_{B \odot C} = S_B$ and $LT(B \odot C) = LT(B)(C)_B$ so that there is a local transition $l \in LT(B)$ with $l' = l(C)_B$, ${}^\bullet l' = {}^\bullet l$ and $l'^\bullet = l^\bullet$. Moreover, by the characterisation of the proof of Theorem 5.4.8, l is maximal (minimal) in $\Phi(B)$ if and only if l' is maximal (minimal) in $\Phi(B \odot C)$. We distinguish four sub-cases:

1. l is neither maximal nor minimal in $\Phi(B)$: whence $l(C)_B = l$, so that ${}^C l(C)_B = {}^C l \leq l^C = l(C)_B{}^C$.
2. l is maximal but not minimal in $\Phi(B)$: whence

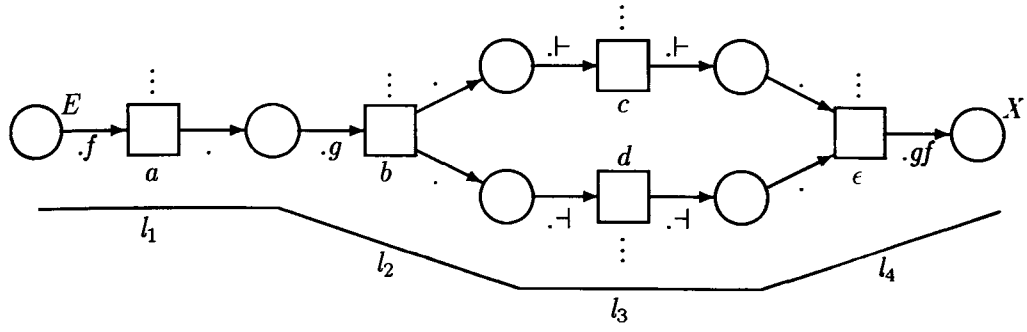
$$\begin{aligned} {}^C l(C)_B &= [l \text{ is not minimal}] \\ &\quad {}^C l \\ &\leq [\text{Induction hypothesis}] \\ &\quad l^C \\ &\leq l^C(C) \\ &= [l \text{ is maximal}] \\ &\quad (l(C)_B)^C. \end{aligned}$$

3. l is minimal but not maximal in $\Phi(B)$: similar, as is
4. l is both maximal and minimal in $\Phi(B)$.

■ 5.4.14

5.4.3.1 η and α

From Section 3.3.4, we see that the effect of firing an abstract transition is, other than creating a label, to push and then pop the contexts annotating a local transition onto a token. From Proposition 5.4.14, annotating contexts are always comparable, so that this manipulation of tokens is always possible in terms of contexts (although we do not yet know that it is sensible in terms of behaviour). We will record the nett effect of this manipulation in the (total) function η.



$$\eta_B(l_1) = (.f) \quad \eta_B(l_2) = (.g) \quad \eta_B(l_3) = (.) \quad \eta_B(l_4) = -(.gf)$$
$$\alpha^K(l_1) = (.f) \quad \alpha^K(l_2) = (.gf) \quad \alpha^K(l_3) = (.gf) \quad \alpha^K(l_4) = (.)$$

$$(a;(b;(c\|d);e)[g])[f]$$

Figure 5.6: An example of the application of the partial function α to a maximal line, $K = l_1 \prec l_2 \prec l_3 \prec l_4$, of a High Level Petri Box. Notice that, for this High Level Petri Box, α is total and, moreover, has value (.) for the maximal local transition l_4 of K.

DEFINITION 5.4.15 Define a function $\eta_B: LT(B) \to \mathcal{SC}$ such that $\eta_B(l) = {}^C l - l^C$.

■ 5.4.15

We will usually omit the subscript B when it is clear by context.

$\eta_B(l)$ records changes to tokens wrought by the local transition l: if l is enabled at a marking M, $M[l\rangle M'$, with $M' = (M \setminus {}^\bullet l \times \{\!|e|\!\}) \cup l^\bullet \times \{\!|f|\!\}$, then

$$f = {}^C l e - l^C = \begin{cases} \eta_B(l) \frown e & l^C \leq {}^C l \\ e - (-\eta_B(l)) & {}^C l \leq l^C \end{cases}$$

'Summing' η over maximal lines characterises the effect of firing the local transitions of which it comprises. The 'partial sums' in this process for a particular maximal line K are recorded in the partial function α^K.

DEFINITION 5.4.16 Let $K = l_1 \prec \cdots \prec l_n \in \Psi(B)$. Define a function $\alpha^K: K \to \mathcal{C}$ such that $\alpha^K(l_1) = \eta_B(l_1)$ and such that, if $\alpha^K(l_i)$ is defined, then

$$\alpha^K(l_{i+1}) = \begin{cases} \eta_B(l_{i+1})\alpha^K(l_i) & \eta_B(l_{i+1}) \in \mathcal{C} \\ \alpha^K(l_i) - (-\eta_B(l_{i+1})) & -\eta_B(l_{i+1}) \in \mathcal{C} \wedge -\eta_B(l_{i+1}) \leq \alpha^K(l_i) \end{cases}$$

leaving $\alpha^K(l_{i+1})$ otherwise undefined. ■ 5.4.16

The range of α^K consists of the 'partial sums' of the contexts which annotate maximal lines of a High Level Petri Box, with the partiality of α reflecting that, at some point, we were not able

to interpret the cumulative effect of η in terms of tokens (behaviour) rather than just signed contexts. An example of the definition of α is shown in Figure 5.6. We note that, for the High Level Petri Box of that figure, α^K is a total function with value on the last local transition in K equal to the trivial context, (.). Proposition 5.4.17 shows that this is, for syntactically generated High Level Petri Boxes, the general case.

PROPOSITION 5.4.17 Let $K = l_1 \prec \cdots \prec l_n \in \Psi(B)$. Then $\alpha^K : K \to C$ is total and $\alpha^K(l_n) = (.)$ □ 5.4.17

Proof: By induction on the structure of B (as usual using auxiliary operators, all lifted relations are sane).

Base Case: for $HLPB(B)$ the result follows by inspection.

Inductive Step: Assume that the result holds for B, B_1 and B_2. For $B^{\dagger}$ and $f(B)$ (f sane) the result follows directly from the induction hypothesis.

$B \odot C$: we use the following lemma:

LEMMA 5.4.18 Let $K = l_1 \prec \cdots \prec l_n \in \Psi(B)$. Then $n = 1$ implies $\eta_{B\odot C}(l_1(C)_B) = \eta_B(l_1)$; $n > 1$ implies

1. $\eta_{B\odot C}(l_1(C)_B) = \eta_B(l_1)(C)$,
2. $\eta_{B\odot C}(l_i(C)_B) = \eta_B(l_i)$ when $1 < i < n$,
3. $\eta_{B\odot C}(l_n(C)_B) = -((-\eta_B(l_n))(C))$. □ 5.4.18

Proof: The proof when $n = 1$ follows by inspection.

Suppose $n > 1$. Note that, as $n > 1$, l_1 is minimal but not maximal in $\Phi(B)$, and l_n is maximal but not minimal in $\Phi(B)$:

We have

$$\begin{aligned}
\eta_{B\odot C}(l_1(C)_B) &= {}^{C}l_1(C)_B - l_1(C)_B{}^{C} \\
&= [l_1 \text{ is minimal in } B \text{ but not maximal}] \\
&\quad {}^{C}l_1(C) - l_1{}^{C} \\
&= [l_1{}^{C} \leq {}^{C}l_1 \leq {}^{C}l_1(C)] \\
&\quad \eta_B(l_1)(C)
\end{aligned}$$

and we have the result. The other cases are similar. ■ 5.4.18

The minor complexity in the statement of Item 3 arises only because we have not defined context concatenation on (strictly) signed contexts.

From the characterisation of $\Phi(B \odot C)$ given in the proof of Theorem 5.4.8, $K' = l'_1 \prec \cdots \prec l'_n \in \Psi(B \odot C)$ if and only if there is $K = l_1 \prec \cdots \prec l_n \in \Psi(B)$ such that $l'_i = l_i(C)_B$, i.e., such that $K' = K(C)_B$. If $n = 1$, the result follows directly from Lemma 5.4.18 and the induction hypothesis.

Suppose $n > 1$. We claim that $i < n$ implies $\alpha^{K'}(l'_i) = \alpha^K(l_i)(C)$, and $i = n$ implies $\alpha^{K'}(l'_i) = \alpha^K(l_i) = (.)$, from which the result follows immediately.

The proof of the claim is by finite induction[8] up to and including n:

Base Case: $i = 1$: whence:

$$\begin{aligned} \alpha^{K'}(l'_1) &= \eta_{B\odot C}(l'_1) \\ &= \text{[Lemma 5.4.18]} \\ &\quad \eta_B(l_1)(C) \\ &= \alpha^K(l_1)(C) \in \mathcal{C} \end{aligned}$$

Inductive Step: Suppose $i \geq 2$ and assume that the property holds for $i-1$, i.e., that $\alpha^{K'}(l'_{i-1}) = \alpha^K(l_{i-1})(C) \in \mathcal{C}$.

For $i < n$ we distinguish two cases:

1. $\eta_{B\odot C}(l'_i)(= \eta_B(l_i)) \in \mathcal{C}$. Hence:

$$\begin{aligned} \alpha^{K'}(l'_i) &= \eta_{B\odot C}(l'_i)\alpha^{K(C)_B}(l_{i-1}(C)_B) \\ &= \eta_B(l_i)\alpha^K(l_{i-1})(C) \\ &= \alpha^K(l_i)(C) \end{aligned}$$

2. $\eta_{B\odot C}(l'_i) \in \mathcal{SC} \setminus \mathcal{C}$ is similar.

For $i = n$, from the (outer) induction hypothesis we may assume that $\alpha^K(l_i) = (.)$ and, as α^K is total on K, that $\alpha^K(l_{i-1}) \in \mathcal{C}$ and, hence, that $\eta_B(l_i) = -\alpha^K(l_{i-1}) \in \mathcal{SC} \setminus \mathcal{C}$. From the (finite) induction hypothesis we may assume that $\alpha^{K'}(l'_{i-1}) = \alpha^K(l_{i-1})(C)$.

From Lemma 5.4.18:

$$\begin{aligned} \eta_{B\odot C}(l'_i) &= -((-\eta_B(l_i))(C)) \\ &= -(\alpha^K(l_{i-1})(C)) \\ &= -\alpha^{K'}(l'_{i-1}) \end{aligned}$$

Hence

$$\begin{aligned} \alpha^{K'}(l'_i) &= \alpha^{K'}(l'_{i-1}) - (-\eta_{B\odot C}(l'_i)) \\ &= \alpha^{K'}(l'_{i-1}) - \alpha^{K'}(l'_{i-1}) \\ &= (.) \end{aligned}$$

and we have the result in this case.

[8] Where the proof by finite induction up to and including m of a property P (in which m does not appear free) is a proof of the property $i \leq m \Rightarrow P$ by 'ordinary' induction.

$B_1 \cup B_2$: we distinguish two subcases:

If ${}^\bullet B_1 \cap B_2{}^\bullet = \emptyset = B_1{}^\bullet \cap {}^\bullet B_2$ then $\Psi(B_1 \cup B_2) = \Psi(B_1) \cup \Psi(B_2)$ and the result follows directly from the characterisation of $\Phi(B_1 \cup B_2)$ of Theorem 5.4.8 for this case and the induction hypothesis.

So suppose that either ${}^\bullet B_1 = B_2{}^\bullet$ or $B_1{}^\bullet = {}^\bullet B_2$: assume, without loss of generality, that $B_1{}^\bullet = {}^\bullet B_2$ (whence ${}^\bullet B_1 \cap B_2{}^\bullet = \emptyset$, as B_1 and B_2 are unionable). From the characterisation of $\Phi(B_1 \cup B_2)$ of Theorem 5.4.8 in this case, $K \in \Psi(B_1 \cup B_2)$ implies $K = K_1 \cup K_2 = l_1 \prec_1 \cdots \prec_1 l_{n_1} \prec' l'_1 \prec_2 \cdots \prec_2 l'_{n_2}$. From the induction hypothesis we may assume that α^{K_1} is total and $\alpha^{K_1}(l_{n_1}) = (.)$ and, similarly, α^{K_2} is total and $\alpha^{K_2}(l'_{n_2}) = (.)$. From the definition we have that $\alpha^K(l_i) = \alpha^{K_1}(l_i)$, for $1 \leq i \leq n_1$ and, in particular, $\alpha^K(l_{n_1}) = (.)$. l'_1 is minimal in $\Phi(B_2)$ so that $\eta(l'_1) \in C$ and

$$\begin{aligned} \alpha^K(l'_1) &= \eta_B(l'_1)\alpha^K(l_{n_1}) \\ &= \eta_{B_2}(l'_1)(.) \\ &= \alpha^{K_2}(l'_1) \end{aligned}$$

and the result follows from the induction hypothesis.

Hence the result. ■ 5.4.17

One useful corollary of Proposition 5.4.17 is that the property which formed the hypothesis of Theorem 3.4.8 (page 56) holds of syntactically generated High Level Petri Boxes. The property was that the token which forms the standard initial marking of a syntactically generated High Level Petri Box forms a postfix of all tokens reachable from that standard initial marking. Comments of its uses in the context of safeness and memorylessness of High Level Petri Boxes are given in the proof of Theorem 7.1.1 in Chapter 7.

COROLLARY 5.4.19 Let B be a syntactically generated High Level Petri Box. Then for all contexts d, and for all $M \in \mathcal{M}_d^B$, $e \in \mathrm{ran}(M)$ implies $d \sqsubseteq e$. □ 5.4.19

Proof: Follows from the totality of α, its independence of the initial marking, Theorem 5.4.8 which implies there are no loops in B, and Corollary 5.4.12, which implies that all local transitions lie on some maximal line. ■ 5.4.19

With Corollary 5.4.19 we may now exclude High Level Petri Boxes such as those appearing in Figure 5.2 as not being syntactically generated, and hence justify rewrite rule 3. Returning to the proof of Lemma 5.3.1:

Proof: We must show that, for all $\hat{M} \in \mathcal{M}_d^{\widehat{B}}$, $s \in B_1{}^\bullet$ and $\epsilon \in \hat{M}(s)$, $\epsilon = (.f)d$ (where $.f$ is as defined earlier).

Let $l \in LT(\widehat{B})$ be such that $l^\bullet \subseteq B_1{}^\bullet$, and let $l \in K \in \Psi(\widehat{B})$. As there are no arcs to entry places in High Level Petri Boxes, tokens in $s \in B_1{}^\bullet$ must have arrived there through the firing of such a local transition.

From Proposition 5.4.17, there is a maximal line $K_1 \in \Psi(B)$ such that $K = K_1(.f)_{B_1 \cup B_2}$ whence

$$\alpha^K(l) = \alpha^{K_1}(l)(.f)$$

But α^K determines the value of the tokens which arrive in $l^\bullet$ so that, for any such token e, $e = (.f)d$, and we have the result. ■ 5.3.1

5.5 Applications

5.5.1 stop as a Behavioural Unit of Choice

Although clearly not a structural unit, flat High Level Petri Boxes provide a simple demonstration that stop forms a behavioural unit of choice composition. For, consider a syntactically generated High Level Petri Box B. Then B+stop contains all abstract transitions of B but also an extra one, $t_{\mathtt{stop}}$ say, labelled **0**. For this transition, and because of its dead label, $index(t_{\mathtt{stop}}) \rightarrow \{d\}$ is never a feasible substitution from a standard initial marking of d. Hence[9] the behaviours of the flattenings of B and B+stop are the same, a property which lifts to B and B+stop.

5.5.2 Commutativity and Associativity of Concurrent and Choice Composition

In the spirit of Section 5.3, we may characterise the structural relationship between the High Level Petri Boxes $(B_1 \odot .\vdash) \cup (B_2 \odot .\dashv)$ and $(B_1 \odot .\dashv) \cup (B_2 \odot .\vdash)$, i.e., between $B_1 \| B_2$ and $B_2 \| B_1$ (which we will abbreviate to B and $\hat{B}$, respectively).

We have the equivalent of Proposition 5.3.2:

[9] Clearly, there are more direct proofs which do not use the properties of flat High Level Petri Boxes.

PROPOSITION 5.5.1 For all $d \in C$, $B \equiv^{d}_{Reach} \widehat{B}$. □ 5.5.1

Proof: We prove the result when $d = (.)$. The other cases are similar. We will show that B and $\widehat{B}$ satisfy the hypotheses of Proposition 3.6.2, whence the result.

That $S_B = S_{\widehat{B}}$ is clear from the definitions.

Define $\xi: LT(B) \to LT(\widehat{B})$ such that

$$\xi(l) = \langle {}^\bullet l, E \frown D, \bar{l}, E' \frown D', l^\bullet \rangle$$

where D and E are such that

$$D = \begin{cases} (.) & {}^\bullet l \cap {}^\bullet B = \emptyset \wedge {}^C l = E \\ (.\mp) & {}^\bullet l \subseteq {}^\bullet B \wedge {}^C l = E(.\pm) \end{cases} \qquad D' = \begin{cases} (.) & l^\bullet \cap B^\bullet = \emptyset \wedge l^C = E' \\ (.\mp) & l^\bullet \subseteq B^\bullet \wedge l^C = E'(.\pm) \end{cases}$$

Then $\xi: LT(B) \equiv LT(\widehat{B})$ such that ${}^\bullet\xi(l) = {}^\bullet l$, $\overline{\xi(l)} = \bar{l}$ and $\xi(l)^\bullet = l^\bullet$. Moreover, ξ lifts to a bijective correspondence between T_B and $T_{\widehat{B}}$ whence, for convenience we will assume that, for $t = \{\!|l_1, \ldots, l_n|\!\}$, $index(t) = index(\xi(t)) = \{x_1, \ldots, x_n\}$ with x_i annotating arcs (s,t) and $(s,\xi(t))$ for $s \in {}^\bullet l_i = {}^\bullet\xi(l_i)$, and y_i annotating arcs (t,s) and $(\xi(t),s)$ for $s \in l_i^\bullet = \xi(l_i)^\bullet$.

For each $s \in S_B$, define $\xi_s: C_{\mathcal{L}} \to C_{\mathcal{L}}$ such that

$$\xi_s(e) = \begin{cases} e & s \in {}^\bullet\widehat{B}^\bullet \\ e'(.\mp) & e = e'(.\pm) \end{cases}$$

Note that for all $s \in S_{\widehat{B}}$, $\xi_s: C_{\mathcal{L}} \equiv C_{\mathcal{L}}$. Moreover, if $s \in {}^\bullet B^\bullet$, then $\xi_s = \mathrm{id}$.

Define $\Xi: \mathcal{M}^B_{(.)} \to \mathcal{M}^{\widehat{B}}_{(.)}$ such that $\Xi(M) = \{s - \xi_s(e) \mid s \to e \in M\}$.

We show that Ξ satisfies Properties 3(a)—3(c) of Proposition 3.6.2, whence the result:

1. $\Xi(M^{I,B}_{(.)}) = M^{I,\widehat{B}}_{(.)}$:

$$\begin{aligned} \Xi(M^{I,B}_{(.)}) &= \{s \to \xi_s(.) \mid s \in {}^\bullet B\} \\ &= \{s \to (.) \mid s \in {}^\bullet\widehat{B}\} \\ &= M^{I,\widehat{B}}_{(.)} \end{aligned}$$

2. Suppose $M, M' \in \mathcal{M}^B_{(.)}$, $t \in T_B$, $\delta \in \mathrm{subs}^B(t)$ such that $M \left[\begin{smallmatrix} a \\ t:\delta \end{smallmatrix} \right\rangle M'$ in B.
Define $\gamma(x_i) = \xi_s(\delta(x_i))$ for $s \in {}^\bullet l_i$, and $\gamma(y_i) = \xi_s(\delta(y_i))$ for $s \in l_i^\bullet$. We first show that γ is a feasible substitution for $\xi(t)$. Suppose not, i.e., either:

(a) there is no consistent value to push onto the tokens populating the pre-set of the local transitions of $\xi(t)$, i.e., $\gamma(x_i) \notin \Xi(M)(s)$ for some $s \in {}^\bullet\xi(l_i) \in \xi(t)$. But then, from the construction, $\alpha(x_i) \notin M(s)$, a contradiction.

(b) we may push onto, but not consistently pop from. tokens, i.e., there is a local transition $\xi(l) \in \xi(t)$ and a context e such that ${}^\bullet\xi(l) \times \{\!|e|\!\} \subseteq \Xi(M)$, but ${}^C\xi(l)e - \xi(l)^C$ is not defined or is a member of $\mathcal{SC} \setminus \mathcal{C}$. From Corollary 5.4.12, we may find a maximal line $K \in \widehat{B}$ such that $\xi(l) \in K$. But then $\alpha^K(\xi(l))$ is not defined or is a member of $\mathcal{SC} \setminus \mathcal{C}$, contradicting Proposition 5.4.17.
Hence ${}^C\xi(l)e - \xi(l)^C \in \mathcal{C}$, and $\xi(l)$ does not disable $\xi(t)$ at γ.

(c) the flow predicate is not satisfied under γ, however:

$$\begin{aligned}
&\mathrm{Flow}(\overline{\xi(l_1)}, \ldots, \overline{\xi(l_n)}, {}^C\xi(l_1)\gamma(x_1), \ldots, {}^C\xi(l_n)\gamma(x_n)) \\
&\quad \Leftrightarrow \text{ [Symmetries of the Flow predicate]} \\
&\qquad \mathrm{Flow}(\overline{l_1}, \ldots, \overline{l_n}, {}^C l_1\delta(x_1), \ldots, {}^C l_n\delta(x_n)) \\
&\quad \Leftrightarrow \textit{ true}
\end{aligned}$$

as δ is a feasible substitution for t.

Moreover, clearly $\overline{\xi(t){:}\gamma} = \overline{t{:}\delta}$; a simple calculation gives that, if $\Xi(M)\left[\begin{smallmatrix} a \\ t{:}\delta \end{smallmatrix}\right\rangle N$ then $N = \Xi(M')$, whence the result.

3. the proof of Property Hypothesis 3(c) is similar.

Hence the result. ■ 5.5.1

Similar arguments may be used to extend the result to associativity of concurrent composition, and the commutativity and associativity of choice composition, giving:

PROPOSITION 5.5.2 Let B_1, B_2 and B_3 be syntactically generated High Level Petri Boxes. Then, for all $d \in C$

1. $B_1 \| B_2 \equiv^d_{Reach} B_2 \| B_1$,
2. $B_1 + B_2 \equiv^d_{Reach} B_2 + B_1$,
3. $(B_1 \| B_2) \| B_3 \equiv^d_{Reach} B_1 \| (B_2 \| B_3)$,
4. $(B_1 + B_2) + B_3 \equiv^d_{Reach} B_1 + (B_2 + B_3)$, □ 5.5.2

5.5.2.1 Extending the Rewrite System

The addition of a rewrite rule based on the above behavioural equivalence, although an attractive option, is, unfortunately, not efficacious—it produces numerous critical pairs which appear, moreover, not to produce 'useful' rewrites which may be added to the system; they appear to lead to an infinite regress.

For instance, suppose we were to introduce a rewrite $\pm$ defined so:

$$(\pm) \quad t_1 \odot .\dashv \cup t_2 \odot .\vdash \quad \rightarrow \quad t_1 \odot .\vdash \cup t_2 \odot .\dashv$$

With the addition of this rule to the rewrite system of Definition 5.2.1 and, in particular, in combination with rewrite β we obtain the critical pair:

Critical Pair: $(\pm, \beta)$

$$\begin{array}{lll} (t_1 \cup t_2) \odot .\vdash \cup\, t_3 \odot .\dashv & \pm\leftarrow & (t_1 \cup t_2) \odot .\dashv \cup\, t_3 \odot .\vdash \quad \rightarrow_\beta \quad t_1 \odot .\dashv \cup\, t_2 \odot .\dashv \cup\, t_3 \odot .\vdash \\ & \rightarrow_\beta & t_1 \odot .\vdash \cup\, t_2 \odot .\vdash \cup\, t_3 \odot .\dashv \end{array}$$

and we require a rewrite $t_1 \odot .\dashv \cup\, t_2 \odot .\dashv \cup\, t_3 \odot .\vdash \rightarrow t_1 \odot .\vdash \cup\, t_2 \odot .\vdash \cup\, t_3 \odot .\dashv$ for conflation of this critical pair.

There are two points of note:

1. the addition of this rule to the system does not resolve the problem: the new rule again forms a critical pair with β. In fact, this process appears not to produce any finitely characterisable collection of rules;

2. the requirement of a behavioural equivalence prevents the introduction of an iterative rule for the confluence[10] of this critical pair: we may not use the intermediate step of, say, $t_1 \odot .\dashv \cup\, t_2 \odot .\dashv \cup\, t_3 \odot .\vdash \rightarrow t_1 \odot .\vdash \cup\, t_2 \odot .\dashv \cup\, t_3 \odot .\vdash$, in the quest for conflation, as the two are not necessarily behaviourally equivalent. This is illustrated in Figure 5.7.

DEFINITION 5.5.3 A term, t, is said to be a *completed binary concurrent composition* if there is a term with a hole, $C[\_]$, and terms t_1 and t_2 such that $t = C[t_1 \| t_2]$. ■ 5.5.3

If we are to permit the addition of conditional rewrites to our system, we may define the following rule, for which the justification is an extended form of Proposition 5.5.2:

[10] In this case, even such an intermediate step would not produce the result, it corresponds to the reduction of an odd to an even permutation using an even number of permutation pairs, which is well known not to be possible.

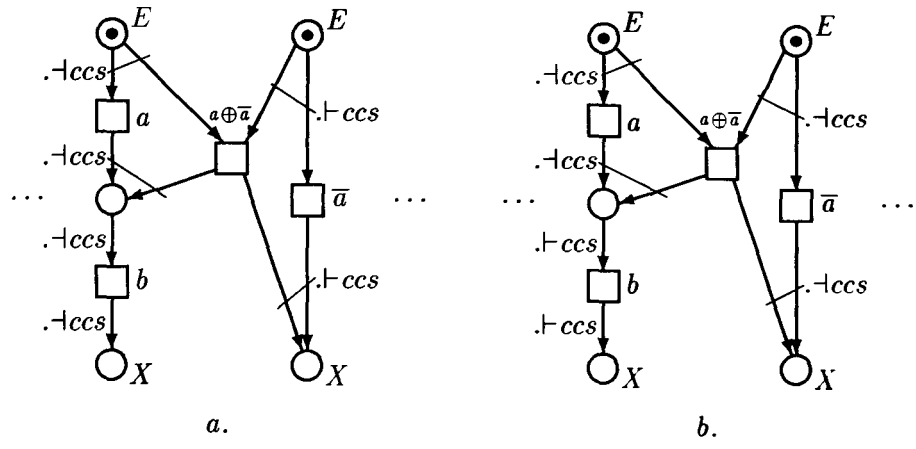


Figure 5.7: Illustrating the non-equivalence of the auxiliary High Level Petri Box terms (*a.*) $(;_1(a) \odot .\dashv ccs \cup ;_2(b) \odot .\dashv ccs \cup \overline{a} \odot .\vdash ccs)^{\oplus}$ and (*b.*) $(;_1(a) \odot .\dashv ccs \cup ;_2(b) \odot .\vdash ccs \cup \overline{a} \odot .\dashv ccs)^{\oplus}$. Whereas the transition labelled $a \oplus \overline{a}$ is enabled in (*a.*) in the label algebra of Example 2.1.1, the corresponding transition of (*b.*) is not.

DEFINITION 5.5.4 Given a completed binary concurrent composition $t = C[t_1 \| t_2]$, we may rewrite t to $C[t_2 \| t_1]$. ■ 5.5.4

Whether a term forms a completed binary concurrent composition is clearly decidable so that the rule is effectively applicable.

5.6 Summary

In this chapter we have shown the existence of a unique normal form for syntactically generated High Level Petri Boxes with equality of normal forms implying equivalence of behaviours from standard initial markings. We have also seen that these equivalences do not necessarily hold for non-standard initial markings, nor for non-syntactically generated High Level Petri Boxes.

The portion of the algebra we have described so far is deficient in that we are not able to describe Turing expressive terms: we may reduce the behaviour of a syntactically generated High Level Petri Box to an essentially finite P/T net, and the class of finite P/T nets is not Turing expressive [Pet81].

To produce Turing expressive High Level Petri Boxes we must forego the normal form, and consider High Level Petri Boxes for which the local transitions are not covered by a partial order. This is done in the next chapter.

Chapter 6

Algebra and Semantics II

In this chapter we complete our description of the High Level Petri Box Algebra with the definitions and properties of the operators of refinement and recursion.

The chapter begins with the development of refinement.

6.1 Refinement

Our motivation for the development of the refinement construct comes from the *free construction* on an algebra. The free construction equips a syntax with the familiar concepts of *variable* and *assignment* (a mapping from variables to terms), together with homomorphism conditions that uniquely determine the lifting of a substitution to the syntax extended with variables. Our refinement operator is a restricted form of assignment, one which has only a single variable in its domain.

To transfer these notions to High Level Petri Boxes, it will be convenient (and appears entirely natural, given the properties of the operator) to identify local transition and variable (in the sense given by Definition 6.3.6), and refinement and assignment (in the sense given by Definition 6.4.13). We thus provide a refinement operator which has the desirable quality that its application commutes with that of all other operators (in the technical sense given by Proposition 6.4.7, Theorem 6.4.15 and Lemma 6.5.9) meaning that it does not extend the semantic domain of the algebra and, consequently, does not require specialised proof techniques for most of its properties.

The definition of the refinement construct on High Level Petri Boxes follows that of [GvG89]

in that transitions are replaced by net structure whilst preserving connectivity through place multiplication of interface places. The idea and the novelty of our approach is that we replace the local transitions and not (abstract) transitions with net structure with the advantage that, although the cardinality of the set of abstract transitions of a syntactically generated High Level Petri Box may be infinite, the set of its local transitions is always finite allowing a fully general iterated derivation of refinement (Definition 6.4.13) from the simpler, auxiliary operator of *local transition refinement* (Definition 6.4.1). (This approach is contrasted with the refinement operators of the PBC in the discussion at the end of the chapter.)

The motivation for refinement stems from algebra and we thus find it convenient to use the language of algebra—in particular, the language of Order Sorted Algebra (OSA), [Gog78]—for its economical description. The use we make of OSAs is, however, for convenience of expression only and is not necessary for the approach.

We begin with a brief resumé of the basic concepts of OSAs.

6.2 Order Sorted Algebra

OSA is the study of algebras with subsorts. An OSA represents a datatype as a signature declaring various data, possibly augmented by equations giving their interdependence. The reader interested in a complete formal development is referred to the excellent [Gog89], from which we have adapted our presentation.

DEFINITION 6.2.1 *[Order Sorted Signature]* An Order Sorted Signature (OSS) is a triple, $\langle S, \leq, \Sigma \rangle$, where:

1. S is a set of Sorts, allowing each constant and operator to be assigned a sort (this corresponds to typing in programming languages),
2. $\leq$ is a partial order on the set of Sorts: for $s_1 \leq s_2$ to appear in the partial order is to say that the sort s_1 is a *subsort* of s_2 or, equivalently, that s_2 is a *supersort* of s_1,
3. Σ is an $(S^* \times S)$-indexed set[1] of operation symbols—where each member of $\Sigma_{s_1 \ldots s_n, s}$. is intended to represent a function of *rank*[2] $s_1 \ldots s_n, s$. For $\lambda \in S^0 (\subseteq S^*)$, the empty string of sorts, we will regard each $d \in \Sigma_{\lambda, s}$ as a constant of type s.

[1] For T a set, a T-indexed set is a set of sets in one to one correspondence with the elements of T. We will write a T-indexed set $A = \{A_t, \ldots\}$ where $T = \{t, \ldots\}$.

[2] I.e., signature. Rank is preferred to distinguish from order sorted 'signature'.

In addition the triple $\langle S, \leq, \Sigma \rangle$ must satisfy the following *monotonicity* condition[3]:

4. $\Sigma_{w_1,s_1} \cap \Sigma_{w_2,s_2} \neq \emptyset$ and $w_1 \leq w_2$ implies $s_1 \leq s_2$. ■ 6.2.1

An OSS is *regular* if and only if given d in Σ_{w_1,s_1} and $w_0 \leq w_1$ in S^*, then there is a least rank $\langle w, s \rangle \in S^* \times S$ such that $w_0 \leq w$ and $d \in \Sigma_{w,s}$. A regular OSS is capable to be extended to include variables as a variable of 'higher' type can hold terms of 'lower' type related to it.

Objects which represent an OSS Σ are called Σ-algebras.

DEFINITION 6.2.2 *[Order Sorted Algebra]* Let $\langle S, \leq, \Sigma \rangle$ be an OSS. A $\langle S, \leq, \Sigma \rangle$-*algebra* A is an S-indexed set of *carrier sets* together with an interpretation that assigns a mapping $A_d^{w,s}: A_w \rightarrow A_s$ to each d in $\Sigma_{w,s}$ (where $A_w = A_{s_1} \times \ldots \times A_{s_n}$ when $w = s_1 \ldots s_n$ and where A_λ is a singleton so that $A_d^{\lambda,s}$ represents a constant) such that

1. $s \leq s'$ implies $A_s \subseteq A_{s'}$

2. $d \in \Sigma_{w_1,s_1} \cap \Sigma_{w_2,s_2}$ and $w_1 \leq w_2$ implies $A_d^{w_1,s_1} = (A_{w_1} \triangleleft A_d^{w_2,s_2})$.

For B another $\langle S, \leq, \Sigma \rangle$-algebra, a $\langle S, \leq, \Sigma \rangle$-*homomorphism*, h, is an S-indexed function $h: A \rightarrow B = \{h_s: A_s \rightarrow B_s\}$ such that

3. $d \in \Sigma_{w,s}$ implies $h_s(A_d^{w,s}(a)) = B_d^{w,s}(h_w(a))$, for all $a \in A_w$

4. $s \leq s'$ implies $h_s = A_s \triangleleft h_{s'}$. ■ 6.2.2

As is usual, we denote the OSS $\langle S, \leq, \Sigma \rangle$ by its third component Σ, when this causes no confusion.

A well-known result is that the class of OSAs on a signature Σ (with Σ-homomorphisms as arrows) forms a category, $\mathbf{OSAlg}_\Sigma$. The 'standard' semantics for an OSS, Σ, is the initial algebra of this category (which is guaranteed to exist, [Gog89]), the 'prototypical' initial object being the *term algebra*, $\mathcal{T}_\Sigma$, for that signature.

DEFINITION 6.2.3 *[Term Algebra]* Given an OSS, $\langle S, \leq, \Sigma \rangle$, define the S-indexed set $\mathcal{T}_\Sigma$ of all Σ-terms to be the smallest set of lists over the alphabet $\Sigma \cup \{(,)\}$ (where $($ and $)$ are special symbols disjoint from Σ) such that

1. $\Sigma_{\lambda,s} \subseteq \mathcal{T}_s$ for all $s \in S$,

[3]Where $\leq$ is extended to S^* in the usual way.

$$\begin{array}{ccc} X & \hookrightarrow & \mathcal{T}_\Sigma(X) \\ \| & & \downarrow !_{a^*} \\ X & \xrightarrow{a} & A \end{array}$$

Figure 6.1: The free construction $\mathcal{T}_\Sigma(X)$ on an algebra Σ is that which uniquely extends an assignment to a Σ-homomorphism a^*, which agrees on the values of variables.

2. $\mathcal{T}_s \subseteq \mathcal{T}_{s'}$ if $s \leq s'$,

3. For $n \geq 1$, $t_i \in \mathcal{T}_{s_i}$, $i = 1, \ldots, n$, and $d \in \Sigma_{s_1,\ldots,s_n,s}$ then $d(t_1, \ldots, t_n) \in \mathcal{T}_s$.

The interpretation of $\mathcal{T}_\Sigma$ as a Σ-algebra is that in which $d(t_1, \ldots, t_n)$ is represented as $d(t_1, \ldots, t_n)$ (with the t_i similarly represented). ■ 6.2.3

It is common to ignore the distinction between the tuple former $(,)$ and the symbols $(,)$.

Terms in the term algebra are *ground*, i.e., contain no variables. The standard construction by which variables are introduced into an algebra is the *free construction* on the algebra. For the free construction on an OSS $\langle S, \leq, \Sigma \rangle$ a collection of disjoint sets of constants is introduced, one for each sort $s \in S$, $X = \{X_s \mid s \in S\}$ say, called the *variable sets*. By adjoining the variables to a signature we create another signature[4], denoted $\Sigma(X)$. The term algebra on the variable extended signature is then $\mathcal{T}_{\Sigma(X)}$. $\mathcal{T}_{\Sigma(X)}$ can also be regarded as a Σ-algebra by forgetting about the variables: this alternative interpretation is called the *free algebra on* Σ, and is denoted $\mathcal{T}_\Sigma(X)$. The following theorem characterises the free construction:

THEOREM 6.2.4 *[GW88]* Given a regular OSS $\langle S, \leq, \Sigma \rangle$, let A be a Σ-algebra and let $a: X \to A$ be an S-sorted function (called an *assignment*). Then there is a unique Σ-homomorphism $a^*: \mathcal{T}_\Sigma(X) \to A$ such that $a^*(x) = a(x)$ for each $x \in X$. □ 6.2.4

The *free construction* on an algebra is often illustrated with the commuting diagram shown in Figure 6.1. (In the figure, $\hookrightarrow$ is an inclusion and ! asserts uniqueness.)

We will base our notion of refinement on a restricted form of assignment on $\mathcal{T}_{\Sigma(X)}$: given a variable set X in which only the sorted variable set corresponding to the sort s is non-empty, $X_s \neq \emptyset$ say, we will define an *s-refinement* as an assignment $a_y: X_s \to \mathcal{T}_{\Sigma(X)}$ such that $a_y(x) = x$ for all $x \neq y \in X_s$. ($a_y(y)$ may, of course, be chosen arbitrarily from $\mathcal{T}_{\Sigma(X)}$.) y is called the

[4]With regularity being inherited; see [Gog89] for the details.

subject variable of the s-refinement. As a_y is, in particular, an assignment, by Theorem 6.2.4 it extends uniquely to a homomorphism between $\mathcal{T}_\Sigma(X)$ and $\mathcal{T}_{\Sigma(X)}$.

As a_y^* is a Σ-homomorphism, we have the important property of a refinement that for each operator d, $a_y^*(d(t_1,\ldots,t_n)) = d(a_y^*(t_1),\ldots,a_y^*(t_n))$. As is clear, the homomorphism properties of a_y^* together with its action on a variable set characterise it amongst the mappings between $\mathcal{T}_\Sigma(X)$ and $\mathcal{T}_{\Sigma(X)}$. We will use this to show that our High Level Petri Box construct corresponding to an s-refinement is the 'correct' one.

6.3 Order Sorted Algebra Encoding of the High Level Petri Box Algebra

In this section, we wrap the machinery developed in Chapter 4 in the language of OSAs. The free construction may then be interpreted in terms of High Level Petri Box Algebras. The remainder of the section is taken with the development of the lifting operation $\_^*$ which produces a unique homomorphism from an assignment as defined for the algebra of High Level Petri Boxes.

DEFINITION 6.3.1 Let $\mathcal{L} = \langle A, R\rangle$ be a label algebra. An Order Sorted High Level Petri Box Algebra (OSHLPB Algebra) on $\mathcal{L}$, $\Sigma_\mathcal{L}$ is an OSS, $\langle S, \leq, \Sigma\rangle$, where:

$$\begin{aligned}
S &= \{Alph, Proc, Rel\}\\
< &= \{(Alph, Proc)\}\\
\Sigma_{\lambda,Alph} &= A\\
\Sigma_{\lambda,Proc} &= \{\texttt{stop}\}\\
\Sigma_{\lambda,Rel} &= R\\
\Sigma_{Proc\ Proc,Proc} &= \{_ \| _, _ ; _, _ + _\}\\
\Sigma_{Proc\ Rel,Proc} &= \{_ [\ _]\}
\end{aligned}$$

and all other $\Sigma_{w,s} = \emptyset$, for $w \in S^*$, $s \in S$. ■ 6.3.1

We note the subsort relationship between *Alph* and *Proc*.

The term algebra of an OSHLPB algebra consists of the terms which arise from all finite applications of correctly typed operators to terms. Examples of terms of the term algebra are as one would expect: for instance[5], $a\|b$, $(a;b)[f]\|c$, are OSHLPB terms.

The object we called $Terms_\mathcal{L}$ in Section 4.8 is the carrier set of sort *Proc* in $\mathcal{T}_{\Sigma_\mathcal{L}}$.

[5] Using infix notation, we use parentheses to denote scope as usual.

EXAMPLE 6.3.2 Consider the label algebra $\mathcal{L}_0$ of Example 2.1.1. The Order Sorted High Level Petri Box Signature (OSHLPB Signature) on $\mathcal{L}_0$, $\Sigma_{\mathcal{L}_0}$, is the signature defined as:

$$\begin{aligned}
S &= \{Alph, Proc, Rel\} \\
< &= \{(Alph, Proc)\} \\
\Sigma_{\lambda,Alph} &= \{\tau, a, \overline{a}\} \\
\Sigma_{\lambda,Proc} &= \{\mathtt{stop}\} \\
\Sigma_{\lambda,Rel} &= \{ccs, r\} \\
\Sigma_{Proc\ Proc,Proc} &= \{\text{-} \parallel \text{-}, \text{-} ; \text{-}, \text{-} + \text{-}\} \\
\Sigma_{Proc\ Rel,Proc} &= \{\text{-}[\text{-}]\}
\end{aligned}$$

As expected, $\mathcal{T}_{\Sigma_{\mathcal{L}_0}}$ has $A_{Proc} = \{\mathtt{stop}, a, (a \| \mathtt{stop})[ccs], a;a, a+\overline{a}, (\tau+\mathtt{stop})[r], a\|(\overline{a}+\mathtt{stop})[r], \ldots\}$.

■ 6.3.2

We now return to the class of Extended High Level Petri Boxes of a label algebra $\mathcal{L}$, $\mathcal{EHLPB}_{\mathcal{L}}$, to find examples of $\Sigma_{\mathcal{L}}$-algebras:

PROPOSITION 6.3.3 Let $\mathcal{L} = \langle A, R \rangle$ be a label algebra. Then

$$\begin{aligned}
A_{Alph} &= \{[Box_{\mathcal{L}}(u)] \mid u \in A\} \\
A_{Proc} &= \mathcal{EHLPB}_{\mathcal{L}} \\
A_{Rel} &= R \\
A_{\mathtt{stop}} &= [\mathtt{stop}] \\
A_{\|}^{Proc\ Proc,Proc} &= \hat{\|} \\
A_{;}^{Proc\ Proc,Proc} &= \hat{;} \\
A_{+}^{Proc\ Proc,Proc} &= \hat{+} \\
A_{\text{-}[\text{-}]}^{Proc\ Rel,Proc} &= \widehat{\text{-}[\text{-}]}
\end{aligned}$$

forms a $\Sigma_{\mathcal{L}}$-algebra. □ 6.3.3

Proof: We have made assignments to the components of the signature. The well-definedness of the operators is given by the congruence theorem, Theorem 4.8.19. ■ 6.3.3

6.3.1 Process Variables

As already mentioned, we approach refinement through the free construction on OSHLPB Algebras. As we refine only process variables, what we call refinements are, in the language of OSA,

$$\begin{array}{ccccc} \mathcal{PV} & \hookrightarrow & \mathcal{T}_{\Sigma_{\mathcal{L}}}(\mathcal{PV}) & \overset{EHLPB_{\mathcal{L}}^{\mathcal{PV}}}{\longrightarrow} & \mathcal{EHLPB}_{\mathcal{L}}(\mathcal{PV}) \\ \| & & \downarrow !_{a_Y^*} & & \downarrow !_{-[Y \leftarrow -]} \\ \mathcal{PV} & \overset{a}{\longrightarrow} & \mathcal{T}_{\Sigma_{\mathcal{L}}(\mathcal{PV})} & \overset{EHLPB_{\mathcal{L}}^{\mathcal{PV}}}{\longrightarrow} & \mathcal{EHLPB}_{\mathcal{L}}(\mathcal{PV}) \end{array}$$

Figure 6.2: The commuting diagram between Free Algebra, the extended semantic domain $\mathcal{EHLPB}_{\mathcal{L}}$ and the extended semantic function $EHLPB_{\mathcal{L}}(\mathcal{PV})$.

actually *Proc*-refinements. (We remind the reader that $\mathbb{L}$ is the label universe, as was defined in Section 2.1.1 (page 7).)

DEFINITION 6.3.4 Let $Y, Z, \ldots$ be symbols not in the label universe $\mathbb{L}^{\mp}$. Define $\mathcal{PV} = \{Y, Z, \ldots\}$ to be the collection of *process variables.* ■ 6.3.4

To interpret $\mathcal{PV}$ as a variable set in the OSA sense, consider it as the sorted family of sets $\{X_{Alph}, X_{Proc}, X_{Rel}\}$ with $X_{Alph} = X_{Rel} = \emptyset$ and $X_{Proc} = \mathcal{PV}$; assignments from $\mathcal{PV}$ to $\mathcal{T}_{\Sigma(\mathcal{PV})}$ are thus *Proc*-refinements.

We extend Example 6.3.2 to illustrate the free construction on $\mathcal{PV}$.

EXAMPLE 6.3.5 With the adjunction of $\mathcal{PV}$ to $\Sigma_{\mathcal{L}_0}$, so that $\Sigma_{\lambda,Proc} = \{\mathtt{stop}\} \cup \mathcal{PV}$ we may form the $\Sigma_{\mathcal{L}_0}(\mathcal{PV})$ term algebra $\mathcal{T}_{\Sigma_{\mathcal{L}_0}(\mathcal{PV})}$. The free algebra on $\mathcal{T}_{\Sigma_{\mathcal{L}_0}}(\mathcal{PV})$ has $A'_{Proc} = A_{Proc} \cup \{Y, \mathtt{stop}\|X, a{+}\overline{a}, a\|Y, Z, \ldots\}$ as the (free-)carrier for *Proc*. As we adjoin only process variables to the signature, other carrier sets are as before.

Then, given a *Proc*-refinement, a_Y, with $a_Y(Y) = t$, a_Y lifts uniquely to the homomorphism a_Y^* between $\mathcal{T}_{\Sigma_{\mathcal{L}_0}}(\mathcal{PV})$ and $\mathcal{T}_{\Sigma_{\mathcal{L}_0}(\mathcal{PV})}$, characterised as:

$$\begin{array}{rclrcl} a_Y^*(\mathtt{stop}) & = & \mathtt{stop} & a_Y^*(\tau) & = & \tau \\ a_Y^*(a) & = & a & a_Y^*(\overline{a}) & = & \overline{a} \\ a_Y^*(Y) & = & a_Y(Y) = t & a_Y^*(X) & = & a_Y(X) = X \\ a_Y^*(t_1 \| t_2) & = & a_Y^*(t_1) \| a_Y^*(t_2) & a_Y^*(t_1 ; t_2) & = & a_Y^*(t_1) ; a_Y^*(t_2) \\ a_Y^*(t_1 + t_2) & = & a_Y^*(t_1) + a_Y^*(t_2) & a_Y^*(t_1[f]) & = & a_Y^*(t_1)[f] \\ a_Y^*(ccs) & = & ccs & a_Y^*(r) & = & r \end{array}$$

■ 6.3.5

6.3.2 Characterisation of Refinement for High Level Petri Boxes

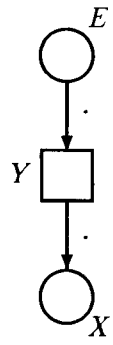
The concept of a refinement has been introduced on the term algebra. To introduce the notion on High Level Petri Boxes we need to:

1. lift the notion of a process variable to High Level Petri Boxes, i.e.. extend the semantic domain to include the denotation of a process variable, and extend the semantic function suitably. We will call the extended semantic function $EHLPB_{\mathcal{L}}^{\mathcal{PV}}$, with $\mathcal{EHLPB}_{\mathcal{L}}^{\mathcal{PV}} = \mathrm{ran}(EHLPB_{\mathcal{L}}^{\mathcal{PV}})$ (so that $EHLPB_{\mathcal{L}}^{\mathcal{PV}}$ is a surjection),

2. identify an operator on the extended semantic domain which corresponds to the (lifting of the) refinement a_Y.

Interpreting the homomorphism condition for a *Proc*-refinement a_Y with $a_Y(Y) = t$ on the extended semantic domain, we arrive at the following characterisation of the refinement construct in terms of Extended High Level Petri Boxes:

$$
\begin{aligned}
EHLPB_{\mathcal{L}}^{\mathcal{PV}}(\mathtt{stop})[Y \leftarrow t] &= EHLPB_{\mathcal{L}}^{\mathcal{PV}}(a_Y^*(\mathtt{stop})) \\
&= EHLPB_{\mathcal{L}}^{\mathcal{PV}}(\mathtt{stop}) \\
EHLPB_{\mathcal{L}}^{\mathcal{PV}}(u)[Y \leftarrow t] &= EHLPB_{\mathcal{L}}^{\mathcal{PV}}(a_Y^*(u)) \\
&= EHLPB_{\mathcal{L}}^{\mathcal{PV}}(u) && u \in A \\
EHLPB_{\mathcal{L}}^{\mathcal{PV}}(Y)[Y \leftarrow t] &= EHLPB_{\mathcal{L}}^{\mathcal{PV}}(a_Y^*(Y)) \\
&= EHLPB_{\mathcal{L}}^{\mathcal{PV}}(a_Y(Y)) \\
&= EHLPB_{\mathcal{L}}^{\mathcal{PV}}(t) \\
EHLPB_{\mathcal{L}}^{\mathcal{PV}}(X)[Y \leftarrow t] &= EHLPB_{\mathcal{L}}^{\mathcal{PV}}(a_Y^*(X)) \\
&= EHLPB_{\mathcal{L}}^{\mathcal{PV}}(X) && X \neq Y \\
EHLPB_{\mathcal{L}}^{\mathcal{PV}}(t_1 \| t_2)[Y \leftarrow t] &= EHLPB_{\mathcal{L}}^{\mathcal{PV}}(a_Y^*(t_1 \| t_2)) \\
&= EHLPB_{\mathcal{L}}^{\mathcal{PV}}(a_Y^*(t_1) \| a_Y^*(t_2)) \\
&= EHLPB_{\mathcal{L}}^{\mathcal{PV}}(a_Y^*(t_1)) \,\hat{\|}\, EHLPB_{\mathcal{L}}^{\mathcal{PV}}(a_Y^*(t_2)) \\
&= EHLPB_{\mathcal{L}}^{\mathcal{PV}}(t_1)[Y \leftarrow t] \,\hat{\|}\, EHLPB_{\mathcal{L}}^{\mathcal{PV}}(t_2)[Y \leftarrow t] \\
EHLPB_{\mathcal{L}}^{\mathcal{PV}}(t_1 ; t_2)[Y \leftarrow t] &= \ldots \\
&= EHLPB_{\mathcal{L}}^{\mathcal{PV}}(t_1)[Y \leftarrow t] \,\hat{;}\, EHLPB_{\mathcal{L}}^{\mathcal{PV}}(t_2)[Y \leftarrow t] \\
EHLPB_{\mathcal{L}}^{\mathcal{PV}}(t_1 + t_2)[Y \leftarrow t] &= \ldots \\
&= EHLPB_{\mathcal{L}}^{\mathcal{PV}}(t_1)[Y \leftarrow t] \,\hat{+}\, EHLPB_{\mathcal{L}}^{\mathcal{PV}}(t_2)[Y \leftarrow t] \\
EHLPB_{\mathcal{L}}^{\mathcal{PV}}(t_1[f])[Y \leftarrow t] &= \ldots \\
&= (EHLPB_{\mathcal{L}}^{\mathcal{PV}}(t_1))[Y \leftarrow t]\widehat{[f]}
\end{aligned}
$$



$Box_{\mathcal{L}}(Y)$

Figure 6.3: The Basic High Level Petri Box for the process variable Y.

As a_Y^* is the unique extension of a_Y, an operator on Extended High Level Petri Boxes which satisfies the above characterisation, and which agrees with a_Y on process variables, will automatically be the correct extension on the extended semantic domain (up to isomorphism).

All that remains is to make a choice for the denotation of a process variable in the extended semantic domain. Although we might use OSAs to limit the choices we wish to make, creativity is still required. In fact the denotation we have chosen for a process variable is essentially that of a Basic Box (cf. Definition 4.7.1), albeit labelled with a process variable. This appears the most natural possibility.

DEFINITION 6.3.6 Given a process variable $Y \in \mathcal{PV}$ define the basic box of Y as the augmented local transition:

$$Box_{\mathcal{L}}(Y) = [\langle\{s_1\}, (.), Y, (.), \{s_2\}\rangle]$$

where s_1 and s_2 are fresh places. Define the denotation of Y in the extended semantic domain to be the equivalence class:

$$EHLPB_{\mathcal{L}}^{\mathcal{PV}}(Y) = [Box_{\mathcal{L}}(Y)]$$

i.e., the equivalence class of the basic box with respect to isomorphism of High Level Petri Boxes. ■ 6.3.6

The denotation of a process variable is illustrated in Figure 6.3.

Given the proximity of the denotation of a process variable to that of a label in the label algebra, the redefinition of the operators on the extended semantic domain is trivial, and we will assume it has been done. This means, for instance, that we will treat $Box_{\mathcal{L}}(Y)$ as a local transition in the application of the auxiliary, and hence top-level, operators.

Note that the application of a relabelling is not defined on a process variable, meaning that we may not produce PrT or P/T net denotations of High Level Petri Boxes in which process variables appear in its original label algebra. Process variables are, thus, a structuring mechanism for High Level Petri Boxes. However, marking theoretic properties of High Level Petri Boxes which contain process variables, can be established by adjoining the collection of process variables to the alphabet and considering the High Level Petri Box over the most permissive label algebra derived from that.

We define our candidate for the lifting of *Proc*-refinements to the semantic domain of High Level Petri Boxes next. Although relatively complex in comparison with the previous operators, it is demonstrably the simplest possible, given the denotation of a process variable and the requirements imposed upon it by the characterisation above. After the definition, we show that the operator satisfies the characteristic equations above, which identifies it as the unique lifting on the extended semantic domain.

6.4 Semantics of Refinement

6.4.1 Local Transition Refinement

That the denotation of a process variable is, essentially, the augment of local transition suggests that we might define a refinement operator on local transitions, and derive full refinement from it.

As usual we will assume we have been given a label algebra $\mathcal{L} = \langle A, R \rangle$.

DEFINITION 6.4.1 *[Local Transition Refinement]* Let B and D be High Level Petri Boxes and l a local transition, not necessarily in $LT(B)$, but such that ${}^\bullet l \cap l^\bullet = \emptyset$ and ${}^\bullet l^\bullet \cap S_D = \emptyset$. Choose a fresh copy of D, D' say[6], which is sociable with B, and define separable auxiliary relations

[6] I.e., $D \equiv D'$ but the places of D' are fresh.

$f_1^l: S_B \to \mathbb{P}S_B \cup \mathbb{P}(S_B \otimes S_{D'})$ and $f_2^l: S'_D \to \mathbb{P}S_{D'} \cup \mathbb{P}(S_B \otimes S_{D'})$ such that

$$f_1^l(s) = \begin{cases} \{s\} \otimes {}^\bullet D' & s \in {}^\bullet l \\ \{s\} \otimes D'^\bullet & s \in l^\bullet \\ \{s\} & \text{otherwise} \end{cases}$$

and

$$f_2^l(s) = \begin{cases} {}^\bullet l \otimes \{s\} & s \in {}^\bullet D' \\ l^\bullet \otimes \{s\} & s \in D'^\bullet \\ \{s\} & \text{otherwise} \end{cases}$$

Define the *local transition refinement of B by D with respect to l*, $B[l \leftarrow D]$, to be the pre-High Level Petri Box such that[7]:

$$\begin{aligned} S_{B[l\leftarrow D]} &= \begin{cases} f_1^l(S_B) \cup f_2^l(S_{D'}) & l \in LT(B) \\ f_1^l(S_B) & \text{otherwise} \end{cases} \\ T_{B[l\leftarrow D]} &= \{f_1^l(t \setminus n.\{\!|l|\!\}) \cup f_2^l(\tau)(l)_{D'} \mid t \in T_B \wedge t(l) = n \wedge \tau \in T_{D'}^{\pm n}\} \\ \lambda_{B[l\leftarrow D]}(s) &= \begin{cases} \lambda_{f_1^l(B)}(s) & s \in f_1^l(S_B) \\ \emptyset & \text{otherwise} \end{cases} \end{aligned}$$

■ 6.4.1

B will be termed the *refined High Level Petri Box*, D the *refining High Level Petri Box* and l the *refined local transition.*

Figures 6.4, 6.5 and 6.6 illustrate the definition of the local transition refinement of a High Level Petri Box. Figure 6.4 gives the generic form of local transition refinement, in terms of the conceptual model of Chapter 1, and shows a High Level Petri Box refined with respect to the local transition which has been affected both by context manipulations and combinational closures.

We note the following of the definition:

1. we choose a fresh copy of D to form the local transitions of the refined High Level Petri Box. The efficacy of this is that by judicious choice of the fresh copy, we may reduce proofs of isomorphism to proofs of equality.

2. the appearance of the pre- and post-contexts of the refined local transition in each of the new transitions formed through local transition refinement ensures that any context

[7] Note that $T_{D'}^{\oplus n} = \{\!|t_1 \oplus \ldots \oplus t_n \mid t_i \in D'|\!\}$, so that $T_{D'}^{\oplus 0} = \{\{\!|\}\!\}$. Using the observation of Definition 4.5.9, we have that $f(\{\!|\}\!) = \{\!|\}\!$. The notation $t(l)_D$ was introduced in Definition 4.5.7.

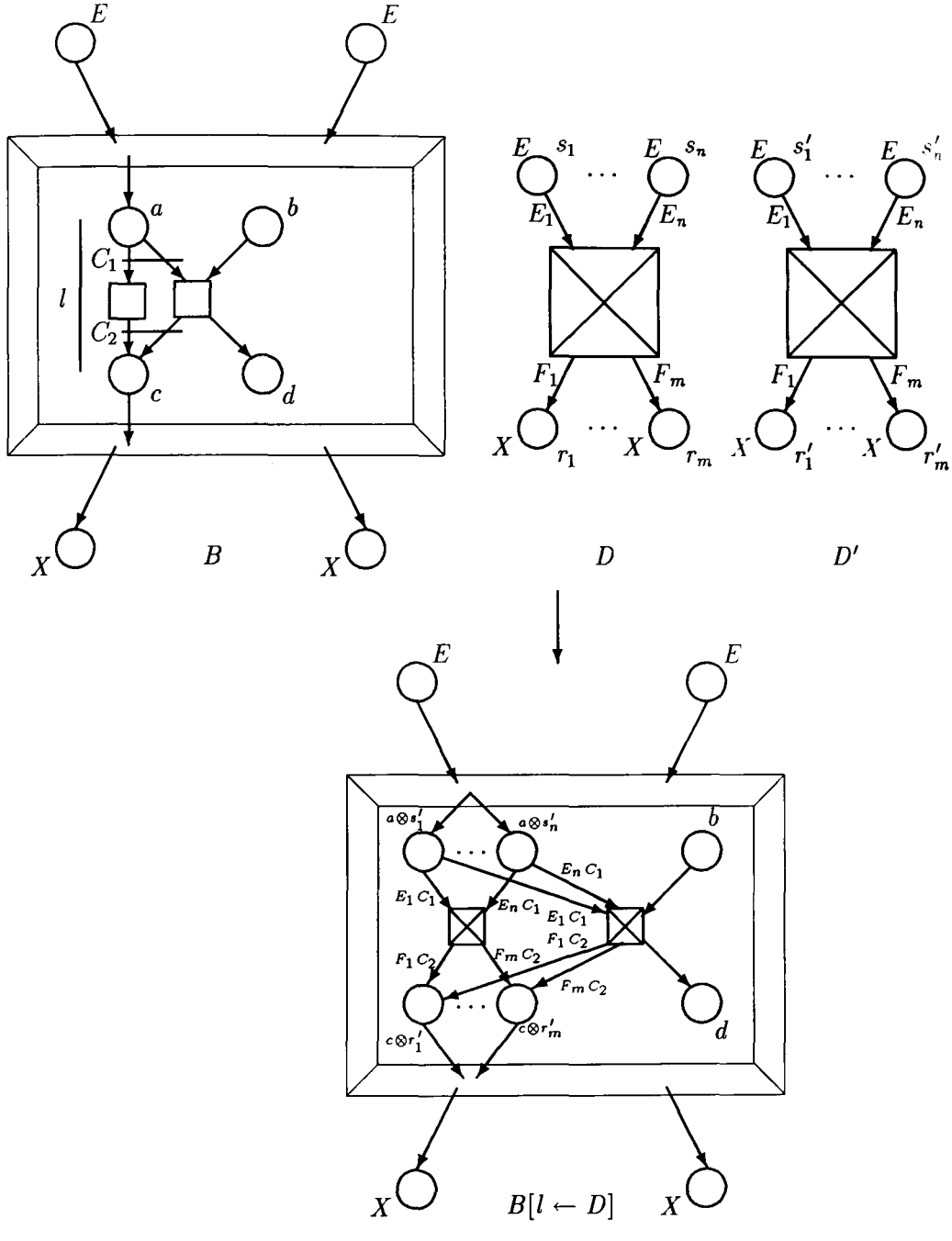


Figure 6.4: Illustration of the local transition refinement of a High Level Petri Box in terms of the conceptual model.

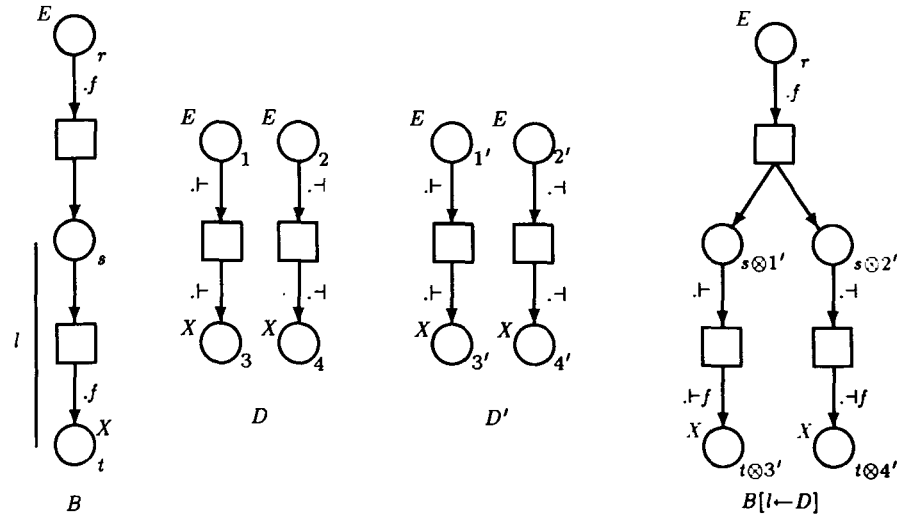


Figure 6.5: Local transition refinement of a High Level Petri Box in which the refining High Level Petri Box contains more than one local transition.

manipulation which has been applied to that local transition has its 'scope' extended to the whole of the refining High Level Petri Box.

3. the appearance of the refined local transition n times in an abstract transition of the refined High Level Petri Box implies that combinations of n abstract transitions from the refining High Level Petri Box are introduced into the refined High Level Petri Box (achieved by requiring that $\tau \in T_{D'}^{\oplus_n}$). Together with Item 2 this will ensure that refinement and relabelling commute (for the extent to which this is true of local transition refinement see Proposition 6.4.11).

4. the assumption of the refined local transition that ${}^\bullet l \cap l^\bullet = \emptyset$ implies that the auxiliary relations are well-defined. Whereas this makes the operator partial on the domain of High Level Petri Boxes (as we do not prevent local transitions from having intersecting pre- and post-sets) the case of refining a syntactically generated High Level Petri Box with respect to such a local transition does not arise.

5. we do not require that the refined local transition appears in the refined High Level Petri Box, nor even that they share places. Such a requirement will be seen to be too restrictive when we come to demonstrate the homomorphic nature of refinement.

6. the place labelling of the refined High Level Petri Box is retained; that of the refining High Level Petri Box is discarded.

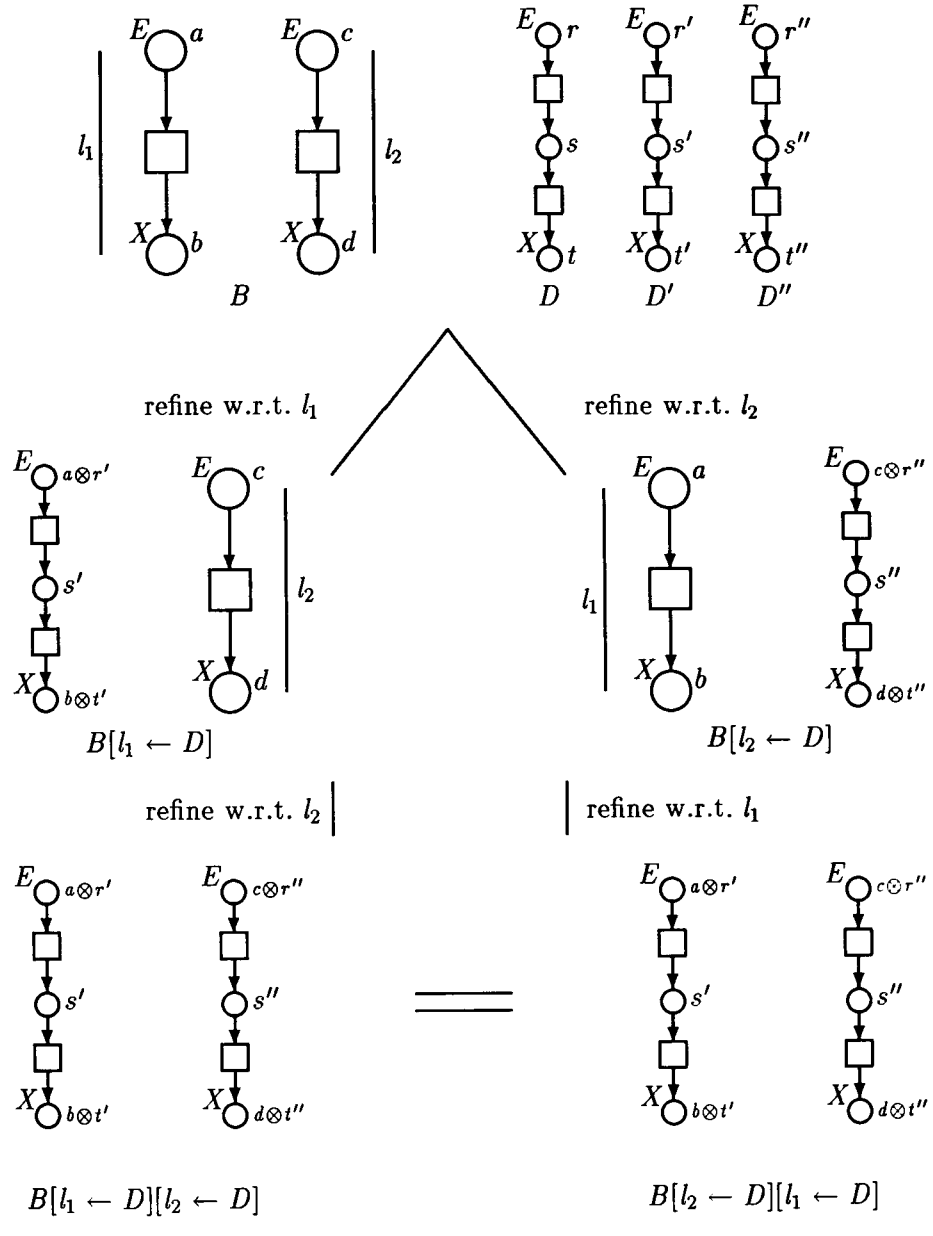


Figure 6.6: Illustrating that (with the judicious choice of the copies (D' and D'') of refining High Level Petri Box D) iterated local transition refinement of a High Level Petri Box is independent of the order.

Given that there will be, in general, many instances of local transition refinement in the following we will often decorate the auxiliary relations with an extra subscript, which is the 'name' of the refined High Level Petri Box. For instance, in the refinement of a High Level Petri Box B_1 by D with respect to the local transition l we will write $f^l_{B_1,1}$ for f^l_1 and $f^l_{B_1,2}$ for f^l_2.

6.4.2 Properties of Local Transition Refinement

In this section we demonstrate a number of technical properties of local transition refinement that enable a simple derivation of (full) refinement and of its syntactic nature.

We will assume, throughout the remainder of this section, that B, B_1, B_2 and D are High Level Petri Boxes and that l is a local transition. Moreover, unless otherwise stated, B, B_1 and B_2 are to be assumed pairwise sociable.

6.4.2.1 Basic Properties

Local transition refinement preserves the High Level Petri Box-ness of its operands. The arguments are similar to those of Chapter 4.

PROPOSITION 6.4.2 $B[l \leftarrow D]$ is a High Level Petri Box whenever B and D are. □6.4.2

Sociability propagates through refinement:

PROPOSITION 6.4.3 $B_1[l \leftarrow D]$ and B_2 are sociable whenever B_1, B_2 and D are pairwise sociable. □6.4.3

6.4.2.2 Places

When the refined local transition does not appear in the refined High Level Petri Box the effect of refinement reduces to the lifting of the first auxiliary relation, f_1^l; when it(s augment) is the refined High Level Petri Box it reduces to the lifting of the second auxiliary relation, f_2^l:

PROPOSITION 6.4.4

1. $l \notin LT(B)$ implies $B[l \leftarrow D] = f_1^l(B)$,
2. $[l][l \leftarrow D] = f_2^l(D')$.
3. ${}^\bullet l^\bullet \cap S_B = \emptyset$ implies $B[l \leftarrow D] = B$. □6.4.4

Proof: Items 1 and 2 follow directly from the definitions. Item 3 is a simple corollary of Item 1, as ${}^\bullet l^\bullet \cap S_B = \emptyset$ implies that f_1^l is the identity. ■6.4.4

6.4.2.3 Local Transition Refinement modulo Isomorphism

That isomorphism is a congruence with respect to the top level operators introduced so far is an important result, allowing us to abstract away from the identities of places. It follows from the assumed sociability of the respective operands, and by careful choice of refining High Level Petri Box:

PROPOSITION 6.4.5

1. Local transition refinement is independent of the choice of copy of D, up to isomorphism, as long as that choice is sociable with B,
2. Let $\sigma: B \equiv B'$. Then[8] $B[l \leftarrow D] \equiv B'[\sigma(l) \leftarrow D]$. □ 6.4.5

6.4.2.4 Local Transitions

The local transitions of a refined High Level Petri Box are those of the original, other than the refined local transition, together with those of the refining High Level Petri Box, if the refined local transition appeared in the refined High Level Petri Box. The following characterisation is evident[9]:

PROPOSITION 6.4.6

$$LT(B[l \leftarrow D]) = \begin{cases} f_1^l(LT(B) \setminus \{l\}) \cup f_2^l(LT(D(l))) & l \in LT(B) \\ f_1^l(LT(B)) & \text{otherwise.} \end{cases}$$

□ 6.4.6

6.4.2.5 Commutativity of Local Transition refinement with Itself

The requirement that the refining and refined High Level Petri Boxes are sociable implies that we should choose a copy of D so as to be able to iterate local transition refinement. If we are sufficiently careful with the choice, we may always ensure that:

[8] Where σ applied outside of its domain is to be considered the identity.
[9] $D(l)$ was defined in Definition 4.5.7.

PROPOSITION 6.4.7 Let B and D be as in the statement of Definition 6.4.1. Let l_1, $l_2 \in LT(B)$ such that ${}^\bullet l_i \cap l_i{}^\bullet = \emptyset$ and ${}^\bullet l_i{}^\bullet \cap S_D = \emptyset$, $i = 1,2$. Then $B[l_1 - D][l_2' \leftarrow D] = B[l_2 \leftarrow D][l_1' \leftarrow D]$ where $l_2' = f_1^{l_1}(l_2)$ and $l_1' = f_1^{l_2}(l_1)$. □ 6.4.7

Proof: The altering of the refined local transitions is necessary so as to compensate for the case when ${}^\bullet l_1{}^\bullet \cap {}^\bullet l_2{}^\bullet \neq \emptyset$. Otherwise the result is straightforward, requiring only a careful choice of the copy of D and the commutativity and associativity of $\otimes$ and multiset union. ■ 6.4.7

We note, in particular, that, if ${}^\bullet l_1{}^\bullet \cap {}^\bullet l_2{}^\bullet = \emptyset$, then $f_1^{l_1}(l_2) = l_2$ and $f_1^{l_2}(l_1) = l_1$. This observation will be useful in the subsequent development of the recursive operator.

6.4.2.6 Commuting Local Transition Refinement with the Top Level Operators

The homomorphism condition required that a (*Proc*-)refinement commutes with the top-level operators of the algebra (at least, up to isomorphism). That our candidate for the refinement operator has this property is shown next:

PROPOSITION 6.4.8 Let $l \in LT(B_1 \| B_2)$ so that either

1. $l = l'(.\vdash)_{B_1}$ for $l' \in LT(B_1)$, or
2. $l = l'(.\dashv)_{B_2}$ for $l' \in LT(B_2)$.

Then $(B_1 \| B_2)[l \leftarrow D] = (B_1[l' \leftarrow D]) \| (B_2[l' \leftarrow D])$. □ 6.4.8

Proof: As B_1 and B_2 are sociable, $LT(B_1) \cap LT(B_2) = \emptyset$ so that, as a simple extension of Proposition 6.4.4, either $(B_1 \odot .\vdash)[l \leftarrow D] = B_1 \odot .\vdash$ or $(B_2 \odot .\dashv)[l \leftarrow D] = B_2 \odot .\dashv$. The result follows. ■ 6.4.8

PROPOSITION 6.4.9 Let $l \in LT(B_1;B_2)$ so that either

1. $l = ;_1(l')$ for $l' \in LT(B_1)$, or
2. $l = ;_2(l')$ for $l' \in LT(B_2)$.

Then $(B_1;B_2)[l \leftarrow D] = (B_1[l' \leftarrow D]);(B_2[l' \leftarrow D])$. □ 6.4.9

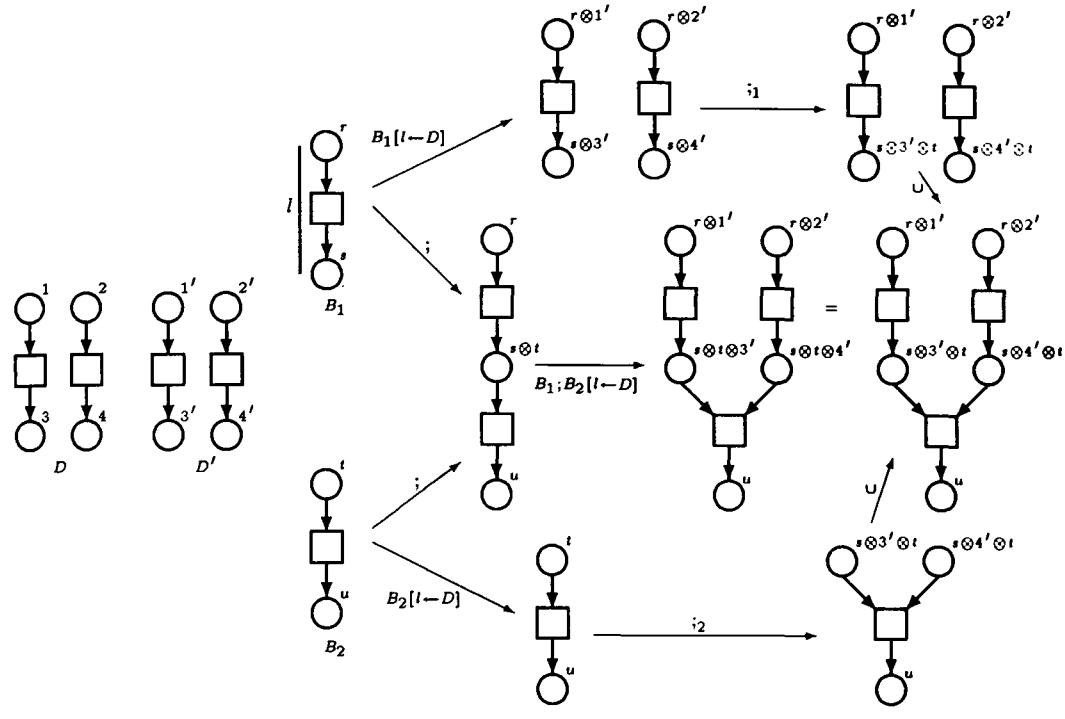


Figure 6.7: Illustrating the proof of Proposition 6.4.9 when $l^\bullet \subseteq B_1{}^\bullet$. Note that we must choose D' simultaneously sociable with both B_1 and B_2. Place annotations have been omitted for clarity.

Proof: Again we have that $LT(B_1) \cap LT(B_2) = \emptyset$. Assume, without loss of generality, that $l \in LT(B_1)$. If $l^\bullet \cap B_1{}^\bullet = \emptyset$, then $B_1[l \leftarrow D]^\bullet = B_1{}^\bullet$ and the result follows easily. When $l^\bullet \subseteq B_1{}^\bullet$ we must be a little careful, as then $B_1[l \leftarrow D]^\bullet = (B_1{}^\bullet \setminus l^\bullet) \cup l^\bullet \otimes D^\bullet$, and we must verify that the application of $;_2$ to B_2 as defined between $B_1[l \leftarrow D]$ and B_2 rather than between B_1 and B_2 is compensated for. However, this follows from the commutativity and associativity of $\otimes$. ■ 6.4.9

PROPOSITION 6.4.10 Let $l \in LT(B_1+B_2)$ so that either

1. $l = +_1(l')(.\rfloor)_{B_1}$ for $l' \in LT(B_1)$, or
2. $l = +_2(l')(.\rfloor)_{B_2}$ for $l' \in LT(B_2)$.

Then $(B_1+B_2)[l \leftarrow D] = (B_1[l' \leftarrow D])+(B_2[l' \leftarrow D])$. □ 6.4.10

Proof: We may assume that B_1 and B_2 are sociable so that $LT(B_1) \cap LT(B_2) = \emptyset$. The result follows as a combination of those of Propositions 6.4.8 and 6.4.9. ■ 6.4.10

The assumption that B_1 and B_2 are sociable is only sufficient to ensure that the internal places of the High Level Petri Boxes are disjoint, after the application of the auxiliary relations $+_1$ and $+_2$, as was illustrated in Figure 4.9 (page 88). This gives the motivation for the use of the nominal relabellings $\lfloor$ and $\rfloor$ which produce distinct local transitions in B_1+B_2 when $B_1 = [l_1]$, $B_2 = [l_2]$ with ${}^C l_1 = {}^C l_2$, $\overline{l_1} = \overline{l_2}$, $l_1{}^C = l_2{}^C$.

PROPOSITION 6.4.11 For $l \in LT(B)$, $B[f][l(.f)_B \leftarrow D] = B[l \leftarrow D][f]$. □ 6.4.11

Proof: The annotation of the entry and exit arcs of D by the contexts annotating the refined local transitions, together with the addition of n copies of abstract transitions of D' to each abstract transition of B in which l appears n times is sufficient to ensure isomorphism. With the careful choice of D', the copy of D used in the refinement, we may strengthen this to equality. ■ 6.4.11

6.4.3 Full Refinement

From Proposition 6.4.7, we may commute the order of applications of local transition refinement and retain the same High Level Petri Box. This allows the convenient derivation of (full) refinement from local transition refinement and permits the simple transfer of the properties between auxiliary and top-level operators.

The derivation of refinement of a (syntactically generated) High Level Petri Box B with respect to (process variable) Y and (refining High Level Petri Box) D, follows from the identification of the set of the local transitions of B annotated by Y and, choosing an arbitrary order, applying local transition refinement to each in turn.

DEFINITION 6.4.12 Let B be a High Level Petri Box and $Y \in \mathcal{PV}(B)$. Define $L_Y(B) = \{l \in LT(B) \mid \overline{l} = Y\}$. ■ 6.4.12

For brevity, we will often refer to $l \in L_Y(B)$ as a *Y-local.*

Full refinement, our candidate for the lifting of a *Proc*-refinement to the semantic domain, is defined as:

DEFINITION 6.4.13 Let B, D be syntactically generated High Level Petri Boxes and suppose $Y \in \mathcal{PV}$. Define

$$B[Y \leftarrow D] = B[L_Y(B) \leftarrow D]$$

where

$$\begin{aligned} B[\emptyset \leftarrow D] &= B \\ B[\{l_1, l_2, \ldots, l_n\} \leftarrow D] &= B[l_1 \leftarrow D][\{l_2', \ldots l_n'\} \leftarrow D] \end{aligned}$$

and $l_i' = f_1^{l_1}(l_i)$. ■ 6.4.13

That Definition 6.4.13 provides a unique value for $B[Y \leftarrow D]$ (up to isomorphism) is a corollary of the commutativity of local transition refinement with itself (Proposition 6.4.7), and the following property of the auxiliary relations of Definition 6.4.1, which implies that using the l_i' to refine $B[l \leftarrow D]$ in the 'continuation' of the iterated refinement does not unduely affect the result.

LEMMA 6.4.14 Let l_1, l_2 and $l_3 \in L_Y(B)$ be distinct local transitions. Then $f_1^{f_1^{l_1}(l_2)}(f_1^{l_1}(l_3)) = f_1^{f_1^{l_2}(l_1)}(f_1^{l_2}(l_3))$. □ 6.4.14

Proof: Follows from a simple consideration of cases, the commutative nature of $\odot$, and a simple technical manipulation of the various auxiliary relations. ■ 6.4.14

6.4.3.1 Commutativity of Refinement with the High Level Petri Box Operators

Another benefit of the iterative definition of refinement may be seen in combination with the properties of $L_Y(B)$, the local transitions of a High Level Petri Box with the (process variable) Y as label. Given that the operand High Level Petri Boxes of a top-level operator are required to be sociable, we may partition the constituent local transitions of the composition into, essentially, those of the operands. For instance, in the case of sequential composition we have that $L_Y(B_1;B_2) = L_Y(;_1(B_1)) \cup L_Y(;_2(B_2))$, *etc.* As L_Y does not depend on the details of the local transitions which carry the label Y, to refine the composition we need only refine the operands and compose:

THEOREM 6.4.15 For B_1, B_2 and D syntactically generated High Level Petri Boxes and for $Y \in \mathcal{PV}$:

1. $(B_1 \| B_2)[Y \leftarrow D] = B_1[Y \leftarrow D] \| B_2[Y \leftarrow D]$
2. $(B_1;B_2)[Y \leftarrow D] = B_1[Y \leftarrow D];B_2[Y \leftarrow D]$
3. $(B_1+B_2)[Y \leftarrow D] = B_1[Y \leftarrow D]+B_2[Y \leftarrow D]$

4. $B[f][Y \leftarrow D] = B[Y \leftarrow D][f]$ □ 6.4.15

Proof: The result follows easily from the corresponding results for local transition refinement, a careful choice of copies of D, and the following characterisations of L_Y for the various cases:

1. $L_Y(B_1 \| B_2) = L_Y(B_1 \odot .\vdash) \cup L_Y(B_2 \odot .\dashv)$.
2. $L_Y(B_1;B_2) = L_Y(;_1(B_1)) \cup L_Y(;_2(B_2))$.
3. $L_Y(B_1+B_2) = L_Y(+_1(B_1) \odot .\lfloor) \cup L_Y(+_2(B_2) \odot .\rfloor)$.
4. $L_Y(B[f]) = L_Y(B \odot .f)$. ■ 6.4.15

With Theorem 6.4.15 we have shown that refinement on the semantic domain of High Level Petri Boxes corresponds precisely to that of a *Proc*-refinement on the free construction.

The reader will note that in the proof of the above, we have not required any properties of refined or refining High Level Petri Boxes (other than the local transition to be refined on, l, is such that ${}^\bullet l \cap l^\bullet = \emptyset$, and that ${}^\bullet l^\bullet \cap S_D = \emptyset$ ensuring the well-definedness of the refinement construct but which, through renaming, is trivially enforceable). The properties of this section will thus extend to other classes of High Level Petri Boxes, such as those introduced in the next section in which the recursive construct has been applied.

6.5 Recursion

Recursion and refinement are closely linked when a fix-point semantics is chosen as together they allow one to answer the question:

is there a solution Q to the equation

$$Q \doteq P[Y \leftarrow Q] \tag{6.1}$$

i.e., by refining Y by Q in P do we again arrive at something with the same meaning (determined by the behavioural or structural equivalence $\doteq$) as Q? Given certain properties of the semantic domain (essentially, that it forms a partially ordered set with directed joins) the fix-point theory of Scott [Sco82] may be used to provide unique (but, in general, infinite) structural solutions to the equation. This is, essentially, the approach used for the low level Petri Box model and more

recently in the A-nets of [DK95]. Throughout the remainder of this chapter, we will refer to Y in Equation 6.1 as the 'recursed-on variable', P as the 'recursive body' and Q as 'the solution'.

Equation 6.1 may be used to finitely encode certain infinite behaviours. In [Min72], Minsky shows that the collection of behaviours defined by such recursively defined terms correspond to those generated by Turing machines. This collection of behaviours is, as Hack [Hac75] first showed, outside of those that a low level Petri net model can describe. A corollary is that any semantics which provides a solution to Equation 6.1 in P/T nets must contain nets with infinite structure.

Whereas there are mechanisms which can be added to low level Petri net models which permit finite denotations of such terms, such as *inhibitor arcs* and *priorities* [JK91], such models have not been widely investigated (generally, the extensions are translated into vanilla P/T nets). Moreover, as they complicate the graph-theoretic structure of Petri nets, they are generally less tractable.

Another solution, investigated in this section, is to look to our high level Petri net model, and to encode those components of control flow which cannot be expressed within finite nets within the individuated tokens.

Such an approach appears first to have been systematically explored by [Tau89]. Taubner also argues there that there is no fully expressive finite semantics of TCSP *even within a high level net model* due to the TCSP communication protocol synchronisations between unbounded numbers of parallel processes may be effected. We concur with Taubner's analysis wishing only to add the observation that it is only the set of transitions of the high level net which is required by his arguments to be infinite; the set of places may be finite.

And indeed, all operators on High Level Petri Boxes, including recursion, preserve the finiteness of the set of places of a High Level Petri Box. Also, all operators preserve the cardinality of the set of local transitions of a High Level Petri Box. It is only relabelling which produces a cardinality change, in producing a countable infinity of abstract transitions.

Moreover, Proposition 4.8.8 (page 94) continues to hold of High Level Petri Boxes to which the recursive construct has been applied, so that we may continue to use the Decoupling Lemma (Lemma 3.4.4, page 55) for finitary proofs of properties which are positive monotonic properties in the permissiveness of a label algebra. Indeed, the behavioural properties of the recursive construct established in the next chapter are proven using such techniques.

6.5.1 Semantics of Recursion

To describe such equations as that which appears above we will introduce a new syntactic construct, recursion (the notation for which should, however, be familiar from the literature). Our Q (our solution to Equation 6.1 for High Level Petri Boxes) we will denote as $\mu Y.P$. It is defined in Section 6.5.7. The proof that it is a solution to Equation 6.1 is given in Chapter 7 (Section 7.3).

The recursive construct is derived from two auxiliary operators: *guarding* which ensures that the recursive operator is *constructive* in its High Level Petri Box operand; and *winding* which performs the folding of structure mentioned above, allowing recursive invocations of a process to use the same High Level Petri Box structure.

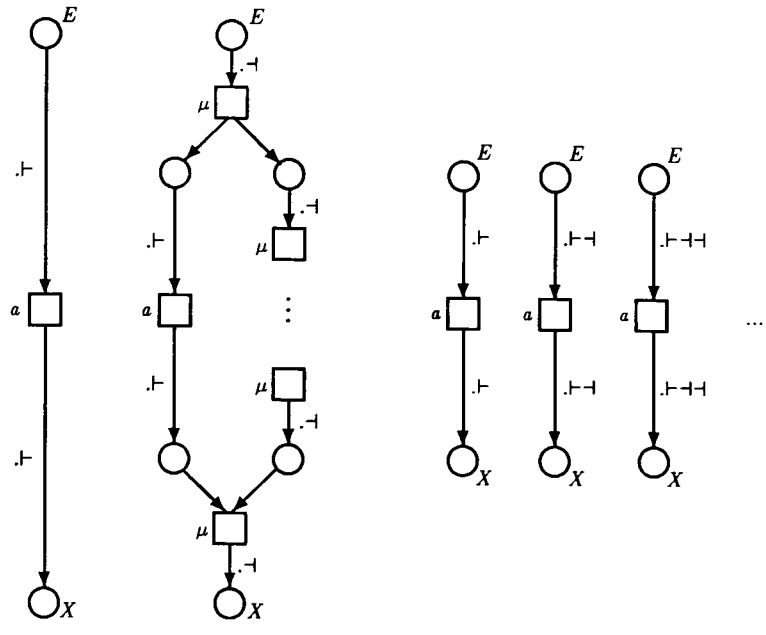
In this section we give the semantics of our recursive operator. We have been careful to provide definitions which preserve as many properties as possible of High Level Petri Boxes (this will explain some slight complications in the definitions). However, we lose the normal form denotation of High Level Petri Boxes which existed of syntactically generated High Level Petri Boxes due to the necessity of the introduction of loops into the semantics.

6.5.2 Pre- and Post-Guarding and Winding

The (pre-)guarding of a recursive call is predicated on the observation that we are able to prefix a silent action to a process without altering its observable behaviour. In Definition 6.5.1 we introduce a silent action, denoted μ (to distinguish it from the τ of CCS), into the alphabets of all label algebras, with communication properties (similar to τ) as defined by Definition 6.5.2.

6.5.2.1 Why Post-Guarding?

The infinite terms $a\|(\mu;(a\|\cdots);\mu)$ and $a\|a\|\cdots$ are behaviourally equivalent (when μ is silent). Their respective denotations are given in Figure 6.8. In Figure 6.8(a) the arc annotations of the abstract transitions labelled μ and a are regular: each pre-guarding μ has a pre-context of $.\dashv$ and an empty post-context; each post-guarding μ an empty pre-context and a post-context of $.\dashv$; each a has pre- and post-contexts $.\vdash$. In Figure 6.8(b) arc annotations (in particular the post-contexts) are not regular: the annotations of the (a-labelled) abstract transitions are of the form $.\vdash$ or $.\vdash\dashv^n$. The regularity of the arc annotations of the first High Level Petri Box will



a. b.

Figure 6.8: (a) Regular and (b) irregular denotations of the same (observable) behaviour.

allow us to 'fold' it into a finite structure with the same behaviour[10]. This is not possible with the second High Level Petri Box.

6.5.3 Winding

As already mentioned, to produce a cardinality preserving semantics of High Level Petri Box terms, we fold together certain repeated structures. One possibility for doing this would be to develop a Scott domain [Sco82] on top of High Level Petri Boxes, produce the approximants to the behaviour of the fix-point, and show that the folding commutes with the formation of chains. However, for our recursive construct we will perform the folding monolithically on the original process term. For obvious reasons we will call this process *winding.*

Winding operators appear naturally in structured operational semantics of *prefix* based approaches. Our operator is a generalisation of this form in which we consider also the terminal state of a process. So, in addition to the practice (of the literature) of identifying the initial state of the recursive body (the P of Equation 6.1) with the state directly preceding the recursed-on variable (Y of Equation 6.1), we will also identify the terminal state of the recursive body with

[10] Actually, we will add more structure first.

the state directly succeeding the recursed-on variable.

This does not, as yet, bring into play any particular properties of high level Petri net models, and we note that if winding were sufficient it would be simple matter to produce a Turing expressive labelled finite 1-safe P/T nets. However, such nets are not Turing expressive so that this cannot be the case. Examples of two ways in which winding can fail (together with solutions) are illustrated in Figure 6.9 when the term to be wound is $(a;Y;b)+c$. Winding this term should produce the (complete) behaviours described by the set of strings $\{a^n cb^n \mid n \in \mathbb{N}\}$. In Figure 6.9(b) a 'best-try' denotation[11] of the winding is given. This net generates the language $\{a^n cb^m \mid n, m \in \mathbb{N}\}$—we are not able to record the depth of the recursive call so that we do not know how may times to allow it to unwind.

By adding (what we will call) an *accounting place* we allow this information to be recorded, but at the cost of moving to unbounded P/T nets. An accounting place is added to Figure 6.9(c), with resulting language $\{a^n cb^m \mid m \leq n \in \mathbb{N}\}$.

Unfortunately the behaviours of Figure 6.9(c) are not preserved through composition: by composing the net 'in sequence' with the action d, as shown in Figure 6.9(d), the language generated is not $\{a^n cb^n d \mid n \in \mathbb{N}\}$ as we might hope, but $\{a^n cb^m d \mid m \leq n \in \mathbb{N}\}$: we are not able to disable the (initial) transition of the causally inferior net until the recursion of the first is *fully* unwound. A solution is given in Figure 6.9(e), at the cost of introducing inhibitor arcs.

The same problems are encountered in High Level Petri Boxes. The solutions we provide are those of the figure: we use a device equivalent to inhibitor arcs, but without moving outside of the High Level Petri Box model.

6.5.4 Silent Labels

As has already been mentioned, we will guard a recursive call with the silent action μ. To this end, we introduce a silent action into the label universe, and extend each label algebra accordingly. Recall that $\mathbb{L}$ is the label universe (as defined in Section 2.1.1) and that $\mathcal{PV}$ is the collection of process variables (as defined in Definition 6.3.4).

DEFINITION 6.5.1 Let $\mu \notin \mathbb{L}^{\oplus} \cup \mathcal{PV}$. Define the *$\mu$-Extended Label Universe*, $\mathbb{L}_\mu$, as $\mathbb{L} \cup \{\mu\}$. ■ 6.5.1

[11] In terms of finite, 1-safe P/T nets.

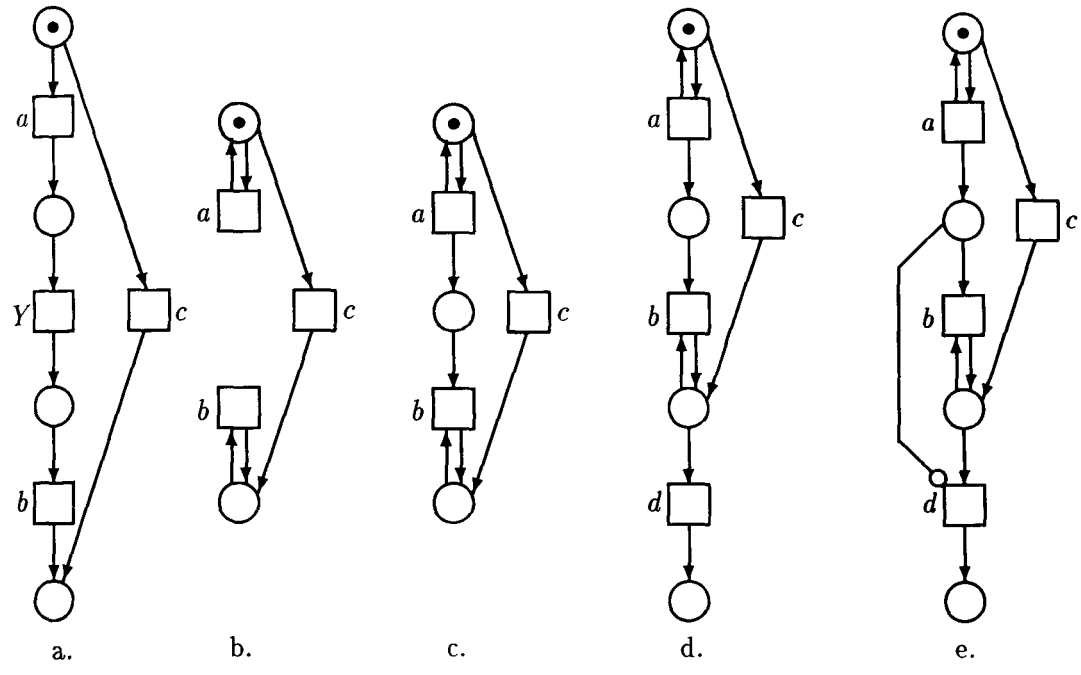


Figure 6.9: (a) A low level net 'denotation' of the term $(a;Y;b)+c$, (b) a labelled 1-safe P/T net which generates the collection of strings $\{a^n cb^m \mid n, m \in \mathbb{N}\}$, (c) a labelled unbounded P/T net which generates the collection of strings $\{a^n cb^m \mid m \leq n \in \mathbb{N}\}$, (d) sequentially composed with another net, generating the language $\{a^n cb^m d \mid m \leq n \in \mathbb{N}\}$ and (e) in which an *inhibitor arc* provides a solution.

As $\mu \notin \mathbb{L}^{\oplus}$, a label algebra may be extended to include μ as a label, and retain its action on non-μ labels:

DEFINITION 6.5.2 Let $\mathcal{L} = \langle A, R \rangle$ be a label algebra. Define the μ-extension of $\mathcal{L}$, $\mathcal{L}_\mu = \langle A_\mu, R_\mu \rangle$ where $A_\mu = A \cup \{\mu\}$ and such that for each $f \in R$, there is a relabelling $f_\mu \in R_\mu$ with:

$$f_\mu(k) = \begin{cases} \mu & k = \mu \\ \mathbf{0} & k = \mu \oplus l_1 \oplus \cdots \oplus l_n,\ n \geq 1,\ \{l_1, \ldots, l_n\} \subseteq A_\mu \\ f(k) & \text{otherwise.} \end{cases}$$

■ 6.5.2

Corresponding to each $f \in R$ there is a relabelling $f_\mu \in R_\mu$ in $\mathcal{L}_\mu$ whose action on labels in which μ does not appear is the same as that of f; which allows μ singly; but prevents any attempted synchronisation in which μ appears.

In common with other approaches, the introduction of silent labels into our algebra allows us to extend notions of behaviour with *observable behaviour*. The construction of an observational transition and step sequence semantics is straightforward, and we do not do it here. However, we must be careful in our algebraic manipulations of the silent action to ensure that the complications associated with it do not arise. We mention and deal with these as they arise in the sequel.

6.5.5 The Guard Auxiliary Operator

Guarding adds a *call/return* mechanism to a High Level Petri Box by both pre- and post-fixing it with a silent action.

For reasons which will become clear, it will be convenient to allow the guarding of a superset of High Level Petri Boxes, those termed *able pre-High Level Petri Boxes*:

DEFINITION 6.5.3 Let P be a pre-High Level Petri Box. Then P is an *able pre-High Level Petri Box* if P satisfies Properties 1 and 3 of Definition 2.5.15 (page 31). ■ 6.5.3

In an able pre-High Level Petri Box entry places may have incoming arcs and exit places may have outgoing arcs. We show later that the guarding of an able pre-High Level Petri Box returns a High Level Petri Box.

Recall that $\perp$ is the nominal relabelling introduced in Chapter 2 and that ϵ is the unit pre-High Level Petri Box of Definition 4.9.1.

DEFINITION 6.5.4 *[Guard]* Let P be an able pre-High Level Petri Box over a label algebra L_μ. Define the *guarding of* P, $\Gamma(P) = (Box_{\mathcal{L}_\mu}(\mu);(P \cup \epsilon);Box_{\mathcal{L}_\mu}(\mu)) \ominus .\perp$. ■ 6.5.4

Note:

1. The mixing of top-level and auxiliary operators in the definition allows it to be abbreviated somewhat and indicates more clearly the structure of the guarded (pre-)High Level Petri Box;

2. the definition of the guarding auxiliary uses the unit pre-High Level Petri Box to provide the accounting mechanism. As we have mentioned, through compositions involving ϵ we are in danger of moving outside of the class of High Level Petri Boxes. This does not happen as we prevent an immediate subsequent context manipulation being applied:

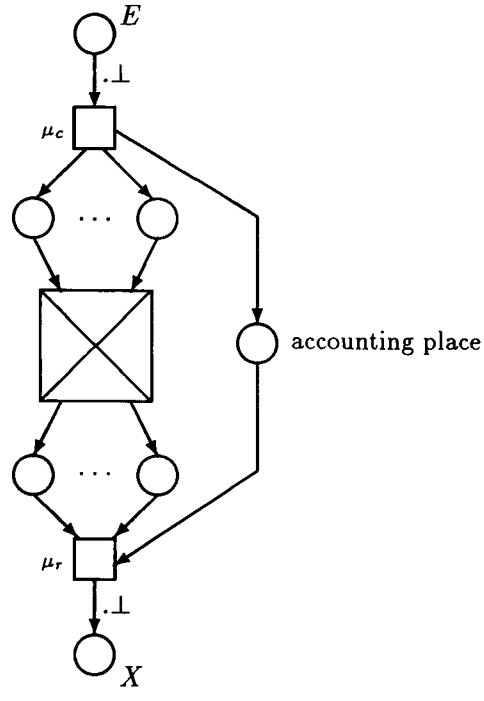


Figure 6.10: The guarding of a (pre-)High Level Petri Box in terms of the conceptual model.

3. it will be convenient to be able to notationally distinguish the two occurrences of μ in the guarding of a High Level Petri Box: the basic box forming the pre-guard of the operand will be referred to as μ_c (for *c*all), the post-guard will be referred to as μ_r (for *r*eturn);

4. guarding provides a single accounting place for each μ_c/μ_r pair. Moreover, as by definition $\perp$ may not appear as a relabelling within a label algebra, it is only through guarding that the nominal relabelling may be introduced into a High Level Petri Box, and then only as a prefix of the pre-context of a μ_c or the post-context of a μ_r;

5. if s the accounting place, $t_c = \{\!|l_c|\!\}$ the call transition, and $t_r = \{\!|l_r|\!\}$ the return transition introduced through an application of the guarding operator, then l_c (respectively, l_r) is the unique local transitions of a guarded High Level Petri Box such that $s \in l_c^\bullet$ (respectively, $s \in {}^\bullet l_r$).

Figure 6.10 illustrates the guarding of a High Level Petri Box in terms of the conceptual model. As is clear from the figure, from a standard initial marking d, the firing of μ_c produces a marking $d(.\perp)$ of the 'entry places' of its operand, together with the marking of the accounting place by the same token. As $\perp$ is a nominal relabelling, the behaviours of a High Level Petri Box from standard initial markings of d and of $d(.\perp)$ are the same. For the termination of the guarded

High Level Petri Box, it is necessary that the operand terminates covering its 'exit places' with $d(.\perp)$ as otherwise the transition labelled μ_r will not be enabled. If this is the case, whence we may fire μ_r and the standard terminal marking d of the guarded High Level Petri Box is restored.

If, for some reason, the initial marking of the guarded High Level Petri Box was unsafe there would be the possibility of the tokens 'over-taking' one another in their passage through the net. However, we show in Chapter 7 that all reachable markings of High Level Petri Boxes are safe from standard initial markings, so that this problem does not arise.

Guarding produces High Level Petri Boxes from able pre-High Level Petri Boxes:

PROPOSITION 6.5.5 For P an able pre-High Level Petri Box, $\Gamma(P)$ is a High Level Petri Box. □ 6.5.5

Proof: Property 1 of Definition 2.5.15 is established anew by the addition of the new entry and exit interface. Property 3 extends to $\Gamma(P)$ as the local transitions we have added share the property.

Property 2 holds of $\Gamma(P)$ as $LT(\Gamma(P)) = LT(P) \cup LT(\mu_c) \cup LT(\mu_r)$ so that ${}^\bullet LT(\Gamma(P)) \setminus LT(\Gamma(P))^\bullet = {}^\bullet\mu_c \setminus \mu_r{}^\bullet = {}^\bullet\Gamma(P)$. The other case is symmetric. ■ 6.5.5

6.5.5.1 Y-guarded High Level Petri Boxes

That observational equivalence does not extend to a congruence without further conditions being imposed on initial actions means we must be careful that the prefixing by a silent action to a process contained in the guarding auxiliary does not alter the behaviour of the process. The restriction we must impose is derived from that well-known from the literature: we ensure that the subject process variable is not in conflict, so that refinement by the silent action does not alter the initial commitment behaviour. Through the annotations of the operands of a choice composition, this property is easily checkable.

DEFINITION 6.5.6 Let B be a High Level Petri Box over some label algebra $\mathcal{L}_\mu$, and let Y be a process variable. Then B is *Y-conflict-free* if, for each $l \in L_Y(B)$, neither $\lfloor$ nor $\rfloor$ appears in ${}^c l$. ■ 6.5.6

We note that Y-conflict-freeness in combination with the properties of the definition of the μ-extended label algebra is sufficient to prevent changes in the commitment behaviour even

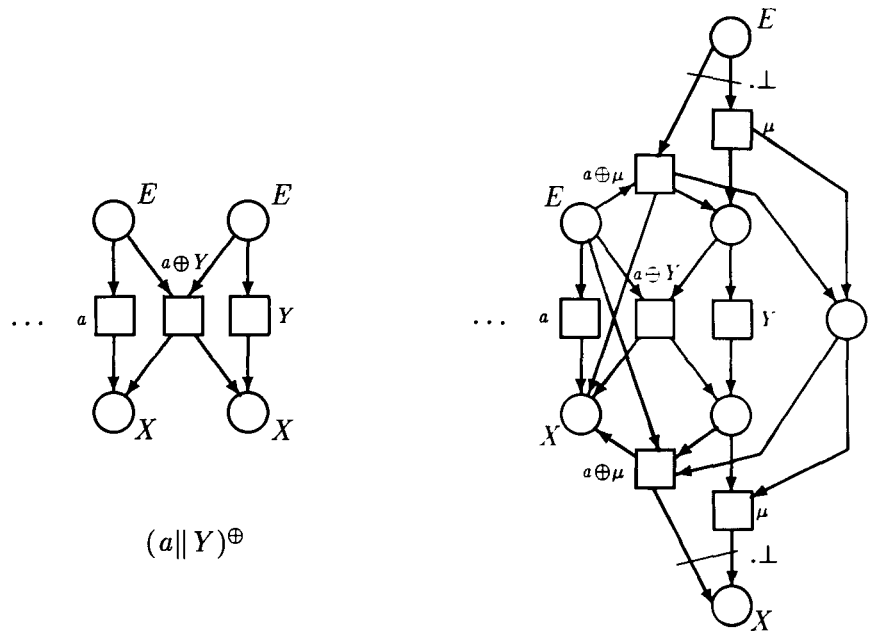


Figure 6.11: The Y-guarding of a High Level Petri Box in which a Y-local has been involved in a synchronisation.

when the process variable appears as part of a synchronisation. Note that we do *not* require Y-conflict-freeness to be able to apply the recursive construct to a High Level Petri Box.

We may guard the process variable of a Y-conflict-free High Level Petri Box without problem:

DEFINITION 6.5.7 Let Y be a process variable and B a Y-conflict-free High Level Petri Box over some label algebra $\mathcal{L}_\mu$. The *Y-guarding* of B is defined as $B[Y \leftarrow \Gamma(Y)]$

A High Level Petri Box which is the Y-guarding of some other High Level Petri Box will be termed *Y-guarded.* ■ 6.5.7

The Y-guarding of a High Level Petri Box produces, for each local transition $l \in L_Y(B)$, an accounting place s^l. We define $Acc_Y(B) = \{s^l \mid l \in L_Y(B)\}$. We will also wish to refer to all accounting places which appear within a High Level Petri Box. As accounting places appear 'between' call/return pairs, we define $Acc(B) = \bigcup\{l^\bullet \cap {}^\bullet l' \mid \bot \leq {}^C l \wedge \bot \leq l'^C\}$.

Note that, if $L_Y(B) = \emptyset$, B is its own Y-guarding. Also, if B is the Y-guarding of B', then for any High Level Petri Boxes D, $B[Y \leftarrow D] \equiv B'[Y \leftarrow \Gamma(D)]$. The second observation is useful in that, if D is only an able pre-High Level Petri Box, then $B[Y \leftarrow D]$ will not be defined whereas.

as $\Gamma(D)$ is a High Level Petri Box, $B'[Y \leftarrow \Gamma(D)]$ will be. In this case we will *define* $B[Y \leftarrow D]$ to be $B'[Y \leftarrow \Gamma(D)]$.

6.5.6 The Winding Auxiliary Operator

As already explained, winding identifies the initial state of a High Level Petri Box with that state directly preceding a recursive call, and the terminal state with that directly following. Clearly, these are, respectively, the entry and exit places of the High Level Petri Box and the pre- and post-sets of the subject process variable(s).

DEFINITION 6.5.8 Let Y be a process variable, and B a Y-guarded High Level Petri Box over the label algebra $\mathcal{L}_\mu$. Define $\Omega_Y(B) = f_\Omega(B[Y \leftarrow \mathtt{stop}])$ where

$$f_\Omega(s) = \begin{cases} {}^\bullet B & s \in {}^\bullet F \\ B^\bullet & s \in F^\bullet \\ \{s\} & \text{otherwise} \end{cases}$$

and ${}^\bullet F = \bigcup_{l \in L_Y(B)} f_1^l({}^\bullet l)$, i.e., the image of ${}^\bullet L_Y(B)$ under the refinement, and $F^\bullet = \bigcup_{l \in L_Y(B)} f_1^l(l^\bullet)$, i.e., the image of $L_Y(B)^\bullet$ under the refinement. ■ 6.5.8

Note:

1. that ${}^\bullet F$ and $F^\bullet$ are well-defined follows from the fact that B is assumed Y-guarded.

2. other than for the trivial case when $L_Y(B) = \emptyset$, f_Ω is not sane: $s \in {}^\bullet B$ and $s' \in {}^\bullet L_Y(B)$ implies that $f_\Omega(s) \cap f_\Omega(s') \neq \emptyset$. ($f_\Omega$ is used to introduce loops into the operand High Level Petri Box, and such loops are not possible using sane relations.) We should not expect the properties upon which rested the normal form of Chapter 5 to hold, and so should not expect the results of that chapter to apply to wound High Level Petri Boxes.

3. however, the definition of f_Ω ensures that a wound High Level Petri Box is at least able as Properties 1 and 3 of Definition 2.5.15 are inherited from the operand High Level Petri Box. When Y appears in B, ${}^\bullet LT(\Omega_Y(B)) \setminus LT(\Omega_Y(B))^\bullet = \emptyset = LT(\Omega_Y(B))^\bullet \setminus {}^\bullet LT(\Omega_Y(B))$. Hence, winding followed by an immediate application of guarding, i.e., $\Gamma(\Omega_Y(B))$, returns a High Level Petri Box.

4. the μ_c and μ_r transitions of a Y-guarded High Level Petri Box may be entry and exit transitions when the operand High Level Petri Box contained Y-locals which bordered on

the entry or exit interface. The application of f_Ω in the definition of the recursive construct may thus produce a local transition, l, such that[12] ${}^\bullet l \cap l^\bullet \neq \emptyset$. We have seen that the commutativity of refinement with the top-level operators assumes that the refined local transition has disjoint pre- and post-set so that there is a potential problem for High Level Petri Boxes involving recursion. However, as $\mathcal{PV} \cap \mathbf{L}_\mu = \emptyset$, we will never be required to refine such local transitions and so do not encounter the problem.

One important corollary is that the proofs of commutativity of refinement and the top-level operators extends to High Level Petri Boxes to which the recursive construct has been applied.

5. the use of **stop** in the definition allows us to replace the refined process variable by a transition which does not contribute to behaviour, simultaneously ensuring that there are no remaining Y-locals in the High Level Petri Box (essentially, all that we do is change the label on the Y-locals to **0**). In figures we will usually omit such transitions, as they contribute nothing to the behaviour of the resulting High Level Petri Box while complicating its illustration. The two versions are illustrated in Figure 6.12. Note that, after winding, all abstract transitions which originally included a Y-local are dead.

6. the reader will note that, as the pre- and post-contexts of all Y-locals are (.) and as the application of the auxiliary relation f_Ω identifies the pre-set of all Y-locals with ${}^\bullet B$ and the post-set with $B^\bullet$, all copies of **stop** introduced into a High Level Petri Boxes through the winding auxiliary have the same pre- and post-sets, pre- and post-contexts and, of course, a **0** label, i.e., are the same local transition.

7. the proof of local strictness (Theorem 4.8.10 of Chapter 4) depended upon the fact that, if $l_1, l_2 \in t \in T_B$ (for B a syntactically generated High Level Petri Box) with ${}^\bullet l_1 \cap {}^\bullet l_2 \neq \emptyset$, and $M \in \mathcal{M}_d^B$ is such that $M[t\rangle$ then ${}^C l_1 = C_1 \lfloor C_1'$ and ${}^C l_2 = C_2 \rfloor C_2'$ (or vice versa). Unfortunately, for wound High Level Petri Boxes this property no longer holds: for instance, when l_1 and l_2 are distinct return transitions on Y, then ${}^\bullet l_1 \cap {}^\bullet l_2 = B^\bullet$, without necessarily having embedded nominal relabellings $\lfloor$ and $\rfloor$. Fortunately, such local transitions never partake in synchronisation, and it is not difficult to see that the property of local strictness continues to hold of wound High Level Petri Boxes (as we do not alter the annotating contexts of other local transitions).

[12] But never such that ${}^\bullet l = l^\bullet$, so that Proposition 3.3.11 (page 52) always applies.

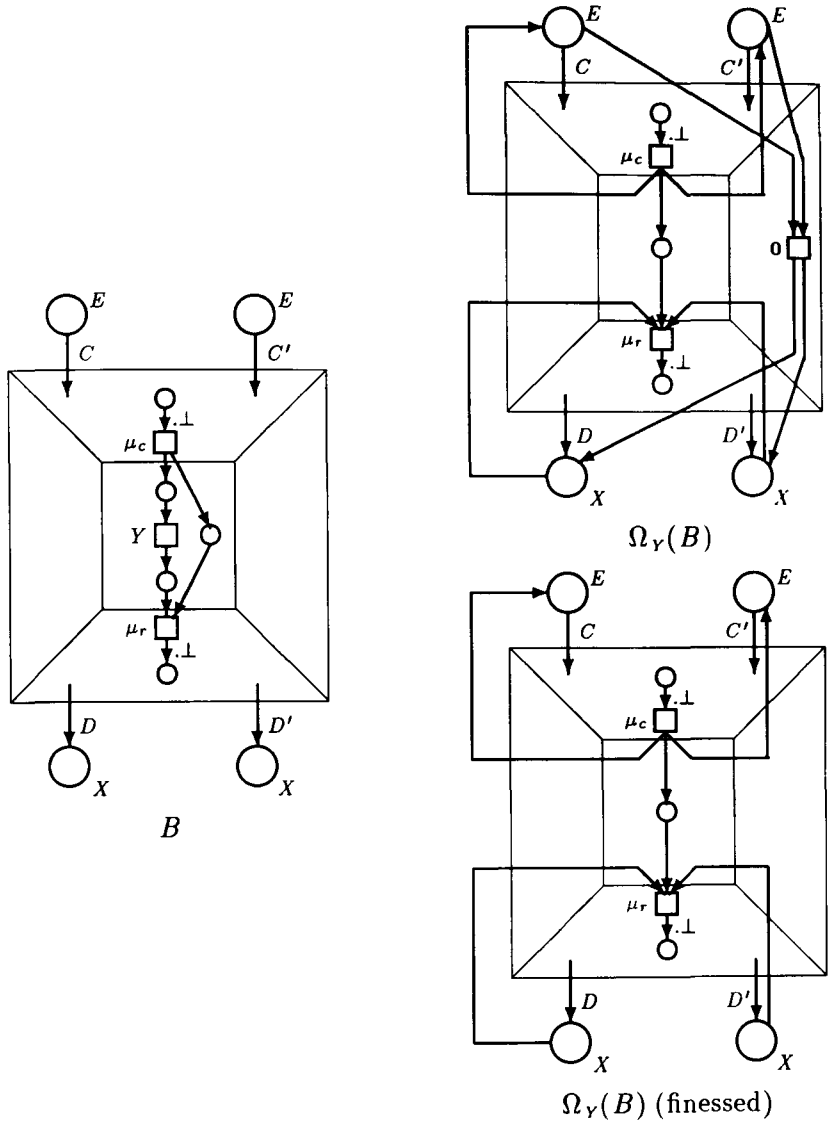


Figure 6.12: Winding a High Level Petri Box with (upper) and without (lower) dead transition.

The following structural properties of the guarding and winding auxiliaries will facilitate the development of their behavioural properties in Chapter 7:

LEMMA 6.5.9 For D a High Level Petri Box

1. $\Gamma(B)[Y \leftarrow D] = \Gamma(B[Y \leftarrow D])$,
2. if $L_Y(D) = \emptyset$ then $Z \in \mathcal{PV}$, $Z \neq Y$ implies $\Omega_Y(B)[Z \leftarrow D] = \Omega_Y(B[Z - D])$ □ 6.5.9

Proof: 1. Follows by inspection from Theorem 6.4.15, given that $\mu \notin \mathcal{PV}$.

2. The result follows from a calculation of the composition of the various auxiliary relations. Note that, as B is Y-guarded, and $Z \neq Y$, $L_Y(B) = L_Y(B[Z - D])$. Moreover, with a careful choice of copy of D for the refinement, ${}^\bullet L_Y(B)^\bullet \cap S_D = \emptyset$. Note also that $B[Z \leftarrow D]$ remains Y-guarded so that the hypotheses of Definition 6.5.8 are satisfied. ■ 6.5.9

Lemma 6.5.9 extends the syntactic nature of refinement as far as may be reasonably expected; i.e., we may not add to or remove from the Y-locals of B and expect the property to hold. Note that, from Proposition 6.4.4, $\Omega_Y(B)[Y \leftarrow D] = \Omega_Y(B)$, as $L_Y(\Omega_Y(B)) = \emptyset$.

If B is syntactically generated then:

LEMMA 6.5.10 For $Y \neq Z \in \mathcal{PV}$ then

1. B is Z-guarded if and only if $\Omega_Y(B)$ is Z-guarded.
2. $Acc_Z(B) = Acc_Z(\Omega_Y(B))$ □ 6.5.10

6.5.7 The Recursive Construct

We derive the recursive construct from the auxiliary operation of guarding and winding:

DEFINITION 6.5.11 *[Recursive Construct]* Let $Y \in \mathcal{PV}$ and let B be a High Level Petri Box, with B' its Y-guarding. Let $B'' = \Gamma(\Omega_Y(B'))$. Define

$$\mu Y.B = B[Y \leftarrow B'']$$

■ 6.5.11

Note:

1. The outer guarding in the definition of the recursive construct is made to ensure that the operand of the embedded refinement is a High Level Petri Box, rather than just a pre-High Level Petri Box.
2. B'' contains no Y-locals as it is derived from the winding of B'. Refining Y in B by B'', therefore, ensures that no Y-locals remain in $\mu Y.B$. In repeated applications of recursion, such as would be required to form the semantics of the term $\mu Y.\mu Y.B$, we do not have erroneous variable capture between inner and outer scopes.

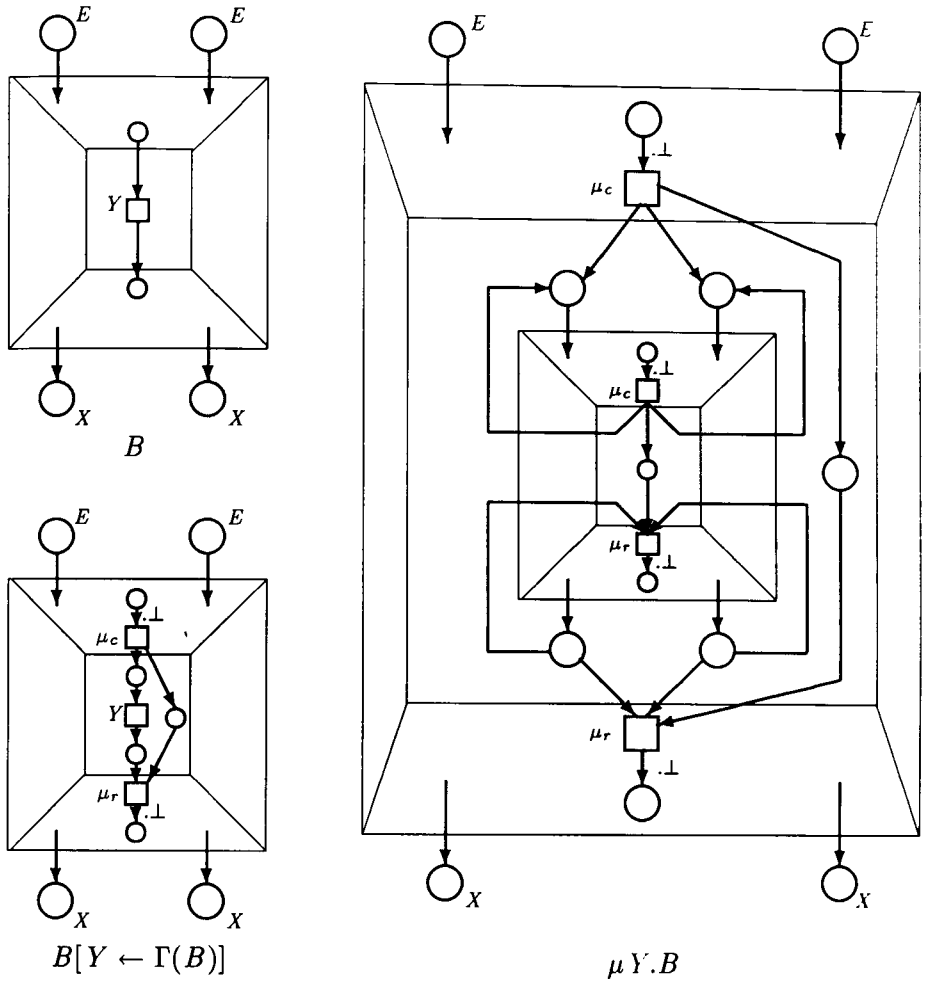


Figure 6.13: The recursive construct applied to the conceptual model (finessed).

3. the inner guarding (the definition proceeds with a Y-guarded version of B) produces a second guarding mechanism inside $\mu Y.B$. This inner guarding mimics the behaviour of the outer guarding, so that their guarding behaviours coincide. It is used in the demonstration that Equation 6.1 holds of our recursive construct in Section 7.3.

The definition of the recursive construct is illustrated in terms of the conceptual model in Figure 6.13. Note that we have elided the dead transitions introduced through the application of the winding auxiliary. Section 6.7 discusses and illustrate the expressive power and the immediate properties of the recursive construct.

From the properties of refinement we have that:

PROPOSITION 6.5.12 $L_Y(B) = \emptyset$ implies $\mu Y.B = B$. □ 6.5.12

Proof: $L_Y(B) = \emptyset$ implies

$$\begin{aligned} \mu Y.B &= B[\emptyset \leftarrow B''] \\ &= \text{[Definition 6.4.13, } B'' \text{ as defined above]} \\ &\quad B \end{aligned}$$

■ 6.5.12

Given that we have introduced the recursive construct, we continue the presentation with the extension of the High Level Petri Box Algebra to include it.

6.6 The High Level Petri Box Syntax

To be able to refer to refinement and recursion in High Level Petri Box Algebra terms we will extend the syntax defined in Definition 4.8.1[13]. For clarity, the presentation of the extended syntax is given in the style of Chapter 4 rather than as an OSA:

DEFINITION 6.6.1 *[High Level Petri Box Syntax]* Given a label algebra $\mathcal{L} = \langle A, R \rangle$, we define the *$\mathcal{L}$-syntax* of the High Level Petri Box Algebra as follows:

E	::=	`stop`	
	|	u	$u \in A$
	|	Y	$Y \in \mathcal{PV}$
	|	$E \Vert E$	Concurrent composition
	|	$E;E$	Causal composition
	|	$E+E$	Choice composition
	|	$E[f]$	Relabelling, where $f \in R$
	|	$E[Y \leftarrow E]$	Refinement, where $Y \in \mathcal{PV}$
	|	$\mu Y.E$	Recursion, where $Y \in \mathcal{PV}$

An *$\mathcal{L}$-term* is a term of the language generated through the $\mathcal{L}$-syntax. ■ 6.6.1

We will assume that operators which also appeared in Definition 4.8.1, i.e., concurrent, causal and choice compositions and relabelling, have the same precedence. Refinement and recursion are defined to have the same precedence as relabelling.

In the sequel, unless otherwise stated, the term *High Level Petri Box Syntax* refers to that of Definition 6.6.1.

[13] For the very last time!

6.6.1 Ground Terms and Free variables

A process variable which is the subject variable of a refinement or recursion will be deemed to be *bound by the refinement (resp. recursion)* (or just *bound*). Variables which are not bound are *free*:

DEFINITION 6.6.2 Given an extended $\mathcal{L}$-term t, define the *free variables* of t, $\mathcal{FV}(t)$. inductively as follows:

$$
\begin{aligned}
\mathcal{FV}(\mathtt{stop}) &= \emptyset \\
\mathcal{FV}(u) &= \emptyset && u \in A \\
\mathcal{FV}(Y) &= Y && Y \in \mathcal{PV} \\
\mathcal{FV}(t_1 \| t_2) &= \mathcal{FV}(t_1) \cup \mathcal{FV}(t_2) \\
\mathcal{FV}(t_1 ; t_2) &= \mathcal{FV}(t_1) \cup \mathcal{FV}(t_2) \\
\mathcal{FV}(t_1 + t_2) &= \mathcal{FV}(t_1) \cup \mathcal{FV}(t_2) \\
\mathcal{FV}(t_1[f]) &= \mathcal{FV}(t_1) \\
\mathcal{FV}(t_1[Y \leftarrow t_2]) &= \mathcal{FV}(t_1) \setminus \{Y\} \cup \mathcal{FV}(t_2) \\
\mathcal{FV}(\mu Y.t_1) &= \mathcal{FV}(t_1) \setminus \{Y\}
\end{aligned}
$$

Term t is *ground* when $\mathcal{FV}(t) = \emptyset$. ■ 6.6.2

By relation to the semantic domain we may define the notions of ground and non-ground High Level Petri Boxes to be the High Level Petri Box denotation of terms with those properties, with $\mathcal{FV}(B) = \mathcal{FV}(t)$, where $B = HLPB(t)$. An important property of ground High Level Petri Boxes is that they contain no process variables so that PrT and P/T net unfoldings, as defined in Chapter 3, exist. Clearly, under the current definitions this cannot be the case of non-ground High Level Petri Boxes under a strict interpretation of process variable: they are place-holders and should have no *labelled* net representation.

6.6.2 Non-ground Terms and Structural Induction

The non-existence of PrT and P/T net unfoldings is a serious disadvantage if we wish to prove behavioural properties by structural induction. For instance, consider an induction step in a proof by structural induction of some property P for the High Level Petri Box term $\mu Y.t$. for which process variable Y appears in t. For the inductive step we are required to be able to

assume property P of the subterm t. But t will not, in general, be ground and so can not, naively, be assigned a behaviour.

We thus introduce *process variable extended label algebras* in which process variables are considered 'ordinary' labels. The construction of the extension is similar to that used in the construction of μ-extended label algebras of Definition 6.5.2. We recall that $\mathcal{PV} \cap (\mathbf{L}^{\oplus} \cup \{\mu\}) = \emptyset$:

DEFINITION 6.6.3 Let $\mathcal{L} = \langle A, R \rangle$ be a label algebra and let $a \in A$. Define a $\mathcal{PV}$-extension of $\mathcal{L}$, $\mathcal{L}^{\mathcal{PV}} = \langle A^{\mathcal{PV}}, R^{\mathcal{PV}} \rangle$ where $A^{\mathcal{PV}} = A \cup \mathcal{PV}$ and such that for each $f \in R$, there is a relabelling $f^{\mathcal{PV}} \in R^{\mathcal{PV}}$ such that

$$f^{\mathcal{PV}}(k) = \begin{cases} f(k) & k \in A^{\oplus} \\ a & k \in (A^{\mathcal{PV}})^{\oplus} \setminus A^{\oplus} \end{cases}$$

■ 6.6.3

Corresponding to each $f \in R$ there is a relabelling $f^{\mathcal{PV}} \in R^{\mathcal{PV}}$ in $\mathcal{L}^{\mathcal{PV}}$ whose action on labels in which a process variable does not appear is the same as that of f; but which generates the label a for any synchronisation label in which a process variable appears. As the relabellings of $\mathcal{L}$ and $\mathcal{L}^{\mathcal{PV}}$ are in bijective correspondence, and as $f^{\mathcal{PV}}\lfloor_{A^{\oplus}} = f$, we will usually use f to represent both f and $f^{\mathcal{PV}}$. This has the benefit of allowing us regard $\mathcal{C}_{\mathcal{L}}$ and $\mathcal{C}_{\mathcal{L}^{\mathcal{PV}}}$ as being equal. Moreover, as for the most permissive label algebra, the important characteristics of the $\mathcal{PV}$-extension are not dependant on the particular label chosen for a so that we will usually speak of *the* $\mathcal{PV}$-extended label algebra.

We note that, for a ground High Level Petri Box, B, the behaviours of B over $\mathcal{L}$ and over $\mathcal{L}^{\mathcal{PV}}$ will coincide.

6.7 Examples of the Recursive Construct

6.7.1 Tail-end Recursion

Figure 6.14 gives the denotation of the tail-end recursive term $\mu Y.(a; Y + b)$, whose complete behaviours generate the strings $\{a^n b \mid n \in \mathbb{N}\}$. The embedded choice composition in the recursive term allows termination as, by taking the action b, the recursive call is then forced (figuratively) to 'unwind'. The structure of the figure makes clear the dependence of the correct behaviour on the presence of *both* outer and inner accounting places: the outer place (3^0 of the figure) allows us to check for correct termination of the 'inner' High Level Petri Box; the

firing of the initial call transition (from the standard initial marking) leaves the token $.\perp\lfloor$ in this accounting place; the final return transition will be enabled if and only if the other place in its pre-set (4^1 in the figure) contains the same $.\perp\lfloor$ token, which can occur if and only if there have been an equal number of firings of the inner call and return mechanisms, i.e., the recursion has fully unwound. The final return transition may then fire, allowing the standard terminal marking of the High Level Petri Box to be reached.

Figure 6.15 gives an example firing sequence of the High Level Petri Box from the standard initial marking (using the combined transition sequence semantics defined in Section 3.3.4). The main points to note are the disabling of transitions v and y at certain markings. This disabling depends crucially on the presence of the accounting place, and the nominal relabelling $\perp$ as an arc annotation.

6.7.2 Unbounded Parallelism

Figure 6.16 illustrates the situation in which there is a concurrent composition under the recursive construct, in which case concurrent processes may be dynamically created, all within a finite net structure. Behaviourally, the High Level Petri Box illustrated in Figure 6.16 is equivalent to that of Figure 6.8(a).

6.7.3 Unbounded Parallelism and Synchronisation

Figure 6.17 illustrates the situation in which there is a concurrent composition under the recursive construct together with a synchronisation across levels of recursion. The correct behaviour and synchronisation of such processes is guaranteed by the careful use of nominal relabellings, loops and accounting places of the recursive construct.

An example of a transition sequence for this High Level Petri Box is given in Figure 6.18.

6.7.4 `Interwoven alignment preambles are not allowed`

Knuth [Knu86, Pg. 299] in the chapter on *Recovery from Errors* provides the following comment on the error message `Interwoven alignment preambles are not allowed` (limiting the complexity of definition of an alignment):

> If you have been so devious as to get this message, you will understand it, and you will deserve no sympathy.

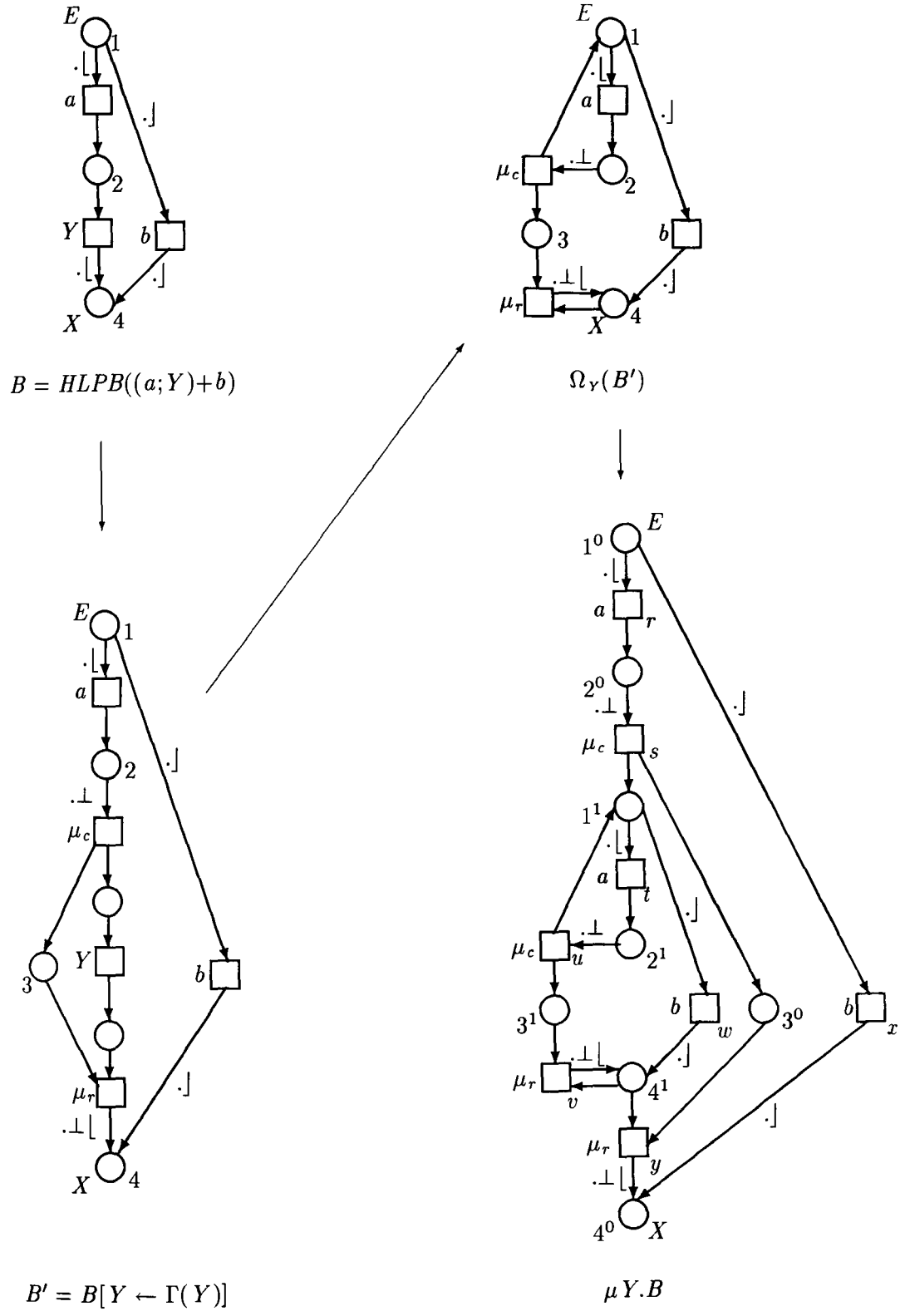


Figure 6.14: The stages in producing the denotation of the tail-end recursive term $\mu Y.((a;Y)+b)$. Note the naming convention used for the places of $\mu Y.B$.

$$\left\{1^0 \to \{.\underline{\rfloor}\}\right\} \quad \left[\begin{matrix}a\\r\end{matrix}\right\rangle \quad \left\{2^0 \to \{.\underline{\rfloor}\}\right\}$$

$$\left[\begin{matrix}\mu_c\\s\end{matrix}\right\rangle \quad \left\{\begin{matrix}1^1 \to \{.\underline{\perp\rfloor}\},\\3^0 \to \{.\perp\rfloor\}\end{matrix}\right\}$$

$$\left[\begin{matrix}a\\t\end{matrix}\right\rangle \quad \left\{\begin{matrix}2^1 \to \{.\underline{\rfloor\perp\rfloor}\},\\3^0 \to \{.\perp\rfloor\}\end{matrix}\right\}$$

$$\left[\begin{matrix}\mu_c\\u\end{matrix}\right\rangle \quad \left\{\begin{matrix}1^1 \to \{.\underline{\perp\rfloor\perp\rfloor}\},\\3^1 \to \{.\perp\rfloor\perp\rfloor\},\\3^0 \to \{.\perp\rfloor\}\end{matrix}\right\}$$

$$\left[\begin{matrix}a\\t\end{matrix}\right\rangle \quad \left\{\begin{matrix}2^1 \to \{.\underline{\rfloor\perp\rfloor\perp\rfloor}\},\\3^1 \to \{.\perp\rfloor\perp\rfloor\},\\3^0 \to \{.\perp\rfloor\}\end{matrix}\right\}$$

$$\left[\begin{matrix}\mu_c\\u\end{matrix}\right\rangle \quad \left\{\begin{matrix}1^1 \to \{.\underline{\perp\rfloor\perp\rfloor\perp\rfloor}\},\\3^1 \to \{.\perp\rfloor\perp\rfloor, .\perp\rfloor\perp\rfloor\perp\rfloor\},\\3^0 \to \{.\perp\rfloor\}\end{matrix}\right\}$$

$$\left[\begin{matrix}b\\w\end{matrix}\right\rangle \quad \left\{\begin{matrix}3^1 \to \{.\perp\rfloor\perp, .\underline{\perp\rfloor\perp\rfloor\perp\rfloor}\},\\3^0 \to \{.\perp\rfloor\},\\4^1 \to \{.\underline{\perp\rfloor\perp\rfloor\perp\rfloor}\}\end{matrix}\right\}$$

transition y not enabled, as $M(3^0) \cap M(4^1) = \emptyset$

$$\left[\begin{matrix}\mu_r\\v\end{matrix}\right\rangle \quad \left\{\begin{matrix}3^1 \to \{.\underline{\perp\rfloor\perp\rfloor}\},\\3^0 \to \{.\perp\rfloor\},\\4^1 \to \{.\underline{\perp\rfloor\perp\rfloor}\}\end{matrix}\right\}$$

transition y not enabled, as $M(3^0) \cap M(4^1) = \emptyset$

$$\left[\begin{matrix}\mu_r\\v\end{matrix}\right\rangle \quad \left\{\begin{matrix}3^0 \to \{.\underline{\perp\rfloor}\},\\4^1 \to \{.\underline{\perp\rfloor}\}\end{matrix}\right\}$$

transition v not enabled, as $M(3^1) = \emptyset$

$$\left[\begin{matrix}\mu_r\\y\end{matrix}\right\rangle \quad \left\{4^0 \to \{.\}\right\}$$

Figure 6.15: A firing sequence for the recursive term $\mu Y.((a;Y)+b)$. To aid the reader, the tokens which partake in the firing of a transition are underlined.

The High Level Petri Box Algebra is more forgiving in the complexity of terms which include recursion. The term $\mu Y.(a;\mu X.((Y \| X) + b);c)$ (of which Figure 6.19 illustrates the denotation of the inner loop $\Omega_Y(a;\mu X.((\Gamma(Y)\|X) + b);c))$ has process variables with overlapping scope. A local transition firing sequence of the inner loop is given in Figure 6.20.

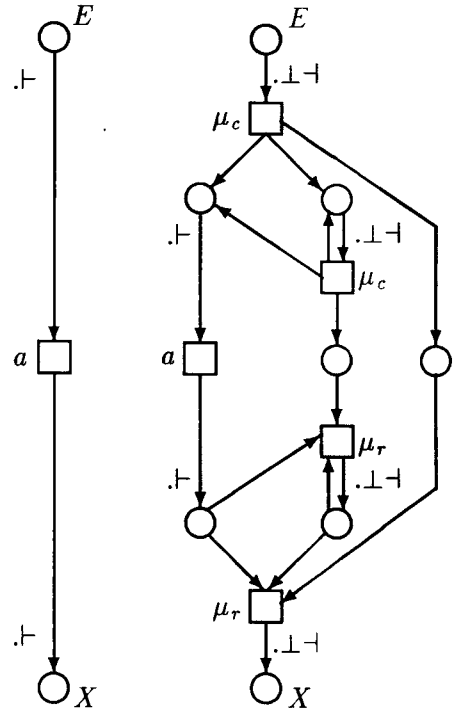


Figure 6.16: Unbounded parallelism in a finite High Level Petri Box: the denotation of the term $\mu Y.a \| Y$.

6.7.5 Turing Expressivity

The demonstration of Turing expressivity of a computational model is given by exhibiting a *register*[14]. The state of a *register* represents a natural number. The number can always be incremented; it can be decremented if non-zero; and it can be tested for zero.

Minsky [Min72] shows that any computation of a Turing Machine may be simulated by a 2 register machine. For the High Level Petri Box Algebra, his construction translates to a term with the following structure:

$$(Register_1 \| Program \| Register_2)[compute]$$

where *Program* encodes the specific computation, and *compute* synchronises the communications of the program with the two registers. (For details of the construction in terms of general counters see [Min72].)

Although we do not give the details of the construction for the *Program* (which is, of course,

[14]At least in folklore of the subject. Taubner makes this more precise in [Tau89]. A register is also known as a *counter* in the literature [Age75].

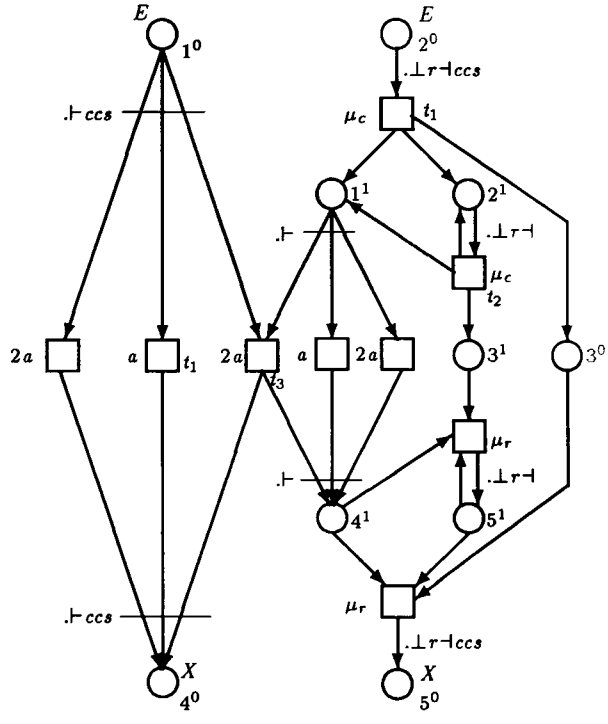


Figure 6.17: Synchronisation and unbounded parallelism in a finite High Level Petri Box: the denotation of the term $\mu Y.(a||(Y[r]))[ccs]$, in the label algebra $\mathcal{L}_0$ of Example 2.1.1.

computation specific) nor the relabelling *compute*, we will define a register as a High Level Petri Box term.

We will say that a μ-extended label algebra, $\mathcal{L} = \langle A, R \rangle$, contains a register if $\{\mu, zero, inc, dec\} \subseteq A$ and $\{zero{-}to{-}dec\} \subseteq R$, where $zero{-}to{-}dec$ is such that

$$zero{-}to{-}dec(k) = \begin{cases} dec & k = zero \\ k & k \in A \wedge k \neq zero \\ \mathbf{0} & \text{otherwise} \end{cases}$$

The definition of the register is then

$$Register = \mu R.(zero{+}inc;R[zero{-}to{-}dec];R)$$

The High Level Petri Box denotation of the inner loop of the *Register* term is shown in Figure 6.21.

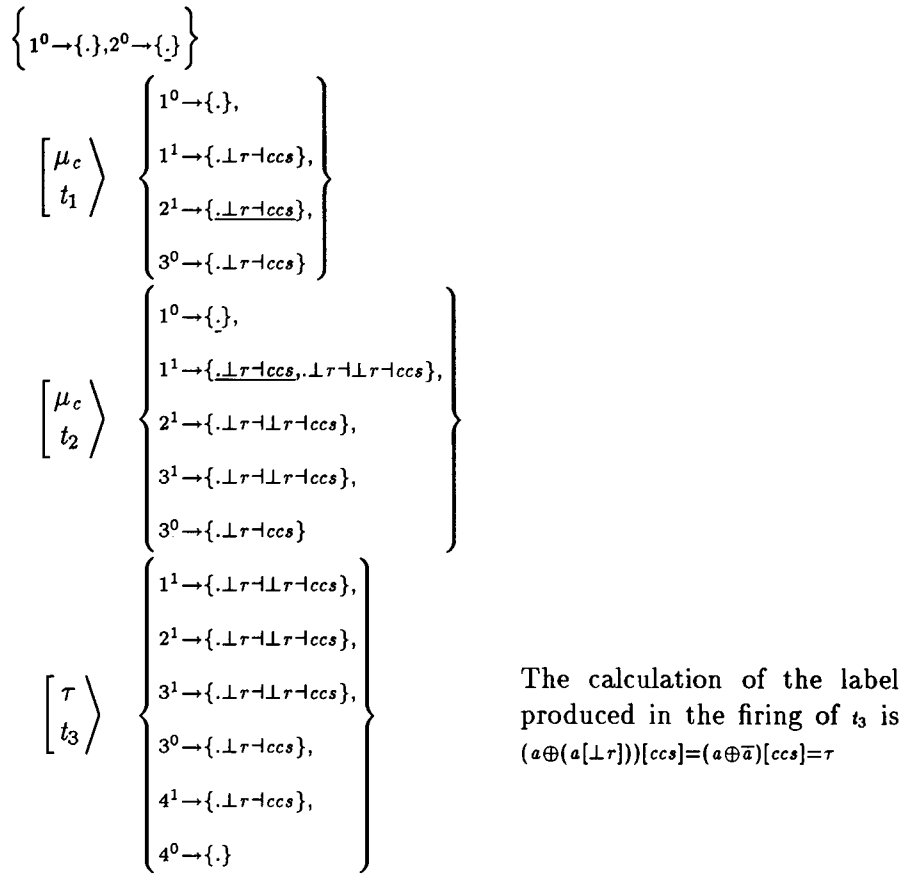


Figure 6.18: A firing sequence for the recursive term $\mu Y.a \| (Y[r])[ccs]$ from the standard initial marking. To aid the reader, the tokens which partake in the firing of a transition are underlined.

6.8 Discussion

6.8.1 Other Net Theoretic Approaches to Variables and their Denotation

Within the labelled net semantics of Taubner [Tau89] (following [Gol88]) variables appear, but only as the subject variable of the recursive construct (refinement does not appear in Taubner's Algebra). In a way similar to the representation of *nil* as discussed in Chapter 4, process variables represent state information, having as semantics a single place which is used to indicate which states should be identified in an iteration-like recursive operator[15]. Hence, there need be no interpretation of variables as place holders for terms. Moreover, as they represent only state information in the semantics they need have no interpretation as actions either.

In Olderog's approach, [Old91], variables again only appear as the subject variable of the recursive construct and his semantic function provides representations of only ground terms using

[15] By this we do not mean that Taubner's recursion is pure tail-end, as non-tail end recursion may be represented through the same device as that of CCS in its derivation of sequential composition.

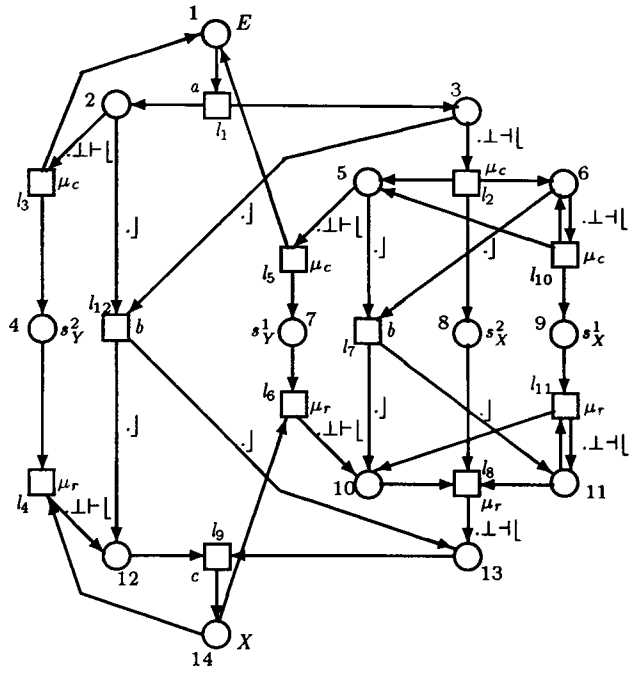


Figure 6.19: The denotation of the inner loop of the High Level Petri Box term $\mu Y.(a;\mu X.((Y\|X)+b);c)$.

the usual SOS rule for recursion which, essentially, identifies the terms $\mu X.P$ and $P[X/\mu X.P]$, whence the behaviour is derived from that of P and need not consider process variables. In his denotational semantics, fix-point techniques are used to provide the denotation of the recursive construct and, again, this removes the need for the explicit representation of process variables.

In the PBC, process variables form *hierarchical labels* to annotate transitions, as in our approach. However, hierarchical labels are structured (unlike here) allowing labels such as $f(X)$ which provide for a compositional approach to relabelling; f is applied to the labels of any box which refines X. Only closed terms are given a behavioural semantics in the PBC, so that process variables need not be considered as having a behaviour.

6.8.2 The Refinement Operators of the Low Level Petri Box Model

The original refinement operator of [BDH92] was defined iteratively although, because the approach did not distinguish local and abstract transitions, the iteration was over ordinary transitions. This meant that the approach did not extend to the refinement of an infinity of transitions. Moreover, it was not clear that the result was independent of the ordering of the iteration, and the operator did not interact well with the *hierarchical interface* (i.e., those transitions labelled

$$\left\{1 \to \{\underline{.}\}\right\} \quad [l_1\rangle \quad \left\{\begin{array}{l} 2 \to \{\underline{.}\} \\ 3 \to \{.\} \end{array}\right\}$$

$$[l_3\rangle \quad \left\{\begin{array}{l} 1 \to \{\underline{.\bot\vdash\lfloor}\} \\ 4 \to \{.\bot\vdash\lfloor\} \\ 3 \to \{.\} \end{array}\right\}$$

$$[l_1\rangle \quad \left\{\begin{array}{l} 2 \to \{.\bot\vdash\lfloor\} \\ 3 \to \{\underline{.}, .\bot\vdash\lfloor\} \\ 4 \to \{.\bot\vdash\lfloor\} \end{array}\right\}$$

$$[l_2\rangle \quad \left\{\begin{array}{l} 5 \to \{\underline{.\bot\dashv\lfloor}\} \\ 6 \to \{\underline{.\bot\dashv\lfloor}\} \\ 8 \to \{.\bot\dashv\lfloor\} \\ 2 \to \{.\bot\vdash\lfloor\} \\ 3 \to \{.\bot\vdash\lfloor\} \\ 4 \to \{.\bot\vdash\lfloor\} \end{array}\right\}$$

$$[l_7\rangle \quad \left\{\begin{array}{l} 10 \to \{\underline{.\bot\dashv\lfloor}\} \\ 11 \to \{\underline{.\bot\dashv\lfloor}\} \\ 2 \to \{.\bot\vdash\lfloor\} \\ 3 \to \{.\bot\vdash\lfloor\} \\ 4 \to \{.\bot\vdash\lfloor\} \\ 8 \to \{\underline{.\bot\dashv\lfloor}\} \end{array}\right\}$$

$$[l_8\rangle \quad \left\{\begin{array}{l} 13 \to \{.\} \\ 2 \to \{\underline{.\bot\vdash\lfloor}\} \\ 3 \to \{\underline{.\bot\vdash\lfloor}\} \\ 4 \to \{.\bot\vdash\lfloor\} \end{array}\right\}$$

$$[l_{12}\rangle \quad \left\{\begin{array}{l} 12 \to \{\underline{.\bot\vdash\lfloor}\} \\ 13 \to \{., \underline{.\bot\vdash\lfloor}\} \\ 4 \to \{.\bot\vdash\lfloor\} \end{array}\right\}$$

$$[l_9\rangle \quad \left\{\begin{array}{l} 14 \to \{\underline{.\bot\vdash\lfloor}\} \\ 13 \to \{.\} \\ 4 \to \{\underline{.\bot\vdash\lfloor}\} \end{array}\right\}$$

$$[l_4\rangle \quad \left\{\begin{array}{l} 12 \to \{\underline{.}\} \\ 13 \to \{\underline{.}\} \end{array}\right\}$$

$$[l_9\rangle \quad \left\{14 \to \{.\}\right\}$$

Figure 6.20: A local transition firing sequence of the High Level Petri Box of Figure 6.19 . To aid the reader, the tokens which enable a local transition are underlined.

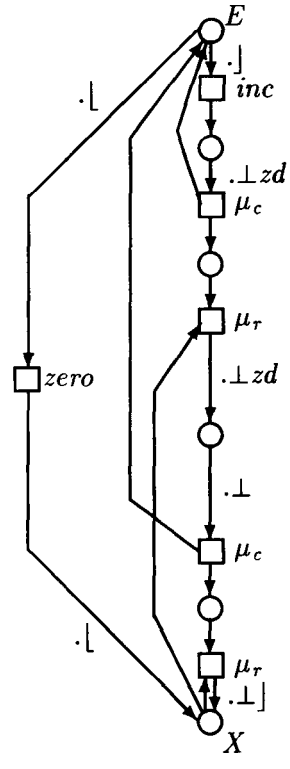


Figure 6.21: A register using the recursive construct (inner 'loop' only). The relabelling *zero−to−dec* is shown as *zd*.

with hierarchical labels) of a Petri box, restricting its syntactic nature in particular with respect to relabelling.

In [BDE93] was defined a more general operator which addressed these problems through the definition of simultaneous refinement. This version uses a *labelled tree device* as a generalisation of the Cartesian product (used here in its commutative form; and elsewhere [GvG89]). This device allows transitions to be refined, as here, by replacement and allows it to work in the very general situations that appear in the low level PBC. For instance, the construction deals with: possibly infinite numbers of refined transition; arbitrary arc weights; a (possibly) continuous infinity of entry/exit places. The tree-based construction also endows the refinement operator with very nice properties, even in its complex setting; these properties include its congruence nature with respect to a number of (structural) equivalences, and the (structural) fix-point nature of the recursive construct derived from it [DK95].

6.8.3 Tail-End and Non-Tail-End Recursion

In comparison with the prefixed-based literature, the register representing term of Section 6.7.5, i.e.,

$$Register = \mu R.(zero{+}inc;R[zero{-}to{-}dec];R)$$

is notable for the absence of an embedded concurrent composition. One of the pragmatic deficiencies of prefixed based approaches is the necessary transformation of non-tail-end recursion into tail-end recursion; one is not able to prefix a process variable to a term. One solution used widely in the literature is based on Milner's derivation of sequential composition from prefix, as described in Chapter 4.

The reader will recall from that chapter that Milner's derivation depends on the postfixing of each term with an action ($\surd$) whose execution indicates termination and whose synchronisation with $\overline{\surd}$ permits the causally inferior process to begin. In the transformation of non-tail-end recursion into tail-end recursion, then, to avoid requiring the 'prefixing' of the process variable onto the $\surd$ (as the $\surd$ should indicate the termination of the recursive call) one must push the terminating action 'through' into the recursive call itself.

Taubner uses this transformation to produce a term which simulates a register[16]:

$$Register_{Tau} = \mu R.(zero;nil{+}inc;((R[\surd zero]\|\overline{\surd};dec;zero;nil)[hide\surd]))$$

where instead of $disable{-}zero$ the $zero$ action is converted to a terminating $\surd$ using the relabelling:

$$\surd zero(k) = \begin{cases} \surd & k = zero \\ k & \text{otherwise} \end{cases}$$

The marked difference in the complexity of the two terms is due precisely to the complexity of the derivation of sequence from prefix. It will not surprise the reader to hear that we prefer the simplicity of the High Level Petri Box term.

6.8.3.1 Refinement and β-Reduction

Finally, process variables as expressed here together with the substitutive nature of refinement provide what appears to be a novel Petri net equivalent of variables in a functional theory; process

[16] Actually, Taubner uses Agerwala's term Counter [Age75].

variables may be substituted for by arbitrary terms. In this sense, our refinement operator has properties similar to the β-reduction of the λ-calculus.

This is not accidental: one of the longer term aims of the work is to be able to investigate the question as to whether a high level Petri net model for concurrency could be developed within a more general, functional framework (following the pioneering categorical approach of Winskel [Win85]). For such an approach it appears that a refinement operator with these properties would be useful.

6.9 Summary

In this chapter we have introduced the top level operators of refinement and recursion. To do this we have:

1. introduced the notion of a process variable,
2. introduced the auxiliary operator of local transition refinement, which will replace a local transition within a High Level Petri Box with another High Level Petri Box, while preserving causality (by respecting place multiplication), distributed scoping information (inherited from the replaced local transition), and from which (full) refinement is derived via a finitary construction based on the finiteness of the local transitions which appear within a syntactically generated High Level Petri Box,
3. extended the syntax to include the concrete representation of refinement as $t_1[X - t_2]$, where t_1 and t_2 are High Level Petri Box terms, and X is a process variable.

The operator of refinement is a semantic version of *syntactic substitution*, common to all algebraic systems. The characteristic property of syntactic substitution is that its application commutes with the application of all other operators; we have shown that, modulo isomorphism of High Level Petri Boxes, the operator of refinement shares this property, thus justifying the adjective 'syntactic'.

We have also introduced a recursive construct which preserves finiteness of state on syntactically generated High Level Petri Boxes, i.e., if B is a syntactically generated High Level Petri Box then $|S_{\mu Y.B}| < \infty$.

We have shown that the addition of the recursive construct to the syntax of the High Level Petri Box Algebra provides a very compact description of both tail-end and non-tail-end recursion.

in contrast with those of the literature. We hope in the future to be able to show that the descriptions are tractable as well.

In the next chapter we explore the behaviours of syntactically generated High Level Petri Boxes, showing that the semantic interpretations of the High Level Petri Box Algebra operators have many desirable behavioural properties.

Chapter 7

Algebra and Behaviour

In this chapter we explore the behaviour of High Level Petri Boxes which include applications of the refinement and recursion operators. The increase in complexity of such High Level Petri Boxes makes the analysis of their behaviours less amenable to structural characterisation, and leads to proofs which proceed by induction over prefixes of behaviours. This approach should be contrasted with that of Chapter 5 in which behavioural arguments were developed, but which were derived from purely structural arguments.

The major result of the chapter is the full behavioural characterisation of the property rich subclass of syntactically generated High Level Petri Boxes (given in Lemma 7.1.2). The result provides many tools for the general manipulation of behaviours of High Level Petri Boxes, which are used in subsequent sections.

Although this result is important in its own right, it also provides justification for many of the choices which have been made in the development of the High Level Petri Box model. In particular, the proof of the property depends upon the annotation of the operands of concurrent composition by the nominal relabellings[1] $\vdash$ and $\dashv$; the careful construction of the guarding auxiliary, including the annotation of arcs with the nominal relabelling $\perp$, and the presence of the accounting place; and the identification of state which is a part of the winding auxiliary and the once unwinding (i.e., the embedded refinement) in the recursive construct.

This behavioural characterisation is stated as a lemma for in the demonstration of the safeness and memorylessness of syntactically generated High Level Petri Boxes (Theorem 7.1.1) in which we find its first use.

[1] The nominal relabellings $\lfloor$ and $\rfloor$ having already been justified in Proposition 6.4.10 (page 157).

Subsequently, in Section 7.2, we define a notion of strong bisimulation, $\approx$, on High Level Petri Boxes and use Lemma 7.1.2 to show $\approx$ to be a congruence for the top-level operators of Chapter 4, and continue to develop the partial congruence nature of the equivalence with respect to the guarding and winding auxiliaries, and the refinement operator, again using Lemma 7.1.2. In Theorem 7.3.8, we combine all these results to show that recursion is a behavioural fixed-point of the refinement operator, i.e., it satisfies Equation 6.1 with $\doteq = \approx$.

We begin the chapter with the demonstration of the safeness and memorylessness of syntactically generated High Level Petri Boxes.

7.1 Safeness and Memorylessness of Syntactically Generated High Level Petri Boxes

We are now in a position to discharge the proof obligation, established in Chapters 4 and 6, that syntactically generated High Level Petri Boxes are both safe and memoryless. The proof proceeds by induction over the structure of a High Level Petri Box. The induction step is to derive a 'compositional behavioural semantics' in characterising the behaviour of a High Level Petri Box term through the behaviours of its immediate operand High Level Petri Boxes terms, as was proffered in Figure 4.16 (page 104). Similar ideas for the method of proof were expounded in [BH92, BK95] over the low level net model contained therein.

As both safeness and memorylessness are positive monotonic in the permissiveness of a label algebra we may use the Decoupling Lemma (Lemma 3.4.4, page 55) to simplify the proof considerably to the consideration of behaviours definable through local transition sequences over the most permissive label algebra. Even with this simplification, however, the proof is rather long as there are still many cases to consider.

The following is a restatement of Theorem 4.8.14 (page 98).

THEOREM 7.1.1 Let B be a syntactically generated High Level Petri Box over a process variable extended label algebra $\mathcal{L}$. Then B is safe and memoryless. □ 7.1.1

The lion's share of the proof is done by the following lemma which provides, essentially, a compositional behavioural semantics[2] of syntactically generated High Level Petri Boxes. As well

[2]Strictly, this would require the explicit development of operators on a behavioural semantic domain which we do not do. We note, however, that the lemma shows that this would be possible (if complicated).

as requiring the operand High Level Petri Boxes to be safe (H1 in Lemma 7.1.2) and memoryless (H2), the lemma also requires that they satisfy three other positive monotonic properties:

H3. that the behaviour of the operands (at least in the most permissive label algebra) is independent of the initial marking,

H4. that accounting places are distinguishable by their markings, and

H5. that accounting places 'record' information as to the identity of the currently active recursive invocations.

For the High Level Petri Boxes of Chapter 4, H3 follows from Corollary 5.4.19 (page 134) and, as there are no accounting places or process variables in such High Level Petri Boxes, each of H4 and H5 follow trivially for the High Level Petri Boxes of that chapter. If the reader wished, then, the proof of Theorem 7.1.1 for such High Level Petri Boxes could be considered under Hypotheses H1 and H2 (and with a considerable simplification of the proof).

LEMMA 7.1.2 Let B_1 and B_2 be sociable High Level Petri Boxes satisfying Hypotheses H1-H5, i.e., they are:

H1. safe,

H2. memoryless,

and such that, for $B = B_1$, $B = B_2$

H3. for all contexts d, if $M \in \mathcal{M}_d^B$, then for all contexts $e \in \mathrm{ran}(M)$, $d \sqsubseteq e$. Moreover, (H3a) for any such M, $M\lfloor_{B^\bullet} \subseteq B^\bullet \times \{\!|d|\!\}$,

H4. for all contexts d, if $M \in \mathcal{M}_d^B$, then for all accounting places $s_1, s_2 \in Acc(B)$, $M(s_1) \cap M(s_2) \neq \emptyset$ implies $s_1 = s_2$,

and if B is Y-guarded then

H5. for all contexts d, markings $M \in \mathcal{M}_d^B$, and $C \in \mathrm{ran}(M)$ such that $d \sqsubset C$ with $\perp \leq C$ there is an accounting place $s \in Acc(B)$ such that $(s, C) \in M$. If, in addition, there is a $C' \in \mathrm{ran}(M)$, such that $C \sqsubset C'$ then there is an accounting place $s \in Acc(B) \setminus Acc_Y(B)$ such that $(s, C) \in M$.

Then:

1. $M \in \mathcal{M}_d^{B_1 \| B_2}$ implies there are markings $M_i \in \mathcal{M}_{(.)}^{B_i}$, $i = 1.2$. such that[3] $M = M_1(.\vdash)_{B_1}(d) \cup M_2(.\dashv)_{B_2}(d)$,

2. $M \in \mathcal{M}_d^{B_1;B_2}$ implies there is a marking $M' \in \mathcal{M}_{(.)}^{B_1}$ such that $M = ;_1(M')(d)$ or $M' \in \mathcal{M}_{(.)}^{B_2}$ such that $M = ;_2(M')(d)$, [4]

3. $M \in \mathcal{M}_d^{B_1+B_2}$ implies that there is a marking $M' \in \mathcal{M}_{(.)}^{B_1}$ such that $M = +_1(M')(.\lfloor)_{B_1}(d)$ or $M' \in \mathcal{M}_{(.)}^{B_2}$ such that $M = +_2(M')(.\rfloor)_{B_2}(d)$,

If we denote B_1 by B, then, in addition, we have:

4. $M \in \mathcal{M}_d^{B[f]}$ implies there is a marking $M' \in \mathcal{M}_{(.)}^{B}$ such that $M = M'(.f)_B(d)$.

5. $M \in \mathcal{M}_d^{\Gamma(B)}$ implies $M = M_d^I$ or $M = M_d^T$ or there is a marking $M' \in \mathcal{M}_{(.)}^{B}$ such that $M = M'(.\perp d) \cup \{s \mapsto (.\perp d)\}$, where s is the accounting place introduced through the guarding of B.

6. if, in addition, B is Y-guarded, then $M \in \mathcal{M}_d^{\Omega_Y(B)}$ implies there exist $n \geq 0$, $s_1, \ldots, s_n \in Acc_Y(B)$, not necessarily distinct places, $C_0 = d$, $C_1, \ldots, C_n \in \mathcal{C}$ pairwise distinct contexts, $M_0, \ldots, M_n \in \mathcal{M}_{(.)}^{B}$ not necessarily distinct markings such that

   (a) $$M = \left(\bigcup_{i=0}^{n} M_i C_i\right)\Big|_{S_B \setminus {}^\bullet L_Y(B)^\bullet}$$

   (b) $$M\lfloor_{Acc_Y(B)} = \{\!|(s_1, C_1), \ldots, (s_n, C_n)|\!\}$$

   and such that there is a labelled tree with nodes $\{0, \ldots, n\}$ (root 0, node i labelled by M_i), such that Properties P1-P3 hold[5]:

   P1 $M_i \# M_j$ (M_i and M_j incomparable in the tree) implies $C_i \# C_j$ (C_i and C_j incomparable as contexts under $\sqsubseteq$),

   P2 $M_i \prec M_j$ (M_j is an immediate child of M_i) implies for all $e \in \mathrm{ran}(M_i C_i)$: $e \# C_j$ or $e \sqsubseteq C_j$,

---

[3] Where $M(C)(s) = M(s)^\frown(C)$ and $M(C)_B(s) = \begin{cases} M(s) & s \in {}^\bullet B^\bullet \\ M(s)^\frown(C) & \text{otherwise.} \end{cases}$

[4] Where, for a sane relation f, $f(M)(s) = M(r)$ when $s \in f(r)$. The well-definedness follows from the sanity of f.

[5] Where, for convenience, by M_i we will mean a marking with concrete index i (equivalent to considering the pair (M_i, i)) so to be able to recover the node i labelled with M_i even though the M_i are not necessarily distinct. This will allow us to speak only of markings, and not of nodes.

P3 $M_i \prec M_j$ implies[6] $M_i C_i \triangleright \{C_j\} = {}^\bullet l_r$, where l_r is the (unique) return transition of B such that $s_j \in {}^\bullet l_r$. □ 7.1.2

A few words of explanation are in order in the last, and most complex, case. For High Level Petri Boxes of the statement, the markings of the winding of a High Level Petri Box and those of the unwound High Level Petri Box are closely related by the recursive construct. In fact, through the ability to distinguish the identities of accounting places, we may distinguish the markings of recursive calls, which are represented by the M_i ($i \geq 1$). The fact that we lose the places in ${}^\bullet L_Y(B)^\bullet$ upon winding explains the slight complication in the statement. The proof of this case is considerably longer than those of the other cases, as might be expected, due to the complexity of the operator definition.

Proof: By induction on the length of the local transition sequence which leads to M, i.e.: $M_d^I = N_0[l_1\rangle \ldots N_{m-1}[l_m\rangle N_m = M$.

When $m = 0$ then $M = M_d^I$ and the result is clear in all cases. Suppose, then, that $m > 0$ and that the result holds for $m - 1$.

1. $M \in \mathcal{M}_d^{B_1 \| B_2}$: from the induction hypothesis there are markings $M_i \in \mathcal{M}_{(.)}^{B_i}$, $i = 1, 2$, such that $N_{m-1} = M_1(.\vdash)_{B_1}(d) \cup M_2(.\dashv)_{B_2}(d)$. As l_m is enabled at N_{m-1}, there are contexts e and f such that ${}^\bullet l_m \times \{\!|e|\!\} \subseteq N_{m-1}$, $f = {}^C l_m e - l_m{}^C$ and

$$N_m = (N_{m-1} \setminus {}^\bullet l_m \times \{\!|e|\!\}) \cup l_m{}^\bullet \times \{\!|f|\!\}$$

From Definition 4.6.1 we may assume, without loss of generality, that there is a local transition $l \in LT(B_1)$ such that $l_m = l(.\vdash)_{B_1}$ with ${}^\bullet l = {}^\bullet l_m$ and $l^\bullet = l_m{}^\bullet$. Moreover, as $l \in LT(B_1)$, and as B_1 and B_2 are sociable, ${}^\bullet l_m \times \{\!|e|\!\} \subseteq M_1(.\vdash)_{B_1}(d)$, and $N_m\lfloor_{S_2} = N_{m-1}\lfloor_{S_2}$.

We show that l in enabled at M_1. There are four cases (which are exhaustive as B_1 is a High Level Petri Box):

(a) ${}^\bullet l_m \subseteq {}^\bullet B_1$ and $l_m{}^\bullet \cap B_1{}^\bullet = \emptyset$: whence ${}^C l_m = {}^C l(.\vdash)$ and $l_m{}^C = l^C$. As B_1 is a High Level Petri Box, there are no arcs into entry places so that $e = d$ and $f = {}^C l_m d - l_m{}^C$.

As ${}^\bullet l_m \times \{\!|d|\!\} \subseteq M_1(.\vdash)_{B_1}(d)$, we have that ${}^\bullet l \times \{\!|.\vdash d|\!\} \subseteq M_1(.\vdash d)$. Moreover, ${}^C l\vdash d - l^C = {}^C l_m d - l_m{}^C = f$, so l is enabled at $M_1(.\vdash d)$ for input token $(.\vdash d)$. Define $M' = (M_1(.\vdash d) \setminus {}^\bullet l \times \{\!|(.\vdash d)|\!\}) \cup l^\bullet \times \{\!|f|\!\}$. Then $M_1(.\vdash d)[l\rangle M'$.

[6] Where $\triangleright$ is range restriction, i.e., for a relation R and a set S, $R \triangleright S = \{r \mid (r, s) \in R \wedge s \in S\}$.

As $M' \in M^{B_1}_{(.\vdash d)}$, by H3 and Proposition 3.6.2, there is a marking $M'' \in \mathcal{M}^{B_1}_{(.)}$ such that $M_1[l\rangle M''$ and $M' = M''(.\vdash d)$. But then, as ${}^\bullet l \subseteq {}^\bullet B_1 \wedge l^\bullet \cap B^\bullet = \emptyset$

$$\begin{aligned} M''(.\vdash)_{B_1}(d) &= (M_1(.\vdash)_{B_1}(d) \setminus {}^\bullet l \times \{\!|d|\!\}) \cup l^\bullet \times \{\!|f|\!\} \\ &= (N_{m-1}\lfloor_{S_1} \setminus {}^\bullet l_m \times \{\!|d|\!\}) \cup l_m{}^\bullet \times \{\!|f|\!\} \\ &= N_m\lfloor_{S_1} \end{aligned}$$

Set $M'_1 = M''$ for the result in this case.

(b) ${}^\bullet l_m{}^\bullet \cap {}^\bullet B_1{}^\bullet = \emptyset$: and

(c) $l_m{}^\bullet \subseteq B_1{}^\bullet$ and ${}^\bullet l_m \cap {}^\bullet B_1 = \emptyset$: are similar. as is:

(d) ${}^\bullet l_m{}^\bullet \subseteq {}^\bullet B_1{}^\bullet$.

2. $M \in \mathcal{M}^{B_1;B_2}_d$. From Definition 4.6.3, we may partition $LT(B_1;B_2)$ into ${};_1(LT(B_1))$ and ${};_2(LT(B_2))$. The proof follows the same lines as that for concurrent composition, noting that we do not have annotating contexts to complicate it. The only extra difficulty in the proof is at the changeover between markings of the operand High Level Petri Boxes, i.e., when l_m is the first local transition to appear in the local transition sequence with ${}^\bullet l_m \subseteq {};_1(B_1)^\bullet$. The result in this case follows from the fact that ${};_1(B_1)^\bullet = {}^\bullet{};_2(B_2)$.

3. $M \in \mathcal{M}^{B_1+B_2}_d$. Is a combination of the previous cases.

Again, denoting B_1 by B

4. $M \in \mathcal{M}^{B[f]}$: is similar to the proof for concurrent composition, although we do not have to consider the partitioning of the local transitions.

5. $M \in \mathcal{M}^{\Gamma(B)}$: clear from the definition of guarding.

6. $M \in \mathcal{M}^{\Omega_Y(B)}_d$: the most complex case. If Y does not appear free in B, i.e., $L_Y(B) = \emptyset$, then the result follows directly from Definition 6.4.13 and the Induction Hypothesis.

   Otherwise, $L_Y(B) \neq \emptyset$. From the induction hypothesis we may assume that

$$N_{m-1} = \bigcup_{i \in \{0,\ldots,n\}} M_i C_i\lfloor_{S_B \setminus {}^\bullet L_Y(B)^\bullet}$$

   with the Properties P1–P3 holding of the $M_i C_i$.

   We first present three lemmata which will be useful in the proof. The first uses the annotating $\perp$ on call and return transitions to show that the $\prec$ relationship between nodes of the tree is reflected in the C_i:

LEMMA 7.1.3

(a) $M_i \prec M_j$ implies $C_i \sqsubset C_j$,

(b) $s \in Acc(B) \cup {}^\bullet L_Y(B)^\bullet$ and $(s, C) \in M_i C_i$ implies $C_i \sqsubset C$. □ 7.1.3

Proof: (a) As $M_i \prec M_j$, by P3, $M_i C_i \triangleright \{C_j\} = {}^\bullet l_r$, where l_r is the unique return transition with $s_j \in {}^\bullet l_r$. As ${}^\bullet l_r \subseteq \mathrm{dom}(M_i C_i)$ there are previous markings K and $K' \in \mathcal{M}^B_{C_i}$ such that, if k is the unique call transition with $s_j \in k^\bullet$, then:

$$M^I_{C_i} \cdots K[k\rangle K' \cdots M_i C_i$$

and $(s_j, C_j) \in K' \setminus K$. Moreover, there is a context $C \in \mathrm{ran}(K)$ such that $C_j = {}^C k C - k^C$, whence $C \sqsubset C_j$, as k is a call transition (i.e., $\perp \leq {}^C k$ and $k^C = (.)$). But then, from H3, $C_i \sqsubseteq C$, and we have the result.

(b) If $s \in Acc(B) \cap {}^\bullet L_Y(B)$ the proof is as above. Suppose $s \in L_Y(B)^\bullet$ and let l_Y be such that $s \in l_Y{}^\bullet$, and k be the call transition such that $k^\bullet \cap {}^\bullet l_Y = \{s'\}$, say. Then there are markings K, K', L, $L' \in \mathcal{M}^B_{C_i}$ such that[7]

$$M^I_{C_i} \cdots K[k\rangle K' \cdots L[l_Y\rangle L' \cdots M_i C_i$$

with $(s, C) \in L' \setminus L$, (s', C) appearing in every marking between K' and L, and $(s', C) \in K' \setminus K$. The remainder of the proof is as above.

■ 7.1.3

The second lemma states that sharing of tokens between different M_i and M_j requires a parent/child relationship between them:

LEMMA 7.1.4 For $i \neq j$, $e \in \mathrm{ran}(M_i C_i) \cap \mathrm{ran}(M_j C_j)$ implies either $M_i \prec M_j$ or $M_j \prec M_i$. In the former case, $e = C_j$, in the latter $e = C_i$. □ 7.1.4

Proof: From H3, there are contexts e_1 and e_2 such that $e = e_1 C_i = e_2 C_j$ so that either $C_i \sqsubseteq C_j$ or $C_j \sqsubseteq C_i$. In particular, $\neg C_i \# C_j$ so that from P1, $\neg M_i \# M_j$. Assume, without loss of generality, that $M_i \prec^+ M_j$ in the tree.

If $M_i \prec M_k \prec^+ M_j$ for some k, then, from P2, $e \# C_k$ or $e \sqsubseteq C_k$. From Lemma 7.1.3(a), $C_k \sqsubset C_j$. But then $e \# C_k$ implies $e \# C_j$, and $e \sqsubseteq C_k$ implies $e \sqsubset C_j$ both of which contradict that $e = e_2 C_j$.

Hence $M_i \prec M_j$. That $e = C_j$ follows from $C_j \sqsubseteq e$ and P2. ■ 7.1.4

The third lemma states that enabling tokens appear in a unique recursive invocation:

[7] The firing of this Y-local is the first use we have made of the process variable extended label algebra.

LEMMA 7.1.5 Let $k \in LT(\Omega_Y(B))$ be such that ${}^\bullet k \times \{\!|e|\!\} \subseteq \bigcup_i M_i C_i$ and let k' be the unique local transition of B such that $k = f_\Omega(k')$. Then there is a unique j such that ${}^\bullet k' \times \{\!|e|\!\} \cap M_j C_j \neq \emptyset$. Moreover, ${}^\bullet k' \times \{\!|e|\!\} \subseteq M_j C_j$. □ 7.1.5

Proof: For the first part suppose the result does not follow, i.e., there are indices $i \neq j$, $r_1, r_2 \in {}^\bullet k'$ such that $(r_1, e) \in M_i C_i$, $(r_2, e) \in M_j C_j$. As $e \in \mathrm{ran}(M_i C_i) \cap \mathrm{ran}(M_j C_j)$ we may apply Lemma 7.1.4 to give that $M_i \prec M_j$ or $M_j \prec M_i$. Without loss of generality, we will assume the former so that $e = C_j$. From P3, $r_1 \in {}^\bullet l_r$ where l_r is the unique return transition such that $s_j \in {}^\bullet l_r$. From the construction, then, $k' = l_r$, $r_2 \in {}^\bullet l_r$ and, from P3, ${}^\bullet k' \subseteq M_i C_i \triangleright \{C_j\}$. But then $r_2 \in Acc_Y(B) \cup {}^\bullet L_Y(B)^\bullet$ and, as $(r_2, e) \in M_j C_j$, Lemma 7.1.3(b) gives that $C_j \sqsubset e$, a contradiction.

For the second part, we must show only that ${}^\bullet k' \times \{\!|e|\!\} \subseteq \bigcup_i M_i C_i$. If k is not a return transition then ${}^\bullet k = {}^\bullet k'$, and the result is immediate from the hypotheses. Otherwise, $k' = l_r$, ${}^\bullet k \cap {}^\bullet k' = \{s\} \in Acc_Y(B)$ whence, by Property 6(b), $s = s_j$ and $e = C_j$ for some $j \geq 1$. Let $M_i \prec M_j$ (exists as $M_0 \prec^+ M_j$), then $M_i C_i \triangleright \{C_j\} = {}^\bullet l_r$, by P3, and ${}^\bullet k' \times \{\!|e|\!\} \subseteq \bigcup_i M_i C_i$ as required. ■ 7.1.5

Returning to the proof of Lemma 7.1.2, as l_m is enabled at N_{m-1} there are contexts e and f such that ${}^\bullet l_m \times \{\!|e|\!\} \subseteq N_{m-1}$, $f = {}^C l_m e - l_m {}^C$ and

$$N_m = (N_{m-1} \setminus {}^\bullet l_m \times \{\!|e|\!\}) \cup l_m{}^\bullet \times \{\!|f|\!\}$$

Let $l \in LT(B)$ be the unique local transition such that $l_m = f_\Omega(l)$.

We will update the tree formed by the M_i using the local transition l (which we show to be enabled in some $M_i C_i$). We distinguish three cases depending upon whether l_m is a call or return transition on Y or neither. We begin with the simplest case:

(a) ${}^\bullet l_m{}^\bullet \cap Acc_Y(B) = \emptyset$: so that l_m is neither a call nor a return transition on Y. From the definition of f_Ω in this case, $l = l_m$, and as l_m is enabled, from the Induction Hypothesis, ${}^\bullet l \times \{\!|e|\!\} \subseteq \bigcup_i M_i C_i$. Apply Lemma 7.1.5 to give the unique j such that ${}^\bullet l \times \{\!|e|\!\} \subseteq M_j C_j$. As ${}^C l = {}^C l_m$ and $l^C = l_m{}^C$, l is enabled at $M_j C_j$ and hence at M_j. We replace the M_j label of node j with M_j' where, if $e' = e - C_j$, $f' = f - C_j$

$$M_j' = (M_j \setminus {}^\bullet l \times \{\!|e'|\!\}) \cup l^\bullet \times \{\!|f'|\!\}$$

i.e., such that $M_j[l\rangle M_j'$. Then $M_j' \in \mathcal{M}^B_{(.)}$.

As ${}^\bullet l \times \{\!|e|\!\} \cap \bigcup_{i \neq j} M_i C_i = \emptyset$ and ${}^\bullet l_m{}^\bullet \cap {}^\bullet L_Y(B)^\bullet = \emptyset$:

$$\begin{aligned}
&\Big(\Big(\bigcup_{\substack{i \in \{0,\ldots,n\} \\ i \neq j}} M_i C_i\Big) \cup M_j' C_j\Big)\lfloor_{S_B \setminus {}^\bullet L_Y(B)^\bullet} \\
&= \Big(\Big(\bigcup_{i \in \{0,\ldots,n\}} M_i C_i \lfloor_{S_B \setminus {}^\bullet L_Y(B)^\bullet}\Big) \setminus {}^\bullet l \times \{\!|e|\!\}\Big) \cup l^\bullet \times \{\!|f|\!\} \\
&= (N_{m-1} \setminus {}^\bullet l_m \times \{\!|e|\!\}) \cup l_m{}^\bullet \times \{\!|f|\!\} \\
&= N_m
\end{aligned}$$

and

$$\begin{aligned}
N_m \lfloor_{Acc_Y(B)} &= N_{m-1} \lfloor_{Acc_Y(B)} \\
&= \{\!|(s_1, C_1), \ldots, (s_n, C_n)|\!\}
\end{aligned}$$

so that Properties 6(a) and 6(b) continue to hold.

This case is illustrated in Figure 7.1(a).

(b) $l_m{}^\bullet \cap Acc_Y(B) = \{s_w\}$: so that l_m is a call transition on Y, i.e., $l_m{}^\bullet = {}^\bullet\Omega_Y(B) \cup \{s_w\}$, ${}^\bullet l = {}^\bullet l_m$, ${}^C l = {}^C l_m$ and $l^C = l_m{}^C$. We will update the tree to reflect that a recursive call is being made. Apply Lemma 7.1.5 to give the unique j such that ${}^\bullet l \times \{\!|e|\!\} \subseteq M_j C_j$. For the parent node we choose not the immediate follower marking of M_j as in the previous case but M_j' defined by $M_j[l\rangle M[l_Y\rangle M_j'$ where l_Y is the Y-local of B corresponding to s_w (that l is enabled at M_i follows as in the previous case; that l_Y is enabled at M is clear) so that, if $e' = e - C_j$, $f' = f - C_j$

$$\begin{aligned}
M_j' &= (((M_j \setminus {}^\bullet l \times \{\!|e'|\!\}) \cup l^\bullet \times \{\!|f'|\!\}) \setminus {}^\bullet l_Y \times \{\!|f'|\!\}) \cup l_Y{}^\bullet \times \{\!|f'|\!\} \\
&= (M_j \setminus {}^\bullet l \times \{\!|e'|\!\}) \cup (l_Y{}^\bullet \cup \{s_w\}) \times \{\!|f'|\!\}
\end{aligned}$$

Set $C_w = f$ and $M_w = M^I_{(.)} \in \mathcal{M}^B_{(.)}$. Clearly $(s_w, C_w) \in M_j' C_j$. Also, $C_j \sqsubset C_w$ and $e \sqsubset C_w$ since $l^C = (.)$, $\bot \leq {}^C l$ and $C_j \sqsubseteq e$ (by H3).

We show that for all j', $C_w \neq C_{j'}$ so that creating the new node does not compromise the pairwise distinctness of the C_i.

Suppose that there is some already existing node $k \leq n$ with $C_k = C_w$. We distinguish five cases:

i. $M_k \# M_j$: whence $C_k \# C_j \sqsubset C_w$, and $C_k \# C_w$, a contradiction,

ii. $M_k = M_j$: whence $C_k = C_j \sqsubset C_w$, a contradiction,

iii. $M_k \prec^+ M_j$: whence $C_k \sqsubset C_j \sqsubset C_w$, by Lemma 7.1.3(a), a contradiction,

iv. $M_j \prec M_k$: whence $(s_k, C_k) \in M_j' C_j \cap M_j C_j$ as M_k was pre-existing. As $(s_w, C_w) \in M_j' C_j$, $C_w = C_k$ implies $s_w = s_k$ by H4, whence $(s_w, C_w) = (s_k, C_k)$. But then $\{\!|(s_w, C_w), (s_k, C_k)|\!\} \subseteq M_j' C_j$, a contradiction of H1.

v. $M_j \prec^+ M_k$, but $\neg M_j \prec M_k$: whence there is a node M_q such that $M_j \prec M_q \prec^+ M_k$, $(s_q, C_q) \in M_j C_j$ and $C_j \sqsubset C_q \sqsubset C_k = C_w$. As $e \in \mathrm{ran}(M_j C_j)$, then P2 implies $e \# C_q$ or $e \sqsubseteq C_q$. If $e \# C_q$, then $e \# C_w$, a contradiction. Hence, $e \sqsubseteq C_q$. We distinguish two cases:

A. $e = C_q$: whence $l = l_r$, where l_r is the unique return transition such that $s_q \in {}^\bullet l_r$, a contradiction as l_m is a call transition,

B. $e \sqsubset C_q$: Then $e \sqsubset C_q \sqsubset C_k = C_w$. Now, noting that each C_i has prefix $\perp$, (following from 6(b) and the construction), $C_q = \perp C \epsilon$, $C_k = \perp C' \perp C \epsilon$, whereas, from the construction, $C_w = \perp C'' e$, with $\perp$ not in C'', a contradiction.

Hence the C_i remain pairwise disjoint.

As ${}^\bullet l \times \{\!|e|\!\} \cap \bigcup_{i \neq j} M_i C_i = \emptyset$, ${}^\bullet l \cap {}^\bullet L_Y(B)^\bullet = \emptyset$ and ${l_Y}^\bullet \subseteq L_Y(B)^\bullet$

$$
\begin{aligned}
&\Big(\Big(\bigcup_{\substack{i \in \{0,\ldots,n\} \\ i \neq j}} M_i C_i\Big) \cup M'_j C_j \cup M^I_{(.)} C_w\Big)\lfloor_{S_B \setminus {}^\bullet L_Y(B)^\bullet} \\
&= \Big(\Big(\bigcup_{i \in \{0,\ldots,n\}} M_i C_i \lfloor_{S_B \setminus {}^\bullet L_Y(B)^\bullet}\Big) \setminus {}^\bullet l \times \{\!|e|\!\}\Big) \cup \{s_w\} \times \{\!|f|\!\} \cup M^I_{(.)} C_w \\
&= (N_{m-1} \setminus {}^\bullet l_m \times \{\!|e|\!\}) \cup {l_m}^\bullet \times \{\!|f|\!\} \\
&= N_m
\end{aligned}
$$

and, as $(s_w, C_w) \in M'_j C_j$, ${}^\bullet l_m \cap Acc_Y(B) = \emptyset$, and ${l_m}^\bullet \cap Acc_Y(B) = \{s_w\}$

$$N_m \lfloor_{Acc_Y(B)} = N_{m-1} \lfloor_{Acc_Y(B)} \cup \{\!|(s_w, C_w)|\!\}$$

Setting $w = n + 1$ re-establishes Properties 6(a) and 6(b) in this case.

This case is illustrated in Figure 7.1(b).

(c) ${}^\bullet l_m \cap Acc_Y(B) = \{s\}$: so that l_m is a return transition on Y, i.e., ${}^\bullet l_m = B^\bullet \cup \{s\} = \Omega_Y(B)^\bullet \cup \{s\}$, ${l_m}^\bullet = l^\bullet$, ${}^C l = {}^C l_m$ and ${l^C} = {l_m}^C$.

As $s \in {}^\bullet l_m$, $(s, e) \in N_{m-1}$ so that, by the Induction Hypothesis (Property 6(b)), $s = s_j$ and $e = C_j$ for some $j \geq 1$. As the C_j are distinct, there is a unique k such that $M_k \prec M_j$ and, from P3, ${}^\bullet l \times \{\!|e|\!\} \subseteq M_k C_k$. As ${}^C l = {}^C l_m$, ${l^C} = {l_m}^C$, and ${}^\bullet l_m \times \{\!|e|\!\} \subseteq N_{m-1}$ there is a marking M'_k such that, if $e' = e - C_k$, $f' = f - C_k$, then

$$M'_k = (M_k \setminus {}^\bullet l \times \{\!|e'|\!\}) \cup l^\bullet \times \{\!|f'|\!\}$$

i.e., such that $M_k[l\rangle M'_k$. Then $M'_k \in \mathcal{M}^B_{(.)}$.

Suppose $B^\bullet \times \{\!|C_j|\!\} \not\subseteq M_j C_j$. Then, as $B^\bullet \times \{\!|C_j|\!\} \subseteq \bigcup_i M_i C_i$, there is a place, $s' \in B^\bullet$, and some other node, $M_{j'}$ say, such that $(s', C_j) \in M_{j'} C_{j'}$. Hence, by H3a, $C_j = C_{j'}$, contradicting the pairwise distinctness of the C_i.

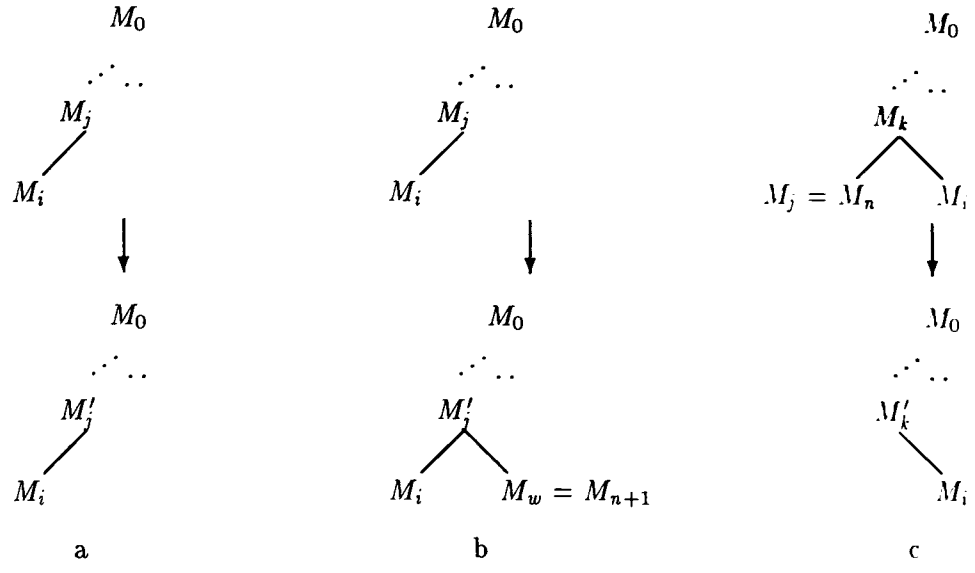


Figure 7.1: Illustrating the proof of Lemma 7.1.2. (a) illustrates the case when l_m is neither a call nor return transition: the unique node in which l is enabled is updated; (b) when l_m is a call transition: a new leaf node is created corresponding to the recursive call and the parent node is updated; (c) when l_m is a return transition: the terminated recursive call is removed from the tree, and the parent node is updated.

Hence $B^\bullet \times \{\!|C_j|\!\} \subseteq M_j C_j$ and thus (by H2) $M_j C_j = B^\bullet \times \{\!|C_j|\!\}$, so that M_j has no children, and may be removed from the tree without problem.

As ${}^\bullet l \setminus L_Y(B)^\bullet = {}^\bullet l_m \setminus \Omega_Y(B)^\bullet = \{s_j\}$

$$
\begin{aligned}
&\Big(\Big(\bigcup_{\substack{i\in\{0,\ldots,n\}\\ i\neq j\\ i\neq k}} M_i C_i\Big) \cup M_k' C_k\Big)\lfloor_{S_B \setminus {}^\bullet L_Y(B)^\bullet} \\
&= \Big(\Big(\bigcup_{\substack{i\in\{0,\ldots,n\}\\ i\neq j}} M_i C_i \lfloor_{S_B \setminus {}^\bullet L_Y(B)^\bullet}\Big) \setminus \{s_j\} \times \{\!|C_j|\!\}\Big) \cup l^\bullet \times \{\!|f|\!\} \\
&= (N_{m-1} \setminus {}^\bullet l_m \times \{\!|C_j|\!\}) \cup l_m{}^\bullet \times \{\!|f|\!\} \\
&= N_m
\end{aligned}
$$

and, as $Acc_Y(B) \cap {}^\bullet l_m = \{s_j\}$ and $Acc_Y(B) \cap l_m{}^\bullet = \emptyset$, then

$$
\begin{aligned}
N_m \lfloor_{Acc_Y(B)} &= N_{m-1} \lfloor_{Acc_Y(B)} \setminus \{(s_j, C_j)\} \\
&= \{\!|(s_1, C_1), \ldots, (s_{j-1}, C_{j-1}), (s_{j+1}, C_{j+1}) \ldots, (s_n, C_n)|\!\}
\end{aligned}
$$

Reindexing the M_i so as to eliminate the gap created by the removal of j re-establishes Properties 6(a) and 6(b) in this case.

This case is illustrated in Figure 7.1(c).

The remainder of the proof is to show that P1-P3 carry forward to the new tree. We consider each case of affected nodes in turn: (a)-(c) corresponding to the respective

cases distinguished in the construction. For nodes not addressed in the cases, the result follows directly from the induction hypotheses.

P1: (a) ${}^\bullet l_m{}^\bullet \cap Acc_Y(B) = \emptyset$: that P1 holds of all nodes of the tree follows directly from the induction hypothesis as we alter neither the tree (with the exception of M_j' replacing M_j) nor C_j in this case.

(b) ${l_m}^\bullet \cap Acc_Y(B) \neq \emptyset$: we should show that the result holds for the new node M_w, i.e., that $M_w \# M_k$ implies $C_w \# C_k$.

If $M_j' \# M_k$ (where $M_j' \prec M_w$) then $M_j \# M_k$ so that $C_j \# C_k$, by the induction hypothesis. As $C_j \sqsubset C_w$, then $C_w \# C_k$.

We next show that $C_w \# C_q$ for each M_q with $M_j' \prec M_q \neq M_w$ as then, by Lemma 7.1.3, we also have that, for all M_k with $M_j' \prec M_q \prec^* M_k$, $C_w \# C_k$, as required.

Suppose, then, that $M_j' \prec M_q \neq M_w$, but that $\neg C_w \# C_q$. From the construction we know that $C_w \neq C_q$. We show that from the two remaining cases, $C_w \sqsubset C_q$ and $C_q \sqsubset C_w$, we may derive contradictions:

i. $C_w \sqsubset C_q$: whence $C_j \sqsubset C_w \sqsubset C_q$ and $\bot \leq C_w$ so that, by H5, there is an accounting place, $s \in Acc(B) \setminus Acc_Y(B)$, and so unequal to $s_w \in Acc_Y(B)$, such that $(s, C_w) \in M_j' C_j$. But then $\{\!|(s_w, C_w), (s, C_w)|\!\} \subseteq M_j' C_j$, contradicting H4.

ii. $C_q \sqsubset C_w$: is similar.

Hence the result in this case.

(c) ${}^\bullet l_m \cap Acc_Y(B) \neq \emptyset$: follows as for P1(a).

P2: (a) ${}^\bullet l_m{}^\bullet \cap Acc_Y(B) = \emptyset$: we should show the result for $M_j' \prec M_i$; that for $M_i \prec M_j'$ following directly from the induction hypothesis as we alter neither $M_i C_i$ nor C_j.

As $M_j \prec M_i$, from P3, $(s_i, C_i) \in M_j C_j$. Moreover, from Lemma 7.1.3, $C_j \sqsubset C_i$ and, as l_m is not a return transition in this case of the construction, $(s_i, C_i) \in M_j' C_j$.

From P2, we may assume that for all contexts $g \in \mathrm{ran}(M_j C_j)$, $g \# C_i$ or $g \sqsubseteq C_i$. Suppose P2 does not hold of the new marking, i.e., $C_i \sqsubset g$, so that $C_j \sqsubset C_i \sqsubset g$. As $\bot \leq C_i$, we may apply H5 to give an accounting place $s \in Acc(B) \setminus Acc_Y(B)$, and so unequal to s_i, such that $(s, C_i) \in M_j' C_j$. But then $\{\!|(s, C_i), (s_i, C_i)|\!\} \subseteq M_j' C_j$, contradicting H4.

(b) $l_m{}^\bullet \cap Acc_Y(B) \neq \emptyset$: We distinguish two cases: $M_j' \prec M_w$ and $M_j' \prec M_i \neq M_w$:

i. $M_j' \prec M_w$: from the construction $C_j \sqsubset C_w$ and $(s_w, C_w) \in M_j' C_j$ where $s_w \in l_m{}^\bullet \cap Acc_Y(B)$. Suppose P2 does not hold of the new marking, i.e., there is a context $g \in \mathrm{ran}(M_j' C_j)$ with $C_w \sqsubset g$. But then $C_j \sqsubset C_w \sqsubset g$ and $\perp \leq C_w$. The remainder of the proof follows that of P2(a).

ii. $M_j' \prec M_i \neq M_w$: is similar.

(c) ${}^\bullet l_m \cap Acc_Y(B) \neq \emptyset$: we should show that P2 continues to hold when $M_k' \prec M_i$ or when $M_i \prec M_k'$. The proof of the former follows that for P2(a); that for the latter follows directly from the induction hypothesis as we do not alter $M_i C_i$ nor C_k.

P3: (a) ${}^\bullet l_m{}^\bullet \cap Acc_Y(B) = \emptyset$: we distinguish two cases, when $M_j' \prec M_i$ and $M_i \prec M_j'$:

i. $M_j' \prec M_i$: as $M_j \prec M_i$, from P3, we have that $M_j C_j \triangleright \{C_i\} = {}^\bullet l_r$, where l_r is the unique return transition such that $s_i \in {}^\bullet l_r$. Clearly, ${}^\bullet l \cap {}^\bullet l_r = \emptyset$, so that ${}^\bullet l_r \subseteq M_j' C_j \triangleright \{C_i\}$.

Suppose the result does not hold, i.e., $M_j' C_j \triangleright \{C_i\} \neq {}^\bullet l_r$. Then $M_j' C_j \triangleright \{C_i\} \supset {}^\bullet l_r$ and $f = C_i$. We distinguish three subcases:

A. ${}^\bullet l_m{}^\bullet \cap Acc(B) = \emptyset$: i.e., l_m is not a call nor return transition on any process variable. Then, as $\perp \leq C_i$, $C_i \sqsubseteq e$. If $C_i = e$ then there is a place $s \in {}^\bullet l \setminus {}^\bullet l_r$, such that $(s, C_i) \in M_j C_j$, contradicting P3. Otherwise, $C_j \sqsubset C_i \sqsubset e$, $\perp \leq C_i$ and we may apply H5 to give $s \in Acc(B) \setminus Acc_Y(B)$ (and so unequal to s_i), with $(s, C_i) \in M_j C_j$. But then $\{\!|(s, C_i), (s_i, C_i)|\!\} \in M_j C_j$, contradicting H4.

B. $l_m{}^\bullet \cap (Acc(B) \setminus Acc_Y(B)) = \{s\}$: then $s \neq s_i$ and $\{\!|(s, C_i), (s_i, C_i)|\!\} \in M_j' C_j$, contradicting H4.

C. ${}^\bullet l_m \cap (Acc(B) \setminus Acc_Y(B)) = \{s\}$: as $\perp \leq {l_m}^C$ we have that $(s, e) \in M_j C_j$, with $C_i \sqsubset e$ whence, as $C_j \sqsubset C_i \sqsubset e$ and $\perp \leq C_i$, by H5, there is an accounting place $r \in Acc(B) \setminus Acc_Y(B)$ (and so unequal to s_i) such that $(r, C_i) \in M_j C_j$. But then $\{\!|(r, C_i), (s_i, C_i)|\!\} \in M_j C_j$, contradicting H4.

ii. $M_i \prec M_j'$: follows directly from the induction hypothesis, as we alter neither $M_i C_i$ nor C_j.

(b) $l_m{}^\bullet \cap Acc_Y(B) = \{s\}$: We distinguish three subcases corresponding to $M_j' \prec M_i$, $M_i \prec M_j'$ and $M_j' \prec M_w$:

i. $M_j' \prec M_i$: is similar to P3(a).

ii. $M_i \prec M_j'$: follows directly from the induction hypothesis, as we alter neither $M_i C_i$ nor C_j.

iii. $M_j' \prec M_w$: whence, from the construction in this case, ${}^\bullet l_r \times \{\!|C_w|\!\} \subseteq M_j' C_j$, where l_r is the unique return transition such that $s_w \in {}^\bullet l_r$.
Suppose the result does not hold of the new marking, i.e., there is an $s \notin {}^\bullet l_r$ such that $(s, C_w) \in M_j' C_j$. But then $(s, C_w) \in M_j' C_j \cap M_j C_j$. As $C_j \sqsubset C_w$ and $\perp \leq C_w$, by H5 we may assume that $s \in Acc(B)$. As $s \notin {}^\bullet l_r$, $s \neq s_w$. But then $\{\!|(s, C_w), (s_w, C_w)|\!\} \subseteq M_j' C_j$, contradicting H4.

(c) ${}^\bullet l_m \cap Acc_Y(B) \neq \emptyset$: as $M_j' \prec M_i$, $l \neq l_r$. The remainder of the proof is similar to P3(a). That of $M_i \prec M_j'$ follows directly from the induction hypothesis, as we alter neither $M_i C_i$ nor C_j. Moreover, these are the only possibilities in this case.

Hence the result. ■ 7.1.2

Given that Lemma 7.1.2 forms the inductive step, we must show that Hypotheses H1-H5 of Lemma 7.1.2 continue to hold of the composed High Level Petri Box. The result is complicated a little since, as has already been mentioned, the winding of a High Level Petri Box is not necessarily memoryless. However, guarding does not require a memoryless High Level Petri Box to produce a memoryless High Level Petri Box, and this will allow us to complete the inductive step:

LEMMA 7.1.6 Let P be an able pre-High Level Petri Box such that for all $d \in C$, and $M \in \mathcal{M}_d^B$, $P^\bullet \times \{\!|d|\!\} \subseteq M$ implies $M = P^\bullet \times \{\!|d|\!\}$. Then $\Gamma(P)$ is memoryless. □ 7.1.6

Proof: For convenience, we will assume that $S_P \subseteq S_{\Gamma(P)}$, $LT(P) \subseteq LT(\Gamma(P))$ and $T_P \subseteq T_{\Gamma(P)}$.

Suppose $M \in \mathcal{M}_d^{\Gamma(P)}$. We claim that either $M = M_d^{I,\Gamma(P)}$, $M = M_d^{T,\Gamma(P)}$ or there is a marking $M' \in \mathcal{M}_{\perp d}^P$ such that $M = M' \cup \{s \rightarrow \{(\perp d)\}\}$, whence the result.

The proof is by induction over the local transition sequence

$$M_d^{I,\Gamma(P)}[l_1\rangle \cdots M_{n-1}[l_n\rangle M$$

which led to M:

Base Case: $n = 0$: the result follows by inspection.

Inductive Step: Suppose that $n > 0$, and that the result holds for $n-1$, i.e., either $M_{n-1} = M_d^{I,\Gamma(P)}$, $M_{n-1} = M_d^{T,\Gamma(P)}$ or there is a marking $M'_{n-1} \in \mathcal{M}^P_{\perp d}$ such that $M_{n-1} = M'_{n-1} \cup \{s \rightarrow \{(.\perp d)\}\}$. We distinguish three corresponding cases:

1. $M_{n-1} = M_d^{I,\Gamma(P)}$, whence l_n is the call transition provided by the guarding auxiliary. Choose $M'_n = M^P_{.\perp d}$ for the result in this case;
2. $M_{n-1} = M_d^{T,\Gamma(P)}$: impossible, as l_n is enabled at M_{n-1};
3. $M_{n-1} = M'_{n-1} \cup \{s \rightarrow \{(.\perp d)\}\}$, for $M'_{n-1} \in \mathcal{M}^P_{\perp d}$. If $l_n \in LT(P)$, choose M'_n such that $M'_{n-1}[l_n\rangle M'_n$ in $\mathcal{M}_d^{\Gamma(P)}$ for the result. Otherwise l_n is the return transition supplied by the guarding whence, as $M(s) = \{\!|.\perp d|\!\}$, and l_n is enabled at M_{n-1}, $P^\bullet \times \{\!|.\perp d|\!\} \subseteq M'_{n-1}$. But then $M'_{n-1} = P^\bullet \times \{\!|.\perp d|\!\}$, from the hypotheses, and $M' = M_d^{T,\Gamma(P)}$.

Hence the result. ■ 7.1.6

Whence:

LEMMA 7.1.7 Let B_i, $i = 1,2$, satisfy the statement of Lemma 7.1.2. Then each of $B_1 \| B_2$, $B_1;B_2$, B_1+B_2 and (denoting B_1 by B) $B[f]$ and $\mu Y.B$ satisfy Hypotheses H1-H5 of that lemma. □ 7.1.7

Proof: For concurrent, choice and causal composition and relabelling the result follows easily from the induction hypotheses and Lemma 7.1.2.

For the recursive construct the result follows from those for guarding and winding (and the syntactic nature of refinement from which winding is derived). For Hypotheses H1 (safeness), H3 (initial marking independence) and H4 (markings of accounting places) the result follows directly.

For H2 (memorylessness) the result is complicated a little since, as has already been mentioned, the winding of a High Level Petri Box is not necessarily memoryless. In this case we must show that $\Omega_Y(B)$ satisfies the hypotheses of Lemma 7.1.6, from which, as $\Omega_Y(B)$ is able (note 2, following Definition 6.5.8), we have that $\Gamma(\Omega_Y(B))$ is memoryless.

That each of Hypotheses H1-H5 follow for guarding follows easily from the induction hypothesis. The proofs for winding follow:

H1: we prove that $\Omega_Y(B)$ is d-safe for all standard initial markings d. For, suppose not, i.e., there is a marking $M \in \mathcal{M}_d^{\Omega_Y(B)}$, a place $s \in \mathrm{dom}(M)$, and a context $e \in \mathrm{ran}(M)$ such that $M(s)(e) \geq 2$. From Lemma 7.1.2 we may assume that M is decomposable as is described in 6(a) with Properties P1-P3 holding of the M_iC_i.

As the M_iC_i are safe, $M(s)(e) \geq 2$ implies there are $i \neq j$ such that $(s,e) \in M_iC_i \cap M_jC_j$, whence Lemma 7.1.4 applies and either $M_i \prec M_j$ or $M_j \prec M_i$. Assume, without loss of generality, the former. Then $e = C_j$ whence $s \in {}^\bullet l_r$ by P3 with l_r the unique return transition such that $s_j \in {}^\bullet l_r$. But then $s \in Acc_Y(B) \cup {}^\bullet L_Y(B)^\bullet$ and $(s_2, e) \in M_jC_j$ so that Lemma 7.1.3 gives that $C_j \sqsubset e$, a contradiction.

Hence $i = j$, a contradiction and we have the result.

H2: from Definition 6.5.8 we have that $\Omega_Y(B)$ is an able pre-High Level Petri Box. We prove that $\Omega_Y(B)$ also satisfies the other hypotheses of Lemma 7.1.6, i.e., that, for all $M \in \mathcal{M}_d^{\Omega_Y(B)}$ with $\Omega_Y(B)^\bullet \times \{\!|d|\!\} \subseteq M$, $M = \Omega_Y(B)^\bullet \times \{\!|d|\!\}$. From Lemma 7.1.2 we may assume that M is decomposable as is described in 6(a) with Properties P1-P3 holding of the M_iC_i.

Now, $\Omega_Y(B)^\bullet \times \{\!|d|\!\} \subseteq M$ implies $B^\bullet \times \{\!|d|\!\} \subseteq M_0C_0$ as for all $j > 0$, $d \sqsubset C_j$ by Lemma 7.1.3. But then $M_0C_0 = B^\bullet \times \{\!|d|\!\}$ as B is memoryless by assumption, whence M_0 has no children, and we have the result.

The memorylessness of $\mu Y.B$ follows from the above argument.

H3: suppose that the hypothesis does not hold, i.e., for $M \in \mathcal{M}_d^B$, there is an $e \in \mathrm{ran}(M)$ such that $d \# e$ or $e \sqsubset d$. From Lemma 7.1.2 we may assume that M is decomposable as is described in 6(a) with Properties P1-P3 holding of the M_iC_i.

Let $(s,e) \in M_iC_i$ for some i. As $M_iC_i \in \mathcal{M}_{C_i}^B$ we may apply P3 to give that $d \sqsubseteq C_i \sqsubseteq e$, a contradiction. Hence the result.

H4: suppose that the hypothesis does not hold, i.e., there is a marking M such that $e \in M(s_1) \cap M(s_2)$ but that $s_1 \neq s_2$ for accounting places s_1 and s_2. From Lemma 7.1.2 we may assume that M is decomposable as is described in 6(a) with Properties P1-P3 holding of the M_iC_i.

Suppose $(s_1, e) \in M_iC_i$, $(s_2, e) \in M_jC_j$, but that $i \neq j$. Then, from Lemma 7.1.4, we may assume that $M_i \prec M_j$ and that $e = C_j$ whence $s_1 \in {}^\bullet l_r$ where $s_j \in {}^\bullet l_r$ where l_r is the unique return transition such that $s_j \in {}^\bullet l_r$. However, $(s_2, e) \in M_jC_j$ so that Lemma 7.1.3 applies to give $C_j \sqsubset e$, a contradiction.

Hence $i = j$ and the result follows from the induction hypothesis.

H5: suppose that $\Omega_Y(B)$ is Z-guarded for some Z in $\mathcal{PV}$ and let M, d and C satisfy the conditions of H5 of Lemma 7.1.2. From Lemma 7.1.2, we may assume that M is decomposable as is described in 6(a) with Properties P1-P3 holding of the M_iC_i. For the first part of H5, we note that, as $d \sqsubset C$ and $M \subseteq \bigcup_i M_iC_i$, there is an index j, and a place s such that $(s, C) \in M_jC_j$. As $M_jC_j \in \mathcal{M}^B_{C_j}$, by H3, $C_j \sqsubseteq C$. If $C_j \sqsubset C$ then $C \in \mathrm{ran}(M_jC_j)$ and $\perp \leq C$, so that we may apply H5 for the result in this case.

Otherwise $C_j = C$. Note that, in this case, $j \neq 0$, as otherwise $C_j = d \sqsubset C$, a contradiction. So we may assume $j > 0$. Then there is a node i such that $M_i \prec M_j$. By P3, $(s_j, C_j) \in M_iC_i$, where $s_j \in {}^\bullet l_r \cap Acc_Y(B)$, with l_r a return transition. Then $(s_j, C) \in M_iC_i$ and we may choose s_j for the result in this case.

For the second part let s', C' be such that $C \sqsubset C'$ and $(s', C') \in \mathrm{ran}(M)$. As $d \sqsubset C$ we may find $s \in Acc(\Omega_Y(B))$ such that $(s, C) \in M$. If $Z = Y$, then $s \notin Acc_Z(\Omega_Y(B)) = \emptyset$ and we are done.

Suppose, then, that $Z \neq Y$ so that, from Lemma 6.5.10, B was also Z-guarded and $Acc_Z(B) = Acc_Z(\Omega_Y(B))$. Let j be as in the first part of the proof, i.e., such that $(s, C) \in M_jC_j$. We distinguish two cases:

1. $C_j = C$: whence we may assume that $s \in Acc_Y(B)$. But $Acc_Y(B) \subseteq Acc(\Omega_Y(B)) \setminus Acc_Z(\Omega_Y(B))$ and we have the result in this case,
2. $C_j \sqsubset C$: if $C' \in \mathrm{ran}(M_jC_j)$ then the the result follows directly from the induction hypothesis. Otherwise $C' \notin \mathrm{ran}(M_jC_j)$ whence, by 6(a) and P1, there is a node M_q and an accounting place $s_q \in Acc_Y(B)$ such that $M_j \prec M_q$, $(s_q, C_q) \in M_jC_j$ and $C_j \sqsubset C \sqsubseteq C_q \sqsubseteq C'$. If $C = C_q$ then $(s_q, C) \in M$ with $s_q \in Acc_Y(\Omega_Y(B)) \setminus Acc_Z(B)$ and we have the result. Otherwise, $C \sqsubset C_q$ and we may apply the induction hypothesis with $C_j \sqsubset C \sqsubset C_q$ to give an $s \in Acc(B) \setminus Acc_Z(B)$ such that $(s, C) \in M_jC_j$. But then $s \in Acc(\Omega_Y(B)) \setminus Acc_Z(\Omega_Y(B))$ and $(s, C) \in M$ and we have the result.

■ 7.1.7

Returning to the proof of Theorem 7.1.1:

Proof: Proof is by induction over the structure of B.

For B a Basic High Level Petri Box the result follows by inspection.

Otherwise, B is one of $B_1 \| B_2$, $B_1;B_2$, B_1+B_2, $B_1[f]$ or $\mu Y.B_1$ whence we may apply Lemma 7.1.7 for the result. ■ 7.1.1

We expect that the extra hypotheses of Lemma 7.1.2, over and above safeness and memorylessness, will be useful tools for reasoning on the behaviours of High Level Petri Boxes, as are the decompositions of the markings which were given therein. We, therefore, state them as corollaries:

COROLLARY 7.1.8 For B a syntactically generated High Level Petri Box, $M \in \mathcal{M}_d^B$ implies for all $e \in \mathrm{ran}(M)$, $d \sqsubseteq e$. □ 7.1.8

COROLLARY 7.1.9 For B a Y-guarded syntactically generated High Level Petri Box, for all contexts d, markings $M \in \mathcal{M}_d^B$, and C, $C' \in \mathrm{ran}(M)$ such that $d \sqsubset C \sqsubseteq C'$ with $\perp \leq C$ there is an accounting place $s \in Acc(B)$ such that $(s, C) \in M$. If, in addition $C \sqsubset C'$, then there is an accounting place $s \in Acc(B) \setminus Acc_Y(B)$ such that $(s, C) \in M$. □ 7.1.9

and

COROLLARY 7.1.10 Let B be a syntactically generated High Level Petri Box, $M \in \mathcal{M}_d^B$ a marking reachable from a standard initial marking d, and s_1 and s_2 accounting places such that $M(s_1) \cap M(s_2) \neq \emptyset$. Then $s_1 = s_2$. □ 7.1.10

Finally for this section, given the highly structured nature of the tokens which occupy the accounting places, we can show that there is a unique tree at any reachable marking which satisfies the hypotheses of Lemma 7.1.2:

COROLLARY 7.1.11 Let $M \in \mathcal{M}_d^{\Omega_Y(B)}$ and suppose that $M = \bigcup_{i=0}^{p} M_i C_i \lfloor_{S_B \setminus {}^\bullet L_Y(B)^\bullet} = \bigcup_{i=0}^{q} N_i D_i \lfloor_{S_B \setminus {}^\bullet L_Y(B)^\bullet}$ with the $M_i C_i$ and the $N_i D_i$ satisfying the hypotheses of Lemma 7.1.2. Then:

1. $p = q$;

Moreover, we may re-index the $N_i D_i$ so that, for each $i \in \{0, \ldots, p\}$.

2. $C_i = D_i$;

3. $M_i \prec M_j$ if and only if $N_i \prec N_j$ (so that the trees are isomorphic); and

4. $M_i C_i = N_i D_i$. □ 7.1.11

Proof: 1. and,

2. follow directly from Property 6(b) of Lemma 7.1.2 as the C_i and D_i are pairwise distinct,

3. follows from P1 and Lemma 7.1.3(a),

4. follows from P1—P3 by induction on the structure of the tree (with base case the leaf nodes) using from the fact that C_j forms a postfix of all tokens in $M_j C_j$ and D_j a postfix of all tokens in $N_j D_j$. ■ 7.1.11

In the sequel it will be convenient to assume that a marking $M \in \mathcal{M}_d^{\Omega_Y(B)}$ has related decomposition under Lemma 7.1.2 $\bigcup_{i=0}^{n} M_i C_i \lfloor_{S_B \setminus {}^\bullet L_Y(B)^\bullet}$, with any decoration on M appearing on the M_i, e.g., $M' = \bigcup_{i=0}^{n} M_i' C_i \lfloor_{S_B \setminus {}^\bullet L_Y(B)^\bullet}$, *etc.*

Moreover, as all markings will be safe, we may omit multi-set brackets when used in markings, e.g., we will write ${}^\bullet l \times \{e\} \subseteq M$, rather than ${}^\bullet l \times \{\!|e|\!\} \subseteq M$, when $M[l\rangle$ (for some enabling ϵ).

7.2 Strong Bisimulation on High Level Petri Boxes

We next develop the notion of a *strong bisimulation* on High Level Petri Boxes, which we show to be a behavioural congruence for syntactically generated High Level Petri Boxes.

The strong bisimulation we define is based on the *concurrency preserving strong bisimulation* of Olderog, [Old91, pg. 25]. A distinguishing property of Olderog's bisimulation is that it 'preserves the concurrency' present in safe P/T nets, in the technical sense that it respects the *causal sub-nets* of a P/T net, where a causal sub-net of a P/T net is a sub-net in which all choices have been resolved. In particular, this implies that causally incomparable transitions are related to causally incomparable transitions under the strong bisimulation so that 'true concurrency' and 'interleaving' interpretations of the same situation are not regarded as bisimilar.

The translation of Olderog's strong bisimulation to High Level Petri Boxes is a straightforward generalisation to the underlying structure of High Level Petri Boxes. Moreover, the definition on High Level Petri Boxes preserves concurrency in the technical sense given by Proposition 7.2.6, although this result is not stated in terms of 'causal sub-High Level Petri Boxes' but only in

terms of concurrent enablings. (The generalisation of 'causal sub-net' to High Level Petri Boxes appears not to be straightforward, due to the individuality of tokens: for a characterisation we would be required to characterise causal sub-nets of the P/T net unfolding of a High Level Petri Box on the structure of the High Level Petri Box, and although this is probably not too difficult, it is anyway unnecessary for the following treatment.)

That strong bisimulation on High Level Petri Boxes preserves concurrency is a necessary condition for it to extend to a congruence over the High Level Petri Boxes. In particular, the demonstration that strong bisimulation and relabelling commute, i.e., if B and B' are strong bisimilar then so are $B[f]$ and $B'[f]$, depends crucially on this property.

The major part of the motivation for the development of strong bisimulation is that it will allow us to discharge the claim that the recursive construct is a *behavioural* fix-point of the refinement operator, i.e., we will be able to show that recursion is a solution to Equation 6.1 when $\doteq$ is strong bisimulation. This means that the 'loops' in High Level Petri Boxes introduced through the recursive construct may be partially unwound without affecting the behaviour of the High Level Petri Box. We note that, in general, no finite structural solution to Equation 6.1 can exist, so that our behavioural solution is a best possible.

The bisimulation is strong in the usual sense: in that it does not consider equivalence modulo silent actions, for instance, a High Level Petri Box B and its Y-guarding may not be strongly bisimilar. We do not formally consider weak bisimulations (observational congruences) in this work. A short discussion on weakenings of strong bisimulation (whilst preserving its strong nature) is given later in this chapter.

7.2.1 Lifting Interface Respecting Relations to High Level Petri Boxes

We begin the development of strong bisimulation with a second form of lifting of a relation to High Level Petri Boxes:

DEFINITION 7.2.1 *[Cf. [Old91, Pg. 25]]* Given safe, able pre-High Level Petri Boxes B_1 and B_2 over the same $\mathcal{PV}$-extended label algebra $\mathcal{L}$ and a relation $\mathcal{R} \subseteq S_1 \times S_2$ we define the *lifting* of $\mathcal{R}$ to be that relation $\widehat{\mathcal{R}}$ such that:

1. for sets $U_i \subseteq S_i$, $i = 1, 2$, we write $U_1 \;\widehat{\mathcal{R}}\; U_2$ when $(U_1 \times U_2) \cap \mathcal{R}$ is a bijection between U_1 and U_2;

2. for local transitions $l_i \in LT(B_i)$, $i = 1,2$, we write $l_1 \,\widehat{\mathcal{R}}\, l_2$ when ${}^\bullet l_1 \,\widehat{\mathcal{R}}\, {}^\bullet l_2$, ${}^C l_1 = {}^C l_2$, $\overline{l_1} = \overline{l_2}$, ${l_1}^C = {l_2}^C$ and ${l_1}^\bullet \,\widehat{\mathcal{R}}\, {l_2}^\bullet$;

3. for finite multisets of local transitions $t_1 = \{\!|l_1, \ldots, l_n|\!\} \in \mathcal{M}_{\mathcal{F}}(LT(B_1))$ and $t_2 = \{\!|k_1, \ldots, k_m|\!\} \in \mathcal{M}_{\mathcal{F}}(LT(B_2))$ we write $t_1 \,\widehat{\mathcal{R}}\, t_2$ when $n = m$ and[8] $l_i \,\widehat{\mathcal{R}}\, k_i$, for each $i = 1, \ldots, n$;

4. for finite relations $M_i \subseteq S_i \times \mathcal{C}_{\mathcal{L}}$ we write $M_1 \,\widehat{\mathcal{R}}\, M_2$ when, for all $d \in \mathcal{C}_{\mathcal{L}}$, $(M_1 \triangleright \{d\}) \,\widehat{\mathcal{R}}\, (M_2 \triangleright \{d\})$.

■ 7.2.1

For any High Level Petri Box B, $T_B \subseteq \mathcal{M}_{\mathcal{F}}(LT(B))$ so that Item 3 allows us to relate the abstract transitions of B_1 and B_2. Between abstract transitions the induced relation is very strong, viz., $t_1 \,\widehat{\mathcal{R}}\, t_2$ implies $t_1 = \{\!|l_1, \ldots, l_n|\!\}$ and $t_2 = \{\!|k_1, \ldots, k_n|\!\}$, with $l_i \,\widehat{\mathcal{R}}\, k_i$ and, hence, ${}^\bullet l_i \,\widehat{\mathcal{R}}\, {}^\bullet k_i$ and ${l_i}^\bullet \,\widehat{\mathcal{R}}\, {k_i}^\bullet$. In this case, we may assume, without loss of generality, that, $index(t_1) = index(t_2) = \{x_1, \ldots, x_n, y_1, \ldots, y_n\}$ with x_i annotating arcs (r, t_1) and (s, t_2) when $r \in {}^\bullet l_i$, $s \in {}^\bullet k_i$ (and $r \,\mathcal{R}\, s$); y_i annotating arcs (t_1, r) and (t_2, s) when $r \in {l_i}^\bullet$, $s \in {k_i}^\bullet$ (and $r \,\mathcal{R}\, s$). Moreover, as ${}^C l_i = {}^C k_i$, $\overline{l_i} = \overline{k_i}$ and ${l_i}^C = {k_i}^C$, under this assumption we have that $\mathrm{subs}^{B_1}(t_1) = \mathrm{subs}^{B_2}(t_2)$.

For any safe (syntactically generated) High Level Petri Box B over $\mathcal{L}$, $\mathcal{M}_d^B \subseteq \mathbb{P}(S_i \times \mathcal{C}_{\mathcal{L}})$ is finite so that Item 3 allows us to relate safe markings under $\widehat{\mathcal{R}}$. In this case, $M_1 \,\widehat{\mathcal{R}}\, M_2$ implies $M_1 = \{(s_1, d_1), \ldots, (s_n, d_n)\}$, $M_2 = \{(r_1, d_1), \ldots, (r_n, d_n)\}$ and[9] $s_i \,\mathcal{R}\, r_i$, $i = 1, \ldots, n$.

DEFINITION 7.2.2 Let B_1 and B_2 be syntactically generated High Level Petri Boxes. We say that a relation $\mathcal{R} \subseteq S_1 \times S_2$ is *proper* if $r \,\mathcal{R}\, s$ implies, for each $Y \in \mathcal{FV}(B_1) \cup \mathcal{FV}(B_2)$

1. $r \in {}^\bullet L_Y(B_1) \Leftrightarrow s \in {}^\bullet L_Y(B_2)$,

2. $r \in L_Y(B_1)^\bullet \Leftrightarrow s \in L_Y(B_2)^\bullet$,

3. $r \in Acc_Y(B_1) \Leftrightarrow s \in Acc_Y(B_2)$,

4. for $l \in L_Y(B_1)$ and $k \in L_Y(B_2)$ with accounting places s^l and s^k respectively, then $s^l \,\mathcal{R}\, s^k$ implies $l \,\widehat{\mathcal{R}}\, k$.

A proper relation $\mathcal{R}$ is *interface respecting* if $r \,\mathcal{R}\, s$ implies $\lambda_1(r) = \lambda_2(s)$. ■ 7.2.2

[8] With the k_i reindexed if necessary.
[9] After reindexing, if necessary.

A proper relation respects the structure introduced through the Y-guarding of a High Level Petri Box, and facilitates the demonstration that strong bisimulation commutes with the recursive construct. That a relation $\mathcal{R}$ is interface respecting implies, in addition, that if, for local transitions $l \in LT(B_1)$ and $k \in LT(B_2)$, $l\ \widehat{\mathcal{R}}\ k$, then $l(.f)_{B_1}\ \widehat{\mathcal{R}}\ l(.f)_{B_2}$.

7.2.2 Strong Bisimulation

We are now in a position to define what it means for two High Level Petri Boxes to be *strongly bisimilar*:

DEFINITION 7.2.3 *[Cf. [Old91, Defn. 2.3.1]]* Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be label algebras and $d \in C_{\mathcal{L}_1}$. We say that safe, able pre-High Level Petri Boxes B_1 and B_2 over $\mathcal{L}_1^{\mathcal{PV}}$ and $\mathcal{L}_2^{\mathcal{PV}}$ respectively are *pre-d-strongly bisimilar* if $\mathcal{L}_1 = \mathcal{L}_2 = \mathcal{L}$ and there exists a proper relation $\mathcal{R} \subseteq S_1 \times S_2$ such that, in the most permissive label algebra $\mathcal{L}_{\mathcal{MP}}^{\mathcal{PV}}$:

1. $M_d^{I,B_1}\ \widehat{\mathcal{R}}\ M_d^{I,B_2}$,
2. for all markings $M_i \in \mathcal{M}_d^{B_i}$, $i = 1,2$, $t_1 \in T_1$, and $\alpha \in \mathrm{subs}^{B_1}(t_1)$, whenever $M_1\ \widehat{\mathcal{R}}\ M_2$, and $M_1[t_1{:}\alpha\rangle N_1$ there is a $t_2 \in T_2$ with $t_1\ \widehat{\mathcal{R}}\ t_2$ and a marking $N_2 \in \mathcal{M}_d^{B_2}$ such that $M_2[t_2{:}\alpha\rangle N_2$ and $N_1\ \widehat{\mathcal{R}}\ N_2$. Moreover, if $N_1 = M_d^{T,B_1}$ then $N_2 = M_d^{T,B_2}$ also,
3. for all markings $M_i \in \mathcal{M}_d^{B_i}$, $i = 1,2$, $t_2 \in T_2$, and $\alpha \in \mathrm{subs}^{B_2}(t_2)$, whenever $M_1\ \widehat{\mathcal{R}}\ M_2$, and $M_2[t_2{:}\alpha\rangle N_2$ there is a $t_1 \in T_1$ with $t_1\ \widehat{\mathcal{R}}\ t_2$ and a marking $N_1 \in \mathcal{M}_d^{B_1}$ such that $M_1[t_1{:}\alpha\rangle N_1$ and $N_1\ \widehat{\mathcal{R}}\ N_2$. Moreover, if $N_2 = M_d^{T,B_2}$ then $N_1 = \mathcal{M}_d^{T,B_1}$ also.

If, in addition to being proper, $\mathcal{R}$ is an interface respecting relation, we say that B_1 and B_2 are *d-strongly bisimilar*, written $B_1 \approx^d B_2$.

B_1 and B_2 are *strongly bisimilar*, written $B_1 \approx B_2$ if they are d-strongly bisimilar for each $d \in C_{\mathcal{L}_1}(= C_{\mathcal{L}_2})$, with the same interface respecting relation $\mathcal{R}$ being used for each d. ■ 7.2.3

Item 1 forms the *initialisation property*; Items 2 and 3 form the *transfer property*. If $B \approx B'$ with underlying relation $\mathcal{R}$, $\mathcal{R}$ is called a *strong bisimulation (between B_1 and B_2)*.

That terminal markings should be reached together in the transfer property implies that we require the same termination behaviour of (pre-)strongly bisimilar High Level Petri Boxes. However, we also note that, if $\mathcal{R}$ is interface respecting, then for syntactically generated (and hence safe and memoryless) High Level Petri Boxes these requirements are trivially satisfied and are

redundant in this case. (In fact, the notion of pre-d-strong bisimulation and the inclusion of the extra requirement is only to provide a slightly more general result in the proof of the congruence nature of strong bisimulation which appears below, which facilitates the demonstration of the fix-point nature of the recursive construct.)

Note, also, that we do not consider the label generated in the firing of related transitions, the justification that this is unnecessary being given by the following, easily checkable result:

LEMMA 7.2.4 Let $B_1 \approx^d B_2$ be safe, able pre-High Level Petri Boxes over some label algebra, $\mathcal{L}^{pv}$, and $M_i \in \mathcal{M}_i^d$, $i = 1,2$, markings such that $M_1 \,\widehat{\mathcal{R}}\, M_2$, and $t_i \in T_i$, $i = 1,2$ abstract transitions with $t_1 \,\widehat{\mathcal{R}}\, t_2$. Then $M_1 \left[\begin{smallmatrix} a_1 \\ t_1 : \alpha \end{smallmatrix} \right\rangle N_1$ and $M_2 \left[\begin{smallmatrix} a_2 \\ t_2 : \alpha \end{smallmatrix} \right\rangle N_2$ implies $a_1 = a_2$. □ 7.2.4

It is not difficult to see that $\approx^d$ and $\approx$ are equivalence relations. Moreover,

PROPOSITION 7.2.5 $B_1 \equiv B_2$ implies $B_1 \approx B_2$. □ 7.2.5

Proof: If $\langle \sigma, \kappa, \tau \rangle : B_1 \equiv B_2$ then σ is a interface respecting relation between S_{B_1} and S_{B_2}. That it forms an interface respecting bisimulation between B_1 and B_2 follows from Proposition 2.5.21. ■ 7.2.5

7.2.2.1 Strong Bisimulation Preserves Concurrency

Suppose $M[l\rangle M'$ and $M[l'\rangle M''$ in some safe High Level Petri Box over some label algebra. Then there are contexts e and e' such that ${}^\bullet l \times \{e\} \cup {}^\bullet l' \times \{e'\} \subseteq M$. If ${}^\bullet l \times \{e\} \cap {}^\bullet l' \times \{e'\} = \emptyset$, we will say that l and l' are *concurrently enabled* at M. As for low level models, concurrent enabling implies causal independence, we may consider the simultaneous (but not necessarily synchronous firing) of concurrently enabled local transitions.

Suppose $B \approx^d B'$ are syntactically generated d-strongly bisimilar High Level Petri Boxes with underlying bisimulation $\mathcal{R}$. Let $M_i \in \mathcal{M}_d^B$, $i = 0,1,2$, l_1, $l_2 \in LT(B)$ and $\alpha_i \in \text{subs}^B(l_i)$, be such that $M_0[l_i : \alpha_i\rangle M_i$, $i = 1,2$, with l_1 and l_2 concurrently enabled at M_0.

As $B \approx^d B'$, there exist N_1, $N_2 \in \mathcal{M}_d^{B'}$, k_1, $k_2 \in LT(B')$ such that $l_i \,\widehat{\mathcal{R}}\, k_i$, $M_i \,\widehat{\mathcal{R}}\, N_i$, and $N_0[k_i : \alpha_i\rangle N_i$, $i = 1,2$. Then

PROPOSITION 7.2.6 k_1 and k_2 are concurrently enabled at N_0. □ 7.2.6

Proof: If $e_i = \alpha_i(x_i)$, $i = 1, 2$, (x_i annotating the input arcs of l_i), then k_1 and k_2 not concurrently enabled at N_0 implies $e_1 = e_2 = e$, say, and ${}^\bullet k_1 \cap {}^\bullet k_2 \neq \emptyset$.

As $M_0 \,\widehat{\mathcal{R}}\, N_0$ we have, in particular, that $(M_0 \triangleright \{e\}) \times (N_0 \triangleright \{e\}) \cap \mathcal{R}$ is a bijection between $(M_0 \triangleright \{e\})$ and $(N_0 \triangleright \{e\})$. As l_1 and l_2 are concurrently enabled, but k_1 and k_2 are not, $|(M_0 \triangleright \{e\}) \setminus ({}^\bullet l_1 \cup {}^\bullet l_2)| < |(N_0 \triangleright \{e\}) \setminus ({}^\bullet k_1 \cup {}^\bullet k_2)|$ so that there are places $r \in {}^\bullet l_1 \cup {}^\bullet l_2$ and $s \in (N_0 \triangleright \{e\}) \setminus ({}^\bullet k_1 \cup {}^\bullet k_2)$ such that $r \,\mathcal{R}\, s$. We will assume, without loss of generality, that $r \in {}^\bullet l_1$.

That $l_1 \,\widehat{\mathcal{R}}\, k_1$ implies, in particular, that $({}^\bullet l_1 \times {}^\bullet k_1) \cap \mathcal{R}$ is a bijection so that, as $r \in {}^\bullet l_1$, there is a place $s' \in {}^\bullet k_1$ (and so unequal to s) such that $r \,\mathcal{R}\, s'$.

But then $r \,\mathcal{R}\, s$ and $r \,\mathcal{R}\, s'$, contradicting that $(M_0 \triangleright \{e\}) \,\widehat{\mathcal{R}}\, (N_0 \triangleright \{\epsilon\})$.

Hence the result. ■ 7.2.6

The main use that Proposition 7.2.6 is put to here stems from the fact that, in the most permissive label algebra, the enabling of an abstract transition of a complete High Level Petri Box implies the concurrent enabling of each of its constituent local transitions. From the proposition we may then find corresponding concurrently enabled local transitions in a strongly bisimilar High Level Petri Box, and if these can be combined into an abstract transition then that abstract transition will be enabled in the most permissive label algebra.

7.2.3 Congruence Nature of Strong Bisimulation

The next three results develop the restricted congruence nature of strong bisimulation:

THEOREM 7.2.7 $\approx$ is a congruence relation with respect to the operators of Definition 4.8.1. □ 7.2.7

We will show the restricted congruence nature of strong bisimulation with respect to refinement and recursion later.

Proof: Throughout the proof we will assume that B_1, B_2, B_3 and B_4 are syntactically generated High Level Petri Boxes over a label algebra $\mathcal{L} = \langle A, R \rangle$ (which, for simplicity, we will assume to be maximally permissive and process variable extended), such that

$B_1 \approx B_2$ with underlying strong bisimulation $\mathcal{R}_{12}$ and $B_3 \approx B_4$ with underlying strong bisimulation $\mathcal{R}_{34}$.

We must show that $B_1 \| B_3 \approx B_2 \| B_4$, $B_1 ; B_3 \approx B_2 ; B_4$, $B_1 + B_3 \approx B_2 + B_4$ and. if B_1 is denoted B and B_2 denoted B', then for $f \in R$ a relabelling, $B[f] \approx B'[f]$.

The case for relabelling contains most of the ideas for the other cases so that we present it first. For notational convenience we will denote B_1 by B, B_2 by B' and $\mathcal{R}_{12}$ by $\mathcal{R}$.

1. $B[f] \approx B'[f]$: we claim that $\mathcal{R}_{[f]} = \mathcal{R}$ is an interface respecting strong bisimulation between $B[f]$ and $B'[f]$.

   That $\mathcal{R}_{[f]}$ is an interface respecting relation and that the initialisation property holds under $\mathcal{R}_{[f]}$ are immediate from the same properties of $\mathcal{R}$, as $S_B = S_{B[f]}$ and $S_{B'} = S_{B'[f]}$.

   For the transfer property, consider an abstract transition $t = \{\!|l_1, \ldots, l_n|\!\} \in T_B$ such that $M[t:\alpha\rangle M'$, for $M \in \mathcal{M}_d^{B[f]}$. As we work in the most permissive label enabled algebra, the local transitions comprising t are concurrently enabled at M. Moreover, from the proof of Lemma 7.1.2 there is a marking $M_0' \in \mathcal{M}_{(.)}^{B}$ such that $M = M_0'(.f)_B(d)$ and to each local transition l_i a corresponding local transition $l_i' \in LT(B)$ with $l_i = l_i'(.f)_B$ such that l_i' is enabled at M_0'. From that proof, the l_i' are also concurrently enabled at M_0'.

   Suppose $N \in \mathcal{M}_d^{B'[f]}$ is such that $M \, \widehat{\mathcal{R}}_{[f]} \, N$. Then, from Lemma 7.1.2, we can find $N_0' \in \mathcal{M}_{(.)}^{B'}$ such that $N = N_0'(.f)_B(d)$. Moreover, as $\mathcal{R}$ is interface respecting then $M \, \widehat{\mathcal{R}}_{[f]} \, N$ implies $M_0' \, \widehat{\mathcal{R}} \, N_0'$.

   We may now use the assumed strong bisimilarity of B and B' to give local transitions $k_1', \ldots, k_n' \in LT(B')$ which are concurrently enabled at N_0' (by Proposition 7.2.6) and such that $l_i'(.f)_B \, \widehat{\mathcal{R}}_{[f]} \, k_i'(.f)_{B'}$ (as $\mathcal{R}_{[f]}$ is interface respecting). From the definition of relabelling, $t' = \{\!|k_1'(.f)_{B'}|\!\} \cup \cdots \cup \{\!|k_n'(.f)_{B'}|\!\} \in T_{B'[f]}$. Moreover, as the k_i' are concurrently enabled at N_0' then there is a marking N' such that $N[t':\alpha\rangle N'$. It is not difficult to check that $t \, \widehat{\mathcal{R}}_{[f]} \, t'$ and $M' \, \widehat{\mathcal{R}}_{[f]} \, N'$, whence the result in this case.

   The method of the proof is illustrated in Figure 7.2.

   The other direction is similar

2. $B_1 \| B_3 \approx B_2 \| B_4$: the relation $\mathcal{R}_{\|} = \mathcal{R}_{12} \cup \mathcal{R}_{34}$ is clearly an interface respecting relation between $B_1 \| B_3$ and $B_2 \| B_4$.

   That the initialisation property holds under $\mathcal{R}_{\|}$ is immediate from the definition.

$M, M' \in \mathcal{M}_d^{B[f]}$		M		$[t:\alpha\rangle$		M'	$\in \mathcal{M}_d^{B[f]}$ in $\mathcal{L}_{\mathcal{MP}}$
$M_0, \ldots, M_n \in \mathcal{M}_d^{B[f]}$		M_0	$[l_1\rangle$	$\cdots$	$[l_n\rangle$	M_n	
$M=M_0'(.f)_B(d)$ $M'=M_n'(.f)_B(d)$							Lem. 7.1.2
$M_0', \ldots, M_n' \in \mathcal{M}_{(.)}^B$	$\widehat{\mathcal{R}}_{[f]}$	M_0'	$[l_1'\rangle$	$\cdots$	$[l_n'\rangle$	M_n'	
		$\widehat{\mathcal{R}}$	$\widehat{\mathcal{R}}$	$\cdots$	$\widehat{\mathcal{R}}$	$\widehat{\mathcal{R}}$	as $B \approx B'$
$N_0', \ldots, N_n' \in \mathcal{M}_{(.)}^{B'}$		N_0'	$[k_1'\rangle$	$\cdots$	$[k_n'\rangle$	N_n'	
$N=N_0'(.f)_B(d)$							Defn. 4.6.8
$N \in \mathcal{M}_d^{B'[f]}$		$N\ [(\{\!\|k_1'(.f)_{B'}\|\!\} \cup \cdots \cup \{\!\|k_n'(.f)_{B'}\|\!\}):\alpha\rangle\ N'$					$\in \mathcal{M}_d^{B'[f]}$

Figure 7.2: Illustrating the proof that $B \approx B'$ implies $B[f] \approx B'[f]$.

That the transfer property holds under $\mathcal{R}_{||}$ follows from a simple application of Lemma 7.1.2, noting that $T_{B_1||B_3} = T_{B_1} \cup T_{B_3}$ and $T_{B_2||B_4} = T_{B_2} \cup T_{B_4}$.

3. $B_1;B_3 \approx B_2;B_4$: and,
4. $B_1+B_3 \approx B_2+B_4$: follow similarly, with underlying bisimulations[10] $\mathcal{R}_; = ;_1(\mathcal{R}_{12}) \cup ;_2(\mathcal{R}_{34})$, and $\mathcal{R}_+ = +_1(\mathcal{R}_{12}) \cup +_2(\mathcal{R}_{34})$, respectively.

■ 7.2.7

PROPOSITION 7.2.8 Let $B \approx B'$ be strongly bisimilar syntactically generated High Level Petri Boxes with underlying strong bisimulation $\mathcal{R}$. Then

1. $\Gamma(B) \approx \Gamma(B')$,
2. if, for all $t \in \mathcal{M}_{\mathcal{F}}(LT(B))$, $t' \in \mathcal{M}_{\mathcal{F}}(LT(B'))$, $t\ \widehat{\mathcal{R}}\ t'$ implies $t \in T_B \Leftrightarrow t \in T_{B'}$ then $\Omega_Y(B) \approx \Omega_Y(B')$.

□ 7.2.8

Proof: 1. $\Gamma(B) \approx \Gamma(B')$: we claim that $\Gamma(B) \approx \Gamma(B')$ with underlying bisimulation, $\mathcal{R}_\Gamma$. defined such that, if s is the accounting place added by the guarding of B and s' that added by the guarding of B', then $\mathcal{R}_\Gamma = ({}^\bullet\Gamma(B) \times {}^\bullet\Gamma(B')) \cup (\Gamma(B)^\bullet \times \Gamma(B')^\bullet) \cup \{(s, s')\} \cup \mathcal{R}$. That $\mathcal{R}_\Gamma$ is interface respecting is clear from its definition.

[10] Where $;_i(\mathcal{R}) = \bigcup\{;_i(r) \times ;_i(s) \mid (r, s) \in R\}$, with the individual $;_i$s being calculated in the obvious way. $+_i(R)$ is defined analogously.

The remainder of the proof is similar to that of Theorem 7.2.7.

2. $\Omega_Y(B) \approx \Omega_Y(B')$: we claim that $\Omega_Y(B) \approx \Omega_Y(B')$, with underlying bisimulation $\mathcal{R}_\Omega = \mathcal{R}\lfloor_{S_B \setminus {}^\bullet L_Y(B)^\bullet}$, when B and B' satisfy the extra hypotheses of the proposition. That $\mathcal{R}_\Omega$ is an interface respecting (and, in particular, proper) relation and that the initialisation property holds under $\mathcal{R}_\Omega$ follow from the same properties of $\mathcal{R}$, that ${}^\bullet L_Y(B)^\bullet \cap {}^\bullet B^\bullet = \emptyset$ and that ${}^\bullet L_Y(B')^\bullet \cap {}^\bullet B'^\bullet = \emptyset$ (as B and B' are Y-guarded).

For the transfer property, suppose $M \in \mathcal{M}_d^{\Omega_Y(B)}$ and $N \in \mathcal{M}_d^{\Omega_Y(B')}$ with $M\ \hat{\mathcal{R}}_\Omega\ N$. Then, from Lemma 7.1.2, we may write

$$M = \bigcup_{i=0}^{p} M_i C_i \lfloor_{S_B \setminus {}^\bullet L_Y(B)^\bullet}$$

and

$$N = \bigcup_{j=0}^{q} N_j D_j \lfloor_{S_{B'} \setminus {}^\bullet L_Y(B')^\bullet}$$

with the properties of that lemma holding of the $M_i C_i$ and $N_j D_j$.

We first show that $M\ \hat{\mathcal{R}}_\Omega\ N$ implies $p = q$, and that[11] $C_i = D_i$ and $M_i C_i\ \hat{\mathcal{R}}\ N_i D_i$ for each $i = 0, \ldots, p$.

That $M\lfloor_{Acc_Y(B)} \hat{\mathcal{R}}_\Omega\ N\lfloor_{Acc_Y(B')}$ follows directly from the fact that $\mathcal{R}_\Omega$ is proper. But then, as $M\lfloor_{Acc_Y(B)} = \{(s_1, C_1), \ldots, (s_p, C_p)\}$, $N\lfloor_{Acc_Y(B')} = \{(r_1, D_1), \ldots, (r_q, D_q)\}$ we have that $p = q$, and that[11] $s_i\ \mathcal{R}_\Omega\ r_i$ and $C_i = D_i$, $i = 1, \ldots, p$. Moreover, $C_0 = D_0 = d$. It follows (from P1 and Lemma 7.1.3(a)) that $M_i \prec M_j$ if and only if $N_i \prec N_j$, so that the respective trees are isomorphic.

We claim that for all $i \in \{0, \ldots, p\}$, $M_i C_i\ \hat{\mathcal{R}}\ N_i C_i$. To see this we will consider the sets $U = \bigcup_{i=0}^{p} M_i C_i$ and $V = \bigcup_{i=0}^{p} N_i C_i$, such that $M \subseteq U$ and $N \subseteq V$.

From the tree construction of Lemma 7.1.2 we have that

$$M\lfloor_{Acc_Y(B)} = \{(s_1, d_1), \ldots, (s_p, d_p)\}$$

and

$$N\lfloor_{Acc_Y(B')} = \{(r_1, d_1), \ldots, (r_p, d_p)\}$$

(with the d_i distinct), whence

$$U = M \cup \{(s'_1, d_1), \ldots, (s'_p, d_p)\}$$

and

$$V = N \cup \{(r'_1, d_1), \ldots, (r'_p, d_p)\}$$

[11] After reindexing if necessary.

where $\{s_i, s_i'\} = {}^\bullet l_r$ (the unique return transition in B corresponding to s_i) and $\{r_i, r_i'\} = {}^\bullet l_r'$ (the unique return transition in B' corresponding to r_i). We note that $M \,\hat{\mathcal{R}}_\Omega\, N$ implies $M \,\hat{\mathcal{R}}\, N$, from the definition.

We will show that $U \,\hat{\mathcal{R}}\, V$, whence the claim follows from the interface respecting nature of $\mathcal{R}$.

Suppose, otherwise, that $\neg U \,\hat{\mathcal{R}}\, V$, i.e., $\exists g \in \mathcal{C}_\mathcal{L}$ such that $\neg(U \triangleright \{g\}) \,\hat{\mathcal{R}}\, (V \triangleright \{g\})$. As $M \,\hat{\mathcal{R}}\, N$, we may assume that $g \in \{d_1, \ldots, d_n\}$; let i be such that $g = d_i$.

Now, $\neg(U \triangleright \{d_i\}) \,\hat{\mathcal{R}}\, (V \triangleright \{d_i\})$ implies either that $\exists r \in N \triangleright \{d_i\}$ such that $s_i' \,\mathcal{R}\, r$, or $\exists s \in M \triangleright \{d_i\}$ such that $s \,\mathcal{R}\, r_i'$, contradicting that $\mathcal{R}$ is proper, or that $\neg s_i' \,\mathcal{R}\, r_i'$, contradicting Item 4 of Definition 7.2.2, as $s_i \,\mathcal{R}\, r_i$, by assumption. Hence $U \,\hat{\mathcal{R}}\, V$, and we have the claim.

As the markings $M_i C_i$ and $N_i D_i$ of B and B' are related under $\mathcal{R}$, we can use the transfer property between them to show that the transfer property holds between $\Omega_Y(B)$ and $\Omega_Y(B')$. So, suppose that $M, M' \in \mathcal{M}_d^{\Omega_Y(B)}$ and $t \in T_{\Omega_Y(B)}$ are such that $M[t{:}\alpha\rangle M'$ for some $\alpha \in \mathrm{subs}^{\Omega_Y(B)}(t)$, and that $N \in \mathcal{M}_d^{\Omega_Y(B')}$ with $M \,\hat{\mathcal{R}}_\Omega\, N$. We must show that there is a $t' \in T_{\Omega_Y(B')}$, $N' \in \mathcal{M}_d^{\Omega_Y(B')}$ such that $t \,\hat{\mathcal{R}}_\Omega\, t'$, $M' \,\hat{\mathcal{R}}_\Omega\, N'$, and $N[t'{:}\alpha\rangle N'$.

We first consider the case when t is an augmented local transition, $t = \{\!|l|\!\}$ say. Let $l' \in LT(B)$ be such that $l = f_\Omega(l')$. Then, from Lemma 7.1.5, there is a unique j such that $M_j C_j[l'\rangle M_j' C_j$. Moreover, from the above, $M_j C_j \,\hat{\mathcal{R}}\, N_j C_j$. Hence, as $B \approx B'$, there is a local transition $k' \in LT(B')$, and a marking $N_j' C_j \in \mathcal{M}_{C_j}^{B'}$, such that $l' \,\hat{\mathcal{R}}\, k'$, $N_j C_j[k'\rangle N_j' C_j$ and $M_j' C_j \,\hat{\mathcal{R}}\, N_j' C_j$.

Define $t' = f_\Omega(k') \in T_{B^\mu}$, whence $t \,\hat{\mathcal{R}}_\Omega\, t'$. Define N' such that $N[t'{:}\alpha\rangle N'$. That $M' \,\hat{\mathcal{R}}_\Omega\, N'$ follows easily.

To extend to the general case we will consider when $t = \{\!|l_1, \ldots, l_n|\!\}$ with $n > 1$. Then, as we work in the most permissive label algebra and as B is complete, M concurrently enables the l_i. Hence, as $B \approx B'$, we may find local transitions $k_i \in LT(B')$, with $l_i \,\hat{\mathcal{R}}\, k_i$, and which are, by Proposition 7.2.6, concurrently enabled at N.

Define $t' = \{\!|k_1, \ldots, k_n|\!\}$. Then $t \,\hat{\mathcal{R}}\, t'$ whence, from the hypotheses, $t' \in T_{B'}$. If $N' \in \mathcal{M}_d^{\Omega_Y(B')}$ is such that $N[t'{:}\alpha\rangle N'$, it is not difficult to check that $M' \,\hat{\mathcal{R}}\, N'$. Hence the result.

The proof of the transfer property in the other direction is similar. ■ 7.2.8

An important observation for the demonstration of the fix-point nature of the recursive construct is that $\mathcal{R}_\Gamma$, as constructed in the guarding case of the above proof, is interface respecting even if $\mathcal{R}$ is proper but not interface respecting. For the fix-point result we will construct a proper (but not, in general, interface respecting) relation which forms a pre-strong bisimulation between (able pre-High Level Petri Box) $\Omega_Y(B)$ and (High Level Petri Box) $\mu Y.B$ for a Y-guarded High Level Petri Box B. Through an immediate subsequent application of the guarding operator (as occurs in the recursive construct) we will then have that $\Gamma(\Omega_Y(B)) \approx \Gamma(\mu Y.B)$.

COROLLARY 7.2.9 For B and B' as in Item 2 of Proposition 7.2.8, and for D a syntactically generated High Level Petri Box, $D[Y \leftarrow B] \approx D[Y \leftarrow B']$. □ 7.2.9

Proof: Suppose $D = HLPB(t)$. From the syntactic nature of refinement there is a (possibly auxiliary) High Level Petri Box term t' such that $D = HLPB(t')$ and which contains no applications of refinement (but in which $\Omega_Z(\_)$ and $\Gamma(\_)$ might appear). Using structural induction on t' rather than t, we may use Theorem 7.2.7 and Proposition 7.2.8 for the result. ■ 7.2.9

As an example of the relationship between t and t' in the proof of Corollary 7.2.9, consider the case when $t = \mu Z.((a\|(Z+Y))[Z \leftarrow \Gamma(Z)])$. Then $t' = a\|(\Gamma(\Omega_Z(a\|(\Gamma(Z)+Y)))+Y)$, whence we may apply structural induction to t'.

We discuss the requirements for a full congruence with respect to refinement and recursion in the Discussion at the end of this chapter.

7.3 The Fix-Point Relationship between Refinement and Recursion

From the definitions, the reader will understand that $\Omega_Y(B)$ and $\mu Y.B$ may have vastly differing (but finite) numbers of places (according to the number of times that the subject process variable Y of the refinement appears free in B). We observe that there will, in general, be no structural equivalence between $\mu Y.B$ and $B[Y \leftarrow \mu Y.B]$ to extend to a behavioural equivalence (unlike Chapter 5). This has motivated the development of the strong bisimulation in the previous section; in this section we will use the congruence properties of strong bisimulation to show that $\mu Y.B \approx B[Y \leftarrow \mu Y.B]$.

The demonstration of the relationship is, as we shall see, technically involved. However, it is also intuitively very simple and is based on the following observation of the relationship between the structures of $\Omega_Y(B)$ and $\mu Y.B$:

OBSERVATION 7.3.1 *[Naming Convention]* As we are concerned with characterising the structural relationship between $\Omega_Y(B)$ and $\mu Y.B$, we will use a convention for the naming of places of the High Level Petri Boxes, which is based on the following observation: that for B a Y-guarded High Level Petri Box with $L_Y(B) = \{l_1, \ldots, l_n\}$, $n \geq 0$, then $\mu Y.B$ consists, essentially, of one copy of B together with n copies of $\Omega_Y(B)$. Fixing an (arbitrary) ordering, $l^1, \ldots, l^n$, of the set $L_Y(B)$, and of the accounting places related to them, $s^1, \ldots, s^n$, we may:

1. annotate the places of $\mu Y.B$ which are also in B by ^{0},
2. annotate the places of $\mu Y.B$ which are also in the copy of $\Omega_Y(B)$ which replaces l^i, with^i.

■ 7.3.1

To illustrate the ideas underlying the proof we refer the reader to Figure 7.3 in which is shown a Y-guarded, syntactically generated High Level Petri Box B (the Y-guarding of the denotation of the High Level Petri Box term $t = (a;Y;c)|||(b;Y;d))$ together with $\mu Y.B$ and $\Omega_Y(B)$. The places have been decorated under the naming convention of Observation 7.3.1. With this example we will show that the existence of the strong bisimulation is crucially dependent upon the characterisation of reachable markings of $\Omega_Y(B)$ provided by Lemma 7.1.2, and so is further justification of the particular choices which have been made in the development of the High Level Petri Box model.

As can be seen in the figure, to each local transition[12] of $\Omega_Y(B)$ there corresponds three local transitions in $\mu Y.B$, one corresponding to each of the two occurrences of the process variable Y in B, and one for the original corresponding local transition of B.

As an example, we will consider behaviours from the respective standard initial markings[13] $M_{(.)}^{I,\Omega_Y(B)}$ and $M_{(.)}^{I,\mu Y.B}$. Fire (in any order possible) the abstract transitions labelled a, μ_c, b, μ_c of $\Omega_Y(B)$ and $\mu Y.B$, respectively, so as to have reached the marking

1. $M \in \mathcal{M}_{(.)}^{\Omega_Y(B)}$, where $M(1) = M(2) = \{.\bot\vdash, .\bot\dashv\}$, $M(7) = \{.\bot\vdash\}$, $M(8) = \{.\bot\dashv\}$ (with all other places empty), and

[12] We have chosen a High Level Petri Box in which no relabelling has taken place, so that abstract transitions correspond to local transitions. This simplifies the illustration somewhat, although the ideas of the illustration are entirely general.

[13] Which, for clarity, do not appear in the figure. A prepared reader will already have equipped themselves with 'token-able' material for the token game which follows.

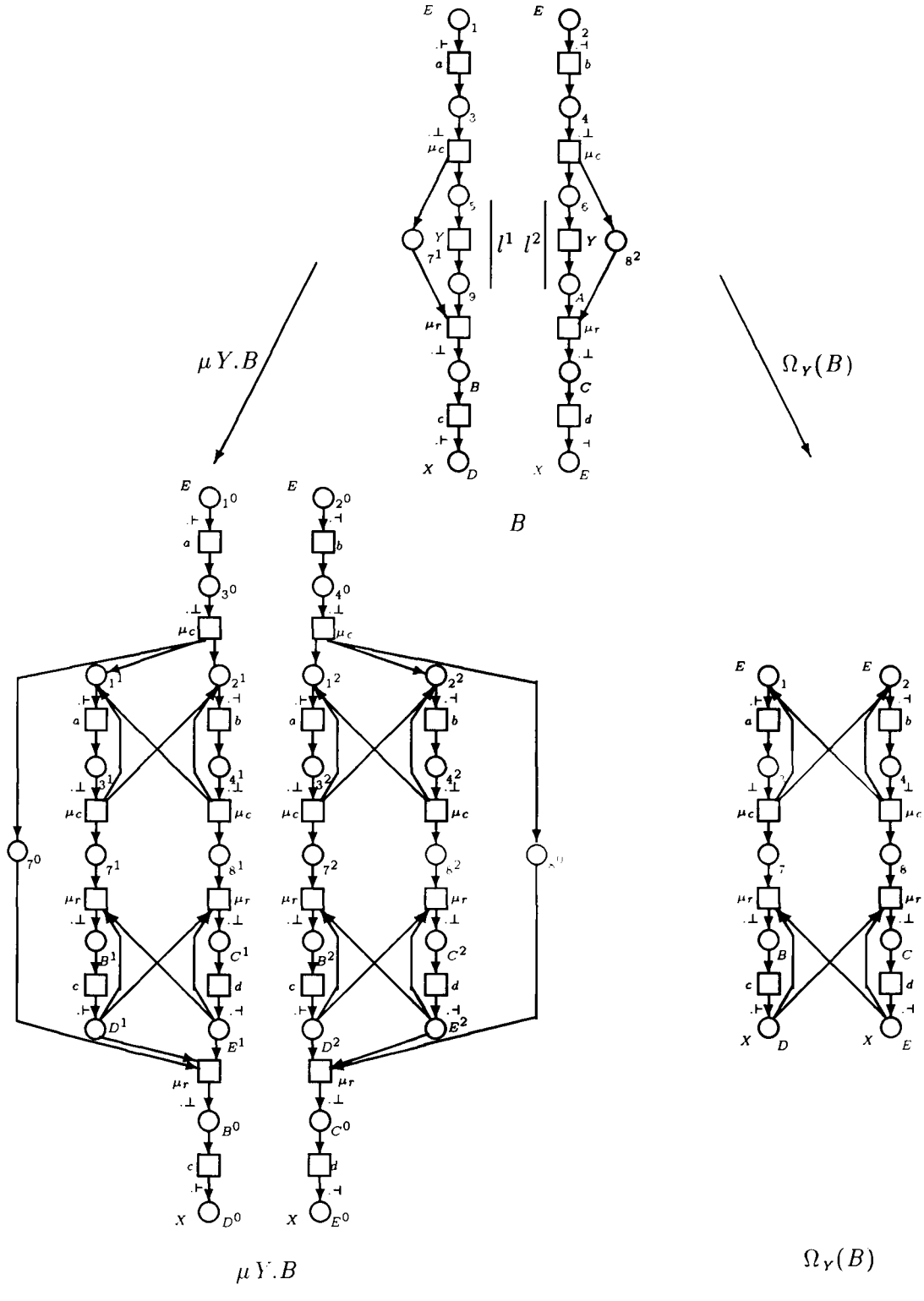


Figure 7.3: The denotation of the High Level Petri Box term $(a;Y;c)\|(b;Y;d)[Y \leftarrow \Gamma(Y)]$, with the recursive and winding operators applied. Places are annotated according to the naming convention of Observation 7.3.1.

2. $M' \in \mathcal{M}_{(.)}^{\mu Y.B}$ such that $M'(1^{(1)}) = M'(2^{(1)}) = \{.\bot\vdash\}$, $M'(1^{(2)}) = M'(2^{(2)}) = \{.\bot\dashv\}$, $M'(7^{(0)}) = \{.\bot\vdash\}$ and $M'(8^{(0)}) = \{.\bot\dashv\}$ (with all other places empty).

The reader will note that for all $s \in S_{\Omega_Y(B)}$, $\bigcup_{i\in\{0,1,2\}} M'(s^{(i)}) = M(s)$ and will quickly see that the obvious generalisation of this property is, in fact, an invariant of the symbolic firing rule. It provides a many-one mapping between reachable markings of $\mathcal{M}_d^{\mu Y.B}$ and $\mathcal{M}_d^{\Omega_Y(B)}$. We wish to show that a bijection exists so that we should provide an inverse mapping to this many-one mapping. The extra information by which this resolution is possible is contained in the markings of the accounting places—as can be seen from the figure, the firing in $\Omega_Y(B)$ of any particular μ_c-transition at a marking reachable from a standard initial marking simultaneously places a token in the initial places of $\Omega_Y(B)$ and in its associated accounting place. The token in that accounting place, by Corollary 7.1.8, forms a postfix of all subsequent markings of the recursive call; it is this token's postfixing of reachable markings of $\mu Y.B$ from which we may determine the copy of $\Omega_Y(B)$ in $\mu Y.B$ in which a token should appear.

For instance, by examining the accounting places (7 and 8) we see that all tokens with postfix $.\bot\vdash$ were generated by the firing of the left-hand side μ_c-transition, i.e., that guarding Y-local l^1 of B, and hence should mark the corresponding places, i.e., those decorated with $^{(1)}$. Similarly, tokens with postfix $.\bot\dashv$ were generated by the firing of the right-hand side μ_c-transition, i.e., that guarding Y-local l^2 of B, and should mark places decorated by $^{(2)}$. A similar device may be used to determine the occupancy of accounting places of $\mu Y.B$. Tokens which are not postfixed by a token in an accounting place appear in places decorated by $^{(0)}$.

By firing the local transitions of $\mu Y.B$ corresponding to those fired in $\Omega_Y(B)$, the reader will rapidly be convinced that this produces the correct correspondence between markings.

7.3.1 The Nature of the Behavioural Relationship Between $\Omega_Y(B)$ and $\mu Y.B$

We will show, in this section, that the naming convention of Observation 7.3.1 is a proper relation which satisfies the weaker precondition of the case for guarding in the proof of Theorem 7.2.7 (as was noted after Proposition 7.2.8).

Throughout this section we will assume that B is a Y-guarded, syntactically generated High Level Petri Box over a maximally permissive $\mathcal{PV}$-extended label algebra $\mathcal{L}$. Moreover, we will assume that B, $\Omega_Y(B)$ and $\mu Y.B$ have had their places decorated as given by Observation 7.3.1 and that $L_Y(B) = \{l^1, \ldots, l^n\}$, $n \geq 0$ (with corresponding accounting places $\{s^1, \ldots, s^n\}$). For notational convenience, we will denote $\Omega_Y(B)$ by B^Ω and $\mu Y.B$ by B^μ.

7.3.1.1 Formalising the Naming Convention

The naming convention is formalised as a proper (but not necessarily interface respecting) relation $\mathcal{K} \subseteq S_{B^\Omega} \times S_{B^\mu}$ defined such that $\mathcal{K} = \{(r, r^{(i)}) \mid r \in S_{B^\Omega} \wedge i = 0, \ldots, n\}$. Note that $\mathcal{K}$ is interface respecting if and only if $n = 0$, i.e., there were no Y-locals in B. As $\mathcal{K} \subseteq S_{B^\Omega} \times S_{B^\mu}$, using Definition 7.2.1, $\mathcal{K}$ lifts to a relation, $\widehat{\mathcal{K}}$, between the components of B^Ω and B^μ, such that:

1. for $U \subseteq S_{B^\Omega}$, $V \subseteq S_{B^\mu}$, $U \;\widehat{\mathcal{K}}\; V$ implies $U = \{s_1, \ldots, s_m\}$ and $V = \{s_1^{(j_1)}, \ldots, s_m^{(j_m)}\}$, where $j_i \in \{0, \ldots, n\}$ for $i = 1, \ldots, m$.

2. for $h \in LT(B^\mu)$, if:

   (a) h is in the 'outer' copy of B, then either:

      i. ${}^\bullet h^\bullet \cap Acc_Y(B) = \emptyset$ whence there is a unique $l' \in LT(B)$ such that $h = l'$. For $l \in LT(\Omega_Y(B))$ the unique local transition such that $l = f_\Omega(l')$ we will write $h = l^{(0)}$ in this case,

      ii. ${}^\bullet h^\bullet \cap Acc_Y(B) = \{s^i\}$ whence there is a unique $l' \in LT(B)$ such that[14] $h = f_1^{l'}(l')$. For $l \in LT(\Omega_Y(B))$ the unique local transition such that $l = f_\Omega(l')$, we will write $h = l^{(0)}$ in this case.

   (b) h is in the 'inner' copy of $\Omega_Y(B)$ which replaces Y-local $l^i \in L_Y(B)$ whence there is a unique $l' \in LT(B)$ such that $h = f_2^{l'}(f_\Omega(l'))$. For $l \in LT(\Omega_Y(B))$ the unique local transition such that $l = f_\Omega(l')$, we will write $h = l^{(i)}$ in this case.

Given this naming of local transitions, we conclude that, for $l \in LT(B^\Omega)$, $h \in LT(B^\mu)$, $l \;\widehat{\mathcal{K}}\; h$ if and only if $h \in \{l^{(i)} \mid i \in \{0, \ldots, n\}\}$.

Moreover, when $l \;\widehat{\mathcal{K}}\; h$, then ${}^C l = {}^C h$, $\overline{l} = \overline{h}$, $l^C = h^C$, and:

(a) if $h = l^{(0)}$, then

$$ {}^\bullet l^{(0)} = \begin{cases} \{s^{(0)} \mid s \in {}^\bullet l\} & {}^\bullet l' \cap Acc_Y(B) = \emptyset \\ \{s^{i(0)}\} \cup \{s^{(i)} \mid s \in B^{\Omega\bullet}\} & {}^\bullet l' \cap Acc_Y(B) = \{s^i\}; \end{cases} $$

and

$$ l^{(0)\bullet} = \begin{cases} \{s^{(0)} \mid s \in l^\bullet\} & l'^\bullet \cap Acc_Y(B) = \emptyset \\ \{s^{i(0)}\} \cup \{s^{(i)} \mid s \in {}^\bullet B^\Omega\} & l'^\bullet \cap Acc_Y(B) = \{s^i\}; \end{cases} $$

[14] Where f_i^l, $i = 1, 2$, are the auxiliary relations defined for the local transition refinement of a High Level Petri Box, and f_Ω is the auxiliary relation defined for the winding auxiliary.

(b) if $h = l^{(i)}$, $i > 0$, then

$$^\bullet l^{(i)} = \{s^{(i)} \mid s \in {}^\bullet l\}$$

and

$$l^{(i)\bullet} = \{s^{(i)} \mid s \in l^\bullet\}$$

3. for $t_1 \in \mathcal{M}_{\mathcal{F}}(LT(B^\Omega))$ and $t_2 \in \mathcal{M}_{\mathcal{F}}(LT(B^\mu))$, $t_1 \; \widehat{\mathcal{K}} \; t_2$ and $t_1 = \{\!|l_1, \ldots, l_m|\!\}$ implies $t_2 = \{\!|l_1^{(j_1)}, \ldots, l_m^{(j_m)}|\!\}$, with $j_i \in \{0, \ldots, n\}$, $i = 1, \ldots, m$.

4. for $M \in \mathbb{P}(S_{B^\Omega} \times \mathcal{C}_{\mathcal{L}})$ and $N \in \mathbb{P}(S_{B^\mu} \times \mathcal{C}_{\mathcal{L}})$ (both finite markings), $M \; \widehat{\mathcal{K}} \; N$ and $M = \{(s_1, d_1), \ldots, (s_p, d_p)\}$ implies $N = \{(s_1^{(j_1)}, d_1), \ldots, (s_p^{(j_p)}, d_p)\}$, for $j_i \in \{0, \ldots, n\}$, $i = 1, \ldots, p$.

We claim that $\widehat{\mathcal{K}} \cap (\mathcal{M}_d^{B^\Omega} \times \mathcal{M}_d^{B^\mu})$ is a bijection between $\mathcal{M}_d^{B^\Omega}$ and $\mathcal{M}_d^{B^\mu}$. To see this we must look more closely at the 'switch tokens' which appear in the accounting places of $\Omega_Y(B)$. As we saw in the above illustration, these 'switches' consisted of the first tokens to arrive in the accounting place. These are the tokens contained in M_{min}:

DEFINITION 7.3.2 Let $M \in \mathcal{M}_d^{B^\Omega}$. From Lemma 7.1.2 there is a unique decomposition $M = \bigcup_i M_i C_i \lfloor_{S_B \setminus {}^\bullet L_Y(B)^\bullet}$, with the properties of that lemma holding. Define

$$M_{min} = M_0 C_0 \lfloor_{Acc_Y(B)}$$

■ 7.3.2

Then:

LEMMA 7.3.3 For all $M \in \mathcal{M}_d^{B^\Omega}$:

1. For all $i \neq j$, if $\{(s_i, C_i), (s_j, C_j)\} \subseteq M_{min}$, then $C_i \# C_j$,

2. For each $i > 0$ there is a unique j with $(s_j, C_j) \in M_{min}$ and $C_j \sqsubseteq C_i$. □ 7.3.3

Proof: 1. We have that $M_0 \prec M_i$ and $M_0 \prec M_j$ and $M_i \# M_j$ in this case, whence P1 of Lemma 7.1.2 gives the result.

2. That there is at least one such j is clear. To show there is at most one, suppose there is $C_k \neq C_j$ such that $(s_k, C_k) \in M_{min}$ with $C_k \sqsubseteq C_i$. By assumption, $C_j \sqsubseteq C_i$.

so that $C_k \sqsubset C_j$ or $C_j \sqsubset C_k$. Assuming the former (the other case being similar) we have that $M_k \prec^+ M_j$, whence $(s_j, C_j) \notin M_{min}$, a contradiction.

Hence the result. ■ 7.3.3

In the sequel for $M \in \mathcal{M}_d^{B^\Omega}$, if $(s,g) \in M_0 C_0$, we will write $\delta_M(s,g) = 0$; if $(s,g) \in M_i C_i$ we will write $\delta_M(s,g) = k$, where s_j (as given by Item 2 of Lemma 7.3.3) is s^k (the accounting place associated with $l^k \in Acc_Y(B)$). Clearly, from 7.1.11, $\delta_M(s,g)$ is well-defined. Moreover, if $|L_Y(B)| = n$ then $\delta_M(s,g) \in \{0,\ldots,n\}$.

The characterisation of the naming relation of reachable markings is approached through $\widehat{\mathcal{J}}$, which will turn out to be, simultaneously, a bijection, and $\widehat{\mathcal{K}} \cap (\mathcal{M}_d^{B^\Omega} \times \mathcal{M}_d^{B^\mu})$.

DEFINITION 7.3.4 Define a relation $\widehat{\mathcal{J}} \subseteq \mathcal{M}_d^{B^\Omega} \times \mathbb{P}(S_{B^\mu} \times C_{\mathcal{L}})$ such that $M_1 \;\widehat{\mathcal{J}}\; M_2$ when $M_1 = \{\!|(s_1,d_1),\ldots,(s_n,d_n)|\!\}$ and $M_2 = \{\!|(s_1^{(j_1)},d_1),\ldots,(s_n^{(j_n)},d_n)|\!\}$ with $j_i = \delta_{M_1}(s_i,d_i)$.

■ 7.3.4

Then:

PROPOSITION 7.3.5

1. $\widehat{\mathcal{J}}: \mathcal{M}_d^{B^\Omega} \equiv \mathcal{M}_d^{B^\mu}$;
2. (a): $\widehat{\mathcal{J}}(M_d^{I,B^\Omega}) = M_d^{I,B^\mu}$ and (b): $\widehat{\mathcal{J}}(M_d^{T,B^\Omega}) = M_d^{T,B^\mu}$;
3. $\widehat{\mathcal{J}} = \widehat{\mathcal{K}} \cap (\mathcal{M}_d^{B^\Omega} \times \mathcal{M}_d^{B^\mu})$. □ 7.3.5

Proof: 1. That $\widehat{\mathcal{J}} \subseteq \mathcal{M}_d^{B^\Omega} \times \mathbb{P}(S_{B^\mu} \to C_{\mathcal{L}})$ is a total, injective function follows from Lemma 7.3.3. As $\widehat{\mathcal{J}}$ is functional, we will write $\widehat{\mathcal{J}}(M)$ for the unique N such that $M \;\widehat{\mathcal{J}}\; N$.

As $\widehat{\mathcal{J}}$ is injective, it is bijective onto its range. For the result, then, we must show only that $\mathrm{ran}(\widehat{\mathcal{J}}) = \mathcal{M}_d^{B^\mu}$.

$\supseteq$: we show that, for each $N \in \mathcal{M}_d^{B^\mu}$, there is a marking $M \in \mathcal{M}_d^{B^\Omega}$ such that $N = \widehat{\mathcal{J}}(M)$. The proof proceeds by induction on the length of the local transition sequence

$$M_d^{I,B^\mu} = N_0[l_1\rangle \cdots N_{n-1}[l_n\rangle N_n = N \in \mathcal{M}_d^{B^\mu}$$

which led to N.

Base Case: $n = 0$, whence $N = M_d^{I,B^\mu}$. Choose $M = M_d^{I,B^\Omega}$: the result follows from the definitions.

Inductive Step: Let $n > 0$ and suppose the result holds for $n-1$, i.e., there exists $M_{n-1} \in \mathcal{M}_d^{B^\Omega}$ such that $N_{n-1} = \hat{\mathcal{J}}(M_{n-1})$. As $N_{n-1}[l_n\rangle N_n$, there are contexts e and f such that $f = {}^C l_n e - l_n{}^C$ and

$$N_n = (N_{n-1} \setminus {}^\bullet l_n \times \{e\}) \cup l_n{}^\bullet \times \{f\}$$

Let $l \in LT(B^\Omega)$ and $i \in \{0, \ldots, n\}$ be such that $l_n = l^{(i)}$, and let $l' \in LT(B)$ be such that $l = f_\Omega(l')$. From the definition of $\hat{\mathcal{J}}$, that $N_{n-1} = \{(s_1^{(j_1)}, d_1), \ldots, (s_m^{(j_m)}, d_m)\}$ implies $M_{n-1} = \{(s_1, d_1), \ldots, (s_m, d_m)\}$. Moreover, as ${}^C l = {}^C l_n$ and $l^C = l_n{}^C$, if

$$M' = (M_{n-1} \setminus {}^\bullet l \times \{e\}) \cup l^\bullet \times \{f\}$$

then $M_{n-1}[l\rangle M' \in \mathcal{M}_d^\Omega$.

We show that $N = \hat{\mathcal{J}}(M')$, whence we may set $M = M'$ for the result.

There are six cases to consider corresponding as to whether l is a call or a return transition, or neither, and as to whether $i = 0$ or $i > 0$. We give the proof for one case only, that when l is a call transition and $i = 0$, the others being broadly similar.

As l is a call transition, we may assume that $l^\bullet \cap Acc_Y(B) = \{s_w\}$ with s_w the accounting place corresponding to $l^k \in L_Y(B)$, and that $f = C_w$ as defined in Lemma 7.1.2. As $i = 0$, ${}^\bullet l_n = {}^\bullet l'^{(0)} = \{s^{(0)} \mid s \in {}^\bullet l'\}$ and $l_n{}^\bullet = l'^{(0)\bullet} = \{s_w^{(0)}\} \cup \{s^{(k)} \mid s \in {}^\bullet B^\Omega\}$.

From the proof of Lemma 7.1.2 and the definition of $\delta_M(s,g)$, it follows that

(a) for all $(s,g) \in M_{n-1} \cap M'$, $\delta_{M'_{n-1}}(s,g) = \delta_{M'}(s,g)$,

(b) ${}^\bullet l_n = \{s^{(m)} \mid s \in {}^\bullet l \wedge m = \delta_{M'_{n-1}}(s,e)\}$,

(c) $l_n{}^\bullet = \{s^{(m)} \mid s \in l^\bullet \wedge m = \delta_{M'}(s,f)\}$

whence

$$\begin{aligned} \hat{\mathcal{J}}(M') &= \hat{\mathcal{J}}((M_{n-1} \setminus {}^\bullet l \times \{e\}) \cup l^\bullet \times \{f\}) \\ &= (\hat{\mathcal{J}}(M_{n-1}) \setminus {}^\bullet l_n \times \{e\}) \cup l_n{}^\bullet \times \{f\} \\ &= N \end{aligned}$$

as required.

$\subseteq$: we show that, for each $\overline{M} \in \mathcal{M}_d^{B^\Omega}$, $\hat{\mathcal{J}}(\overline{M}) \in \mathcal{M}_d^{B^\mu}$. As for the previous case, the proof proceeds by induction on the length of the local transition sequence

$$M_d^{I,B^\Omega} = \overline{M}_0[l_1\rangle \cdots \overline{M}_{n-1}[l_n\rangle \overline{M}_n = \overline{M} \in \mathcal{M}_d^{B^\Omega}$$

which led to $\overline{M}$.

Base Case: $n = 0$, immediate, as $\widehat{\mathcal{J}}(M_d^{I,B^\Omega}) = M_d^{I,B^\mu}$.

Inductive Step: Let $n > 0$, and suppose the result holds for $n-1$, i.e., $\widehat{\mathcal{J}}(\overline{M}_{n-1}) \in M_d^{B^\mu}$. As $\overline{M}_{n-1}[l_n\rangle\overline{M}_n$, there are contexts ϵ and f such that $f = {}^C l_n e - l_n{}^C$ and

$$\overline{M}_n = (\overline{M}_{n-1} \setminus {}^\bullet l_n \times \{e\}) \cup l_n{}^\bullet \times \{f\}$$

We claim that $\widehat{\mathcal{J}}(\overline{M}_{n-1})[l_n^{(j)}\rangle\widehat{\mathcal{J}}(\overline{M}_n)$, where $j = \min\{\delta_{\overline{M}_{n-1}}(s,\epsilon) \mid s \in {}^\bullet l_n\}$ (the min appearing for return transitions in the outer copy of $\Omega_Y(B)$ in $\mu Y.B$; these transitions have presets which include a $^{(0)}$ decorated accounting place). The result follows by a purely technical manipulation, for which we present only the case when l_n is a return transition on Y, which is typical (and is the source of the min). From Lemma 7.1.2, we may write $\overline{M}_{n-1} = \bigcup M_i C_i \lfloor_{S_B \setminus {}^\bullet L_{Y(B)}{}^\bullet}$. As $\overline{M}_{n-1}[l_n\rangle\overline{M}_n$, by Lemma 7.1.5, there is a unique j such that $M_j C_j[l'\rangle M_j' C_j$ where $l_n = f_\Omega(l')$. Whence, for $s \in {}^\bullet l'$, $(s,e) \in M_j C_j$. Suppose ${}^\bullet l_n \cap Acc_Y(B) = \{s^k\}$ (the accounting place associated with $l^k \in L_Y(B)$). We distinguish two cases:

(a) $j = 0$: so that $\delta_{\overline{M}_{n-1}}(s^k, e) = 0$ and for $s \in B^\bullet$, $\delta_{\overline{M}_{n-1}}(s,e) = k$ whence $\min\{\delta_{\overline{M}_{n-1}}(s,e) \mid s \in {}^\bullet l_n\} = 0$ and ${}^\bullet l_n^{(0)} \times \{e\} \subseteq \widehat{\mathcal{J}}(\overline{M}_{n-1})$,

(b) $j > 0$: whence $\delta_{\overline{M}_{n-1}}(s,e) = k$, $\min\{\delta_{\overline{M}_{n-1}}(s,e) \mid s \in {}^\bullet l_n\} = k$ and ${}^\bullet l_n^{(k)} \times \{e\} \subseteq \widehat{\mathcal{J}}(\overline{M}_{n-1})$.

In either case, there is an $M' \in M_d^{B^\mu}$ such that $\widehat{\mathcal{J}}(\overline{M}_{n-1})[l_n^{(j)}\rangle M'$, with $j = \min\{\delta_{\overline{M}_{n-1}}(s,e) \mid s \in {}^\bullet l_n\}$. That $M' = \widehat{\mathcal{J}}(M_n)$ follows from a purely technical manipulation.

Whence the result.

2. (a) follows from the proof for Item 1. (b) follows from the definitions and the proof of Lemma 7.1.2.

3. Denote $\widehat{\mathcal{K}} \cap (\mathcal{M}_d^{B^\Omega} \times \mathcal{M}_d^{B^\mu})$ by $\widehat{\mathcal{U}}$. Then:

   $\subseteq$: that $\widehat{\mathcal{J}} \subseteq \widehat{\mathcal{U}}$ is clear.

   $\supseteq$: as $\widehat{\mathcal{J}}$ is a bijection, if $\widehat{\mathcal{J}} \neq \widehat{\mathcal{U}}$ and $N\ \widehat{\mathcal{J}}\ M$, then either:

   (a) there is an $N' \in \mathcal{M}_d^{B^\Omega}$ such that $N' \neq N$ with $N'\ \widehat{\mathcal{U}}\ M$. But $N'\ \widehat{\mathcal{U}}\ M$ and $N\ \widehat{\mathcal{U}}\ M$ implies $N' = N$, a contradiction, or

   (b) there is an $M' \neq M \in \mathcal{M}_d^{B^\mu}$ such that $N\ \widehat{\mathcal{U}}\ M'$. As $\widehat{\mathcal{J}}$ is a bijection there is an $N' \in \mathcal{M}_d^{B^\Omega}$ such that $N' \neq N$ but $N'\ \widehat{\mathcal{U}}\ M'$. But, again, $N\ \widehat{\mathcal{U}}\ M'$ and

$N' \widehat{\mathcal{U}} M'$ implies $N = N'$, a contradiction.

■ 7.3.5

As we have seen, if $|L_Y(B)| > 0$ then $\mathcal{K}$ is not interface respecting. However, $\mathcal{K}$ is proper and satisfies the following:

COROLLARY 7.3.6 For $\mathcal{K} \subseteq S_{B^\Omega} \times S_{B^\mu}$ as defined above, and for each $d \in \mathcal{C}_\mathcal{L}$:

1. $M_d^{I,B^\Omega} \widehat{\mathcal{K}} M_d^{I,B^\mu}$,
2. for all markings M, $M' \in \mathcal{M}_d^{B^\Omega}$, $N \in \mathcal{M}_d^{B^\mu}$, $t_1 \in T^\Omega$ and $\alpha \in \text{subs}^{B^\Omega}(t_1)$, whenever $M \widehat{\mathcal{K}} N$, and $M[t_1{:}\alpha\rangle M'$ there is a $t_2 \in T^\mu$ with $t_1 \widehat{\mathcal{K}} t_2$ and a marking $N' \in \mathcal{M}_d^{B^\mu}$ such that $N[t_2{:}\alpha\rangle N'$ and $M' \widehat{\mathcal{K}} N'$. Moreover, if $M' = M_d^{T,B^\Omega}$ then $N' = M_d^{T,B^\mu}$ also,
3. for all markings M, $M' \in \mathcal{M}_d^{B^\mu}$, $N \in \mathcal{M}_d^{B^\Omega}$, $t_1 \in T^\mu$ and $\alpha \in \text{subs}^{B^\mu}(t_1)$, whenever $M \widehat{\mathcal{K}} N$, and $M[t_1{:}\alpha\rangle M'$ there is a $t_2 \in T^\Omega$ with $t_1 \widehat{\mathcal{K}} t_2$ and a marking $N' \in \mathcal{M}_d^{B^\Omega}$ such that $N[t_2{:}\alpha\rangle N'$ and $M' \widehat{\mathcal{K}} N'$. Moreover, if $M' = M_d^{T,B^\mu}$ then $N' = M_d^{T,B^\Omega}$ also.

□ 7.3.6

Proof: Follows from the proof of Lemma 7.3.5, given that $\widehat{\mathcal{K}} \cap (\mathcal{M}_d^{B^\Omega} \times \mathcal{M}_d^{B^\mu})$ is a bijection between $\mathcal{M}_d^{B^\Omega}$ and $\mathcal{M}_d^{B^\mu}$. ■ 7.3.6

$\mathcal{K}$ is, thus, a pre-d-strong bisimulation between $\Omega_Y(B)$ and $\mu Y.B$ for each $d \in \mathcal{C}_\mathcal{L}$.

The only other result we need is that $\Omega_Y(B)$ and $\mu Y.B$ (as B and B', respectively) satisfy the hypotheses of Item 2 of Proposition 7.2.8, :

LEMMA 7.3.7 Let $t \in \mathcal{M}_\mathcal{F}(LT(B^\Omega))$, $t' \in \mathcal{M}_\mathcal{F}(LT(B^\mu))$ with $t \widehat{\mathcal{R}} t'$. Then $t \in T_{B^\Omega}$ if and only if $t' \in T_{B^\mu}$. □ 7.3.7

Proof: Follows easily from the construction. ■ 7.3.7

Whence we have:

THEOREM 7.3.8 Let B be a Y-guarded syntactically generated High Level Petri Box. Then $\mu Y.B \approx B[Y \leftarrow \mu Y.B]$. □ 7.3.8

Proof: From Corollary 7.3.6, $\mathcal{K}$ satisfies the weaker conditions for $\mathcal{K}_\Gamma$, as defined for the guarding case of the proof of Proposition 7.2.8, to be an interface preserving strong bisimilarity between $\Gamma(\Omega_Y(B))$ and $\Gamma(\mu Y.B)$. As B is Y-guarded there is a B' such that $B = B'[Y \leftarrow \Gamma(Y)]$, $\mu Y.B = B'[Y \leftarrow \Gamma(\Omega_Y(B))]$ and $B[Y \leftarrow \mu Y.B] = B'[Y \leftarrow \Gamma(\mu Y.B)]$, and the result follows from the syntactic nature of refinement and the congruence nature of strong bisimulation (Theorem 7.2.7, Proposition 7.2.8 and Corollary 7.2.9). ■ 7.3.8

7.4 Discussion

7.4.1 Strong Bisimulation of High Level Petri Boxes

We have demonstrated the fix-point nature of the relationship between the denotations of $\mu Y.t$ and $t[Y \leftarrow \mu Y.t]$. Although there was no simple structural equivalence between the High Level Petri Boxes which formed the respective denotations, the result went through due to the ability to define an operational behavioural equivalence on the High Level Petri Boxes in question.

As was mentioned, this operational definition is based on the concurrency preserving strong bisimulation of Olderog, [Old91]. Olderog's definition of strong bisimulation is stronger than those of [Pom85, vGV87], for instance, in that it uses structure (i.e., places) rather than behaviour (i.e., marking) for the basis of the state-relationship. From the proof of Theorem 7.2.7 (and, in particular, the case for relabelling), we see that this feature of the strong bisimulation is necessary for the extension of strong bisimulation to a congruence; marking based notions of strong bisimulation would not, in general, suffice.

As an example, compare the terms $t_1 = a||\overline{a}$, and its 'interleaved' version $t_2 = (a;\overline{a})+(\overline{a};a)$. We will assume that we have a marking based notion of strong bisimilarity which identifies these terms. Consider now relabelled versions $t_1[ccs]$ and $t_2[ccs]$ (with ccs the CCS-style relabelling of Example 2.1.1, page 8) which are clearly not strongly bisimilar: t_1 can perform a silent action stemming from the synchronisation of a and $\overline{a}$ whereas t_2 cannot.

7.4.2 Strong Bisimulation as a Full Congruence

The congruence nature of strong bisimulation as given by Proposition 7.2.8 and Corollary 7.2.9 is partial in that: for winding (and hence recursion), we require that High Level Petri Boxes have the same 'synchronisation structure' (i.e., abstract transitions in the High Level Petri Boxes must correspond); for refinement the congruence holds only with respect only to the second operand.

That full congruence in the case of refinement does not follow from its syntactic nature can be seen from considering syntactically generated High Level Petri Boxes B, $B_1 = HLPB(t_1)$ and $B_2 = HLPB(t_2)$ with $B_1 \approx B_2$. To use the syntactic nature of B_1 and B_2 to show that $B_1[Y \leftarrow B] \approx B_2[Y \leftarrow B]$, would require $t_1 = t_2$ which, of course, is not the case in general (cf. Theorem 7.3.8). However, we conjecture that a more direct proof of the congruence with respect to the first operand could be developed from a characterisation of the behaviours of $B[Y \leftarrow B']$ in terms of those of B and B' (as was given by Lemma 7.1.2 for the other operators). This leads us to the conjecture that:

CONJECTURE 7.4.1 $B_1 \approx B_2$ and $B_3 \approx B_4$ implies $B_1[Y \leftarrow B_3] \approx B_2[Y \leftarrow B_4]$. □ 7.4.1

It would appear that strong bisimulation is a full congruence for winding when the strong bisimilar High Level Petri Boxes are syntactically generated. To see why this is we will consider the High Level Petri Boxes B and B' which appear in Figure 7.4: although $B \approx B'$, $\Omega_Y(B) \not\approx \Omega_Y(B)$. The non-congruence is a corollary of the fact that the abstract transition labelled $a \oplus b$ is dead at every reachable marking of B' (even in the most permissive label algebra) in the unwound High Level Petri Boxes, whereas the marking $\{1 \rightarrow \{.\bot \vdash f\}, 2 \rightarrow \{.\bot \vdash f\}, 3 \rightarrow \{.\dashv f\}\} \in \mathcal{M}_{(.)}^{\Omega_Y(B)} \cap \mathcal{M}_{(.)}^{\Omega_Y(B')}$ enables it in $\Omega_Y(B')$ whereas, in $\Omega_Y(B)$, there is no corresponding abstract transition to be enabled. The reader will note, however, that B is not syntactically generated: the f in the arc annotations must appear through a relabelling, and that relabelling would produce the required abstract transition. This leads us to conjecture that:

CONJECTURE 7.4.2 For $B \approx B'$ syntactically generated High Level Petri Boxes, $\Omega_Y(B) \approx \Omega_Y(B')$. □ 7.4.2

We note that the full congruence nature of strong bisimulation was not required for the fix-point nature of the recursive construct.

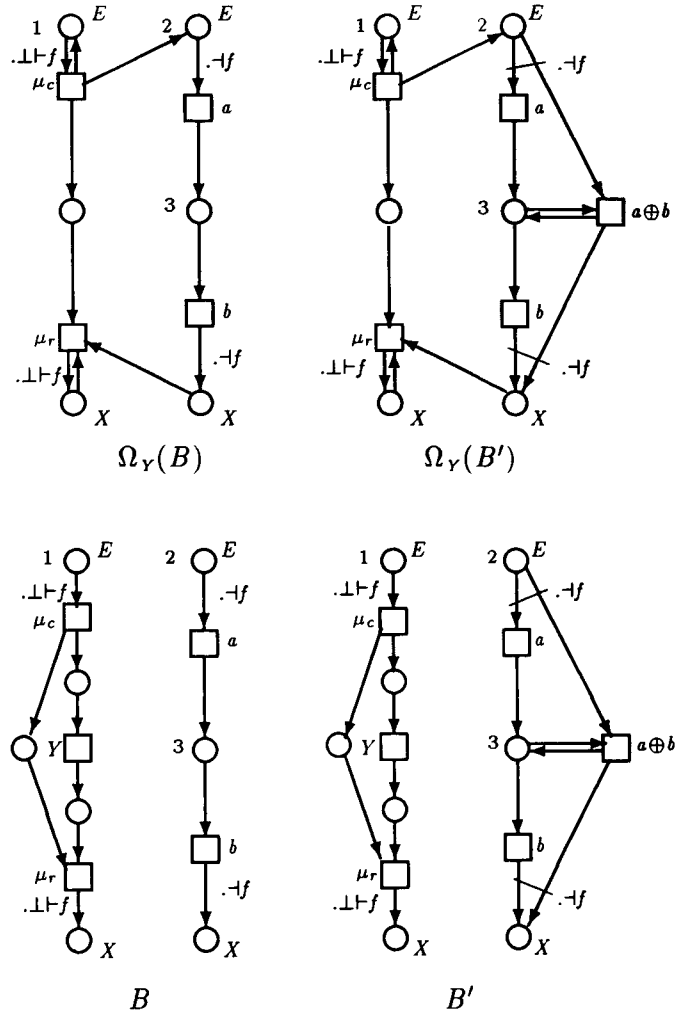


Figure 7.4: In which $B \approx B'$ but $\Omega_Y(B) \not\approx \Omega_Y(B')$. The marking $\{1 \rightarrow \{.\bot\vdash f\}, 2 \rightarrow \{.\bot\vdash f\}, 3 \rightarrow \{.\dashv f\}\}$ enables transition labelled $a \oplus b$ in $\Omega_Y(B')$ but there is no corresponding transition in $\Omega_Y(B')$ to be enabled. This marking is reachable in neither B nor B'. Note that B is not syntactically generated.

7.4.3 Weaker Bisimulations

The definition of strong bisimulation is just about the strongest one could imagine (while still allowing loops to be unwound in High Level Petri Boxes). Whereas this was sufficient for the purposes to which it was put, i.e., the demonstration that the recursive construct is a fix-point of the recursive operator, it does mean that we will distinguish many High Level Petri Boxes which are behaviourally indistinguishable, and which cannot be distinguished through composition.

For instance, the commutativity and associativity of concurrent and choice compositions are not included under the strong bisimulation, i.e., for syntactically generated High Level Petri Boxes

B_1 and B_2, $B_1\|B_2 \not\approx B_2\|B_1$. This would, of course, be a desirable property, as (we conjecture) $B_1\|B_2$ and $B_2\|B_1$ are not distinguishable through composition.

The reason that strong bisimulation does not include the above case is that we have been very strict in preserving the 'absolute' rather than the 'relative' identities of processes involved in the High Level Petri Boxes, i.e., strong bisimulation does not consider that the swapping of $\vdash/\dashv$ pairs in the arc annotations of local transitions as 'harmless'. Intuitively, all that would be required to accommodate such changes is not to force the pre- and post-contexts of a local transition to be equal: equality modulo $\vdash/\dashv$ pairs would be sufficient.

Technically, at least definitionally, this change would not be too difficult. However, it would require a more formal treatment of the symmetries of the Flow predicate and, as a brief inspection will confirm, would complicate many of the proofs of the congruence nature of the resulting strong bisimulation.

Another example in which the weakening would be attractive is that we have defined strong bisimulation using the most permissive label algebra. In particular, this forces 'dormant' labels (i.e., those with order strictly greater than 1) to be considered as part of behaviour. An interesting question is, then:

> *Is it possible to define a strong bisimulation which is a congruence with respect to the High Level Petri Box Algebra, which does not require the firing of transitions labelled with dormant labels?*

Unfortunately, the answer appears to be 'no'. To illustrate this, we will briefly consider the properties of the following weaker flavour of strong bisimulation which considers behaviours in the actual label algebras over which its operands are defined (and which are, of course, not necessarily maximally permissive):

DEFINITION 7.4.3 Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be label algebras and $d \in \mathcal{C}_{\mathcal{L}_1}$. We say that High Level Petri Boxes B_1 and B_2 over[15] $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively are *label algebra d-strongly bisimilar*, written $B_1 \approx^d_{la} B_2$, if $\mathcal{L}_1 = \mathcal{L}_2$ and there exists an interface respecting relation $\mathcal{R} \subseteq S_1 \times S_2$ such that:

1. $M_d^{I,B_1}\ \widehat{\mathcal{R}}\ M_d^{I,B_2}$,

[15] Not $\mathcal{L}_1^{pv}$ and $\mathcal{L}_2^{pv}$ as previously.

2. for all markings $M_i \in \mathcal{M}_d^{B_i}$, $i = 1,2$, $t_1 \in T_1$, and $\alpha \in \mathrm{subs}^{B_1}(t_1)$, whenever $M_1 \widehat{\mathcal{R}} M_2$, and $M_1[t_1:\alpha\rangle N_1$ there is a $t_2 \in T_2$ with $t_1 \widehat{\mathcal{R}} t_2$ and a marking $N_2 \in \mathcal{M}_d^{B_2}$ such that $M_2[t_2:\alpha\rangle N_2$ and $N_1 \widehat{\mathcal{R}} N_2$. Moreover, if $N_1 = M_d^{T,B_1}$ then $N_2 = M_d^{T,B_2}$ also,

3. for all markings $M_i \in \mathcal{M}_d^{B_i}$, $i = 1,2$, $t_2 \in T_2$, and $\alpha \in \mathrm{subs}^{B_2}(t_2)$, whenever $M_1 \widehat{\mathcal{R}} M_2$, and $M_2[t_2:\alpha\rangle N_2$ there is a $t_1 \in T_1$ with $t_1 \widehat{\mathcal{R}} t_2$ and a marking $N_1 \in \mathcal{M}_d^{B_1}$ such that $M_1[t_1:\alpha\rangle N_1$ and $N_1 \widehat{\mathcal{R}} N_2$. Moreover, if $N_2 = M_d^{T,B_2}$ then $N_1 = M_d^{T,B_1}$ also. ■ 7.4.3

We will assume that $\approx_{la}$ bears the same relationship to $\approx_{la}^d$ as $\approx$ does to $\approx^d$.

That $\approx_{la}$ is not a congruence is shown by the example of Figure 7.5 in which are illustrated High Level Petri Boxes B and B' (the denotations of High Level Petri Box terms $\mathtt{stop}\|(b+Y)[f]$ and $a\|(b+Y)[f]$, respectively) and for which $B \approx_{la} B'$ in any label algebra in which $f(b) \notin A$, but for which $\mu Y.B \not\approx_{la} \mu Y.B'$ if $f(a \oplus f(b)) \in A$. (We note that $B \not\approx B'$.)

Prohibiting the synchronisation behaviour which is seen in the example of the figure would be one way of recovering the congruence nature of the weaker notion of strong bisimulation. More generally we conjecture that, for even weaker notions of bisimilarity, the congruence nature may be determined through the restriction of the synchronisation behaviour allowed in label algebras. A taxonomy of label algebras and their congruence inducing properties would be an interesting topic for future work.

7.4.4 Finite Approximants to Recursive Behaviours

Using Theorem 7.3.8, we may develop the notion of a finite approximant to a recursive behaviour. Given a syntactically generated Y-guarded High Level Petri Box B, we may construct a class of High Level Petri Boxes, the *finite approximants to* $\mu Y.B$, $\{B_{ref}^{(i)} \mid i \in \mathbb{N}\}$ such that

$$\begin{aligned} B_{ref}^{(0)} &= B[Y \leftarrow \mathtt{stop}] \\ B_{ref}^{(i+1)} &= B[Y \leftarrow B_{ref}^{(i)}] \end{aligned}$$

We note that $B_{ref}^{(n)}$ is a syntactically generated High Level Petri Box in which winding does not appear, and so which fits within the syntax of Definition 4.8.1 (modulo the possible appearance of guarding and process variables other than Y). Moreover, from Theorem 7.3.8, it is not difficult to see that $B_{ref}^{(n)}$ contains (at least) the markings of $\mu Y.B$ which can be reached from a standard initial marking by firing n of fewer local transitions.

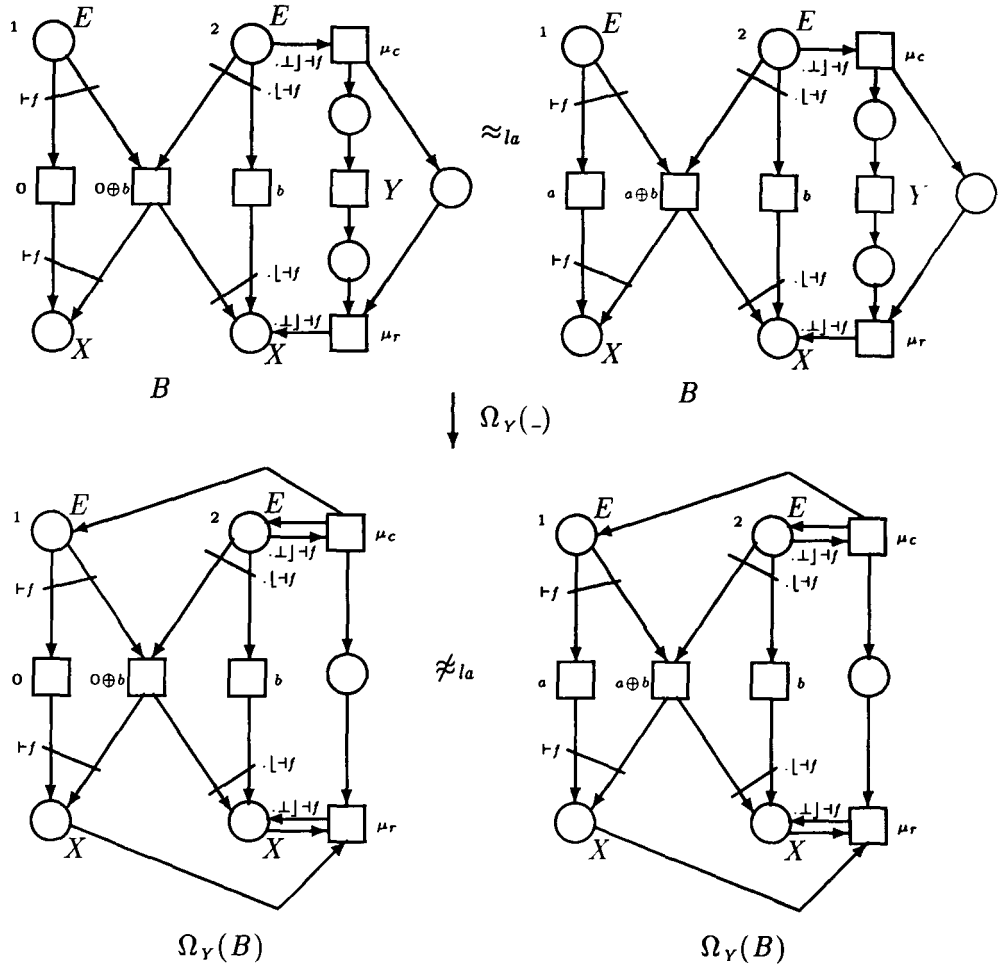


Figure 7.5: In which $B \approx_{la} B'$ but $\mu Y.B \not\approx_{la} \mu Y.B'$. The figure illustrates that $\Omega_Y(B) \not\approx_{la} \Omega_Y(B')$ in any label algebra in which $f(a) = \mathbf{0}$, $f(a \oplus b) = \mathbf{0}$ but $f(a \oplus f(b)) \in A$. The marking $\{1 \to \{(.)\}, 2 \to \{(.\lfloor \dashv f \perp \rfloor \vdash f)\}\}$ does not enable transition t in $\Omega_{Y(B)}$ but enables transition t' in $\Omega_{Y(B')}$. This marking is not reachable in B nor B'.

As $B^{(n)}_{ref}$ contains no applications of the winding auxiliary, (with minor extensions for the guarding operator and process variables) the constructions of Chapter 5 will apply (*mutatis mutandis*, giving a normal form which applies to all finite behaviours of the recursive construct. This may give a simple method for showing that the commutative and associative nature of parallel and choice compositions, based upon the symmetries of the H-synchrony rule, extend to all syntactically generated High Level Petri Boxes.

Finally, we note that the accounting places used in the recursive construct serve no purpose in the finite approximants, but are 'harmless'.

7.5 Summary

In this chapter we have demonstrated certain important behavioural properties of syntactically generated High Level Petri Boxes. The demonstration of these properties is by no means trivial, but has justified each of the choices which have taken in the development of the High Level Petri Box model.

More specifically, we have shown developed a full behavioural characterisation of syntactically generated High Level Petri Boxes in Lemma 7.1.2. This characterisation is most interesting in the case of recursion, in which we have shown that any reachable marking of an application of the recursive construct can be modelled by a tree, and behaviours correspond to 'tree'-morphisms. Indeed, with suitable generalisations, the behaviour of applications of each operator might be interpreted this way.

We have also developed the basis of a family of behavioural congruences (concurrency preserving strong bisimulations) on High Level Petri Boxes. Moreover, and importantly for the ultimate goal of the work, such bisimulations will form the basis of a rich algebraic characterisation of the High Level Petri Box model.

We have also shown the fix-point nature of the recursive construct. With this result comes the possibility of producing finite approximants to recursive behaviours, and with them the possibility of extending the normal form of Chapter 5 to the behaviours of all syntactically generated High Level Petri Boxes.

It is interesting to note that the proofs of the congruence nature and of the fix point nature relied crucially upon the behavioural characterisations of Lemma 7.1.2. This lemma and its corollaries would, then, seem to be important tools for the manipulation of the behaviours of High Level Petri Boxes.

In the next chapter we summarise the results and contributions of this work, and consider how the current state of the High Level Petri Box Algebra and of the High Level Petri Box model may subsequently be developed.

Chapter 8

Conclusions

The goal of this work was to define an algebra of high level Petri nets and to begin the characterisation of its properties. In this chapter we will review the contribution the work reported here may be seen to have made towards this goal.

8.1 Summary

8.1.1 Label Algebras

Label algebras encode communication models as commutative, associative structures. Unlike the synchronisation algebras of Winskel which they extend, label algebras are not limited to inherently binary synchronisations, allowing communication models in which, for instance, synchronisation is 3-way, but in which no 2-way synchronisations are allowed (cf. Example 2.1.2).

We have provided label algebras for the most well-known of the communication models of the literature: i.e., CCS and TCSP, together with those for the low level Petri Box Calculus and Taubner's algebra, **A**.

8.1.1.1 H-Synchrony

We have introduced a new SOS-style rule called H-synchrony which combines aspects of the SOS-rules for the expression of Asynchrony, Synchrony and Morphism (Section 2.2.2). The H-synchrony rule synchronises actions from concurrently executing processes (if that synchronisation is allowed by the chosen label algebra), and is used to characterise the conditions under

which atomic actions of a High Level Petri Box may be executed (i.e., the conditions under which abstract transitions may fire).

8.1.2 The High Level Petri Box Model

We have defined a highly modular class of high level Petri nets. High Level Petri Boxes, consisting of entry and exit, and communication interfaces through which all composition between High Level Petri Boxes is performed. The claim to modularity is justified through the fact that the innards of a High Level Petri Box are hidden during composition. We have shown that High Level Petri Boxes are equivalent to a subclass of the PrT nets of Genrich, in terms of which is defined their behaviour (Section 3.3). Incorporated in the translation is the logical encoding of the H-synchrony rule as the selectors of the PrT net transitions, which ensures that their behaviour is derived therefrom.

The notions of completeness, decoupling and most permissive label algebras were introduced in Section 3.4, and in that section we have given important tools for the characterisation of behaviours of High Level Petri Boxes, underlying much of the subsequent work:

1. that we may consider only local behaviour in the determination of *positive monotonic properties* (Lemma 3.4.4), and
2. that for an important subclass of High Level Petri Boxes (including those forming the denotation of the algebra), behaviour is independent of the initial marking when behaviours are considered in the most permissive label algebra (Theorems 3.4.7 and 3.4.8).

8.1.3 The High Level Petri Box Algebra

We have introduced an algebra of high level Petri nets, the High Level Petri Box Algebra, with underlying model a subclass of High Level Petri Boxes.

The organisation of High Level Petri Boxes into the semantic domain of the High Level Petri Box Algebra was achieved in Chapters 4 and Chapter 6 through the definition of a compositional semantics on the following syntax:

DEFINITION 8.1.1 *[High Level Petri Box Syntax]* Given a label algebra $\mathcal{L} = \langle A, R \rangle$, the $\mathcal{L}$-

Syntax of the High Level Petri Box Algebra is defined as follows:

t ::=	stop		
	u	$u \in A$	
	Y	$Y \in \mathcal{PV}$	
	$t \| \| t$		Concurrent composition
	$t;t$		Causal composition
	$t+t$		Choice composition
	$t[f]$	$f \in R$	Relabelling
	$t[Y \leftarrow t]$	$Y \in \mathcal{PV}$	Refinement
	$\mu Y.t$	$Y \in \mathcal{PV}$	Recursion

■ 8.1.1

The syntax is parametrised by a label algebra, $\mathcal{L}$, whence for each communication model there is a High Level Petri Box Algebra. There are, therefore, High Level Petri Box Algebras for CCS, TCSP, the PBC, *etc.*

Which operators to include in the algebra was, more or less, determined by the existing process algebra literature. We have operators for the expression of asynchrony (concurrent composition), causal dependence (causal composition), conflict (choice composition), synchronisation (relabelling), modularity (refinement), and infinite behaviours (recursion).

We note only that:

1. The use of causal composition rather than prefix (which is the more traditional expression of causality in action based approaches and therefore many net based approaches) was motivated (in Chapter 4) to be the pragmatic choice for the expression of causality seeing the equal prominence given to state and action in the net setting. As we have seen, prefix and sequence are interderivable; however, as we have also seen the derivation of prefix from sequence is very much easier than the derivation of sequence from prefix. In particular, the expression of non-tail-end recursive terms is very much facilitated by using causal composition rather than prefix (Section 6.8.3).

2. Relabelling is an entirely novel operator. Relabelling allows us to remove from concurrent composition any synchronisation component, allowing that operator to be both commutative *and* associative. Relabelling is, in consequence, the only operator of the algebra in which are found details of the parametrising label algebra $\mathcal{L}$.

3. Refinement is a purely syntactic operator, by which we mean that its application commutes with the application of all other operators. The High Level Petri Box Algebra with and without refinement is, therefore, equally expressive. Refinement operators are known to provide for modularity and economy of expression, and we have included one such for this reason.

4. It is also notable that concurrent, causal, and choice compositions have become disjoint unions of 'processes' (as should surely be the case), each with a different method of distinguishing the operands of the union: concurrent and choice composition through the nominal relabellings $\vdash$, $\dashv$, $\lfloor$ and $\rfloor$, and causal composition through manipulations on places.

The semantic interpretation of the operators was, of course, determined by the High Level Petri Box model, although we have been guided in their definition by operators of the literature and, in particular, those based upon place multiplication.

8.1.4 Structural and Behavioural Characterisation of the Semantic Domain

In Chapters 4, 5, 6 and 7 we have partially characterised the structural and behavioural interrelation of elements of the semantic domain in terms of $=$ and $\equiv_{Reach}$ (defined in Chapter 3) and $\approx$ (defined in Chapter 7). Table 8.1 summarises (and names) the established equivalences.

In addition we have provided an elementary, but full, behavioural characterisation of the semantic interpretations of the High Level Petri Box operators in Lemma 7.1.2. With this result we have determined that:

1. concurrent composition, $B_1 \| B_2$, allows B_1 and B_2 to proceed independently;

2. causal composition, $B_1;B_2$, behaves first like B_1 and then, when B_1 has terminated, like B_2;

3. choice composition, B_1+B_2, behaves either like B_1 or B_2, depending as to whether the first action executed belonged to B_1 or B_2;

4. relabelling, $B[f]$, for a relabelling f, allows concurrent processes of B to proceed independently or synchronise if allowed under the H-synchrony rule;

5. refinement, $B_1[Y \leftarrow B_2]$, behaves as would the term $B_1[B_2/Y]$;

Concurrent composition

[CON–COM]	$B_1 \| B_2$	$\equiv_{Reach}$	$B_2 \| B_1$	
[CON–ASS]	$(B_1 \| B_2) \| B_3$	$\equiv_{Reach}$	$B_1 \| (B_2 \| B_3)$	

Causal composition

[CAU–ASS]	$(B_1;B_2);B_3$	$=$	$B_1;(B_2;B_3)$	

Choice composition

[CHO–UNIT]	$B + \mathtt{stop}$	$\equiv_{Reach}$	B	
[CHO–COM]	$B_1 + B_2$	$\equiv_{Reach}$	$B_2 + B_1$	
[CHO–ASS]	$(B_1 + B_2) + B_3$	$\equiv_{Reach}$	$B_1 + (B_2 + B_3)$	

Refinement

[REF–ID]	$B[Y \leftarrow B']$	$=$	B	$Y \notin \mathcal{FV}(B)$
[CAU–REF–DIST]	$(B_1;B_2)[Y \leftarrow B']$	$=$	$B_1[Y \leftarrow B'];B_2[Y \leftarrow B']$	
[CHO–REF–DIST]	$(B_1 + B_2)[Y \leftarrow B']$	$=$	$B_1[Y \leftarrow B'] + B_2[Y \leftarrow B']$	
[CON–REF–DIST]	$(B_1 \| B_2)[Y \leftarrow B']$	$=$	$B_1[Y \leftarrow B'] \| B_2[Y \leftarrow B']$	
[REL–REF–DIST]	$B[f][Y \leftarrow B']$	$=$	$B_1[Y \leftarrow B'][f]$	

Recursion

[REC–CON]	$\mu Y.B$	$\approx$	$B[Y \leftarrow \mu Y.B]$	when B is Y-guarded

Table 8.1: Equivalences of High Level Petri Box Terms.

6. recursion, $\mu Y.B$, behaves as $B[Y \leftarrow \mu Y.B]$ (when B is Y-guarded).

With Lemma 7.1.2 we thus claim to have shown that the operators of the High Level Petri Box Algebra have the behaviour that would be expected of them, given the description of their counterparts in the process algebra literature.

8.2 Future Work

Many aspects of the work presented in this thesis are, we feel, essentially complete in themselves. These include the basic notions underlying High Level Petri Boxes such as, for instance, the definition of label algebras (Section 2.1); the PrT semantics and the behaviour of High Level Petri Boxes (Chapter 3); the algebra (Chapters 4 and 6); and the structure of its semantic domain (Definition 2.5.15 and Chapters 4 and 6).

However, there are many areas in which the work is incomplete, and would provide the basis for productive future work. Among the nearer-term goals should be:

1. The introduction of 'restricted' combinational closure operators into the algebra. For instance, consider the class of operators $\{\_^{\oplus_i}\}$, each of which introduces multisets of transitions into a High Level Petri Box, but restricted so that the multisets introduced have order less than or equal to i. The important property of such operators is that, unlike combinational closure, they preserve the cardinality of the set of transitions of their operand High Level Petri Boxes.

   However, only for the special case when $i = 1$ is $\_^{\oplus_i}$ idempotent; indeed, $B^{\oplus_i \oplus_j} = B^{\oplus_{i+j}}$. Rewrite θ (Definition 5.2.1), and the normal form based thereon, fails to hold for an algebra based on these operators. Moreover, because of the (multiplicative) nature of the properties of $\_^{\oplus_i}$ a naive extension of this rewrite would not exist.

   The single case when $\_^{\oplus_i}$ is idempotent, i.e., when $i = 1$, is harmless, and perhaps even useful (and, in effect, it already appears as the context manipulation component of relabelling). If we were to consider promoting context manipulation to the top level operators, we would have the extra laws for a renaming (see Example 2.1.1), following directly from Proposition 5.4.17 and as expressed in Table 8.2.

2. The weakening of strong bisimulation to a congruence modulo the commutative and associative nature of concurrent and choice compositions (if such exists, which we strongly believe to be the case).

Relabelling (Extra Laws)

$$\begin{array}{lrcl}
\text{[CAU-REL-COM]} & (B_1;B_2)\odot.f & \equiv & (B_1\odot.f;B_2\odot.f) \\
\text{[CHO-REL-COM]} & (B_1+B_2)\odot.f & \equiv & (B_1\odot.f+B_2\odot.f) \\
\text{[CON-REL-COM]} & (B_1\|B_2)\odot.f & \equiv & (B_1\odot.f\|B_2\odot.f)
\end{array}$$

Table 8.2: Partial characterisation of transformation rules for a restricted combinational closure operator in combination with a renaming f.

The goal of this weakening would be the development of observational congruences, the existence of which would offer the possibility of sound (and ultimately complete, if they exist) axiomatisations of instances of the High Level Petri Box Algebra. It would appear that the obvious way of characterising sub-algebras is in terms of the label algebras over which instances of the High Level Petri Box Algebra are defined. Initially, we would aim to follow the work of the literature on the provision of observational congruences for, for example, CCS, attempting to derive the same results which appear therein in a High Level Petri Box setting.

Related to this, the equivalences in terms of which Table 8.1 is expressed are not ordered by inclusion ($= \subseteq \equiv_{Reach}$ and $= \subseteq \approx$, but $\equiv_{Reach}$ and $\approx$ are incomparable). Consequently, the table is not expressible in terms of any one equivalence we have defined in this work; a weaker bisimulation may provide such an equivalence.

3. The development of a full, compositional behavioural semantics of the algebra as was implied possible in the comments following Lemma 7.1.2. We suggest that a semantic domain based on the notion of reachability graphs of High Level Petri Boxes would be initially feasible: the current proof of Lemma 7.1.2 (*mutatis mutandis*) provides the description of this new semantic domain, together with the justification of the homomorphism condition (required for compositionality).

   Of course, a truly concurrent behavioural semantics should be our goal. In fact, and as we have seen in Chapter 5, many aspects of the structural semantics appear to lend themselves to analysis through partial orders based on local causality. Moreover, using the finite approximants which were developed from the fix-point nature of the recursive operator we may be able to extend such techniques based to the reachable behaviours of all syntactically generated High Level Petri Boxes.

Of a more speculative and long-term nature are the following:

1. We should develop an operational semantics, almost certainly in the Structured Operational Semantics-style of Plotkin. Initially this is envisaged to be an interleaving (single global state) semantics, although of real value would be a fully distributed operational semantics (à la Degano *et al*). The development of an interleaving semantics would be relatively straightforward (based on ideas similar to those expressed in the operational semantics of [BK95]); a fully distributed semantics would present more of a challenge.

   The obvious result to look for here would be to show that the operational semantics and behaviour of the denotational semantics coincide.

2. In the spirit of Taubner, we should attempt to characterise sub-algebras that correspond to the traditional (and stable) process algebras of the literature, CCS, TCSP, ACP, and Taubner's **A**, for instance. This would require us to find suitable derivations of the operators in the various process algebras in terms of those of the High Level Petri Box Algebra (we suspect this is relatively straightforward). More challenging and importantly, we should show that this characterisation extends to behaviours. We note that, initially, it would be sufficient to show that Taubner's **A** is contained as an instance of the High Level Petri Box Algebra as this includes large portions of the other approaches.

   This would most likely be approached through the operational semantics proposed by the previous item, as the abstractions possible from the particular details of our net-based approach (in particular, the explicit representation of state) may reduce the complexity of the arguments required.

3. In the spirit of Devillers, we should investigate an S-invariant analysis of High Level Petri Boxes. The work in Chapter 5 on the definition of flat High Level Petri Boxes (perhaps extended to work on finite approximants of recursive behaviours, as mentioned above) will be of relevance here as it states, essentially, that higher and lower level Petri Box models are closely linked.

4. The syntax (Definition 6.6.1) of the High Level Petri Box Algebra might be used as a rudimentary specification language for concurrent and distributed systems. However, a longer-term aim should be to provide tools for the end-user of High Level Petri Boxes. Initially, in the spirit of Best *et al.* ([BH93, BF93]), this would be through a linear design language allowing the algebra to be encoded using more or less traditional and

familiar programming constructs, hopefully making the formalism accessible to engineers of concurrent and distributed systems.

That the underlying graphical and true concurrency semantics of the design notation (as provided by High Level Petri Boxes) is important in this regard is clear. As has been understood since their inception, the graphical representation of Petri nets provides much insight into the workings (or failings) of concurrent systems and we should make the most of this for engineers of such systems. For instance, tools which allow the animation of a specified system at the net level would, we feel, increase dramatically the accessibility of the formalism.

Appendix A

Index of Notation

Abstract Transitions

$\bar{t}$ 28
$|t|$ 27
$^\bullet t^\bullet$ 28
$^\bullet t$ 28
$t(C)_B$ 79
$t^\bullet$ 28
$t(l)_B$ 79
$\mathcal{T}_\mathcal{L}$ 27

Contexts

$\pm, \mp$ 95
$-(C)$ 18
$C_1 - C_2$ 19
$(.)$ 17
$C_1 \leq C_2$ 19
$C_1 < C_2$ 19
$C_1 \sqsubseteq C_2$ 19
$C_1 \sqsubset C_2$ 19
$a[C]$ 19
$\perp$ 17
$\mathcal{C}_\mathcal{L}$ 17
$C_1 \frown C_2$ 18
$\lfloor$ 17
$\vdash$ 17
$\rfloor$ 17
$\dashv$ 17

High Level Petri Boxes

$^\bullet B^\bullet$ 29
$^\bullet B$ 29
$+_1, +_2$ 87
$;_1, ;_2$ 85
$;_1^{(t_1,t_2)}(t_1)$ 108
$M[\rangle$ 50
$M[t\rangle M'$ 50
$M[t\rangle$ 50
$Acc(B)$ 169
$Acc_Y(B)$ 169
$\alpha^{(i)}$ 55
α 47
$\mathcal{A}_\mathcal{P}$ 26
$B^\bullet$ 29
$B_1 \| B_2$ 83
$[B_1] \hat{\|} [B_2]$ 101
$B_1 + B_2$ 87
$[B_1] \hat{+} [B_2]$ 101
$B_1 ; B_2$ 85
$[B_1] \hat{;} [B_2]$ 101
$B_1 =_{P/T} B_2$ 61
$B_1 =_{PrT} B_2$ 61
$B_1 \cup B_2$ 73
$B_1 \equiv_{P/T} B_2$ 61
$B_1 \equiv_{PrT} B_2$ 61
$B_1 \equiv B_2$ 36
$B_1 \equiv^d_{Reach} B_2$ 61
$B_1 \subseteq B_2$ 30
$B[f]$ 89
$[B]\widehat{[f]}$ 101
B^C 29
B^d 119
$B^\oplus$ 77
$B[l - D]$ 150
$B[L_Y(B) - D]$ 158
$B(l)$ 79
$\overset{\bullet}{B}$ 29
$B \bar{\ } (C, C')$ 120
$B \ddot{\ } (C, C')$ 79
$B \ddot{\ } C$ 79
$Box_\mathcal{L}(u)$ 90
$Box_\mathcal{L}(Y)$ 148
$[B]$ 101
$C[\_]$ 94
$^C B$ 29
$\mathcal{SHLPB}^{\mathcal{N}}_{\Sigma}$ 147

$\mathcal{EHLPB}_{\mathcal{L}}$ 101
$EHLPB_{\mathcal{L}}^{\mathcal{PV}}$ 147
$EHLPB_{\mathcal{L}}^{\mathcal{PV}}(Y)$ 148
ϵ 102
$E[Y \leftarrow t]$ 147
$\hat{f}$ 80
f_1^I 150
f_2^I 150
f_Ω 170
$\Gamma(P)$ 166
$\mathcal{HLPB}_{\mathcal{L}}^*$ 31
$\mathcal{HLPB}_{\mathcal{L}}$ 92
$HLPB_{\mathcal{L}}(t)$ 92
$\mathrm{id}(B)$ 30
$index(t)$ 47
$\mathbf{L}_{\mathcal{L}}$ 42
$LT(B)$ 29
$L_Y(B)$ 158
$M(d)$ 56
$M[l\rangle N$ 55
M 33
$\mathcal{M}_d^B$ 50
$M(C)_B$ 194
$M_d^{I,B}$ 34
$M[\overline{t:\alpha}\rangle M'$ 50
$M[t:\alpha\rangle M'$ 49
$M\left[\begin{smallmatrix}\overline{t:\alpha}\\ t:\alpha\end{smallmatrix}\right\rangle M'$ 50
$M_d^{T,B}$ 34
$\mu Y.B$ 173
$\mathcal{O}:\alpha$ 47
Ω 42
$\Omega^{(n)}$ 42
$\Omega_Y(B)$ 170
$\Pi^{(n)}$ 42
Π 42
$\mathcal{PHLPB}_{\mathcal{L}}$ 28
$PrT(B)$ 44
$\Phi(B)$ 125
$P/T(P)$ 60
$\mathcal{PV}$ 146
$\mathcal{P}$ 24
$\mathcal{R}_{\mathcal{L}}$ 43
S^d 61
$\hat{s}d$ 60
s^I 169
stop 90
$\mathrm{subs}^P(t)$ 48
$t_1 \equiv_\rightarrow t_2$ 117
$t{:}\alpha$ 59
t_{aux} 108
$Terms_{\mathcal{L}}$ 91
t^{nf} 117
$T_{D'}^{\oplus_r}$ 150
$t^\oplus \odot .f_{t^\oplus}$ 108
$t \Rightarrow t'$ 115
$t \rightarrow_{aux} t_{aux}$ 115
$t \rightarrow_\varpi t'$ 115
$t \rightarrow t'$ 109
$[t]$ 117
$\mathcal{V}$ 42
W 44
W_F 44
W_N 44
W_S 44
W_T 44
$\Psi(B)$ 127

Label Algebras

$\mathbf{0}$ 8
$\mathcal{A}_{\mathbf{A}}$ 11
$\mathcal{A}_{CCS}$ 11
A_μ 165
$a \oplus b$ 8
$\mathcal{A}_{PBC}$ 12
$A^{\mathcal{PV}}$ 177
$\mathcal{A}_{TCSP}$ 12
f_μ 165
$f^{\mathcal{PV}}$ 177
$\mathcal{L}_{\mathbf{A}}$ 11
$\mathcal{L}_{CCS}$ 11
$\mathcal{L}_\mu$ 165
$\mathcal{L}_{\mathcal{MP}}$ 54
$\mathbb{L}_\mu$ 164
$\mathcal{L}_{PBC}$ 12
$\mathcal{L}^{\mathcal{PV}}$ 177
$\mathcal{L}_{TCSP}$ 12
$\mathbb{L}$ 7
$\mathcal{L}$ 8
μ_c 167
μ_r 167
μ 162
$\mathcal{R}_{\mathbf{A}}$ 11
$\mathcal{R}_{CCS}$ 11
R_μ 165
$\mathcal{R}_{PBC}$ 12
$R^{\mathcal{PV}}$ 177
$\mathcal{R}_{TCSP}$ 12
τ 8

Local Transitions

$\overline{l}$ 26
$\bullet l \bullet$ 27
$\bullet l$ 26
$[l]$ 30
α^K 131
$^C l$ 26
$l(C)_B$ 79
$l^\bullet$ 26

$l_1 \preceq l_2$ 125
$l_1 \prec l_2$ 125
l^C 26
$loc_{\mathcal{L}}$ 26
η_B 131

Miscellaneous

$\circ$ 108
$\triangleleft$ 50
$\triangleright$ 194
$A[C/B]$ 18
nil 66
$\mathbb{N}$ 14
$o^{(n)}$ 42
P_f 71
$\surd$ 66
$T^{\oplus}$ 77

Order Sorted Algebra

$\hookrightarrow$ 143
$!$ 143
$s_1 \leq s_2$ 141
$a^*: \mathcal{T}_\Sigma(X) \to A$ 143
$A_d^{w,s}: A_w \to A_s$ 142
$a_y: X_s \to \mathcal{T}_{\Sigma(X)}$ 143
$\mathbf{OSAlg}_\Sigma$ 142
$\Sigma_{s_1 \ldots s_n, s}$ 141
$\Sigma(X)$ 143
$\mathcal{T}_\Sigma(X)$ 143
$\mathcal{T}_{\Sigma(X)}$ 143
$\mathcal{T}_\Sigma$ 142
X_s 143

Sets

$A \otimes B$ 24
$\{\!|s_1, \ldots, s_n|\!\}$ 12
$\langle S, \preceq \rangle$ 123
$\langle S, \preceq \rangle$ 123
$f \lfloor w$ 55
$\mathrm{id}(S)$ 25
λ 141
$\mathcal{M}_{\mathcal{F}}(S)$ 8
$n.\{\!|l|\!\}$ 150
$\mathbb{P}S$ 26
$\mathrm{ran}(R)$ 92
S^* 141
S^0 141
$s \# s'$ 124

Bibliography

[Age75] T. K. M. Agerwala. *Towards a theory for the analysis and synthesis of systems exhibiting concurrency.* PhD thesis, John Hopkins University, Baltimore, Maryland, 1975.

[Bar84] H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics (Revised Edition)*, volume 103 of *Studies in Logic and The Foundations of Mathematics*. North-Holland, 1984.

[BCC+86] B. Berthomieu, N. Choquet, C. Colin, B. Loyer, J. M. Martin, and A. Mauboussin. Abstract Data Nets: combining Petri nets and abstract data types for high level specification of distributed systems. In *Proceedings of the 7th European Workshop on Application and Theory of Petri nets*, 1986.

[BdCM88] E. Battiston, F. de Cindio, and G. Mauri. OBJSA nets: a class of high level nets having objects as domains. In G. Rozenberg, editor, *Advances in Petri Nets 1988*, volume 340 of *Lecture Notes in Computer Science*, pages 20–43. Springer-Verlag, 1988.

[BDE93] E. Best, R. Devillers, and J. Esparza. General refinement and recursion operators for the Petri Box Calculus. In *Proceedings of STACS'93*, volume 665 of *Lecture Notes in Computer Science*, pages 130–140. Springer-Verlag, 1993.

[BDH92] E. Best, R. Devillers, and J. G. Hall. The Box Calculus: A new causal algebra with multilabel communication. In Rozenberg [Roz92], pages 21–69.

[BDKP91] E. Best, R. Devillers, A. Kiehn, and L. Pomello. Concurrent bisimulation in Petri nets. *Acta Informatica*, 28:231–264, 1991.

[Bes87] E. Best. COSY: its relationship to CSP. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Central Models and their Properties*, volume 255 of *Lecture Notes in Computer Science*, pages 416–440. Springer-Verlag, 1987.

[BF86] E. Best and C. Fernández. Notations and terminology of Petri nets. Technical report, Arbeitspapiere der GMD 195. Bonn, 1986.

[BF93] E. Best and H. Fleischhack. PEP - Programming environment based on Petri nets, 1993.

[BG91a] D. Buchs and N. Guelfi. CO-OPN: A Concurrent Object Oriented Petri Net approach for system specification. In *Proceedings of the 12th International Conference on Application and Theory of Petri Nets*, pages 432–454, 1991.

[BG91b] D. Buchs and N. Guelfi. Open distributed programming using the object oriented specification formalism CO-OPN. Technical Report 700, LRI, 1991.

[BH92] E. Best and J. G. Hall. The Box Calculus: A new causal algebra with multilabel communication. Technical Report 369, University of Newcastle upon Tyne, 1992.

[BH93] E. Best and R. Hopkins. $B(PN)^2$ - a basic Petri net programming notation. In *Proceedings of PARLE'93*, volume 694 of *Lecture Notes in Computer Science*, pages 379–390. Springer-Verlag, 1993.

[BHR84] S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of Communicating Sequential Processes. *Journal of the Annals of Computing Machinery*, 31, 1984.

[Bil89] J. Billington. Many-sorted high level nets. In *Proceedings of the* 3rd *International Workshop on Petri nets and Performance Models.* IEEE CS Press, Washington D.C., 1989.

[BK95] E. Best and M. Koutny. Operational and denotational semantics of the box algebra. Technical report, Universität Hildesheim, 1995.

[BRR87] W. Brauer, W. Reisig, and G. Rozenberg, editors. *Petri Nets: Central Models and their Properties*, volume 254 of *Lecture Notes in Computer Science.* Springer-Verlag, 1987.

[BRW85] S. D. Brookes, A. W. Roscoe, and G. Winskel, editors. *Seminar on Concurrency*, volume 197 of *Lecture Notes in Computer Science.* Springer-Verlag, 1985.

[BW90] J. C. M. Baeten and W. P. Weijland. *Process Algebra.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1990.

[CM81] W. F. Clocksin and C. S. Mellish. *Programming in Prolog.* Springer-Verlag, 1981.

[Dai93] J. Daintith. *Anagram Finder.* Bloomsbury, 1993.

[dCdMPS83] F. de Cindio, G. de Michelis, L. Pomello, and C. Simone. Milner's communicating systems and Petri nets. In A. Pagnoni and G. Rozenberg, editors, *Applications and Theory of Petri Nets*, volume 66 of *Informatik-Fachberichte*, pages 40–59. Springer-Verlag, 1983.

[DdNM85] P. Degano, R. de Nicola, and U. Montanari. Partial ordering derivations for CCS. In L. Budach, editor, *Fundamentals of Computation Theory*, volume 199 of *Lecture Notes in Computer Science.* Springer-Verlag, 1985.

[DdNM87] P. Degano, R. de Nicola, and U. Montanari. CCS is an (augmented) contact free C/E system. In M. V. Zilli, editor, *Mathematical Models for the Semantics of Parallelism*, volume 280 of *Lecture Notes in Computer Science*, pages 144–165. Springer-Verlag, 1987.

[DdNM88] P. Degano, R. de Nicola, and U. Montanari. A distributed operational semantics for CCS based on Condition/Event systems. *Acta Informatica*, 26:59–91, 1988.

[DdNM90] P. Degano, R. de Nicola, and U. Montanari. A partial order semantics for CCS. *Theoretical Computer Science*, 75, 1990.

[Dev92] R. Devillers. Maximality preservation and the ST-idea for action refinements. In Rozenberg [Roz92], pages 108–151.

[Dev93] R. Devillers. Analysis of general refined boxes. In *Proceedings of the XIII$^{\text{th}}$ International Conference of the Chilean Society of Computer Science*, pages 419–434, 1993.

[Dev95] R. Devillers. S-invariant analysis of general recursive Petri boxes. *Acta Informatica*, 32:313–345, 1995.

[DK95] R. Devillers and H. Klaudel. Refinement and recursion in a high level Petri box calculus. In J. Desel, editor, *Structures in Concurrency Theory*, pages 144–159. Springer-Verlag, 1995.

[Gen87] H. J. Genrich. Predicate/Transition nets. In Brauer et al. [BRR87], pages 207–247.

[Gen89] H. J. Genrich. Equivalence transformation of PrT nets. In G. Rozenberg, editor, *Advances in Petri Nets 1989*, volume 424 of *Lecture Notes in Computer Science*. Springer-Verlag, 1989.

[GL81] H. J. Genrich and K. Lautenbach. System modelling with high-level Petri nets. *Theoretical Computer Science*, 13:109–136, 1981.

[GLT89] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1989.

[GM84] U. Goltz and A. Mycroft. On the relation of CCS and Petri nets. In J. Paredaens, editor, *ICALP 84*, volume 172 of *Lecture Notes in Computer Science*, pages 196–208. Springer-Verlag, 1984.

[Gog78] J. A. Goguen. Order sorted algebra. Technical Report 14, UCLA Computer Science Department Semantics and Theory Computation Series, 1978.

[Gog89] J. A. Goguen. Order-Sorted Algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partial operations. Technical Report SRI-CSL-89-10, SRI International, Computer Science Lab, July 1989.

[Gol88] U. Goltz. On representing CCS programs by finite Petri nets. In M. Chytil, L. Janiga, and V. Koubek, editors, *Proceedings Mathematical Foundations of Computing Science*, volume 324 of *Lecture Notes in Computer Science*, pages 339–350. Springer-Verlag, 1988.

[GvG89] U. Goltz and R. J. van Glabbeek. Refinement of actions in causality based models. In *Proceedings of the REX Workshop on Stepwise Refinement of Distributed Systems*, volume 430 of *Lecture Notes in Computer Science*, pages 267–300. Springer-Verlag, 1989.

[GW88] J. A. Goguen and T. Winkler. Introducing OBJ. Technical Report SRI-CSL-88-9. Computer Science Lab., SRI International, 1988.

[Hac75] M. Hack. *Decidability Questions for Petri Nets*. PhD thesis, MIT, Cambridge, MA, December 1975.

[Hac76] M. Hack. Petri net languages. Technical Report 159, Laboratory for Computer Science, MIT, Cambridge, 1976.

[HH91] J. G. Hall and R. P. Hopkins. PN^3: Preliminary notions for a Petri net based programming notation: Generalised communication. Technical report, DEMON. 1991.

[HHB92] R. P. Hopkins, J. G. Hall, and O. Botti. A basic-net algebra for program semantics and its application to OCCAM. In Rozenberg [Roz92], pages 179–214.

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice-Hall International, 1985.

[Jen87] K. Jensen. Coloured Petri nets. In Brauer et al. [BRR87], pages 248–299.

[Jen92] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*, volume 1: Basic Concepts of *Monographs in Theoretical Computer Science.* Springer-Verlag, 1992. ISBN: 3-540-60943-1.

[JK91] R. Janicki and M. Koutny. Invariant semantics of nets with inhibitor arcs. In *Proceedings of CONCUR'91*, volume 527 of *Lecture Notes in Computer Science.* Springer-Verlag, 1991.

[KB70] D. E. Knuth and P. B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra.* Pergamon Press, 1970.

[Kle52] S. C. Kleene. *Introduction to Metamathematics.* North-Holland Publishing Co., 1952.

[Knu86] D. E. Knuth. *Computers & Typesetting / A: The TEXbook.* Addison Wesley, 1986.

[Kot78] V. Kotov. An algebra for parallelism based on Petri nets. In J. Winkowski, editor, *Mathematical Foundations of Computing Science*, volume 64 of *Lecture Notes in Computer Science.* Springer-Verlag, 1978.

[KP95] H. Klaudel and E. Pelz. Communication as unification in the Petri Box Calculus. Technical Report LRI-TR 967, Université Paris Sud, Orsay, 1995.

[Kra85] B. Kramer. Stepwise construction of non-sequential software systems using a net-based specification language. In G. Rozenberg, editor, *Advances in Petri Nets 1984*, volume 188 of *Lecture Notes in Computer Science*. Springer-Verlag, 1985.

[Kra87] B. Kramer. A formal and semigraphical language combining Petri nets and abstract data types for the specification of distributed systems. In *Proceedings of the 9th International Conference on Software Engineering*, 1987.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.

[Mil85] A. J. C. Milner. Lectures on a Calculus of Communicating Systems. In Brookes et al. [BRW85], pages 197–220.

[Mil89] A. J. C. Milner. *Communication and Concurrency*. Prentice-Hall International, 1989.

[Min72] M. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall International, 1972.

[MM89] J. Meseguer and U. Montanari. Petri Nets are Monoids. Technical report, SRI International, 1989.

[Old87] E.-R. Olderog. Operational Petri net semantics of CCSP. In Rozenberg [Roz87], pages 196–223.

[Old91] E.-R. Olderog. *Nets, Terms and Formulas: Three Views of Concurrent Processes and Their Relationship*. Cambridge University Press, 1991.

[Pet62] C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, Technische Hochscule Darmstadt, 1962.

[Pet76] C. A. Petri. Interpretations of net theory. Technical Report ISF 75-07, 2nd edition, GMD, 1976.

[Pet81] J. Peterson. *Petri net theory and the modelling of systems*. Prentice-Hall International, 1981.

[Pet91] L. Petrucci. *Techniques d'Analyse des Réseau de Petri Algébraiques*. PhD thesis, Paris-Orsay, January 1991.

[Plo81] G. D. Plotkin. Structured approach to operational semantics. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

[Pom85] L. Pomello. Some equivalence notions for concurrent systems—an overview. In G. Rozenberg, editor, *Advances in Petri Nets 1985*, volume 222 of *Lecture Notes in Computer Science*, pages 381–400. Springer-Verlag. 1985.

[Rei91] W. Reisig. Petri nets and algebraic specifications. *Theoretical Computer Science*, 80:1–34, 1991. Also SFB-Berichte 342/1/90/B, (March 90).

[Roz87] G. Rozenberg, editor. *Advances in Petri Nets 1987*, volume 266 of *Lecture Notes in Computer Science*. Springer-Verlag, 1987.

[Roz92] G. Rozenberg, editor. *Special Issue of Advances in Petri Nets*, volume 609 of *Lecture Notes in Computer Science*. Springer-Verlag, 1992.

[RT86] G. Rozenberg and P. S. Thiagarajan. Petri nets: Basic notions, structure, behaviour. In J. W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Current Trends in Concurrency*, volume 224 of *Lecture Notes in Computer Science*, pages 585–668. Springer-Verlag, 1986.

[RV87] W. Reisig and J. Vautherin. An algebraic approach to high level Petri nets. In *Proceedings of the* 8^{th} *European Workshop on Applications and Theory of Petri Nets*, 1987.

[Sco82] D. S. Scott. Domains for denotational semantics. In *Automata, Languages and Programming*, volume 140 of *Lecture Notes in Computer Science*, pages 577–613. Springer-verlag, 1982.

[Sho77] R. E. Shostak. On the SUP-INF method for proving Presburger formulae. *Journal of the Association of Computing Machinery*, 24(4), 1977.

[Sho79] R. E. Shostak. A practical decision procedure for arithmetic with function symbols. *Journal of the Association of Computing Machinery*, 26(2), 1979.

[TA86] A. M. Tenenbaum and M. J. Augenstein. *Data Structure Using Pascal.* Prentice-Hall Software Series. Prentice-Hall International, second edition, 1986.

[Tau89] D. Taubner. *Finite Representations of CCS and TCSP Programs by Automata and Petri Nets*, volume 369 of *Lecture Notes in Computer Science*. Springer-Verlag, 1989.

[Thi87] P. S. Thiagarajan. Elementary net systems. In Brauer et al. [BRR87], pages 26–59.

[Vau87] J. Vautherin. Parallel systems specification with coloured Petri nets and algebraic specification. In Rozenberg [Roz87].

[vGV87] R. J. van Glabbeek and F. W. Vaandrager. Petri nets models for algebraic theories of concurrency. In J. W. de Bakker, A. J. Nijman, and P. C. Treleaven, editors, *Proceedings PARLE Conference, Eindhoven*, volume 259(II) of *Lecture Notes in Computer Science*, pages 224–242. Springer-Verlag, 1987.

[Win84] G. Winskel. A new definition of morphism on Petri nets. In M. Fontet and K. Mehlhorn, editors, *Proceedings of the Symposium of Theoretical Aspects of Computer Science*, volume 166 of *Lecture Notes in Computer Science*, pages 140–150. Springer-Verlag, 1984.

[Win85] G. Winskel. Categories of models for concurrency. In Brookes et al. [BRW85].