BIGRAPHS WITH SHARING AND APPLICATIONS IN WIRELESS NETWORKS

MICHELE SEVEGNANI

SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
Doctor of Philosophy

SCHOOL OF COMPUTING SCIENCE

COLLEGE OF SCIENCE AND ENGINEERING
UNIVERSITY OF GLASGOW

First submission: 22ND MAY 2012
Final submission: 29TH SEPTEMBER 2012

Abstract

Bigraphs are a fully graphical process algebraic formalism, capable of representing both the position in space of agents and their inter-connections. However, they assume a topology based on sets of trees and thus cannot represent spatial locations that are shared among several entities in a simple or intuitive way. This is a problem, because shared locations are often a requirement, for example, when modelling scenarios in the physical world or in modern complex computer systems such as wireless networks and spatial-aware applications in ubiquitous computing.

We propose *bigraphs with sharing*, a generalisation of the original definition of bigraphs, to allow for overlapping topologies. The new locality model is based on directed acyclic graphs.

We demonstrate the new formalism can be defined in the general framework of bigraphical theories and wide reactive systems, as originally devised by Robin Milner. We do so by defining a categorical interpretation of bigraphs with sharing, an axiomatisation derived from the equations of a bialgebra over finite ordinals, and a normal form to express bigraphical terms. We illustrate how sharing is essential for modelling overlapping localities by presenting two example case studies in the field of wireless networking. We show that bigraphs with sharing can be used realistically in a production environment by describing the implementation of an efficient matching algorithm and a software tool for the definition, simulation, visualisation and analysis of bigraphical reactive systems.

Acknowledgements

I am deeply grateful to my supervisor, Muffy Calder, for being extremely helpful and supportive throughout the years of my PhD and for providing me with valuable insight and enlightening suggestions whenever I encountered difficulties. Muffy's constant encouragement and sincere enthusiasm were a driving force in my research.

I also owe much to the late Robin Milner for his guidance early in my studies. His ideas have been a great inspiration for this thesis and helped establish my research directions.

I also thank all those I worked with in the Homework project, especially: Alexandros Koliousis, Joseph Sventek, Tom Rodden, Dimosthenis Pediaditakis, Naranker Dulay, Morris Sloman.

More personally, I thank my family for their unconditional love and support and all the good friends I have made during my stay in Glasgow: Andrea, Sedgil, Oberdan, Robin, Chris, Roberta, Raffaele, Erica, Maurizio, Irene, Luis, Gowri, Mirko, Nuno.

* * *

This is a revised version of my thesis; I have benefited a lot from the detailed criticism of my examiners Peter Sewell and Simon Gay.

to My Family: Gianluca, Adriana and Erica

Declaration

This thesis is submitted in accordance with the rules for the degree of Doctor of Philosophy at the University of Glasgow in the College of Science and Engineering. None of the material contained herein has been submitted for any other degree. The material contained herein is the work of myself except if stated otherwise.

Contents

Chapter 1

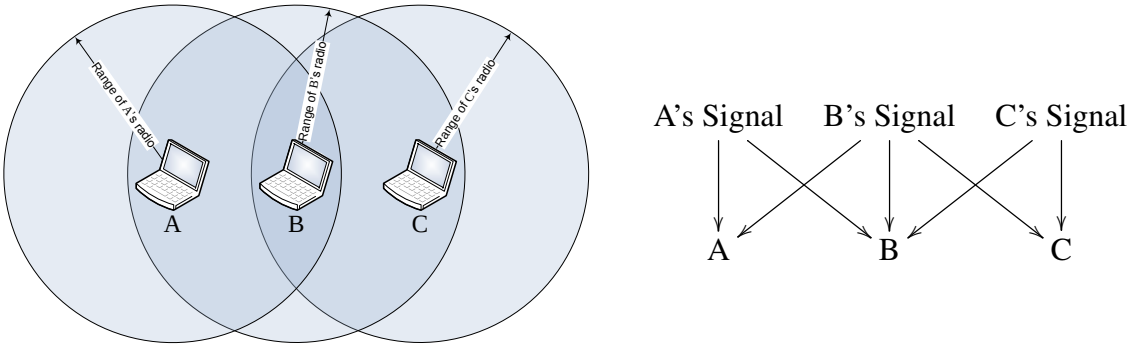
Introduction

1.1 Motivation: space

Ubiquitous computing is a vision of future computer systems conceived about two decades ago by Weiser [65] at the Electronics and Imaging Laboratory of the Xerox Palo Alto Research Center. His model of human computer interaction foresees a huge variety of interacting smart devices that will pervade our lives. These include sensors, situated displays, mobile devices, etc that will be seamlessly integrated into everyday objects and activities. Technology advancements will enable us to build increasingly complex systems in which devices are hidden from its users and are capable of autonomously managing the environment with little or no user input. Example studies in this area are the analysis of social interactions in future domestic environments [23], the development of innovative systems for the monitoring and control of private vehicles on the public highway [24], and the application to future health monitoring systems given in [44].

This vision involves scenarios that often exhibit a complexity that is almost intractable, given current software engineering practice. This can be explained by the fact that ubiquitous systems are less structured than traditional distributed systems and *spatial locations* assume a paramount rôle by influencing the behaviour of context-aware devices. Moreover, applications may present critical requirements from the point of view of functional correctness, reliability, availability, security, and safety. Therefore, a strong need is emerging for a *formal modelling* framework capable of representing the movement of agents, their intercommunications and their locations. This is essential in order to guide the specification and programming of these systems, understand their behaviour and facilitate repair when a malfunction occurs or when a component fails.

Spatial models are formalisms for the representation of the containment relationship between physical/virtual objects, computing devices, human users and places. The authors



(a) WLAN of three stations

(b) DAG topology

Figure 1.1: A network diagram of a WLAN of three stations (a) and a DAG representing its topology (b).

of [7] argue that location hierarchies based on directed acyclic graphs (DAGs) are more suitable than traditional models based on trees for the description of the spatial features of ubiquitous systems, especially when space is shared among several entities or the spatial topology is not neatly bounded. This is the case, for instance, when modelling devices interacting wirelessly as in the system drawn in Figure 1.1a. The expressive power of this spatial model becomes evident in the corresponding topology in Figure 1.1b. Unlike trees, the underlying DAG can natively represent the fact that machine A is currently occupying a space location covered by both A's signal and B's signal, but not by C's signal. Besides wireless communications, other scenarios requiring this kind of location concept have been studied. Two examples are a system of presence detectors situated in the rooms of a house [50] and a model of overlapping attention spaces [8].

In this thesis we will develop a process algebra formalism for representing systems whose behaviours are affected by spatial location. The formalism is based on Milner's *bigraphical reactive systems* but crucially, our underlying representation of spatial location is DAGs (instead of trees).

1.2 Related work

The quest for new conceptual models of computation has characterised informatics since its early days. Traditionally, basic formalisms such as automata, Turing machines [62] and the λ-calculus [18] are recognised as the precursors in this field. In this section, we give a brief historical background to the work in this thesis by recalling some related theoretical frameworks.

Process calculi

One of the first theories of concurrent processes was Petri nets [54] in 1962. This formalism is of particular relevance because it is perhaps the earliest example in which a rigorous mathematical treatment is used to describe discrete events. Research on the algebraic structure of processes began in the following decade. Alongside, the framework of *Structural Operational Semantics* (SOS), as pioneered by Plotkin [55], emerged as the preferred approach to define formally the behaviour of processes. This opened the way to the introduction of numerous process calculi (or process algebrae) such as Milner's Calculus of Communicating Systems (CCS) [46] in 1982. A popular paradigm among process calculi consisted of providing syntactical constructs to specify interactions, communications and synchronisations among independent processes and semantics interpretation by means of *labelled transition systems* (LTS). In this context, the behaviour of processes began to be studied with the introduction of *bisimulation* relations [52]. CCS served as the basis of the π-calculus of Milner, Parrow and Walker [48] in the early 1990's. This involved the definition of primitives to support *mobility*, i.e. the capability of changing the topology of the interconnections among processes during computation. Note that this does not include a notion of *locality* of processes. Another notable innovation was the reaction semantics of the Chemical Abstract Machine proposed by Berry and Boudol [9], in which SOS inference rules were replaced by rewrite rules on structurally congruent processes. An early model for the spatial aspects of computation was an enrichment of the π-calculus with locations [58] in which operations can be indexed by the location where they are executed. Another example in this direction was Cardelli and Gordon's calculus of mobile ambients [16]. In both cases, the location hierarchy is organised into a tree structure.

Term rewriting

Term rewriting is the branch of theoretical computer science concerned with the transformation of algebraic terms by means of rewrite rules application. Terms are built up from variables and constants using function symbols or operations. Two of the most important properties of term rewriting systems are *termination* and *confluence*. The former is satisfied when every term in the system has at least one normal form, i.e. an expression to which no more rules apply. The latter holds if the terms can be rewritten in more than one way, to yield the same result. This formalism has been proven to be particularly suited for tasks like symbolic computation, program analysis and program transformation. A comprehensive introduction to the field is given in [3]. A generalisation of term rewriting is the theory of *graph rewriting* [28] in which terms are represented as finite DAGs labelled over a signature and rewriting is implemented by graph transformation. It has been successfully applied to various problems in software engineering and for the implementation of functional languages

and database systems [29]. Categorical characterisations of term and graph rewriting were given in [60, 22].

Bigraphs

Bigraphs are a mathematical model for systems of interacting agents (real or virtual) introduced by Milner [47]. A bigraph consists of two independent structures: a set of nodes that can be nested one inside another, and a set of hyper-edges linking the nodes. The intended interpretation is that nodes represent *locality*, i.e. the spatial placement of agents, while links encode *connectivity*, i.e. their communication capabilities. The definition of bigraphs was partly inspired by the formalisms described above. In particular, the link structure recalls channels in the π-calculus, and the tree structure induced by the nesting of nodes is reminiscent of the location hierarchy of mobile ambients and the concrete syntax tree of terms. Another similarity with term rewriting is a formulation of the formalism within the framework of category theory. Bigraphs admit both an algebraic and an equivalent graphical representation. Their behaviour is specified by a set of *reaction rules*. In this case, the analogy is with rewrite rules in term rewriting and in process calculi equipped with reduction semantics. The two principal motivations that led to the development of bigraphs are:

- to model directly ubiquitous systems by focusing on mobile connectivity and mobile locality;
- to provide a unification of existing theories by developing a general theory in which many existing calculi for concurrency and mobility may be represented, with a uniform behavioural theory.

From trees to graphs

Historically, the idea of using graphs instead of trees has emerged in several fields of computer science. An example is the adoption of *ordered binary decision trees* (OBDDs) [14] for the representation of boolean formulae in the context of model checking. In this approach, the DAG data structure implementing an OBDD is obtained by merging the isomorphic subtrees occurring in the equivalent tree encoding of a given boolean formula. Another example is the representation of terms by means of DAGs in graph term rewriting. Here, nodes with more than one parent are used to encode shared sub-terms explicitly. In both cases, the motivation for moving away from trees was to reduce the size of the data structure storing in order to speed up the verification and rewrite process, respectively. To the best of our knowledge, we are not aware of any applications of graphs for the direct modelling of space as demanded by the case studies analysed in Section 1.1.

1.3 Thesis statement

Bigraphs are a fully graphical process algebraic formalism, capable of representing both the position in space of agents and their inter-connections. However, they assume a topology based on sets of trees and thus cannot represent spatial locations that are shared among several entities in a simple or intuitive way. This is a problem, because shared locations are often a requirement, for example, when modelling scenarios in the physical world or in modern complex computer systems such as wireless networks and spatial-aware applications in ubiquitous computing.

We propose bigraphs with sharing, a generalisation of the original definition of bigraphs, to allow for overlapping topologies. The new locality model is based on directed acyclic graphs.

We demonstrate the new formalism can be defined in the general framework of bigraphical theories and wide reactive systems, as originally devised by Milner. We do so by defining a categorical interpretation of bigraphs with sharing, an axiomatisation derived from the equations of a bialgebra over finite ordinals, and a normal form to express bigraphical terms. We illustrate how sharing is essential for modelling overlapping localities by presenting two example case studies in the field of wireless networking. We show that bigraphs with sharing can be used realistically in a production environment by describing the implementation of an efficient matching algorithm and a software tool for the definition, simulation, visualisation and analysis of bigraphical reactive systems.

1.4 Overview of the thesis

This thesis is arranged into two distinct parts: Part I presents the theory of bigraphs with sharing; Part II investigates their adoption as a practical modelling tool. The subsequent chapters are organised in the following way.

In Part I, Chapter 2 introduces Milner's definition of bigraphs, together with the operations that build them, i.e. composition and tensor product. First, it defines a graphical notation for bigraphs and an algebra for bigraphical terms. Second, it presents the dynamics of bigraphs based on the notion of Bigraphical Reactive System (BRS). Third, it introduces the various kinds of category used to develop the theory of bigraphs. Finally, it discusses the spatial logic for bigraphs, **BiLog**, and other extensions of the basic formalism such as stochastic bigraphs.

Chapter 3 formally defines bigraphs with sharing. First, it analyses the motivations behind the development of a new locality concept in which space is represented by directed acyclic graphs. Second, it introduces an unambiguous graphical notation for bigraphs with

sharing, an algebraic axiomatisation and a normal form. Finally, it shows how bigraphs with sharing fit in to the general categorical interpretation of Milner's bigraphs.

Chapter 4 concludes the first part of the thesis. It defines an efficient matching algorithm for bigraphs with sharing based on a reduction to the sub-graph isomorphism problem. It also contains proofs of soundness and completeness of the matching algorithm.

In Part II, Chapter 5 describes BigraphER, an implementation of BRS that natively supports place graphs with sharing. In particular, it analyses the manipulation and visualisation routines, the matching engine based on a SAT encoding of the matching algorithm defined in Chapter 4, and the rewriting engine for the computation of a reaction relation in a BRS. It also introduces a method based on matching for the verification of a decidable fragment of **BiLog**.

Chapter 6 tests the adequacy of bigraphs with sharing by presenting a model of a non-trivial communication protocol for wireless networks that supports arbitrary topologies. The idea of the model is to represent overlapping wireless signals as shared nodes and the various phases of the protocol with stochastic reaction rules organised into priority classes. Quantitative analysis is carried out by using a probabilistic model checker Prism.

Chapter 7 reports on an application of bigraphs with sharing for real-time verification of domestic wireless network management. First, it describes how network topologies are modelled as sorted bigraphs and how network events such as moving in and out of the router's range, and granting and revoking of DHCP leases are encoded as reaction rules. Second, it defines reaction rules to represent *enforce/drop policy* events. Finally, it discusses the rôle of **BiLog** predicates in the analysis of network configurations and compliance with policies.

Chapter 8 concludes the thesis. It contains a summary of the previous chapters and suggests some directions for future development of bigraphs with sharing, both in practice and theory.

Appendix A serves as a reference on category theory. All the categorical concepts utilised in the rest of the thesis (mainly in chapters 2 and 3) are summarised and defined here.

Appendix B formally defines Continuous Time Markov Chains (CTMCs) and the syntax and semantics of the logic **CSL** (Continuous Stochastic Logic).

Appendix C contains algebraic definitions of all the bigraphs and bigraphical rules used in the model of the communication protocol described in Chapter 6.

Appendix D shows an example of event-driven generation of bigraphical models of the current configuration of a domestic wireless network, as specified in Chapter 7. The chapter contains both graphical and algebraic definitions of the reaction rules.

1.5 Contribution

This thesis makes the following main contributions:

- The definition of bigraphs with sharing, a novel generalisation of Milner's bigraphs in which locality is modelled by directed acyclic graphs instead of forests. This includes a formal definition of place graphs with sharing and their operations (i.e. composition and tensor product), an unambiguous graphical notation that allows for an explicit representation of shared nodes, and an axiomatisation equipped with a normal form to express algebraically bigraphical terms. It is also shown how the new formalism fits the general categorical interpretation of bigraphs based on symmetric (partial) monoidal categories.

- The definition of a graph theoretic matching algorithm for bigraphs with sharing that is based on a reduction to the sub-graph isomorphism problem. The new algorithm is proven sound and complete.

- A prototype implementation of BRS with sharing. This consists of an OCaml library and a command-line tool that provide a matching engine based on an efficient SAT encoding of the matching algorithm, a rewriting engine for the computation of the reaction relation and the state space of a BRS, and a visualisation component for the automatic generation of the graphical representation of a bigraph. The software also supports rule priorities and stochastic reaction rules.

- A reasoning technique for a class of **BiLog** predicates based on bigraph matching.

- Two example applications of bigraphs with sharing in real-world scenarios that highlight how the new formalism facilitates the specification of complex yet compact models. Three different strategies are employed to obtain fruitful and efficient models. First, rule priorities and instantaneous rules are used to discard intermediate interleavings that are confluent, thus considerably reducing the state space of a BRS. Second, sequences of rule application to tag/untag entities offer an elegant way to overcome the limitation of matching. They also allow us to control the number of times a rule is applied and to verify predicates involving universal quantifiers or negated existential quantifiers. A further benefit includes avoiding the introduction of duplicates, i.e. nodes of the same control. This permits us to effectively consider unique controls as node identifiers and track nodes through reaction. Third, the use of non parameterised reaction rules leads to instances of matching that are solvable in polynomial time.

1.6 Publications

Some of the material in this thesis has been previously reported in the following papers:

- Many of the definitions and proofs in Chaper 3 appear in [56].
- A preliminary version of the matching algorithm described in Chapter 4 and the implementation discussed in Chapter 5 appear in [57].
- The application of real-time verification of domestic wireless network management given in Chapter 7 and Appendix D appears in [15].

Part I

Bigraphs with sharing

Chapter 2

Background: bigraphs

This chapter summarises the state of the art in bigraphs. The main reference is Milner's text [47].

In Section 2.1, we give an informal introduction to the formalism, we introduce the graphical notation for the representation of bigraphs and we set some conventions and terminology. In Section 2.2 bigraphs and their constituents, namely place and link graphs, are formally defined. Section 2.3 is devoted to the definition of the fundamental operations on bigraphs, i.e. composition and tensor product. Section 2.4 introduces an algebra for bigraphical terms, while a typing discipline on bigraphs, called sorting, is presented in Section 2.5. The part of the theory dealing with the dynamic evolution of bigraphs is in Section 2.6, where Bigraphical Reactive Systems are defined. An overview of the spatial logic for bigraphs, **BiLog**, is given in Section 2.7. Section 2.8 briefly summarises how bigraphs can be defined in the general setting of category theory. Stochastic bigraphs are presented in Section 2.9. Other extensions of the basic formalism and known software tools for the manipulation of bigraphs and Bigraphical Reactive Systems are discussed in Section 2.10. Finally, some concluding remarks are given in Section 2.11.

2.1 Introduction

Bigraphs are a recent formalism conceived for modelling agents, their spatial arrangement and their inter-connections. A bigraph has a graphical form and an equivalent algebraic form. For now, we focus on the former, since it allows for a more intuitive description of the formalism.

Let us begin by considering example bigraph B shown Figure 2.1. In the graphical form, agents, or entities (real or virtual) are encoded by *nodes*, represented as ovals and circles. Their spatial placement is described by node nesting. Nodes are assigned a type,

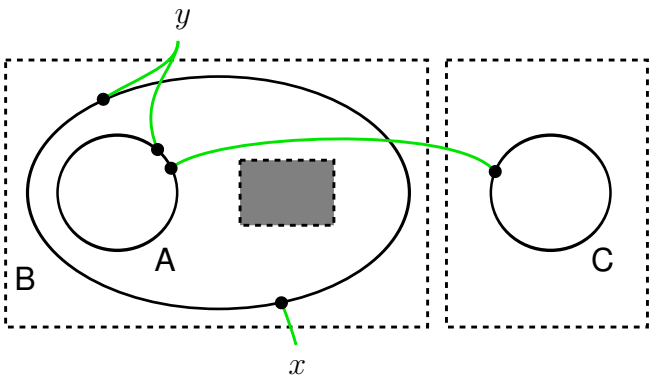
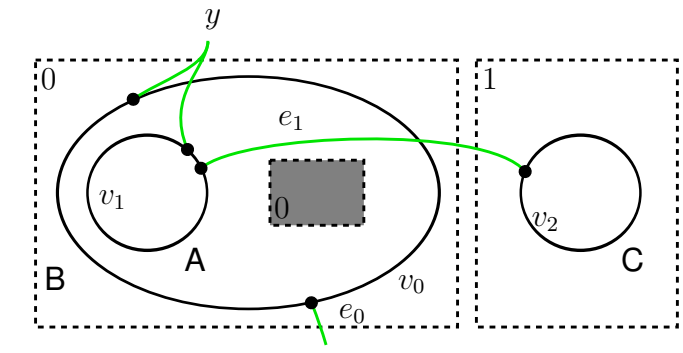


Figure 2.1: Example bigraph B.

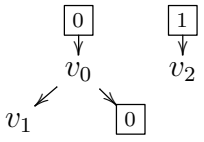
called *control*, denoted here by the labels A, B and C. The set of controls of a bigraphs is called the *signature*. A node is said to be *atomic* if it does not contain any sites or nodes. Interactions between agents are represented by *links* like, for instance, the edge connecting the A-node and the C-node. Each node can have zero, one or many *ports*, indicated by bullets. They can be thought of as sockets into which links can be plugged. Observe that nodes of the same control have also the same number of ports. Dashed rectangles denote *regions* (sometimes called *roots*). The rôle of a region is to describe adjacent parts of the system. Grey squares are called *sites*. They encode parts of the model that have been abstracted away. Regions and sites are indexed by natural numbers (starting from 0) form left to right. These are crucial in the definition of composition for bigraphs, as we will see in greater detail in Section 2.3. Nodes, sites and roots are the *places* of a bigraph. Note that there is no significance in where a link crosses the boundary of a place in a bigraph. A bigraph can have *inner names* and *outer names*. In our example, y is an outer name while x is an inner name. By convention in the graphical form, inner names and outer names are drawn below and above the bigraph, respectively. They encode links (or potential links) to other bigraphs representing the external environment or context. Inner names and ports are the *points* of a bigraph. A link is said to be *idle* when it has no points. Similarly, a place is idle when it does not contain any nodes or sites. Two places with the same parent, or two points with the same link, are called *siblings*.

The capabilities of a bigraph to interact with the external environment are recorded in its *interface*. For example, we write $B : \langle 1, \{x\}\rangle \to \langle 2, \{y\}\rangle$ to indicate that B has one site, two regions and its inner and outer names are sets of names $\{x\}$ and $\{y\}$, respectively. Pair $\langle 1, \{x\}\rangle$ is called the *inner face* of B, while $\langle 2, \{y\}\rangle$ is its *outer face*. We will see in greater detail in Section 2.3 how interfaces allow bigraphs to be composed and how they allow one bigraph to be considered as a component of another bigraph.

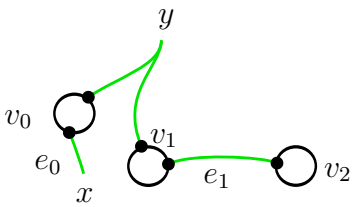
Elements forming a bigraphs, namely nodes and edges, can be assigned unique identifiers, collectively called the *support* of a bigraph. A bigraph with identifiers is said to be *concrete*. An example concrete bigraph, $\widetilde{B}$, is drawn in Figure 2.2a. Observe that $\widetilde{B}$ corres-



(a) $\widetilde{B} : \langle 1, \{x\}\rangle \to \langle 2, \{y\}\rangle$



(b) $\widetilde{B}^{\mathsf{P}} : 1 \to 2$



(c) $\widetilde{B}^{\mathsf{L}} : \{x\} \to \{y\}$

Figure 2.2: Concrete bigraph $\widetilde{B}$ and its constituents: concrete place graph $\widetilde{B}^{\mathsf{P}}$ and concrete link graph $\widetilde{B}^{\mathsf{L}}$.

ponds to bigraph B given in Figure 2.1 with the addition of identifiers. Nodes are indicated with v_0, v_1 and v_2, while edges are e_0 and e_1.

Summarising, *locality* is represented by node placement while *connectivity* is encoded by the links of the bigraph. This characteristic can be made explicit by defining bigraphs in terms of the constituent notions of *place graph* and *link graph*. A place graph is a structure defined over the places of a bigraph able to capture their nesting. More precisely, the structure is a forest whose roots are the regions of the corresponding bigraph and leaves are its sites and atomic nodes. A link graph is a structure describing the linkage of a bigraph. It consists of a hyper-graph whose vertices are the names and nodes of the corresponding bigraph and hyper-edges are its links. The place graph $\widetilde{B}^{\mathsf{P}}$ and link graph $\widetilde{B}^{\mathsf{L}}$ for concrete bigraph $\widetilde{B}$ are shown in Figure 2.2[1]. The two structures are independent and they only share the set of nodes. This allows to define distinct interfaces for place graphs and link graphs. In the example, $\widetilde{B}^{\mathsf{P}} : 1 \to 2$ and $\widetilde{B}^{\mathsf{L}} : \{x\} \to \{y\}$.

Notation and conventions

We frequently interpret a natural number as a finite ordinal, namely $m = \{0, 1, \ldots, m-1\}$. We write $S \uplus T$ to indicate the union of sets known or assumed to be disjoint. If a function f has domain S and $S' \subseteq S$, then $f \restriction S'$ denotes the restriction of f to S'. For two functions

[1]Controls are omitted.

f and g with disjoint domains S and T we write $f \uplus g$ for the function with domain $S \uplus T$ such that $(f \uplus g) \restriction S = f$ and $(f \uplus g) \restriction T = g$. We write Id_S for the identity function on the set S.

In defining bigraphs we assume that names, node-identifiers and edge-identifiers are drawn from three infinite sets, respectively $\mathcal{X}$, $\mathcal{V}$ and $\mathcal{E}$, disjoint from each other. We denote identifiers by lower-case letter: x, y, z for names, v, u for nodes, and $e_0, e_1, \ldots$ for edges. Upper-case letters $A, B, \ldots$ are used to denote bigraphs and their constituents. Concrete bigraphs are usually indicated with $\widetilde{A}, \widetilde{B}, \ldots$. We call the trivial interface $\epsilon \stackrel{\text{def}}{=} \langle 0, \emptyset \rangle$ the *origin*.

2.2 Constituents of bigraphs

As mentioned previously, every bigraph consists of two orthogonal structures: a place graph specifying the spatial relation among its entities and a link graph defining their connectivity. We begin by defining signatures.

Definition 2.2.1 (signature)**.** A *signature* takes the form $(\mathcal{K}, ar)$. It has a set $\mathcal{K}$ whose elements are controls, and map $ar : \mathcal{K} \to \mathbb{N}$ assigning an *arity* to each control. A bigraph over $\mathcal{K}$ assigns to each node a control, whose arity indexes the ports of a node.

A signature suitable for the examples in Figures 2.1 and 2.2a is $\mathcal{K} = \{\mathsf{A} : 2, \mathsf{B} : 2, \mathsf{C} : 1\}$. Now, we are ready to define formally place graphs and link graphs.

Definition 2.2.2 (concrete place graph)**.** A *concrete place graph*

$$F = (V_F, ctrl_F, prnt_F) : m \to n$$

is a triple having an inner face m and an outer face n. These index respectively the sites and roots of the place graph. F has a finite set $V_F \subset \mathcal{V}$ of nodes, a *control map* $ctrl_F : V_F \to \mathcal{K}$, and a *parent map*

$$prnt_F : m \uplus V_F \to V_F \uplus n$$

which is acyclic, i.e. if $prnt_F^i(v) = v$ for some $v \in V_F$ then $i = 0$.

The defining triple for example place graph $\widetilde{B}^{\mathsf{P}}$ drawn in Figure 2.2b is given by

$$(V, ctrl, prnt) = (\{v_0, v_1, v_2\}, \{v_0 : \mathsf{B}, v_1 : \mathsf{A}, v_2 : \mathsf{C}\}, \{(0, v_0), (v_1, v_0), (v_0, 0), (v_2, 1)\}) \ .$$

Definition 2.2.3 (concrete link graph)**.** A *concrete link graph*

$$F = (V_F, E_F, ctrl_F, link_F) : X \to Y$$

is a quadruple having an inner face X and an outer face Y, both finite subsets of $\mathcal{X}$, called respectively the *inner* and *outer names* of the link graph. F has finite sets $V_F \subset \mathcal{V}$ of *nodes* and $E_F \subset \mathcal{E}$ of *edges*, a *control map* $ctrl_F : V_F \to \mathcal{K}$ and a *link map*

$$link_F : X \uplus P_F \to E_F \uplus Y$$

where $P_F \stackrel{\text{def}}{=} \{(v,i) | i \in ar(ctrl_F(v))\}$ is the set of *ports* of F. Thus (v,i) is the ith port of node v. We shall call $X \uplus P_F$ the points of F, and $E_F \uplus Y$ its *links*.

The edges and link map for example link graph $\widetilde{B}^{\mathsf{L}}$ in Figure 2.2c are $E = \{e_0, e_1\}$ and

$$link = \{(x, e_0), ((v_0, 0), e_0)), ((v_1, 0), e_1), ((v_2, 0), e_1), ((v_0, 1), y), ((v_1, 1), y)\} .$$

A concrete bigraph simply consists of a concrete place graph and a concrete link graph.

Definition 2.2.4 (concrete bigraph)**.** A *concrete bigraph*

$$F = (V_F, E_F, ctrl_F, prnt_F, link_F) : \langle k, X\rangle \to \langle m, Y\rangle$$

consists of a concrete place graph $F^{\mathsf{P}} = (V_F, ctrl_F, prnt_F) : k \to m$ and a concrete link graph $F^{\mathsf{L}} = (V_F, E_F, ctrl_F, link_F) : X \to Y$. We write the concrete bigraph as $F = \langle F^{\mathsf{P}}, F^{\mathsf{L}}\rangle$.

Diagrams for example bigraph $\widetilde{B} : \langle 1, \{x\}\rangle \to \langle 2, \{y\}\rangle$ and its constituents $\widetilde{B}^{\mathsf{P}} : 1 \to 2$ and $\widetilde{B}^{\mathsf{L}}\{x\} \to \{y\}$ are given in Figure 2.2.

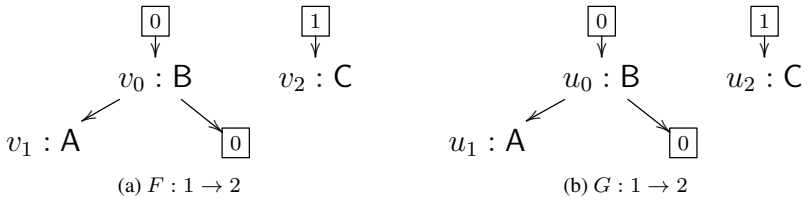
The structures introduced above are called concrete because their nodes and edges are uniquely identified by members of $\mathcal{V}$ and $\mathcal{E}$. Formally, given a concrete bigraph F, its support is $|F| = V_F \uplus E_F$. The identifiers of nodes and edges can be varied in a disciplined way by means of a *support translation*.

Definition 2.2.5 (support translation)**.** Given two bigraphs $F, G : \langle k, X\rangle \to \langle m, Y\rangle$, a support translation $\rho : |F| \to |G|$ from F to G consists of a pair of bijections $\rho_V : V_F \to V_G$ and $\rho_E : E_F \to E_G$ that respect structure in the following sense:

1. ρ preserves controls, i.e. $ctrl_G \circ \rho_V = ctrl_F$. It follows that ρ induces a bijection $\rho_P : P_F \to P_G$ on ports, defined by $\rho_P((v,i)) \stackrel{\text{def}}{=} (\rho_V(v), i)$.

2. ρ commutes with the structural maps as follows:

$$prnt_G \circ (\mathsf{Id}_m \uplus \rho_V) = (\mathsf{Id}_n \uplus \rho_V) \circ prnt_F$$
$$link_G \circ (\mathsf{Id}_X \uplus \rho_P) = (\mathsf{Id}_Y \uplus \rho_E) \circ link_F.$$



(a) $F : 1 \to 2$ (b) $G : 1 \to 2$

Figure 2.3: Support equivalent place graphs.

Given F and the bijection ρ, these conditions uniquely determine G. We therefore denote G by $\rho{\cdot}F$, and call it the *support translation of F by ρ*. We call F and G *support equivalent*, and we write $F \eqsim G$, if such a translation exists. Support translation is defined similarly for place graphs and link graphs.

An example of support equivalent place graphs is given in Figure 2.3. The support translation can be made explicit by writing $G = \rho{\cdot}F$, where $\rho(v_i) = u_i$ with $i \in \{0, 1, 2\}$.

Support is essential for the formal definition of bigraphs and their operations, as will become clear in Section 2.3. However, in most applications, support is irrelevant and thus it is often dropped. This is because one wishes to abstract and not to regard support-equivalent bigraphs as different. Formally, *abstract* bigraphs, i.e. without identifiers, can be interpreted as equivalence classes of support equivalent bigraphs. An example abstract bigraph, B, is shown in Figure 2.1. It represents the class of concrete bigraphs obtainable by applying any valid support translation to concrete bigraph $\widetilde{B}$ drawn in Figure 2.2a. One technicality arises when bigraphs have idle edges. Therefore, we need to quotient by a slighter larger equivalence, as follows:

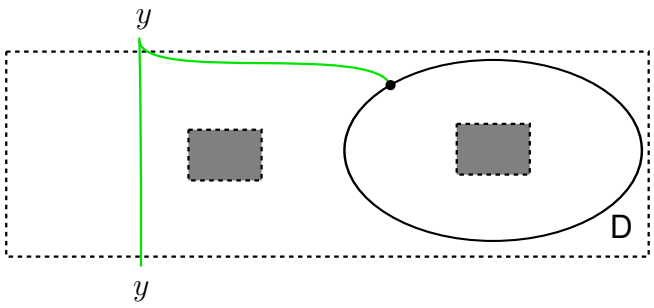
Definition 2.2.6 (leanness)**.** A bigraph is *lean* if it has no idle edges. Two bigraphs F and G are *lean-support equivalent*, written $F \eqsim G$, if they are support-equivalent after discarding any idle edges.

Definition 2.2.7 (abstract bigraph, abstraction)**.** An abstract bigraph $B : \langle k, X\rangle \to \langle m, Y\rangle$ consists of a $\eqsim$-equivalence class of concrete bigraphs: $B = [\widetilde{F}]_{\eqsim}$ with $\widetilde{F} : \langle k, X\rangle \to \langle m, Y\rangle$ a concrete bigraph. Bigraph B is also called the *abstraction* of $\widetilde{F}$.

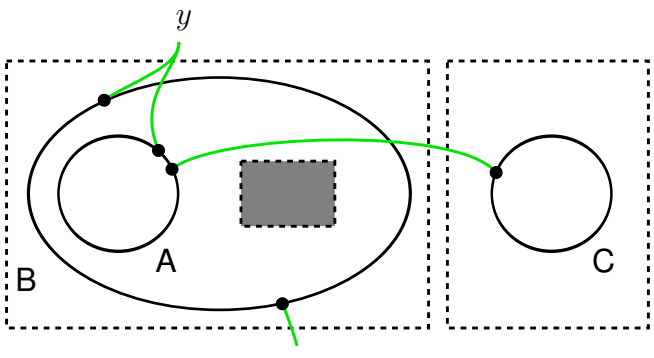
Sometimes it is also useful to move in the opposite direction, namely from abstract bigraphs to concrete ones:

Definition 2.2.8 (Concretion)**.** If G is an abstract bigraph, then a concrete bigraph $\widetilde{G}$, called a *concretion* of G, is obtained by assigning to each node a unique identifier $v \in \mathcal{V}$ and to each edge a unique identifier $e \in \mathcal{E}$.

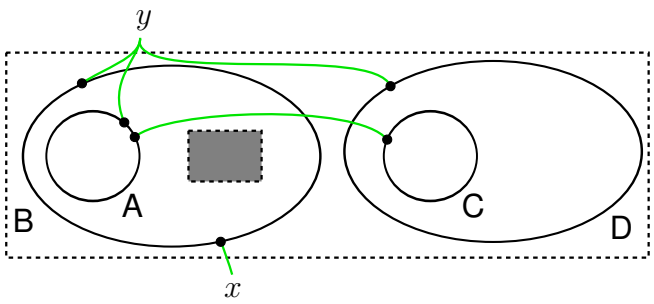
Observe that any two concretions $\widetilde{A}, \widetilde{B}$ of the same bigraph are always support equivalent.



(a) $G : \langle 2, \{y\}\rangle \to \langle 1, \{y\}\rangle$



(b) $B : \langle 1, \{x\}\rangle \to \langle 2, \{y\}\rangle$



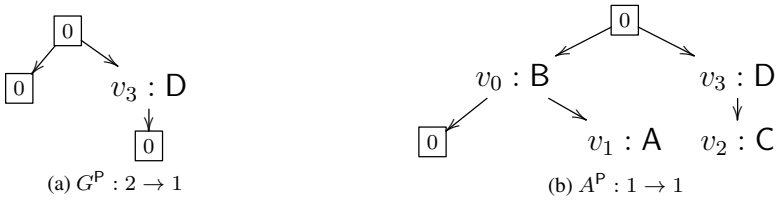
(c) $A : \langle 1, \{x\}\rangle \to \langle 1, \{y\}\rangle$

Figure 2.4: Composition for bigraphs: $A = G \circ B$.

2.3 Operations on bigraphs

We can now describe how to make larger bigraphs from smaller ones. An example of composition is shown in Figure 2.4c: bigraph A is the result of the composition of bigraphs G and B, given in Figures 2.4a and 2.4b, respectively.

Algorithmically, we can think of composition as placing one bigraph, e.g. B, in the context represented by another, e.g. G. For this we require the outer face of B to equal the inner face of G. We write $G \circ B$ to indicate composition, i.e. that bigraph B is inserted into bigraph G. In more detail, when a bigraph is inserted into another, its outer names are merged with the corresponding inner names of the host bigraph, and its roots are merged with the corresponding sites of the host bigraph. In the example, the root in B containing the C-node is merged with the site inside the D-node in G. Moreover, the D-node is linked



(a) $G^{\mathsf{P}} : 2 \to 1$ (b) $A^{\mathsf{P}} : 1 \to 1$

Figure 2.5: Composition for place graphs: $A^{\mathsf{P}} = G^{\mathsf{P}} \circ \widetilde{B}^{\mathsf{P}}$.

over common name y to nodes of control A and B. Sometimes G is called the *context* or the *environment* for B.

We now define separately composition for concrete place graphs and link graphs.

Definition 2.3.1 (composition for concrete place graphs)**.** If $F : k \to m$ and $G : m \to n$ are two concrete place graphs with disjoint supports, their composite

$$G \circ F = (V, ctrl, prnt) : k \to n$$

has nodes $V = V_F \uplus V_G$ and control map $ctrl = ctrl_F \uplus ctrl_G$. Its parent map $prnt$ is defined as follows: If $w \in k \uplus V$ is a site or a node in $G \circ F$ then

$$prnt(w) \stackrel{\text{def}}{=} \begin{cases} prnt_F(w) & \text{if } w \in k \uplus V_F \text{ and } prnt_F(w) \in V_F, \\ prnt_G(j) & \text{if } w \in k \uplus V_F \text{ and } prnt_F(w) = j \in m, \\ prnt_G(w) & \text{if } w \in V_G\ . \end{cases}$$

An example of composition for place graphs is given in Figure 2.5. Note that place graph $\widetilde{B}^{\mathsf{P}}$ is drawn in Figure 2.2b.

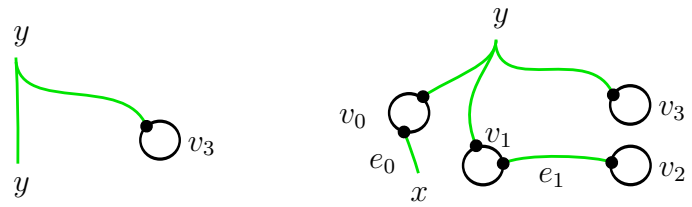
Definition 2.3.2 (composition for concrete link graphs)**.** If $F : X \to Y$ and $G : Y \to Z$ are two concrete link graphs with disjoint support, their composite

$$G \circ F = (V, E, ctrl, link) : X \to Z$$

has $V = V_F \uplus V_G$, $E = E_F \uplus E_G$, $ctrl = ctrl_F \uplus ctrl_G$. Its link map $link$ is defined as follows: If $q \in X \uplus P_F \uplus P_G$ is a point of $G \circ F$ then

$$link(q) \stackrel{\text{def}}{=} \begin{cases} link_F(q) & \text{if } q \in X \uplus P_F \text{ and } link_F(q) \in E_F, \\ link_G(y) & \text{if } q \in X \uplus P_F \text{ and } link_F(q) = y \in Y, \\ link_G(q) & \text{if } q \in P_G\ . \end{cases}$$

An example of composition for link graphs is in Figure 2.6. Link graph $\widetilde{B}^{\mathsf{L}}$ is depicted in Figure 2.2c.



(a) $G^{\mathsf{L}} : \{y\} \to \{y\}$ (b) $A^{\mathsf{L}} : \{x\} \to \{y\}$

Figure 2.6: Composition for concrete link graphs: $A^{\mathsf{L}} = G^{\mathsf{L}} \circ \widetilde{B}^{\mathsf{L}}$.

By combination of the previous two definitions, it is possible to define composition for concrete bigraphs.

Definition 2.3.3 (composition for concrete bigraphs)**.** If $F : \langle k, X\rangle \to \langle m, Y\rangle$ and $G : \langle m, Y\rangle \to \langle n, Z\rangle$ are two concrete bigraphs with disjoint supports, their composite is

$$G \circ F \stackrel{\text{def}}{=} \langle G^{\mathsf{P}} \circ F^{\mathsf{P}}, G^{\mathsf{L}} \circ F^{\mathsf{L}}\rangle : \langle k, X\rangle \to \langle n, Z\rangle \ .$$

Identities are node-free elementary bigraphs which are neutral for the composition operation:

Definition 2.3.4 (identities)**.** Identities at ordinal m, set of names X and interface $\langle m, X\rangle$ are given by

$$\begin{aligned} \mathsf{id}_m &= (\emptyset, \emptyset, \mathsf{Id}_m) : m \to m \ , \\ \mathsf{id}_X &= (\emptyset, \emptyset, \emptyset, \mathsf{Id}_X) : X \to X \ , \\ \mathsf{id}_{\langle m,X\rangle} &= \langle \mathsf{id}_m, \mathsf{id}_X\rangle : \langle m, X\rangle \to \langle m, X\rangle \ , \end{aligned}$$

respectively.

The other fundamental operation on bigraphs is *tensor product*, denoted by $\otimes$. It is only defined over bigraphs with disjoint interfaces and disjoint supports and it consists of putting two bigraphs side-by-side. More precisely, we say that two bigraphs $F_i : \langle m_i, X_i\rangle \to \langle n_i, Y_i\rangle$, with $(i = 0, 1)$, have disjoint interfaces if $X_0 \cap X_1 = \emptyset$ and $Y_0 \cap Y_1 = \emptyset$. Tensor product over interfaces $I = \langle m, X\rangle$ and $J = \langle n, Y\rangle$ is defined as $I \otimes J = \langle m + n, X \uplus Y\rangle$. As for composition, we first define tensor product independently for concrete place and link graphs.

Definition 2.3.5 (tensor for concrete place graphs)**.** If $F_i = (V_i, ctrl_i, prnt_i) : m_i \to n_i$ are disjoint place graphs $(i = 0, 1)$ their tensor product $F_0 \otimes F_1 : m_0 + m_1 \to n_0 + n_1$ is given by

$$F_0 \otimes F_1 \stackrel{\text{def}}{=} (V_0 \uplus V_1, ctrl_0 \uplus ctrl_1, prnt_0 \uplus prnt_1') \ ,$$

where $prnt_1'(m_0 + i) = n_0 + j$ whenever $prnt_1(i) = j$.

Definition 2.3.6 (tensor for concrete link graphs)**.** If $F_i = (V_i, E_i, ctrl_i, link_i) : X_i \to Y_i$ are disjoint link graphs $(i = 0, 1)$ their tensor product

$$F_0 \otimes F_1 : X_0 \uplus X_1 \to Y_0 \uplus Y_1$$

is given by

$$F_0 \otimes F_1 \stackrel{\text{def}}{=} (V_0 \uplus V_1, E_0 \uplus E_1, ctrl_0 \uplus ctrl_1, link_0 \uplus link_1) \ .$$

Again, by combining the previous two definitions, we obtain tensor product for concrete bigraphs:

Definition 2.3.7 (tensor for concrete bigraphs)**.** If $F_i : \langle m_i, X_i \rangle \to \langle n_i, Y_i \rangle$ are disjoint bigraphs $(i = 0, 1)$ their tensor product

$$F_0 \otimes F_1 : \langle m_0 + m_1, X_0 \uplus X_1 \rangle \to \langle n_0 + n_1, Y_0 \uplus Y_1 \rangle$$

is defined as

$$F_0 \otimes F_1 \stackrel{\text{def}}{=} \langle {F_0}^{\mathsf{P}} \otimes {F_1}^{\mathsf{P}}, {F_0}^{\mathsf{L}} \otimes {F_1}^{\mathsf{L}} \rangle \ .$$

Identity id_ϵ is neutral for tensor product.

Operations on abstract bigraphs are defined over congruence classes as follows:

Definition 2.3.8 (operations for abstract bigraphs)**.** Operations on abstract bigraphs are defined as operations on the corresponding concretions to which is applied the $\eqcirc$-equivalence afterwards:

- If $[F]_\eqcirc : \langle k, X \rangle \to \langle m, Y \rangle$ and $[G]_\eqcirc : \langle m, Y \rangle \to \langle n, Z \rangle$ are two abstract bigraphs, their composite is

$$[G]_\eqcirc \circ [F]_\eqcirc \stackrel{\text{def}}{=} [G \circ F]_\eqcirc : \langle k, X \rangle \to \langle n, Z \rangle \ .$$

- If $[F_i]_\eqcirc : \langle m_i, X_i \rangle \to \langle n_i, Y_i \rangle$ are two abstract bigraphs $(i = 0, 1)$ with disjoint interfaces, their tensor product is

$$[F_0]_\eqcirc \otimes [F_1]_\eqcirc \stackrel{\text{def}}{=} [F_0 \otimes F_1]_\eqcirc : \langle m_0 + m_1, X_0 \uplus X_1 \rangle \to \langle n_0 + n_1, Y_0 \uplus Y_1 \rangle \ .$$

Finally, sub-components of a bigraph are specified as follows:

Definition 2.3.9 (occurrence, matching)**.** A bigraph F *occurs* in a bigraph G if the equation $G = C_1 \circ (F \otimes \mathsf{id}_I) \circ C_0$ holds for some interface I and bigraphs C_0 and C_1. The computational problem of determining whether a given graph occurs in another bigraph is called the

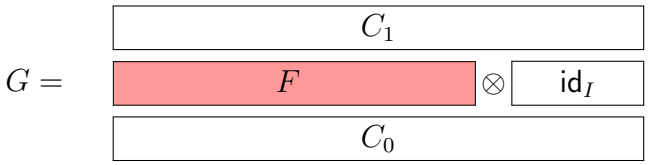


Figure 2.7: Match $G = C_1 \circ (F \otimes \mathsf{id}_I) \circ C_0$.

matching problem (for bigraphs). We say that F *matches* G or F is a *match* in G when F occurs in G.

The decomposition of a bigraph G induced by match F is described graphically in Figure 2.7. Observe that identity id_I is necessary to allow nodes in C_1 to have children also in C_0, and to allow C_1 and C_0 to share links that do not involve F. When considering concrete bigraphs, the supports of G and F have to be compatible in order to find a valid occurrence.

Definition 2.3.10 (concrete occurrence)**.** Let F and G be two concrete bigraphs. We say there is a *concrete occurrence* of F in G if the equation $G = C_1 \circ (F' \otimes \mathsf{id}_I) \circ C_0$ holds for some interface I, and concrete bigraphs F', C_0 and C_1, where $F' \eqsim F$. Two concrete occurrences are equal if they differ only by a permutation or a bijective renaming on the inner interface of C_1 and the outer interface of C_0.

An important property is that it is possible to determine an (abstract) occurrence starting from a concrete one. In other words, a bigraph F occurs in G only if an arbitrary concretion of F occurs in an arbitrary concretion of G. In both the abstract and concrete case, a match F may induce more than one decomposition of G. Since composition and tensor product are defined independently over the constituents of a bigraphs, the previous two definitions can be reformulated by using place graphs and link graphs only, i.e. F occurs in G only if F^{P} occurs in G^{P} and F^{L} occurs in G^{L}.

2.4 Algebraic form

All bigraphs can be expressed in terms of elementary bigraphs by means of composition and tensor product. This allows us to represent bigraphical terms structurally as in process algebra. The algebraic structure we will introduce in the following specifies abstract bigraphs.

A bigraph is called *discrete* if it has no closed links, i.e. edges, and its link map is bijective. A bigraph is *prime* if it has no inner names and an outer face in the form $\langle 1, X \rangle$. These two kinds of bigraph are important for the algebraic structure of bigraphs and will be used in Section 2.4.2 to define a normal form. We denote the interfaces of bigraphs by I, J, K. When there is no ambiguity, we shall often write a name set $\{x, y, z, \dots\}$ as $\{xyz \cdots\}$ or

(a) $\gamma_{1,1} : 2 \to 2$ (b) $1 : 0 \to 1$ (c) $join : 2 \to 1$

Figure 2.8: Elementary placings. Symmetry $\gamma_{1,1}$ in (a) swaps the order of two sites, bigraph 1 in (b) is an idle root and bigraph $join$ in (c) is a root with two sites.

even abbreviate $\{x\}$ to x. We write n to indicate interface $\langle n, \emptyset \rangle$ and Y for $\langle 0, Y \rangle$. We write $\bigotimes_{i<n} F_i$ for the iterated tensor product $F_0 \otimes \cdots \otimes F_{n-1}$. This is id_ϵ when $n = 0$. We sometimes write $F_0 F_1$ for composition, letting it bind tighter than tensor product. All operations are assumed to be defined.

2.4.1 Axioms and elementary Bigraphs

Basic axioms for a theory of bigraphical terms are induced from the definitions of composition and tensor product:

$$\begin{aligned} A \circ \mathsf{id}_X = A = \mathsf{id}_Y \circ A \qquad & A : X \to Y \\ A \circ (B \circ C) = (A \circ B) \circ C & \\ A \otimes \mathsf{id}_\epsilon = A = \mathsf{id}_\epsilon \otimes A & \\ \mathsf{id}_I \otimes \mathsf{id}_J = \mathsf{id}_{I \otimes J} & \\ A \otimes (B \otimes C) = (A \otimes B) \otimes C & \\ (A_0 \otimes B_0) \circ (A_1 \otimes B_1) = (A_0 \circ A_1) \otimes (B_0 \circ B_1) & \end{aligned} \tag{2.1}$$

They formalise the rôle of identities and associativity of $\circ$ and $\otimes$, and the interplay between the two operators.

Elementary place graphs are identities $\mathsf{id}_n : n \to n$, *symmetries* $\gamma_{m,n} : m + n \to n + m$, $1 : 0 \to 1$ and $join : 2 \to 1$. The corresponding diagrams are given in Figure 2.8. Node-free bigraphs with no links are called *placings*. They can all be built from the elementary place graphs. A placing that is bijective from sites to roots is a *permutation*. We shall use ϕ and π to denote placings and permutations, respectively. It is useful to indicate a placing with one root and n sites by $merge_n$. Note that $merge_0 = 1$, $merge_1 = \mathsf{id}_1$ and $merge_2 = join$. Axioms for placing are:

$$\begin{aligned} join \circ (1 \otimes \mathsf{id}_1) &= \mathsf{id}_1 \\ join \circ (join \otimes \mathsf{id}_1) &= join \circ (\mathsf{id}_1 \otimes join) \\ join \circ \gamma_{1,1} &= join \end{aligned} \tag{2.2}$$

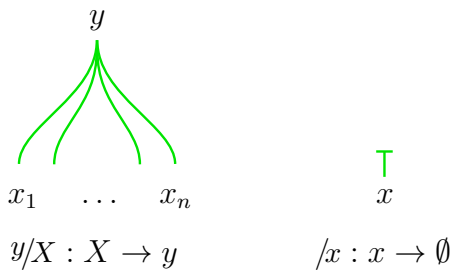


Figure 2.9: Elementary link graphs.

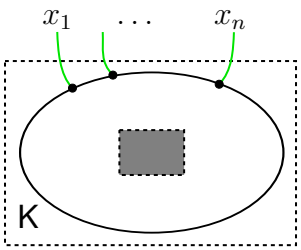


Figure 2.10: Ion $\mathsf{K}_{\vec{x}}$.

They say that $(1 = \{0\}, join : 2 \to 1, 1 : 0 \to 1)$ is a symmetric monoid[2] in which placings $join$ and 1 are the *multiplication* and the *unit*, respectively.

Elementary link graphs are *substitutions* $y/X : X \to y$ and *closures* $/x : x \to \emptyset$ as shown in Figure 2.9. A node-free bigraph with no places is a *linking*. Linkings are generated by composition, tensor product and identities from substitutions and closures. A bijective substitution is called a *renaming*. We use λ, σ, α to denote linkings, substitutions and renamings, respectively. We also indicate the empty substitution from $\emptyset$ to x by $x : \emptyset \to x$. Axioms for linkings are:

$$\begin{aligned} x/x &= \mathsf{id}_x \\ /x \circ x &= \mathsf{id}_0 \\ /y \circ y/x &= /x \\ z/Y \uplus y \circ (\mathsf{id}_Y \otimes y/X) &= z/Y \uplus X \ . \end{aligned} \tag{2.3}$$

Nodes are introduced by only one kind of elementary bigraph called *ions*. An example ion of control K is drawn in Figure 2.10. For each control $\mathsf{K} : n$, a ion consists of a single K-node containing a site with ports linked bijectively to n distinct names $\vec{x}$. Notation $\mathsf{K}_{\vec{x}} : \langle 1, \emptyset \rangle \to \langle 1, \{\vec{x}\} \rangle$ expresses this bigraph. There is one node axiom stating that ports can be named arbitrarily:

$$(\mathsf{id}_1 \otimes \alpha) \circ \mathsf{K}_{\vec{x}} = \mathsf{K}_{\alpha(\vec{x})} \tag{2.4}$$

[2]See Definition A.18 in Appendix A

Finally, symmetries can be generalised as follows:

$$\gamma_{\langle m,X\rangle,\langle n,Y\rangle} \stackrel{\text{def}}{=} \gamma_{m,n} \otimes \mathsf{id}_{X \uplus Y} \ . \tag{2.5}$$

Therefore, if we apply a symmetry to a bigraph, then $\gamma \circ G$ reorders the roots of G but leaves its names unchanged. Similarly, if a bigraph is applied to a symmetry, then $G \circ \gamma$ reorders the sites of G. Also in this case, names are unaffected. A set of axioms specifying the properties of symmetries for bigraph is:

$$\begin{aligned} \gamma_{I,\epsilon} &= \mathsf{id}_I \\ \gamma_{J,I} \circ \gamma_{I,J} &= \mathsf{id}_{I \otimes J} \\ \gamma_{I,K} \circ (A \otimes B) &= (B \otimes A) \circ \gamma_{H,J} \qquad\qquad (A : H \to I, B : J \to K) \\ \gamma_{X \otimes Y,Z} &= (\gamma_{X,Z} \otimes \mathsf{id}_Y) \circ (\mathsf{id}_X \otimes \gamma_{Y,Z}) \ . \end{aligned} \tag{2.6}$$

The set of Axioms (2.1) – (2.6) is complete for equations between bigraphical expressions.

2.4.2 Normal form

Bigraphical terms can be decomposed uniquely into sub-terms with some standard properties. This way of expressing bigraphs is a kind of normal form called *discrete normal form* (DNF). We begin by presenting a factorisation for prime and discrete bigraphs.

Proposition 2.4.1. *Every prime and discrete bigraph P can be expressed uniquely, up to permutations, as*

$$\begin{aligned} P &= (merge_{m+n} \otimes \mathsf{id}_Y) \circ (\mathsf{id}_m \otimes \bigotimes_{i<n} M_i) \circ \pi \\ M &= (\mathsf{K}_{\vec{x}} \otimes \mathsf{id}_Y) \circ P \end{aligned}$$

where π is a permutation.

Observe that idle root 1 occurs as a special case of P when $m = n = 0$. Similarly, id_1 occurs when $m = 1$ and $n = 0$. The DNF for arbitrary bigraphs is as follows:

Proposition 2.4.2 (discrete normal form)**.** *Every bigraph $G : \langle m, X\rangle \to \langle n, Z\rangle$ can be expressed uniquely, up to renaming on Y, as*

$$G = (\mathsf{id}_n \otimes \lambda) \circ D$$

where $\lambda : Y \to Z$ is a linking and $D : \langle m, X\rangle \to \langle n, Y\rangle$ is discrete. Further, every discrete

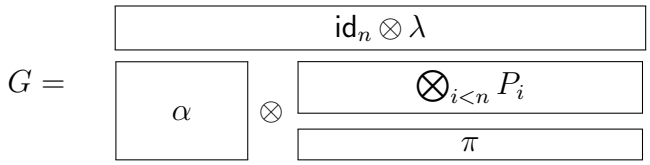


Figure 2.11: DNF for bigraph G.

D may be factored uniquely, up to permutation of the sites of each factor, as

$$D = \alpha \otimes ((\bigotimes_{i<n} P_i) \circ \pi)$$

with α a renaming, each P_i prime and discrete, and π a permutation of all the sites.

We describe DNF graphically in Figure 2.11.

We now show how bigraph $A : \langle 1, x\rangle \to \langle 1, y\rangle$, depicted in Figure 2.4c, can be written in DNF:

$$\begin{aligned}
A &= (\mathsf{id} \otimes \lambda) \circ D \\
\lambda &= y/Y \otimes /Z \otimes /x \qquad\qquad Y = \{y_0, y_1, y_2\}\ Z = \{z_0, z_1\} \\
D &= \mathsf{id}_x \otimes ((P_0 \otimes P_1) \circ \mathsf{id}) \qquad (2.7) \\
P_0 &= (\mathsf{B}_{y_0 x} \otimes \mathsf{id}_{y_1,z_0}) \circ (merge \circ (\mathsf{id} \otimes \mathsf{A}_{y_1,z_0} \circ 1)) \\
P_1 &= (\mathsf{D}_{y_2} \otimes \mathsf{id}_{z_1}) \circ \mathsf{C}_{z_1} \circ 1
\end{aligned}$$

2.4.3 Algebraic operators

Starting from composition and tensor product, it is possible to derive additional operators closely resembling the traditional operators found in process algebra, such as parallel composition in CCS. We will see that this allows for a more natural and succinct specification of bigraphs.

The first derived operator is *parallel product*, indicated by $\|$. Intuitively, it consists of a special kind of tensor product in which names are allowed to be shared between the two terms being composed.

Definition 2.4.1 (parallel product)**.** We begin by defining parallel product for place and link graphs. Then, the operation is defined for concrete and abstract bigraphs.

- Let $F_i = (V_i, ctrl_i, prnt_i) : m_i \to n_i$ be two concrete place graphs $(i = 0, 1)$ with disjoint supports. Then their parallel product $F_0 \parallel F_1 : m_0 + m_1 \to n_0 + n_1$ is given by

$$F_0 \parallel F_1 \stackrel{\text{def}}{=} F_0 \otimes F_1 \ .$$

- If $F_i = (V_i, E_i, ctrl_i, link_i) : X_i \to Y_i$ are link graphs $(i = 0, 1)$ with disjoint support and $link_0 \cup link_1$ is a function, their parallel product $F_0 \parallel F_1 : X_0 \cup X_1 \to Y_0 \cup Y_1$ is given by

$$F_0 \parallel F_1 \stackrel{\text{def}}{=} (V_0 \uplus V_1, E_0 \uplus E_1, ctrl_0 \uplus ctrl_1, link_0 \cup link_1) \; .$$

- Let $G_i : I_i \to J_i$ be two concrete bigraphs $(i = 0, 1)$ with disjoint supports. Then their parallel product $G_0 \parallel G_1 : \langle m_0 + m_1, X_0 \cup X_1 \rangle \to \langle n_0 + n_1, Y_0 \cup Y_1 \rangle$ is

$$G_0 \parallel G_1 \stackrel{\text{def}}{=} \langle {G_0}^{\mathsf{P}} \parallel {G_1}^{\mathsf{P}}, {G_0}^{\mathsf{L}} \parallel {G_1}^{\mathsf{L}} \rangle \; .$$

- If $[F_i]_{\eqsim} : \langle m_i, X_i \rangle \to \langle n_i, Y_i \rangle$ are two abstract bigraphs $(i = 0, 1)$, and $F_0 \parallel F_1$ is defined, their parallel product is

$$[F_0]_{\eqsim} \parallel [F_1]_{\eqsim} \stackrel{\text{def}}{=} [F_0 \parallel F_1]_{\eqsim} : \langle m_0 + m_1, X_0 \cup X_1 \rangle \to \langle n_0 + n_1, Y_0 \cup Y_1 \rangle \; .$$

Note that the condition of $link_0 \cup link_1$ being a function is required because it avoids a shared inner name to be mapped to more than one outer name. For example expression $\mathsf{id}_x \parallel y/x$ is not allowed because $link(x) = \{x, y\}$. However, $\mathsf{id}_x \parallel x/y$ is valid since the requirement is satisfied, i.e. $link(x) = x$ and $link(y) = x$.

Another derived operator is *merge product*, written $|$. It is a form of parallel product that produces bigraphs with only one root.

Definition 2.4.2 (merge product)**.** Let $G_i : \langle m_i, X_i \rangle \to \langle n_i, Y_i \rangle$ be two bigraphs $(i = 0, 1)$, and assume further that $G_0 \parallel G_1$ is defined. Their merge product is given by:

$$G_0 \mid G_1 \stackrel{\text{def}}{=} (merge_{n_0+n_1} \otimes \mathsf{id}_{Y_0 \cup Y_1}) \circ (G_0 \parallel G_1)$$

with $G_0 \mid G_1 : \langle m_0 + m_1, X_0 \cup X_1 \rangle \to \langle 1, Y_0 \cup Y_1 \rangle$.

Finally, nodes can be organised spatially by means of *nesting*. Additionally, their names are allowed to be shared.

Definition 2.4.3 (nesting)**.** Let $F : I \to \langle m, X \rangle$ and $G : m \to \langle n, Y \rangle$ be bigraphs. Define the nesting $G.F : I \to \langle n, X \cup Y \rangle$ by:

$$G.F \stackrel{\text{def}}{=} (G \parallel \mathsf{id}_X) \circ F \; .$$

Derived operators allow further convenient abbreviations. For example, if G has outer face $\langle n, X \uplus Z \rangle$, we write $y/X \circ G$ to mean $(y/X \parallel \mathsf{id}_I) \circ G$, where $I = \langle n, Z \rangle$. We usually write id for id_1 and we omit the subscript n for $merge$.

Arbitrary bigraphs have also been shown to be expressible by using the algebraic operators. This alternative normal form is called *connected normal form* (CNF). An example bigraphical expression in CNF for $A : \langle 1, x\rangle \to \langle 1, y\rangle$ in Figure 2.4c is

$$A = /x/z\,((\mathsf{B}_{xy}.(\mathsf{A}_{yz}.1 \mid \mathsf{id}) \mid \mathsf{D}_y.\mathsf{C}_z.1) \parallel \mathsf{id}_x)\ .$$

A quick comparison between this expression and the equivalent DNF given in Equation (2.7) shows how derived operators yield algebraic expressions that are more compact, intuitive and closer to the form of expression found in process calculi. The higher complexity of DNF is due to the fact that this normal form was introduced by Milner mainly to prove the completeness theorem for bigraphical expressions and not to provide a friendly notation for bigraphical terms. For this reason, we will always adopt algebraic operators in the expressions for the bigraphs used in the applications described in Part II.

2.5 Sorting

In most applications of bigraphs, it is useful to restrict the set of admissible bigraphs. This is achieved by classifying controls by means of *sorts*. Take for instance a simple scenario in which bigraphs are used to model hospitals. We have nodes of control B to represent buildings, F-nodes to model floors, R-nodes to model recovery rooms, O-nodes to encode operating rooms, etc. An example bigraph would then be

$$H = \mathsf{B}.(\mathsf{F}.(\mathsf{R}.1 \mid \mathsf{O}.1) \mid \mathsf{F} \mid \mathsf{F}.(\mathsf{id} \mid \mathsf{R}.1))\ . \tag{2.8}$$

Note that the intended semantics of the model forbids R-nodes from containing B-nodes, i.e. rooms cannot contain buildings. Hence, a bigraph like $H_{wrong} = \mathsf{R}.(\mathsf{F}.1 \mid \mathsf{B})$ should be regarded as not admissible. In the following, we will describe formally how to specify this kind of restriction.

Let us first establish some notational conventions. Sorts are ranged over by a, b, ... Disjunctive sorts are written as $\widehat{\mathsf{ab}}$, meaning that a node can either be of sort a or sort b. A bigraph satisfying a sorting Σ is called Σ-*sorted*.

Let us begin with the classification of places.

Definition 2.5.1 (place sorting)**.** A *place sorting* $\Sigma = (\Theta, \mathcal{K}, \Phi)$ has a non-empty set Φ of sorts and a signature $\mathcal{K}$ *place-sorted* over Θ, i.e. assigning a sort to each control. Component Φ is the *formation rule* of Σ. It is a property of Σ-sorted bigraphs that is satisfied by the identities and symmetries, and preserved by composition and tensor product.

When applying sorting Σ to interface n, we write $\vec{\theta}$, where $\vec{\theta} = \theta_1 \cdots \theta_n$ lists the sorts θ_i assigned to each $i \in n$.

We now formalise the hospital modelling scenario presented above by specifying a place-sorting as follows:

$$\Theta = \{\mathsf{b}, \mathsf{f}, \mathsf{r}\}, \qquad \mathcal{K} = \{\mathsf{B} : \mathsf{b}, \mathsf{F} : \mathsf{f}, \mathsf{R} : \mathsf{r}, \mathsf{O} : \mathsf{r}\},$$

and Φ requires:

an r-node is atomic;
all children of a b-node or f-root have sort f;
all children of an f-node or r-root have sort r;
all children of a θ-root have sort θ, where $\theta \in \Theta$.

Observe that bigraph H_{wrong} is indeed not admissible because the r-node is not atomic. The interface of the model given in Equation (2.8) is $H : \langle \mathsf{rr}, \emptyset \rangle \to \langle \mathsf{b}, \emptyset \rangle$. It can be composed with sorted bigraph $R : \epsilon \to \langle \mathsf{rr}, \emptyset \rangle$ given by $R = (\mathsf{R}.1 \mid \mathsf{R}.1) \parallel \mathsf{O}.1$. The composite $H \circ R$ is also sorted.

Links can be sorted in a similar fashion.

Definition 2.5.2 (link sorting)**.** A *link sorting* is a triple $\Sigma = (\Theta, \mathcal{K}, \Phi)$ where Θ is a non-empty set of sorts, and $\mathcal{K}$ is a *link-sorted* signature, i.e. assigning a sort to each port of each control. Component Φ is the *formation rule* of Σ. It is a property of Σ-sorted bigraphs that is satisfied by the identities and symmetries, and preserved by composition and tensor product.

When applying sorting Σ to interface $\{\vec{x}\}$, we write $\{x_1 : \theta_1, \ldots, x_n : \theta_n\}$, where each $\theta_i \in \Theta$. An example formation rule Φ is given by:

a link has sort l if it has an l-point;
no link has more than 2 j-points.

In some applications of bigraphs, place and link sorting are often combined. It is interesting to note that several extensions of bigraphs can be expressed by an appropriate combined place-link sorting (see [13]).

2.6 Bigraphical reactive systems

So far we have discussed only the static structure of bigraphical systems. However, they admit a dynamical theory too, involving ways in which bigraphs may reconfigure their own

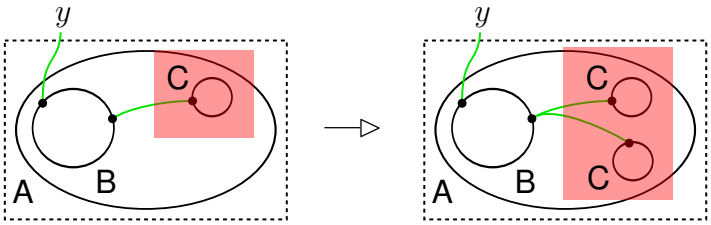


Figure 2.12: Reaction $S_0 \longrightarrow\!\!\triangleright\, S_1$.

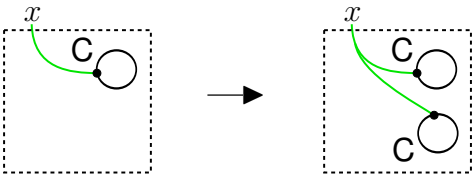


Figure 2.13: Reaction rule $R \longrightarrow\!\!\blacktriangleright\, R'$.

placing and linking. Such reconfigurations are defined in terms of rewrite rules, called *reaction rules*, which induce a *reaction relation* on bigraphs. An example of reaction is drawn in Figure 2.12. The diagram shows how bigraph S_0 on the left-hand side evolves to become bigraph S_1 on the right-hand side. The sub-parts of the two bigraphs being affected by the reaction are shaded in red. A reaction rule $R \longrightarrow\!\!\blacktriangleright\, R'$ consists of pairs of bigraphs that can be inserted into the same host bigraph. Left-hand side R specifies the pattern to be changed, while right-hand side R' specifies the changed pattern. An example reaction rule is given in Figure 2.13. Observe that the two bigraphs R and R' correspond to the shaded parts in S_0 and S_1, respectively. In fact, reaction $S_0 \longrightarrow\!\!\triangleright\, S_1$ is derived by checking if R matches S_0 and by substituting it with R' to obtain the new system S_1.

Before presenting the formal definitions, we set some terminology. Given a set of reaction rules, we refer to the configurations that a system may adopt as *states*. Hence, in the example above, S_0 and S_1 are states. A site is *guarding* if its parent is a node. A bigraph with inner face ϵ is called *ground*. We shall call ground bigraphs *agents*, since we ascribe dynamic behaviour to them. They are ranged over by lower case letters $g, r, d, \ldots$. In order to uniquely identify an occurrence of a redex, we introduce the following property of bigraphs:

Definition 2.6.1 (solid bigraph)**.** A bigraph is *solid* if these conditions hold:

1. no roots or outer names are idle;
2. no two sites or inner names are siblings;
3. every site is guarding;
4. no outer name is linked to an inner name.

At this point, we have all the notions required to formally define reaction rules. Note that we adopt the simplified definition introduced in [39][3]. All the bigraphs are assumed abstract.

Definition 2.6.2 (reaction rule)**.** A *reaction rule*[4] is a pair

$$\mathsf{R} = (R : m \to J, R' : m \to J) \ ,$$

sometimes written as $R \longrightarrow\!\!\blacktriangleright R'$, where R is the *redex* and R' the *reactum*, and R is solid. The rule generates all the *ground reaction rules* (r, r'), where $r = (R \otimes \mathsf{id}_Y) \circ d$ and $r' = (R' \otimes \mathsf{id}_Y) \circ d$ for some discrete ground *parameter* $d : \epsilon \to \langle m, Y \rangle$. The *reaction relation* $\longrightarrow\!\!\triangleright_{\mathsf{R}}$ over ground bigraphs is defined by

$$g \longrightarrow\!\!\triangleright_{\mathsf{R}} g' \text{ iff } g = Dr \text{ and } g' = Dr'$$

for some bigraph D and some ground reaction rule (r, r') generated from R.

Observe that condition 4 in Definition 2.6.1 holds because R has no inner names. We also remark that the constraints on d and R do not limit the rules that can be expressed. For example, whenever d is non-discrete, it can be replaced by a discrete parameter by adjusting the context D. Finally, note that d can only have nodes that are descendants of places in R. This is because id_Y in the definition of r.

We are now ready for the main definition:

Definition 2.6.3 (bigraphical reactive system (BRS))**.** A *bigraphical reactive system* consists of a pair $(\mathcal{B}, \mathcal{R})$ where $\mathcal{B}$ is a set of agents and $\mathcal{R}$ is a set or reaction rules defined over $\mathcal{B}$. It has a reaction relation

$$\longrightarrow\!\!\triangleright_{\mathcal{R}} \stackrel{\text{def}}{=} \bigcup_{\mathsf{R} \in \mathcal{R}} \longrightarrow\!\!\triangleright_{\mathsf{R}}$$

which will be written $\longrightarrow\!\!\triangleright$ when $\mathcal{R}$ is understood. We write $(\mathcal{B}(\Sigma), \mathcal{R})$ when the elements of $\mathcal{B}$ are Σ-sorted.

In [37], the authors introduce a different transition relation giving rise to minimal labelled transition systems. This theory is a refinement of BRS in which *labelled transitions* describe the reactions an agent may perform, possibly with assistance from its environment. Given a reaction rule R, a labelled transition $a \xrightarrow{L} \triangleright_{\mathsf{R}} a'$ defines the minimal context L such that

$$L \circ a = D \circ r \qquad\qquad a' = D \circ r'$$

[3] In this version, left-hand sides are constrained to be solid and only linear reaction rules are allowed, i.e. sub-terms of parameter d cannot be reordered, discarded or duplicated. This corresponds to having instantiation map $\eta = \mathsf{Id}$ in Milner's original definition.

[4] Note that R is overloaded. It was also used to indicate the control of nodes modelling recovery rooms in the previous section.

for some rule (r, r') generated from R and some context D. Informally, label L describes how the environment of an agent a can assist it in triggering reaction R. Observe that there is an underlying reaction $L \circ a \longrightarrow\!\!\triangleright_{\mathsf{R}} a'$ for every transition $a \xrightarrow{L} \!\!\triangleright_{\mathsf{R}} a'$. Conversely, every reaction $a \longrightarrow\!\!\triangleright_{\mathsf{R}} a'$ can be interpreted as transition $a \xrightarrow{\mathsf{id}_J} \!\!\triangleright_{\mathsf{R}} a'$, where J is the outer interface of a. This confirms the intuition that there is a reaction only when an agent provides enough context to trigger it, i.e. the redex matches the agent. Labels turn out to be important to define bismilarity over agents. We will not use this theory in this thesis.

2.7 BiLog

Bigraphs can be described by **BiLog**, a spatial logic whose modal operators are capable of expressing the placing and linking structure of bigraphical terms, in a fashion common to logics for process calculi, e.g. mobile ambients and the π-calculus. In **BiLog**, we can express formulae like $\mathbf{A} \circ \varphi$ to describe a node of control A acting as context for a bigraph satisfying φ and $(\mathbf{A}_a \circ \top) \otimes (\mathbf{B}_b \circ \top)$ to describe two places next to each other, with anything inside and with different names.

The logic was introduced by Conforti, Macedonio and Sassone in [21]. We give a brief overview in the following. We indicate the set of elementary bigraphs with $\Omega(\mathcal{K})$. We often omit $\mathcal{K}$ when an arbitrary signature is presumed. Recall that equality over bigraphical terms is defined by Axioms (2.1) – (2.6). Formulae in **BiLog** are given by

$$\begin{array}{rcl} \boldsymbol{\Omega} & ::= & \mathbf{id}_I \;\big|\; \mathbf{K}_{\vec{x}} \;\big|\; \mathbf{join} \;\big|\; \mathbf{1} \;\big|\; \boldsymbol{\gamma}_{m,n} \;\big|\; y/X \;\big|\; /x \\ \varphi, \psi & ::= & \bot \;\big|\; \varphi \Rightarrow \psi \;\big|\; \mathbf{id} \;\big|\; \boldsymbol{\Omega} \;\big|\; \varphi \circ \psi \;\big|\; \varphi \otimes \psi \end{array}$$

for each interface I and for each $\mathsf{K}_{\vec{x}}, \gamma_{m,n}, y/X, /x \in \Omega$. The $\models$ relation is defined inductively as follows:

$$\begin{array}{lcl} G \models \bot & \Leftrightarrow & \text{never} \\ G \models \varphi \Rightarrow \psi & \Leftrightarrow & G \models \varphi \text{ implies } G \models \psi \\ G \models \boldsymbol{\Omega} & \Leftrightarrow & G = E \text{ and } E \in \Omega \\ G \models \mathbf{id} & \Leftrightarrow & \text{exists } I \text{ such that } G = \mathsf{id}_I \\ G \models \varphi \circ \psi & \Leftrightarrow & \text{exists } G_0, G_1 \text{ such that } G = G_0 \circ G_1, \text{ with } G_0 \models \varphi \text{ and } \\ & & G_1 \models \psi \\ G \models \varphi \otimes \psi & \Leftrightarrow & \text{exists } G_0, G_1 \text{ such that } G = G_0 \otimes G_1, \text{ with } G_0 \models \varphi \\ & & \text{and } G_1 \models \psi \end{array}$$

where all operations on bigraphs are assumed defined. Classical operators $\top, \wedge, \vee, \neg$ can be

derived as well as the following modalities:

$$\varphi_I \stackrel{\text{def}}{=} \varphi \circ \mathbf{id}_I \qquad \varphi_{\to J} \stackrel{\text{def}}{=} \mathbf{id}_J \circ \varphi$$
$$\varphi_{I \to J} \stackrel{\text{def}}{=} (\varphi_I)_{\to J} \qquad \varphi \circ_I \psi \stackrel{\text{def}}{=} \varphi \circ \mathbf{id}_I \circ \psi$$

More advanced operators called *adjuncts* can also be defined. For example, the left adjunct $\varphi \circ\!\!- \psi$ expresses the property of a term satisfying ψ whenever inserted in a context satisfying φ. The decidability of **BiLog** is an open question.

2.8 Categorical semantics for bigraphs

Bigraphical structures and their operations can also be expressed in the general framework of category theory. A short introduction to elementary category theory and its basic notions is given in Appendix A. In this section we will only present the results needed to cast bigraphs and their constituents, place and link graphs, as categories.

Let us first recall some notational conventions. We write $\mathbf{C}, \mathbf{D}$ for *categories* and $\widetilde{\mathbf{C}}, \widetilde{\mathbf{D}}$ for *precategories*. We often denote *objects* by I, J, K and *arrows* by f, g, h. If an arrow f has a *domain* I and a *codomain* F, both objects, we write $f : I \to J$. We use $\mathbf{C}(I \to J)$ to indicate the *homset* of I and J, i.e. the set of arrows in $\mathbf{C}$ in the form $f : I \to J$. *Functors* between categories are written as $\mathfrak{F} : \mathbf{C} \to \mathbf{D}$.

We also recall the definition of *s-category*, a non-standard kind of category introduced by Milner to define concrete bigraphical structures and their support within a categorical semantics. The distinctive feature of an s-category is that composition is a partial operation and arrows and objects are equipped with symmetries and a tensor product.

Definition 2.1 (s-category)**.** An *s-category* $\widetilde{\mathbf{C}}$ is a precategory in which each arrow f is assigned a finite *support* $|f| \subset \mathcal{S}$. Further, $\widetilde{\mathbf{C}}$ possesses a partial tensor product, unit and symmetries, as in an spm category. The identities id_I and symmetries $\gamma_{I,J}$ are assigned empty support. In addition:

- For $f : I \to J$ and $g : J' \to K$, the composition $g \circ f$ is defined iff $J = J'$ and $|f| \cap |g| = \emptyset$; then $|g \circ f| = |f| \uplus |f|$.
- For $f : I_0 \to I_1$ and $g : J_0 \to J_1$, the tensor product $f \otimes g$ is defined iff $I_i \otimes J_i$ is defined ($i = 0, 1$) and $|f| \cap |g| = \emptyset$; then $|f \otimes g| = |f| \uplus |g|$.

We begin by defining concrete place graphs, concrete link graphs and concrete bigraphs as s-categories.

Definition 2.8.1 (bigraphical s-categories)**.** Given a signature $\mathcal{K}$, bigraphical s-categories are defined as follows:

- Natural numbers and concrete place graphs as given in Definition 2.2.2 are the objects and the arrows of s-category $\widetilde{\mathbf{Pg}}(\mathcal{K})$, respectively. The finite support of each arrow F is V_F. Composition and tensor product are set out in definitions 2.3.1 and 2.3.5, respectively. The identities are id_m and the symmetries are $\gamma_{m,n}$.
- The s-category $\widetilde{\mathbf{Lg}}(\mathcal{K})$ has finite name-sets as objects and concrete link graphs defined in Definition 2.2.3 as arrows. The finite support of each arrow F is $V_F \uplus E_F$. Composition and tensor product are given in definitions 2.3.2 and 2.3.6, respectively. The identities are id_X and the symmetries are $\gamma_{X,Y} \stackrel{\text{def}}{=} \mathsf{id}_{X \uplus Y}$.
- Interfaces $\langle m, X \rangle$ and concrete bigraphs as defined in Definition 2.2.4 are the objects and the arrows of s-category $\widetilde{\mathbf{Bg}}(\mathcal{K})$, respectively. The finite support of each arrow F is $V_F \uplus E_F$. Composition and tensor product are given in definitions 2.3.3 and 2.3.7, respectively. The identities are $\mathsf{id}_{\langle m,X \rangle}$ and the symmetries are $\gamma_{\langle m,X \rangle,\langle n,Y \rangle}$.

The relationships between bigraphical categories can be made formal by means of functors. For instance it is possible to define a forgetful functor $\mathfrak{F}^{\mathsf{P}} : \widetilde{\mathbf{Bg}}(\mathcal{K}) \to \widetilde{\mathbf{Pg}}(\mathcal{K})$ that drops the names from every interface and the link graph from every bigraph in $\widetilde{\mathbf{Bg}}(\mathcal{K})$. The result is s-category $\widetilde{\mathbf{Pg}}(\mathcal{K})$. Forgetful functor $\mathfrak{F}^{\mathsf{L}} : \widetilde{\mathbf{Bg}}(\mathcal{K}) \to \widetilde{\mathbf{Lg}}(\mathcal{K})$ is defined in a similar fashion. An important functor is $\mathsf{width} : \widetilde{\mathbf{Bg}}(\mathcal{K}) \to \mathbf{Finord}$, which transforms bigraphs into functions between finite ordinals. In more detail, for each interface $I = \langle m, X \rangle$, it defines $\mathsf{width}(I) = m$, and for every bigraph F and any site i of F, it defines $\mathsf{width}(F)(i)$ to be the unique root that is an ancestor of i in F.

Another functor based on $\eqslantless$-equivalence is used to pass from concrete bigraphs to abstract bigraphs.

Definition 2.8.2 (abstract bigraphs)**.** The $\eqslantless$-equivalent quotient $\mathbf{Bg}(\mathcal{K}) = \widetilde{\mathbf{Bg}}(\mathcal{K})/\eqslantless$ is an spm category whose objects are those in $\widetilde{\mathbf{Bg}}(\mathcal{K})$ and whose arrows $\mathbf{Bg}(\mathcal{K})(I \to J)$ are $\eqslantless$-equivalence classes of the homset $\widetilde{\mathbf{Bg}}(\mathcal{K})(I \to J)$. Its construction defines a functor $[\cdot] : \widetilde{\mathbf{Bg}}(\mathcal{K}) \to \mathbf{Bg}(\mathcal{K})$ called the lean-support quotient functor.

Similar functors for concrete place graphs and concrete link graphs can be defined. Observe that the properties defining spm categories are captured by Axioms (2.1) and Axioms (2.6).

When a sorting Σ is considered, we write $\mathbf{Bg}(\Sigma)$ to denote the spm category of abstract Σ-sorted bigraphs. It is shown in [13] that sortings can be interpreted as functors.

The isomorphisms (isos) in $\mathbf{Bg}(\mathcal{K})$ are pairs $\langle \pi, \alpha \rangle$ with π a permutation and α a renaming. The epimorphisms (epis) are the bigraphs in which no root and no outer name is idle. Monomorphisms (monos) are the bigraphs in which no two sites and no two inner names are siblings.

The theory of BRSs can be interpreted categorically as a special case of *wide reactive systems*. See Leifer's PhD Thesis [42] for a complete account. Moreover, *labelled transition systems* over s-categories can also be applied to $\widetilde{\mathbf{Bg}}(\mathcal{K})$. This allows for an elegant definition of a bisimulation that is a congruence. In order to prove this possible, bigraphical precategories are shown to have *relative push-outs* (RPOs) and *Idem push-outs* (IPOs). See Appendix A for more details.

2.9 Stochastic bigraphs

A stochastic extension of BRSs is introduced in [39] by attaching a stochastic rate to reaction rules. In this manner, the state space generated by a *stochastic BRS* (SBRS) can be naturally transformed into a *Continuous Time Markov Chain* (CTMC), and quantitative reasoning carried out with tools available for CTMC analysis, e.g. stochastic model checker Prism. Refer to Appendix B for a formal definition of CTMC. Let us give a more precise idea of the approach, omitting a few details.

Definition 2.9.1 (stochastic reaction rule)**.** A *stochastic reaction rule* R is a triple (R, R', ρ), sometimes written $R \overset{\rho}{\longrightarrow\!\!\blacktriangleright} R'$ where (R, R') is a reaction rule and $\rho > 0$ is a stochastic rate.

The next step is to associate a rate with each reaction in a given BRS. Since a reaction may occur with different underlying reaction rules, the contribution of each rule has to be taken into account. The reaction rate for a rule is obtained as the product of the rate and the number of *distinct* occurrences of the rule. This corresponds to the number of distinct concrete occurrences (see Definition 2.3.10) of a concretion of the rule's redex in a concretion of the agent. For a given stochastic reaction rule $\mathsf{R} = (R, R', \rho)$, we define $\mu_{\mathsf{R}}[g, g']$ to be the number of distinct occurrences of $\widetilde{R}$ in $\widetilde{g}$ such that $g \longrightarrow\!\!\triangleright_{\mathsf{R}} g'$. Note that it is always possible to count the occurrences in g because R is solid, and therefore epi and mono. This forces context D and parameter d to be unique (up to isomorphisms on the mediating interfaces) in the decomposition $g = D \circ (R \otimes \mathsf{id}_I) \circ d$.

Definition 2.9.2 (stochastic reaction)**.** Given agents g, g' and a family of stochastic reaction rules $\mathcal{R}$, the rate of a reaction $g \longrightarrow\!\!\triangleright_{\mathcal{R}} g'$ is given by

$$rate_{\mathcal{R}}[g, g'] \stackrel{\text{def}}{=} \sum_{\mathsf{R} \in \mathcal{R}} rate_{\mathsf{R}}[g, g'] \,,$$

where the reaction rate of $g \longrightarrow\!\!\!\!\triangleright_{\mathsf{R}} g'$ is defined formally by

$$rate_{\mathsf{R}}[g, g'] \stackrel{\text{def}}{=} \rho\, \mu_{\mathsf{R}}[g, g'] \,.$$

We often write $g \stackrel{\rho}{\longrightarrow\!\!\!\!\triangleright}_{\mathcal{R}} g'$ when $rate_{\mathcal{R}}[g, g'] = \rho$.

2.10 Other extensions and applications

Other extensions and refinements to Milner's definition of bigraphs have been proposed since their first introduction. We briefly describe two of them in the remainder of this section.

In the first one, the full independence of placing and linking is relaxed by assigning a locality to some links in a bigraph. Therefore, in this new setting, it matters whether a link crosses the boundaries of a place. For example, an edge bound to a certain node can only link ports that lie within that node. The structures defined by this extension are called *binding bigraphs* and were formalised in [25]. Their characteristics have been proven adequate to encode calculi with name binding (e.g. π-calculus) and to model protocols with security features that enforce communications to take place only within certain boundaries (e.g. private channels). To uniformly define binding of links, names are also allowed to be located at roots and sites. Hence, interfaces in binding bigraphs take the form $\langle m, \vec{X}, X \rangle$ where m is an ordinal, X a set of names (both *local* and *global*) and $\vec{X}$ is a vector indicating the *locations* of each local name. For instance, $\vec{X} = [\{\}, \{x, y\}]$ occurring in an outer interface says that local names x, y are located at the second root of the bigraph. Global names are those that are not assigned to any root or site. They behave like in standard bigraphs. Composition, tensor product and a normal form were defined for binding bigraphs.

The second extension is presented in [34] and it is called *directed bigraphs*. It consists of a new definition of link graphs in which edges are regarded as resources and names are interpreted as requests of resources. The main novelty is to assign a direction to edges in order to capture the "resource request flow" which starts from ports, goes through names and terminates in edges. Consider for instance the composition $A \circ F \circ B$. An inward link in F from an outer name x to an edge e means that e is a resource offered through x to the context A. Conversely, an outward link in F form a port p to an inner name y encodes the request of p through y of a resource in B. Note that other combinations are possible. Directed bigraph have been used to encode the fusion calculus [35].

2.10.1 Implementation

One of the primary goals of the development of bigraphs was to model directly phenomena in ubiquitous computing and biology. However, even prototypical applications tend to be extremely complex for manual manipulation and analysis. Therefore, software tools are essential to investigate whether bigraphical models are suitable for use in real deployments. Milner [47] proposed that such tools should aid modellers and users in three different ways:

- in programming and specifying the model for a system,
- in analysing the behaviour of a system by means of model checking and stochastic simulations,
- in visualising the components of a system at various levels of abstraction.

The first work in this direction is the the design and implementation of BPL Tool [32], a prototype implementation of bigraphical reactive systems. It can be used for experimenting with bigraphical models specified in a generic programming language that closely resembles the algebraic form of bigraphical terms. Computation is carried out by a matching engine [12], based on inference rules, capable of detecting the occurrences of a redex in a given agent and rewriting it. The BPL tool consists of parser, matching engine, and includes web and command line user interfaces. The tool has been written in Standard ML. It has been used to simulate a model of a mobile phone system in a bigraphical representation of the polyadic π-calculus. Note that binding bigraphs are also supported. A recent extension is BigMC [53], an explicit-state model checking tool for bigraphical reactive systems based on the BPL matching engine. A significant limitation of these tools is given by the performance of the matching engine as well as the lack of support for stochastic bigraphs and automatic generation of graphical representation of bigraphs. We will explain how we propose to overcome these issues in Chapter 4 where we introduce an efficient matching engine based on a SAT encoding and in Chapter 5 where we describe BigraphER, our implementation of bigraphical reactive systems.

Another tool supporting textual representation, graphical visualization of bigraphs, and the calculation of redex matchings is DBtk [4], a Tool-Kit for directed bigraphs.

2.11 Summary

In this chapter, we gave an overview of the fundamentals of the theory of bigraphs. We presented the graphical notation for the representation of bigraphs through several examples in Section 2.1. Sections 2.2 and 2.3 rigorously defined place graphs, link graphs, bigraphs

and their operations. A complete axiomatisation for an algebra of bigraphical terms was given in Section 2.4, where DNF was introduced. Sorting was defined in Section 2.5. Reaction rules and BRSs were discussed in Section 2.6, while **BiLog** was introduced in Section 2.7. Categories for bigraphical structure were explained in Section 2.8. A stochastic extension of bigraphs was described in Section 2.9. Other extensions and various implementations were discussed in Section 2.10.

In the next chapter we will introduce a novel generalisation of place graphs in which parent map $prnt$ is not restricted to be a function but can also be an acyclic binary relation. Hence, the underlying spatial model is a DAG instead of a forest. This allows for the representation of shared localities and overlapping space.

Chapter 3

Bigraphs with sharing

In this chapter we introduce *bigraphs with sharing*, a novel generalisation of Milner's bigraphs in which places may have several parents. This leads to a formulation of place graphs based on acyclic binary relations instead of the acyclic functions in the original definition. A preliminary version of this work was presented in [56].

In Section 3.1, we analyse some of the motivations behind the development of bigraphs with sharing. A comparison with other possible approaches is also provided. Section 3.2 formally presents bigraphs with sharing. We first introduce concrete place graphs with sharing and their operations. Then, we show how to build bigraphs with sharing by using the new definition of place graphs. Section 3.3 is devoted to the definition of an unambiguous graphical notation for bigraphs with sharing called stratified notation. Section 3.4 shows how bigraphs with sharing fit in the general categorical interpretation of bigraphs presented in Chapter 2. In Section 3.5 we give an axiomatisation and a normal form for bigraphs with sharing. We also introduce a new algebraic form to express shared places. We discuss how the introduction of a new formulation of locality affects the other aspects of the theory of bigraphs in Section 3.6. Some concluding remarks are in Section 3.7.

3.1 Motivation

In this section, we justify the introduction of sharing in three ways. First, we show that a locality model based on DAGs is often more natural when modelling the physical world. Second, we observe that roots and sites are treated asymmetrically in the standard definition of bigraphs. Third, we expose the disadvantages of two possible encodings of sharing that do not require a full extension of the formalism.

Scenarios in which space is shared among several entities are often encountered in the literature and even more frequently in the deployments of real applications. For instance,

Becker and Dürr in [7] argue that DAGs are a more appropriate spatial model for ubiquitous systems by presenting an example in which the location hierarchy of a building cannot be easily expressed by a forest. In particular, their model has entities modelling rooms that belong both to a floor and to a wing i.e. rooms are shared. Other examples are overlapping wireless signals, attention space of different individuals and biological processes. We will show in detail some models using sharing in Part II.

The asymmetry in standard place graphs becomes evident by observing that a node can have zero or more children, while it can only have one parent. From a more abstract point of view, this limitation implies that category $\mathbf{Pg}(\mathcal{K})$ is not self-dual. Namely, if we take a place graph $F : m \to n$, we swap the roots with the sites, and we take the inverse of the parent relation, we obtain a structure that is not a valid place graph. This asymmetry also arises in the definition of epi and mono place graphs.

We now explain in detail why we choose to extend the original definition of bigraphs, rather than encode sharing within the standard formalism. There are two possible encodings.

The first is to introduce dummy controls to represent intersections of nodes. For instance, if nodes A and B share a region, their intersection is represented as a separate node of control $\mathsf{A} \cap \mathsf{B}$. A graphical representation is given in Figure 3.1b. The immediate consequence of this approach is that place graphs are still representable by forests. However, a major disadvantage is that the number of dummy nodes to be added grows exponentially with the number of intersecting nodes. Moreover, this encoding is not complete because it cannot represent sharing when no nodes are involved. To prove this, consider the example of a C-node shared by two regions. It is possible to create a dummy region containing C, however we cannot assign a control to it. Hence, we lose track of who was sharing the node. This can be a limiting factor especially in the definition of reaction rules. Another shortcoming is that a node shared between A and B is placed inside the dummy node $\mathsf{A} \cap \mathsf{B}$, thus both A and B appear as if they do not have a child. This is not desirable especially when formation rules in place sortings or **BiLog** predicates specify properties on the absence or presence of parents or children.

The second encoding consists of keeping a copy of a shared node inside each of its parents and connecting the copies with a special link. For example, when a node C is shared between A and B, both A and B contain a node of control $\overline{\mathsf{C}}$ and the two $\overline{\mathsf{C}}$s are linked together. This is drawn in Figure 3.1c. Note that control $\overline{\mathsf{C}}$ is defined exactly as C but with an extra port to handle the special link. However, this approach does not allow one to express sharing without nodes, e.g. two nodes sharing a site, and nodes or sites with no parents. Another disadvantage of this approach is that links are used to represent locality instead of connectivity, going against the spirit of bigraphs, in which the two notions are kept distinct. Finally, a problem arises when occurrences have to be counted, for example

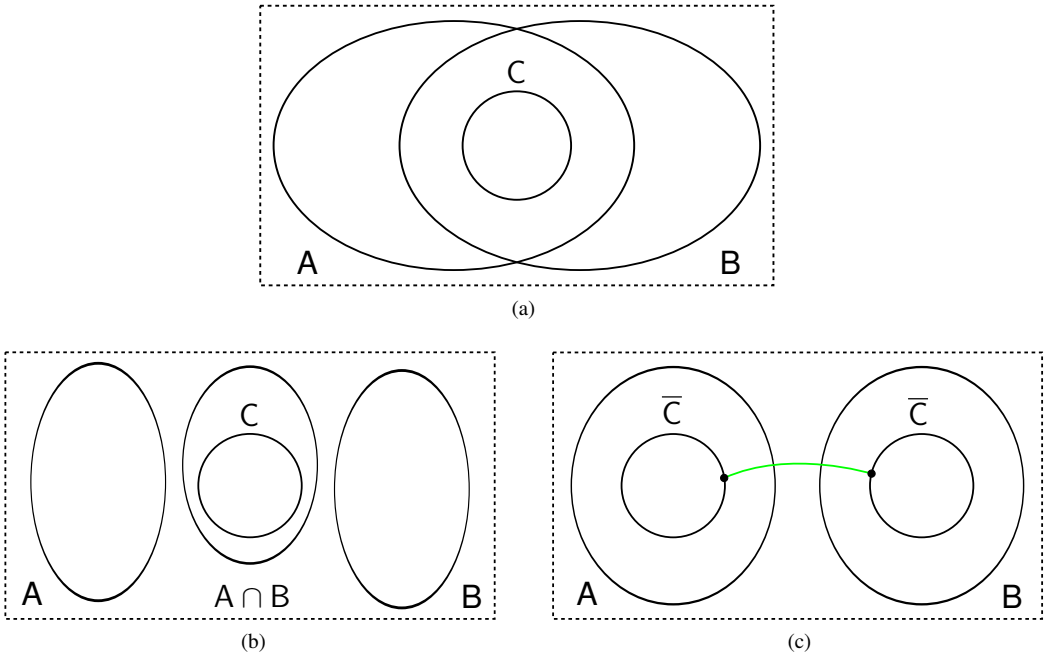


Figure 3.1: An example of a bigraph with sharing (a) and two possible encodings (b) and (c)

for the computation of a reaction rate in a stochastic BRS. In stochastic BRS, a shared node is copied (thus counted) n times, where n is the number of sharing nodes. Therefore, a new definition of reaction rate would be required.

Our extension yields several advantages. First, its completeness allows for the representation of arbitrary place graphs with sharing. Second, the modelling phase is more natural and immediate, because no additional links, copies of nodes and controls have to be introduced. Third, the structure of place graphs with sharing appears to have many similarities with standard categorical notions as we will show in the next sections. In particular, it makes places graphs self-dual.

3.2 Formal definition

In the previous section we justified why the locality concept of standard bigraphs should be updated by allowing sharing of places. In this section we present formally our extension.

We start off by introducing some notational conventions. Given a binary relation $rel \subseteq A \times B$, we denote the *domain restriction* of rel over $S \subseteq A$ by $S \lhd rel$. Similarly, we write $rel \rhd S$ for the *range restriction* of rel over $S \subseteq B$. Finally, it is worthwhile to recall the definition of composition between two binary relations $rel_0 \subseteq B \times C$ and $rel_1 \subseteq A \times B$:

$$rel_0 \circ rel_1 \stackrel{\text{def}}{=} \{(a, c) \in A \times C \mid \exists b.((a, b) \in rel_1 \wedge (b, c) \in rel_0)\} .$$

3.2.1 Concrete place graphs with sharing

The first step towards a formal definition of bigraphs with sharing is the introduction of a generalised version of concrete place graphs in which a place may have several parents. This is defined as follows:

Definition 3.2.1 (concrete place graph with sharing)**.** A *concrete place graph with sharing*

$$F = (V_F, ctrl_F, prnt_F) : m \to n$$

is a triple having an inner face m and an outer face n. These index respectively the sites and roots of the place graph. F has a finite set $V_F \subset \mathcal{V}$ of nodes, a control map $ctrl_F : V_F \to \mathcal{K}$, and a *parent relation*

$$prnt_F \subseteq (m \uplus V_F) \times (V_F \uplus n)$$

which is acyclic i.e. if $(v, v) \in prnt_F^i$ for some $v \in V_F$ then $i = 0$.

Note that the only difference with Definition 2.2.2 is that $prnt_F$ is now a binary relation instead of a function. This modification is sufficient to allow for places to have zero or more parents. Namely, it is possible to have a place $v \in m \uplus V_F$ such that $a \neq v$ for every $(a, b) \in prnt_F$. It is also allowed to have $(v, a), (v, b) \in prnt_F$ with $a \neq b$ for some places $a, b \in V_F \uplus n$. Any place having more than one parent is said to be *shared*. A place with no parents is called an *orphan*. Two places with a common parent and two places with a common child, are called *siblings* and *partners*, respectively. We define the *ancestor* relation $prnt_F^+$ to be the transitive closure of $prnt_F$.

Consider example place graph with sharing $B^{\mathsf{P}} : 2 \to 3$ drawn in Figure 3.2. The graphical representation highlights the fact that the underlying spatial model is a DAG. The node set is $V_G = \{v_0, v_1, v_2\}$ and the parent relation is

$$prnt = \{(0, v_0), (0, v_2), (1, v_2), (v_2, v_0), (v_2, v_1), (v_2, 0), (v_0, 0), (v_0, 2)\} .$$

Nodes v_0, v_1 are partners because they share node v_2. Node v_1 is an orphan. Observe that $prnt$ is not necessarily transitive since the parent of a place v may not be the parent of v's children. In the example, for instance, a parent of site 0 is v_0 and root 0 is a parent of v_0. However, root 0 is not a parent of site 0. On the other hand, we also have that $(v_2, v_0), (v_0, 0) \in prnt$ and also $(v_2, 0) \in prnt$.

3.2.2 Operations for place graphs with sharing

The second step is to define composition and tensor product for place graphs with sharing. We also have to prove that these operations have the same properties enjoyed by their coun-

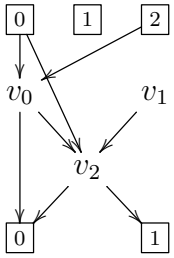


Figure 3.2: Example concrete place graph with sharing $B^{\mathsf{P}} : 2 \to 3$.

terparts in the standard definition of bigraphs, i.e. composition and tensor product are associative and identities are their neutral elements. Composition for place graphs with sharing is based on composition for binary relations.

Definition 3.2.2 (composition for place graphs with sharing)**.** If $F : k \to m$ and $G : m \to n$ are two concrete place graphs with sharing with $V_F \cap V_G = \emptyset$, their composite

$$G \circ F = (V, ctrl, prnt) : k \to n$$

has nodes $V = V_F \uplus V_G$ and control map $ctrl = ctrl_F \uplus ctrl_G$. Its parent relation $prnt \subseteq (k \uplus V) \times (V \uplus n)$ is given by:

$$prnt \stackrel{\text{def}}{=} prnt_G^{\triangleleft} \uplus prnt_{\circ} \uplus prnt_F^{\triangleright}$$

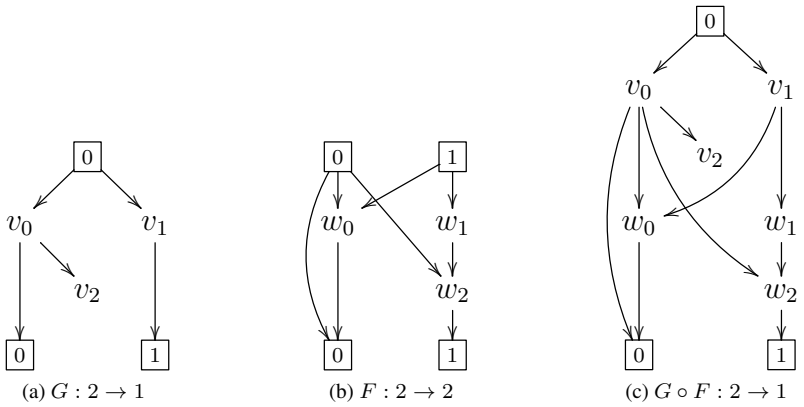
where

$$prnt_F^{\triangleright} = prnt_F \rhd V_F \qquad prnt_G^{\triangleleft} = V_G \lhd prnt_G$$
$$prnt_{\circ} = (m \lhd prnt_G) \circ (prnt_F \rhd m) \ .$$

We describe in words the three components of $prnt$. Relation $prnt_F^{\triangleright}$ is the set of edges in F from a place to a node. Since no roots are present in these edges, the relation is left unchanged by the composition. Similarly, $prnt_G^{\triangleleft}$ is the set of edges in G from a node to a place. In this case, there are no edges from a site in m, thus the relation is unaffected by the composition. Finally, relation $prnt_{\circ}$ is the composition in which the edges of F terminating in m and the edges of G originating in m are fused together. Consider the following example of composition.

Example 3.2.1. Let place graphs $G : 2 \to 1$, $F : 2 \to 2$ and their composition $G \circ F$ as in Figure 3.3. The parent relations for place graphs G and F are

$$prnt_G = \{(0, v_0), (1, v_1), (v_2, v_0), (v_0, 0), (v_1, 0)\} \ ,$$
$$prnt_F = \{(0, 0), (0, w_0), (1, w_2), (w_0, 0), (w_0, 1), (w_2, 0), (w_2, w_1), (w_1, 1)\} \ .$$



(a) $G : 2 \to 1$ (b) $F : 2 \to 2$ (c) $G \circ F : 2 \to 1$

Figure 3.3: Example of composition for concrete place graphs with sharing.

To construct the parent relation for the composed place graph $G \circ F$ we first define the following relations:

$$\begin{aligned} prnt^{\triangleleft}_G &\subseteq V_G \times (V_G \uplus 1) = \{(v_2, v_0), (v_0, 0), (v_1, 0)\} \ , \\ prnt^{\triangleright}_F &\subseteq (2 \uplus V_F) \times V_F = \{(0, w_0), (1, w_2), (w_2, w_1)\} \ , \\ prnt_{\circ} &\subseteq (2 \uplus V_F) \times (V_G \uplus 1) = \{(0, v_0), (w_0, v_0), (w_0, v_1), (w_2, v_0), (w_1, v_1)\} \ . \end{aligned}$$

Their union gives rise to the parent relation

$$prnt = prnt^{\triangleleft}_G \uplus prnt_{\circ} \uplus prnt^{\triangleright}_F \ . \qquad \blacksquare$$

We now prove that composition for place graphs with sharing enjoys the same properties of composition for standard place graphs. These results will be used in Section 3.4. We begin by proving associativity.

Proposition 3.2.1 (associativity of composition)**.** *If $A : m \to n$, $B : k \to m$, $C : h \to k$ are three concrete place graphs with sharing with disjoint node sets, then*

$$A \circ (B \circ C) = (A \circ B) \circ C \ .$$

Proof. Let us define $A \circ (B \circ C) = G_0$ and $(A \circ B) \circ C = G_1$. Since the node sets V_A, V_B, V_C are all disjoint and the domains are all compatible, then by Definition 3.2.2 all the composition are defined. We have to prove that $G_0 = G_1$. Again by Definition 3.2.2, $G_0, G_1 : h \to n$,

$$\begin{aligned} V_{G_0} = V_{G_1} &= V_A \uplus V_B \uplus V_C \\ ctrl_{G_0} = ctrl_{G_1} &= ctrl_A \uplus ctrl_B \uplus ctrl_C \ . \end{aligned}$$

It remains to prove that $prnt_{G_0} = prnt_{G_1}$. Since both relations are sub-sets of $(h \uplus V_{G_0}) \times (V_{G_0} \uplus n)$, we have to show that $(v, w) \in prnt_{G_0}$ if and only if $(v, w) \in prnt_{G_1}$ for every element (v, w). The parent relations are defined as

$$prnt_{G_0} = prnt_A^{\triangleleft} \uplus prnt_{\circ} \uplus prnt_{B \circ C}^{\triangleright} \tag{3.1}$$
$$prnt_{G_1} = prnt_{A \circ B}^{\triangleleft} \uplus prnt'_{\circ} \uplus prnt_C^{\triangleright} \ . \tag{3.2}$$

To analyse the single components we also compute the parent relations for the compositions $B \circ C$ and $A \circ B$:

$$prnt_{B \circ C} = prnt_B^{\triangleleft} \uplus prnt_{\circ}^{BC} \uplus prnt_C^{\triangleright}$$
$$prnt_{A \circ B} = prnt_A^{\triangleleft} \uplus prnt_{\circ}^{AB} \uplus prnt_B^{\triangleright} \ .$$

Therefore,

$$\begin{aligned} prnt_{B \circ C}^{\triangleright} &= prnt_{B \circ C} \rhd (V_B \uplus V_C) \\ &= (prnt_B^{\triangleleft} \uplus prnt_{\circ}^{BC} \uplus prnt_C^{\triangleright}) \rhd (V_B \uplus V_C) \\ &= (prnt_B^{\triangleleft} \rhd (V_B \uplus V_C)) \uplus (prnt_{\circ}^{BC} \rhd (V_B \uplus V_C)) \\ &\quad \uplus (prnt_C^{\triangleright} \rhd (V_B \uplus V_C)) \\ &= (V_B \lhd prnt_B \rhd V_B) \uplus (prnt_{\circ}^{BC} \rhd V_B) \uplus prnt_C^{\triangleright} \end{aligned}$$

and similarly

$$prnt_{A \circ B}^{\triangleleft} = prnt_A^{\triangleleft} \uplus (V_B \lhd prnt_{\circ}^{AB}) \uplus (V_B \lhd prnt_B \rhd V_B) \ .$$

Then (3.1) and (3.2) can be rewritten as

$$prnt_{G_0} = prnt_A^{\triangleleft} \uplus prnt_{\circ} \uplus (V_B \lhd prnt_B \rhd V_B) \uplus (prnt_{\circ}^{BC} \rhd V_B) \uplus prnt_C^{\triangleright}$$
$$prnt_{G_1} = prnt_A^{\triangleleft} \uplus (V_B \lhd prnt_{\circ}^{AB}) \uplus (V_B \lhd prnt_B \rhd V_B) \uplus prnt'_{\circ} \uplus prnt_C^{\triangleright} \ .$$

Hence, to prove that $prnt_{G_0} = prnt_{G_1}$ we have to show that

$$prnt_{\circ} \uplus (prnt_{\circ}^{BC} \rhd V_B) = (V_B \lhd prnt_{\circ}^{AB}) \uplus prnt'_{\circ}$$

holds. We start by proving $\Rightarrow$. We have the following cases:

1. If $(v, w) \in prnt_{\circ}$ then there exists a $w' \in m$ such that $(v, w') \in prnt_{B \circ C}$ and $(w', w) \in prnt_A$, where $v \in h \uplus V_C \uplus V_B$ and $w \in V_A \uplus n$. There are two sub-cases:

   (a) If $v \in V_B$ then $(v, w') \in prnt_B$. Therefore, $(v, w) \in (V_B \lhd prnt_{\circ}^{AB})$.

(b) If $v \in h \uplus V_C$ then there exists a $w'' \in k$ such that $(v, w'') \in prnt_C$ and $(w'', w') \in prnt_B$. But then $(w'', w) \in prnt_{A \circ B}$. It follows that $(v, w) \in prnt'_\circ$.

2. If $(v, w) \in (prnt_\circ^{BC} \rhd V_B)$ then there exists a $w' \in k$ such that $(v, w') \in prnt_C$ and $(w', w) \in prnt_B$, where $v \in h \uplus V_C$ and $w \in V_B$. But we also have that $(w', w) \in prnt_{A \circ B}$. Hence, $(v, w) \in prnt'_\circ$.

The proof for $\Leftarrow$ is symmetric. This concludes the proof. □

Identities are given in Definition 2.3.4. In the presence of sharing, the parent relation $\mathsf{Id_m}$ is interpreted as a binary relation. We now prove that identities are the neutral elements for composition of concrete place graphs with sharing.

Proposition 3.2.2 (neutral elements for composition)**.** *For any concrete place graph with sharing $G : m \to n$ the following holds*

$$G \circ \mathsf{id}_m = G = \mathsf{id}_n \circ G \ .$$

Proof. All the compositions are defined and $V_{\mathsf{id}_m} = V_{\mathsf{id}_n} = \emptyset$. The composite $G \circ \mathsf{id}_m = (V, ctrl, prnt)$ is defined according to 3.2.2. In particular, we have $V = V_G \uplus \emptyset$, $ctrl = ctrl_G$ and

$$prnt = prnt_G^{\lhd} \uplus prnt_\circ \uplus (\mathsf{Id}_m \rhd \emptyset) \ .$$

But, $\mathsf{Id_m} \rhd \emptyset = \emptyset$ and

$$prnt_\circ = (m \lhd prnt_G) \circ (\mathsf{Id} \rhd m) = m \lhd prnt_G \ .$$

It follows that $prnt = prnt_G$ and then $G \circ \mathsf{id}_m = G$. The proof for $\mathsf{id}_n \circ G = G$ is similar. □

Also the formal definition of tensor product for concrete place graphs with sharing has to be slightly changed.

Definition 3.2.3 (tensor product for place graphs)**.** If $G_0 : m_0 \to n_0$ and $G_1 : m_1 \to n_1$ are two concrete place graphs with sharing with disjoint supports, their tensor product

$$G_0 \otimes G_1 = (V, ctrl, prnt) : m_0 + m_1 \to n_0 + n_1$$

has nodes $V = V_{G_0} \uplus V_{G_1}$ and control map $ctrl \stackrel{\text{def}}{=} ctrl_{G_0} \uplus ctrl_{G_1}$. Its parent relation $prnt \subseteq ((m_0 + m_1) \uplus V) \times (V \uplus (n_0 + n_1))$ is defined as follows:

$$prnt_{G_0} \uplus prnt_{G_1}^{(m_0, n_0)}$$

where,

$$\begin{aligned} prnt_{G_1}^{(m_0,n_0)} = &\{(v,w) \mid (v,w) \in prnt_{G_1} \text{ and } v,w \in V_{G_1}\} \\ &\uplus \{(m_0+i,w) \mid (i,w) \in prnt_{G_1}\ , w \in V_{G_1} \text{ and } i \in m_1\} \\ &\uplus \{(v,n_0+i) \mid (v,i) \in prnt_{G_1}\ , v \in V_{G_1} \text{ and } i \in n_1\} \\ &\uplus \{(m_0+i,n_0+j) \mid (i,j) \in prnt_{G_1}\ , i \in m_1 \text{ and } j \in n_1\}\ . \end{aligned}$$

In words, it states that the parent relation of the place graph on the right-hand side (i.e. G_1) is shifted to the right by adding m_0 to every site and n_0 to every root. Therefore, tensor product is not commutative. As for composition, we prove that tensor product enjoys associative property and has neutral elements.

Proposition 3.2.3 (associativity of tensor product)**.** *If $A : m_0 \to n_0$, $B : m_1 \to n_1$, $C : m_2 \to n_2$ are three concrete place graphs with sharing with disjoint node sets, then*

$$A \otimes (B \otimes C) = (A \otimes B) \otimes C\ .$$

Proof. Let us define $A \otimes (B \otimes C) = G_0$ and $(A \otimes B) \otimes C = G_1$. Since the node sets are all disjoint, by Definition 3.2.3, all the products are defined. We have to prove that $G_0 = G_1$. Associativity of $\uplus$ and $+$ assures that

$$\begin{aligned} V_{G_0} = V_{G_1} &= V_A \uplus V_B \uplus V_C \\ ctrl_{G_0} = ctrl_{G_1} &= ctrl_A \uplus ctrl_B \uplus ctrl_C \\ m = m_0 + (m_1 + m_2) &= (m_0 + m_1) + m_2 \\ n = n_0 + (n_1 + n_2) &= (n_0 + n_1) + n_2\ . \end{aligned}$$

It remains to prove that $prnt_{G_0} = prnt_{G_1}$. By construction the following equalities hold:

$$prnt_{G_0} = prnt_A \uplus prnt_{B\otimes C}^{(m_0,n_0)} \qquad prnt_{G_1} = prnt_{A\otimes B} \uplus prnt_C^{(m_0+m_1,n_0+n_1)}$$

with

$$prnt_{B\otimes C} = prnt_B \uplus prnt_C^{(m_1,n_1)} \qquad prnt_{A\otimes B} = prnt_A \uplus prnt_B^{(m_0,n_0)}\ .$$

Hence, we have

$$prnt_{G_0} = prnt_A \uplus prnt_B^{(m_0,n_0)} \uplus prnt_C^{(m_0+m_1,n_0+n_1)} = prnt_{G_1}\ . \qquad \square$$

Proposition 3.2.4 (neutral element for tensor product)**.** *For any concrete place graph with*

sharing $G : m \to n$ *the following holds*

$$G \otimes \mathsf{id}_0 = G = \mathsf{id}_0 \otimes G \ .$$

Proof. Immediate from $prnt_{\mathsf{id}_0} = \emptyset$. □

Summarising, we proved that composition for concrete place graphs with sharing is associative with identities as neutral elements. Moreover, tensor product for concrete place graphs with sharing is also associative with id_0 as neutral element.

3.2.3 Bigraphs with sharing

At this point we have all the elements to define formally bigraphs with sharing. Observe that links and names are unaffected by the introduction of overlapping places. Therefore, link graphs in bigraphs with sharing are defined in the standard way. A concrete bigraph with sharing $G : \langle m, X \rangle \to \langle n, Y \rangle$ is a pair $\langle G^{\mathsf{P}}, G^{\mathsf{L}} \rangle$ with G^{P} a place graph with sharing and G^{L} a link graph. Operations on bigraphs with sharing are given by the corresponding operations on their constituents. Also the definition of support for bigraphs with sharing is analogous to the one presented in the previous chapter. We can then define abstract bigraphs with sharing as $\eqcirc$-equivalence classes of concrete bigraphs with sharing. Finally, symmetries for bigraphs with sharing are defined as for standard bigraphs (see (2.5)) but parent map $prnt$ is interpreted as a binary relation.

3.3 Graphical notation

Bigraphs with sharing can be represented by using a modified version of the nesting diagrams for standard bigraphs introduced in Chapter 2 in which intersecting nodes and roots are allowed.

An example nesting diagram of bigraph $B : \epsilon \to \langle 1, y \rangle$ is drawn in Figure 3.4a. The linkage between ports and names is represented by green edges as in the standard graphical notation. The node of control D is shared by the B and C-nodes. This is shown graphically by placing D within the intersection of B and C. Analogously, the E-node is shared by the A, B and C-nodes, so it is in the corresponding intersection. The F-node is not shared thus it is placed within the boundaries of its only parent C but outside the intersections with B and A. Finally, the G-node is an orphan and it is placed outside the bigraph. Hence, it has no parents.

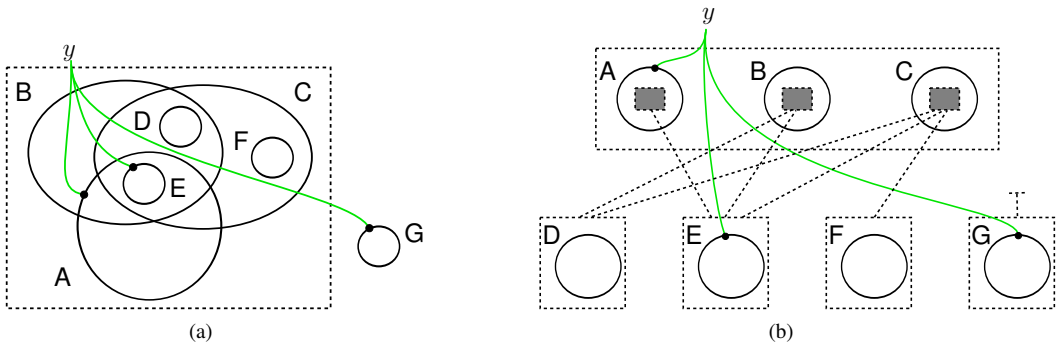


Figure 3.4: Graphical forms for bigraph $B : \epsilon \to \langle 1, y \rangle$: nesting diagram (a) and stratified diagram (b).

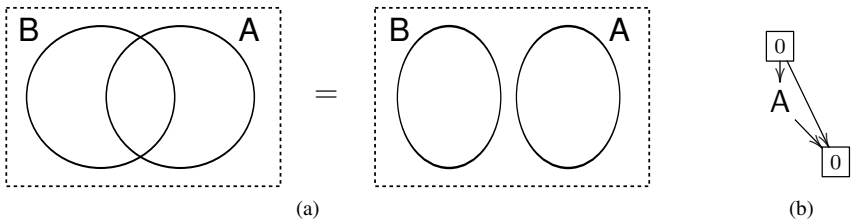


Figure 3.5: Disadvantages of Venn diagram style notation.

This notation recalls Venn diagrams and it is adequate to quickly specify simple models. However, it can be difficult to draw all the possible intersections when there are several (more than three) intersecting nodes. Moreover, there are other two major drawbacks:

- Empty intersections are meaningless (see 3.5a). These arise especially when there are more than two nodes overlapping. For instance, the intersection between the A and C-nodes in the previous example.
- It is impossible to represent a node sharing a place with one of its ancestors (see 3.5b).

To overcome these limitations, a new graphical form, called *stratified notation*, is introduced. In this notation, we make explicit how places are shared. An example stratified representation of bigraph $B : \epsilon \to \langle 1, y \rangle$ is given in Figure 3.4b. Nodes are organised in two different layers. The bottom one contains the children (nodes and sites), while the top one contains their parents (nodes and roots). In the example, nodes D, E, F, G and A, B, C, respectively. Each node in the bottom layer is placed into a different region. This allows the nodes to be shared by different combinations of parents. Each parent node in the top layer contains a site. The nesting of the nodes is specified by the dashed edges connecting the two layers. In the example, the first region containing the D-node is connected to the sites in the B and C-nodes. This models the fact that D is shared by B and C. The orphan node G has a *place closure*, hence there are no edges connecting its region to the top layer. The representation of links is not changed. Note that with this notation meaningless intersections are not

represented. In particular they do not arise because empty regions are not allowed to occur in the bottom layer. When a bigraph has a more complex nesting structure, it is possible to use more than two layers. This allows, for instance, for the representation of bigraphs with place graphs similar the one in Figure 3.5b.

The stratified notation is also important because it is related to the normal form for bigraphs with sharing we will introduce in Section 3.5. Moreover, automated drawing tools for bigraphs with sharing are based on this notation (refer to Chapter 5). In the following, we will use nesting diagrams for the representation of bigraphs with sharing. We assume that empty intersections mean nothing and that nodes cannot share a place with their ancestors. We will use the stratified notation only when particularly complex bigraphs arise or when these assumptions have to be dropped.

3.4 Categories of bigraphs with sharing

As for standard bigraphs, it is possible to define bigraphs with sharing in the general framework of category theory. We begin by proving that concrete place graphs with sharing form an s-category. To do so, we first have to prove that they can be interpreted as a precategory.

Definition 3.4.1 (precategory of concrete place graphs with sharing)**.** Given a signature $\mathcal{K}$, $\widetilde{\mathbf{SPg}}(\mathcal{K})$ is the precategory whose arrows are concrete place graphs with sharing as given in Definition 3.2.1 and objects are finite ordinals. Identities are place graphs $\mathsf{id}_m : m \to m$ and composition is set out in Definition 3.2.2.

Proof. By Definition 3.2.2, composition is a partial operation. Moreover, given $G : m \to n$ and $F : k \to m'$, when $G \circ F$ is defined then $m = m'$. Additionally, Propositions 3.2.1 and 3.2.2 prove that composition is associative and identities are its neutral elements, respectively. □

Before presenting the main result, we prove bifunctoriality of tensor product for concrete place graphs with sharing. We have two propositions. The first is the following:

Proposition 3.4.1 (bifunctoriality 1)**.** *If $A_0 : n_0 \to n_1$, $A_1 : n_1 \to n_2$, $B_0 : m_0 \to m_1$ and $B_1 : m_1 \to m_2$ are four concrete place graphs with sharing with disjoint node sets, then*

$$(A_1 \otimes B_1) \circ (A_0 \otimes B_0) = (A_1 \circ A_0) \otimes (B_1 \circ B_0) \ .$$

Proof. Define $(A_1 \otimes B_1) \circ (A_0 \otimes B_0) = G_0$ and $(A_1 \circ A_0) \otimes (B_1 \circ B_0) = G_1$. By Defini-

tions 3.2.2 and 3.2.3, we have

$$V_{G_0} = V_{G_1} = V_{A_0} \uplus V_{A_1} \uplus V_{B_0} \uplus V_{B_1}$$
$$ctrl_{G_0} = ctrl_{G_1} = ctrl_{A_0} \uplus ctrl_{A_1} \uplus ctrl_{B_0} \uplus ctrl_{B_1} \quad .$$

It remains to prove that $prnt_{G_0} = prnt_{G_1}$. By construction we have

$$prnt_{G_0} = prnt^{\triangleleft}_{A_1 \otimes B_1} \uplus prnt_{\circ} \uplus prnt^{\triangleright}_{A_0 \otimes B_0} \tag{3.3}$$

where the sub-sets are given by

$$prnt^{\triangleleft}_{A_1 \otimes B_1} = prnt^{\triangleleft}_{A_1} \uplus prnt^{(\_,n_2)\triangleleft}_{B_1} \qquad prnt^{\triangleright}_{A_0 \otimes B_0} = prnt^{\triangleright}_{A_0} \uplus prnt^{(n_0,\_)\triangleright}_{B_0}$$

and

$$prnt_{G_1} = prnt_{A_1 \circ A_0} \uplus prnt^{(n_0,n_2)}_{B_1 \circ B_0} \tag{3.4}$$

with

$$prnt_{A_1 \circ A_0} = prnt^{\triangleleft}_{A_1} \uplus prnt^{l}_{\circ} \uplus prnt^{\triangleright}_{A_0}$$
$$prnt^{(n_0,n_2)}_{B_1 \circ B_0} = prnt^{(\_,n_2)\triangleleft}_{B_1} \uplus prnt^{(n_0,n_2)r}_{\circ} \uplus prnt^{(n_0,\_)\triangleright}_{B_0} \quad .$$

We write $prnt^{(\_,n_2)\triangleleft}_{B_1}$ and $prnt^{(n_0,\_)\triangleright}_{B_0}$ to highlight the fact that these sets do not contain pairs with sites and roots to be shifted to the right. By equating the left-hand sides of (3.3) and (3.4), we obtain

$$prnt_{\circ} = prnt^{l}_{\circ} \uplus prnt^{(n_0,n_2)r}_{\circ} \tag{3.5}$$

By expanding the left-hand side, the equation can be rewritten as follows

$$\begin{aligned}
prnt_{\circ} &= ((n_1 + m_1) \triangleleft prnt_{A_1 \otimes B_1}) \circ (prnt_{A_0 \otimes B_0} \triangleright (n_1 + m_1)) \\
&= ((n_1 + m_1) \triangleleft (prnt_{A_1} \uplus prnt^{(n_1,n_2)}_{B_1})) \\
&\quad \circ ((prnt_{A_0} \uplus prnt^{(n_0,n_1)}_{B_0}) \triangleright (n_1 + m_1)) \\
&= ((n_1 \triangleleft prnt_{A_1}) \uplus ((n_1 + m_1) \triangleleft prnt^{(n_1,n_2)}_{B_1})) \\
&\quad \circ ((prnt_{A_0} \triangleright n_1) \uplus (prnt^{(n_0,n_1)}_{B_0} \triangleright (n_1 + m_1))) \\
&= ((n_1 \triangleleft prnt_{A_1}) \circ (prnt_{A_0} \triangleright n_1)) \\
&\quad \uplus (((n_1 + m_1) \triangleleft prnt^{(n_1,n_2)}_{B_1}) \circ (prnt^{(n_0,n_1)}_{B_0} \triangleright (n_1 + m_1))) \\
&= prnt^{l}_{\circ} \uplus prnt^{(n_0,n_2)r}_{\circ} \quad .
\end{aligned}$$

This concludes the proof. □

The second proposition is as follows:

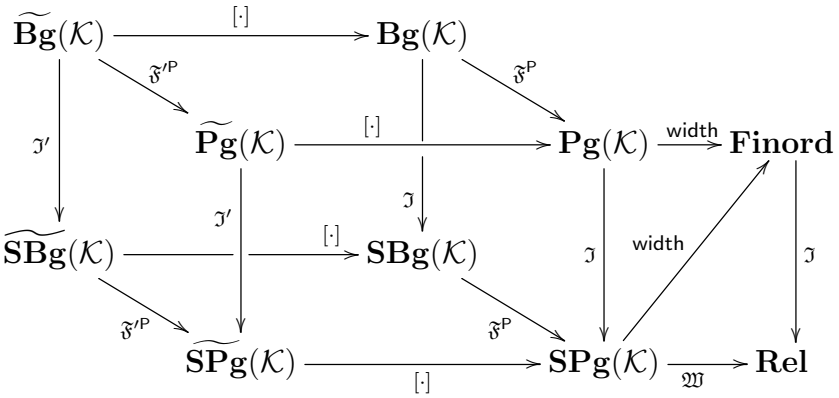


Figure 3.6: Bigraphical categories.

Proposition 3.4.2 (bifunctoriality 2)**.** *If* id_m *and* id_n *are two place graphs with sharing then*

$$\mathsf{id}_m \otimes \mathsf{id}_n = \mathsf{id}_{m \otimes n} \ .$$

Proof. The parent relations are Id_{m+n} and $\mathsf{Id}_{m \otimes n}$. Since $m \otimes n = m + n$ we have equality. □

We now can cast concrete place graphs with sharing to s-categories:

Proposition 3.4.3. *Given a signature* $\mathcal{K}$*, precategory* $\widetilde{\mathbf{SPg}}(\mathcal{K})$ *equipped with a finite support* V_F *for every arrow* $F : m \to n$*, a partial tensor product like in Definition 3.2.3 and symmetries* $\gamma_{m,n}$*, is an s-category.*

Proof. Immediate. We already proved bifunctoriality of tensor product. Since symmetries are defined like in $\widetilde{\mathbf{Pg}}(\mathcal{K})$, it is routine to prove that the same properties hold. □

We proved that concrete place graphs with sharing belong to the same kind of category of $\widetilde{\mathbf{Pg}}(\mathcal{K})$. Therefore, $\widetilde{\mathbf{SPg}}(\mathcal{K})$ together with $\widetilde{\mathbf{Lg}}(\mathcal{K})$ can be used to form s-category $\widetilde{\mathbf{SBg}}(\mathcal{K})$ of concrete bigraphs with sharing. The spm category of abstract bigraphs with sharing is then obtained by applying a lean-support quotient functor $[\cdot] : \widetilde{\mathbf{SBg}}(\mathcal{K}) \to \mathbf{SBg}(\mathcal{K})$ defined like in standard bigraphs. Usual forgetful functors like $\mathfrak{F}^{\mathsf{P}} : \mathbf{SBg}(\mathcal{K}) \to \mathbf{SPg}(\mathcal{K})$ and $\mathfrak{F}^{\mathsf{L}} : \mathbf{SBg}(\mathcal{K}) \to \mathbf{Lg}(\mathcal{K})$ can also be defined. Since functions are binary relations, $\mathbf{Pg}(\mathcal{K})$ is a sub-category of $\mathbf{SPg}(\mathcal{K})$. This relationship between the two categories is made explicit by inclusion functor $\mathfrak{I} : \mathbf{Pg}(\mathcal{K}) \to \mathbf{SPg}(\mathcal{K})$. Following the same argument, $\widetilde{\mathbf{Pg}}(\mathcal{K})$ is a sub-category of $\widetilde{\mathbf{SPg}}(\mathcal{K})$ with $\mathfrak{I}'$ acting as inclusion functor. The relationships between $\mathbf{SPg}(\mathcal{K})$ and other categories are summarised in Figure 3.6. By an abuse of notation, we write $\mathfrak{I}, \mathfrak{I}'$ to indicate inclusion functors, $\mathfrak{F}^{\mathsf{P}}, \mathfrak{F}'^{\mathsf{P}}$ for functors projecting into place graphs, and $[\cdot]$ to denote $\eqslantless$-quotient functors.

In the previous chapter, we discussed the close relationship between categories $\mathbf{Bg}(\mathcal{K})$ and $\mathbf{Finord}$. In particular, we defined a functor width which allows to interpret any bigraph

as a function between ordinals. When considering $\mathbf{SBg}(\mathcal{K})$, we can build a similar functor to move from bigraphs with sharing to the category of binary relations between ordinals $\mathbf{Rel}$. Formally, we define a functor $\mathfrak{W} : \mathbf{SPg}(\mathcal{K}) \to \mathbf{Rel}$ that acts as the identity on objects and is defined on every arrow $G^{\mathsf{P}} : m \to n$ as follows:

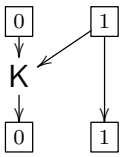
$$\mathfrak{W}(G) = \{(i,j) \mid i \in m \wedge j \in n \wedge (i,j) \in prnt_G^+\} \ .$$

Note that $\mathbf{Rel}$ corresponds to the class of node-free place graphs with sharing. This fact will be exploited in the following section when we will define an axiomatisation for bigraphs with sharing. It is also possible to move directly to $\mathbf{Finord}$ by mapping every site of a bigraph with sharing to the set of its root ancestors. Formally, for every arrow $G^{\mathsf{P}} : m \to n$ and for every site $i \in m$ we define

$$\mathsf{width}(G)(i) = j \qquad \text{such that } j \in 2^n \text{ and } \forall x \in j.(i,x) \in prnt_G^+ \ .$$

Consider for instance the place graph with sharing given below

$$F : 2 \to 2$$



By applying width we obtain

$$\mathsf{width}(F)(0) = \{0,1\} \qquad \mathsf{width}(F)(1) = \{1\} \ .$$

Since ordinal $2^2 = \{\emptyset, \{0\}, \{1\}, \{0,1\}\} = \{0,1,2,3\}$, the previous result can be rewritten as a function between ordinals:

$$\mathsf{width}(F)(0) = 3 \qquad \mathsf{width}(F)(1) = 2 \ .$$

Therefore, the existence of width and $\mathfrak{F}^{\mathsf{P}} : \mathbf{SBg}(\mathcal{K}) \to \mathbf{SPg}(\mathcal{K})$ proves that $\mathbf{SBg}(\mathcal{K})$ is a wide category.

An important property of $\mathbf{SPg}(\mathcal{K})$ is self-duality. This can be proved by showing that the inverse of any arrow is still a place graph with sharing. In more detail, for every arrow $G : m \to n$ we can construct its inverse arrow $G^{\mathsf{op}} : n \to m$ in which sites and roots are swapped, $prnt_{G^{\mathsf{op}}} = prnt_G^{-1}$, and G^{op} is itself an arrow of $\mathbf{SPg}(\mathcal{K})$. Note that place graphs without sharing are not self-dual. If we try for instance to construct the inverse of $join : 2 \to 1$, we obtain $prnt^{-1}(0) = 0$ and $prnt^{-1}(0) = 1$ which is not a function. Therefore $join^{\mathsf{op}}$ is not an arrow of $\mathbf{Pg}(\mathcal{K})$.

We can now characterise epis and monos in $\mathbf{SPg}(\mathcal{K})$. There are two considerations that help understanding their definitions. The first one is that epis are the inverse arrows of monos, and vice versa, due to self-duality of place graphs with sharing. The second consideration is that epis (monos) in place graphs with sharing are also epis (monos) in standard place graphs, since $\mathbf{Pg}(\mathcal{K})$ is a sub-category of $\mathbf{SPg}(\mathcal{K})$. Also recall that epis in $\mathbf{Rel}$ are surjective partial functions.

Proposition 3.4.4 (epis for place graphs with sharing)**.** *A concrete place graph with sharing is epi iff no root is idle and no two roots are partners.*

Proof. Recall that $B : m \to n$ is epi if $B_0 \circ B = B_1 \circ B$ implies $B_0 = B_1$ for any $B_0, B_1 : n \to h$.

We start by proving $\Rightarrow$. Assume B is epi; we prove that it has no idle roots (i) and no two roots are partners (ii).

(i) Suppose there is an idle root. Without loss of generality we set root $0 \in n$ idle. Pick $B_0, B_1 : n \to n-1$ as below

Then $B_0 \neq B_1$ but $B_0 \circ B = B_1 \circ B$, contradicting B is epi.

(ii) Suppose roots $0, 1 \in n$ are partners. Pick B_0, B_1 as in the previous case. Then $B_0 \neq B_1$ but $B_0 \circ B = B_1 \circ B$, contradicting B is epi.

Now we prove $\Leftarrow$. Assume B has no idle roots and no two roots are partners; we prove it to be epi. Let $B_0 \circ B = B_1 \circ B$. Then B_0 and B_1 have the same nodes and control map; so to prove $B_0 = B_1$ we have to show that $prnt_{B_0} = prnt_{B_1}$. By definition of composition for place graphs with sharing, equality $prnt_{B_0 \circ B} = prnt_{B_1 \circ B}$ leads to

$$prnt^{\triangleleft}_{B_0} \uplus prnt_{\circ} = prnt^{\triangleleft}_{B_1} \uplus prnt'_{\circ} \quad . \tag{3.6}$$

Assume $prnt^{\triangleleft}_{B_0} \neq prnt^{\triangleleft}_{B_1}$. Then we have two cases:

1. There exists an element $(v, w) \in prnt^{\triangleleft}_{B_0}$ (i.e. with $v \in V_{B_0}$) such that $(v, w) \notin prnt^{\triangleleft}_{B_1}$. Then by (3.6), $(v, w) \in prnt'_{\circ}$. But by construction of $prnt'_{\circ}$, $v \in V_B \uplus m$. Hence, $v \notin V_{B_0}$. This is a contradiction.

2. There is an element $(v, w) \in prnt^{\triangleleft}_{B_1}$ such that $(v, w) \notin prnt^{\triangleleft}_{B_0}$. By contradiction as in the previous case.

This proves $prnt^{\lhd}_{B_0} = prnt^{\lhd}_{B_1}$. Together with (3.6), it implies $prnt_\circ = prnt'_\circ$. By expanding the two terms, we can write

$$(n \lhd prnt_{B_0}) \circ (prnt_B \rhd n) = (n \lhd prnt_{B_1}) \circ (prnt_B \rhd n) \ .$$

By hypothesis, B has no idle roots and no two roots are partners. Therefore $prnt_B \rhd n$ is a surjective partial function. Thus, it is epi in $\mathbf{Rel}$. It follows that $(n \lhd prnt_{B_0}) = (n \lhd prnt_{B_1})$. Since

$$(n \lhd prnt_{B_i}) \uplus prnt^{\lhd}_{B_i} = prnt_{B_i} \qquad i = 0, 1 \ ,$$

we proved $prnt_{B_0} = prnt_{B_1}$. This concludes the proof. □

Proposition 3.4.5 (monos for place graphs with sharing)**.** *A concrete place graph with sharing is mono iff no two sites are siblings and no site is an orphan.*

Proof. Immediate by invoking self-duality of $\mathbf{SPg}(\mathcal{K})$. Observe that orphan sites and siblings correspond to idle roots and partners, respectively. □

Epis and monos in $\mathbf{SBg}(\mathcal{K})$ are the arrows in which both the place graph and the link graph are epi and mono, respectively.

We conclude this section by showing, with an example, that it is not always possible to build RPOs in concrete place graphs with sharing. Before presenting this result, we recall some notation and terminology. We use $\vec{F}$ to denote a pair $F_0 : m_0 \to n_0$, $F_1 : m_1 \to n_1$ of concrete place graphs with sharing. If $m_0 = m_1 = m$ the pair is a *span*, if $n_0 = n_1 = n$ it is a *cospan*. This is indicated by writing $\vec{F} : m \to \vec{n}$ and $\vec{F} : \vec{m} \to n$, respectively. We call cospan $\vec{G}$ a *bound* for span $\vec{F}$ if $G_0 \circ F_0 = G_1 \circ F_1$. Refer to Appendix A for the definition of RPOs.

Example 3.4.1. Consider span $\vec{A} : 1 \to \vec{n}$ bounded by cospan $\vec{D} : \vec{n} \to 1$ as given in Figure 3.7. Interfaces n_0 and n_1 are 3 and 2, respectively. By applying a construction procedure similar to the one defined by Milner for standard concrete place graphs, it is possible to build a candidate RPO $(\vec{H}, H)$ for $\vec{A}$ relative to $\vec{D}$ as given in Figure 3.8. Informally, H is obtained by "cutting" D_0 and D_1 as low as possible so that the two bigraphs above the cut line are the same. Hence, place graph H is the biggest common part of D_0 and D_1:

$$D_0 = H \circ H_0 \qquad\qquad D_1 = H \circ H_1 \ .$$

Now pick relative bound $(\vec{K}, K)$ shown in Figure 3.9. By construction, we have that

$$D_0 = K \circ K_0 \qquad\qquad D_1 = K \circ K_1 \ .$$

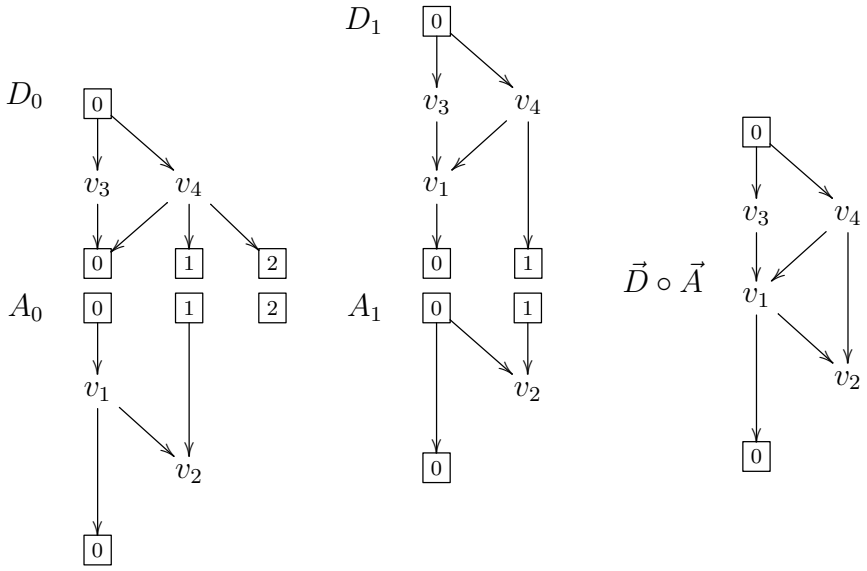


Figure 3.7: Span $\vec{A}$, bound $\vec{D}$ and their composite $D_0 \circ A_0 = D_1 \circ A_1$.

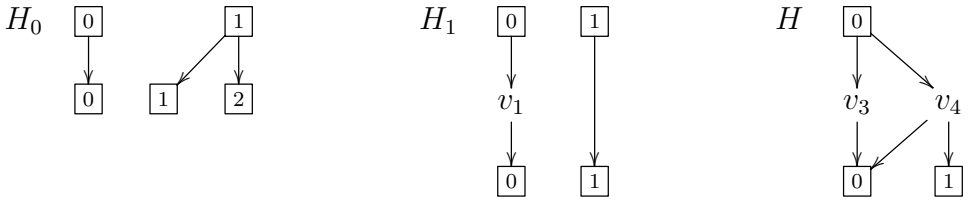


Figure 3.8: Candidate RPO $(\vec{H}, H)$ from $\vec{A}$ to $\vec{D}$.

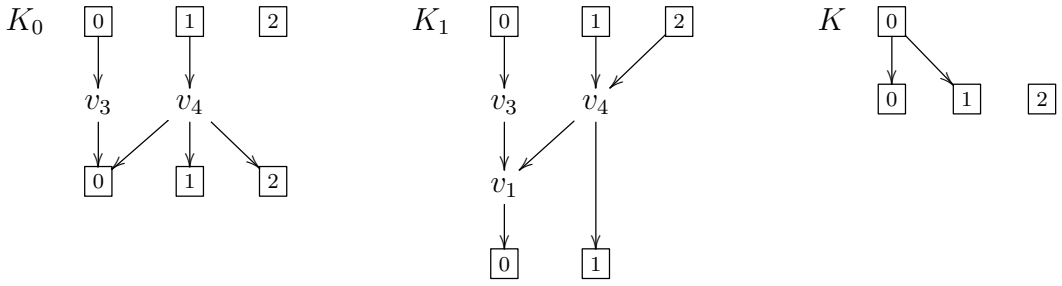


Figure 3.9: Relative bound $(\vec{K}, K)$ from $\vec{A}$ to $\vec{D}$.

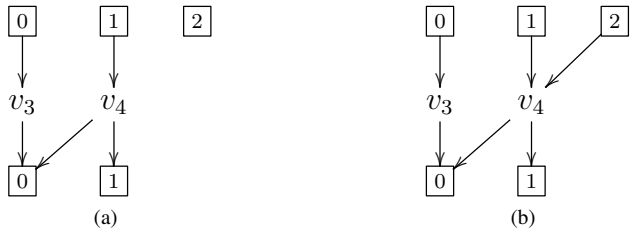


Figure 3.10: Tentative bigraphs for $\widehat{K}$.

In order for $(\vec{H}, H)$ to be an RPO, there must be a unique place graph $\widehat{K}$ such that

$$K_0 = \widehat{K} \circ H_0 \qquad K_1 = \widehat{K} \circ H_1 \qquad H = K \circ \widehat{K} \ .$$

But such $\widehat{K}$ cannot exist. If we chose to pick the bigraph in Figure 3.10a, then $K_1 \neq \widehat{K} \circ H_1$. On the other hand, if the bigraph given in Figure 3.10b is chosen instead, then $K_0 \neq \widehat{K} \circ H_0$. This is because the pair $(v_4, 2)$ has to be in $prnt_{\widehat{K}}$ in order to be compatible with K_1, while it must be absent in order to have K_0's idle root 2. Therefore, $(\vec{H}, H)$ is not an RPO. Observe that even by choosing a different candidate, it is always possible to build a bound such that the construction of $\widehat{K}$ is impossible. Thus, we can conclude that $\vec{A}$ does not have RPOs relative to $\vec{D}$. ■

Proposition 3.4.6 (concrete place graphs with sharing lack RPOs)**.** *In $\widetilde{\mathbf{SPg}}(\mathcal{K})$ there is a span $\vec{A}$ of concrete place graphs with sharing, and a bound $\vec{D}$ for it, such that no RPO exists from $\vec{A}$ to $\vec{D}$.*

Proof. Immediate by taking $\vec{A}$ and $\vec{D}$ as in the previous example. □

It follows that RPOs do not exist in general for $\widetilde{\mathbf{SBg}}(\mathcal{K})$. This may appear as a severe disadvantage of bigraphs with sharing since RPOs are essential to derive labels in labelled transition systems and to handling behavioural equivalence in bigraphs. Note, however, that this is not the case because RPOs do always exist in the sub-precategory of epimorphic place graphs with sharing, $\widetilde{\mathbf{SPg}}^{e}(\mathcal{K})$. This and the fact that the redex in a reaction rule must be epimorphic allows for the derivation of labelled transition systems also in the presence of sharing.

3.5 Algebraic form

In this section we show that bigraphs with sharing can be expressed by means of composition and tensor product starting from a minimal set of elementary building blocks. This results in an algebraic form similar to the one introduced for standard bigraphs in the previous chapter.

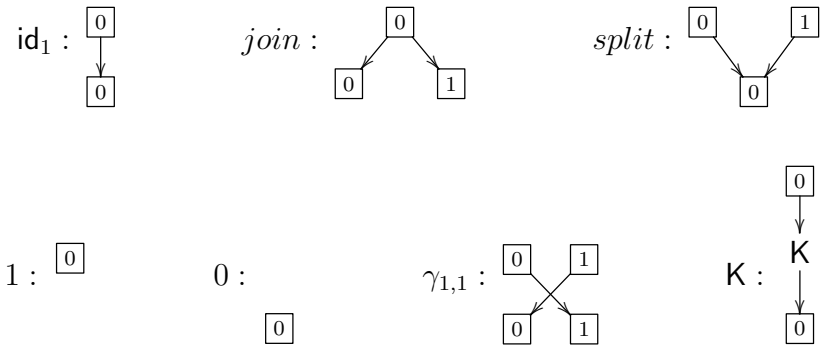


Figure 3.11: Elementary place graphs.

Two new elementary place graphs are required: $0 : 1 \to 0$ and $split : 1 \to 2$. They are the dual bigraphs of $1 : 0 \to 1$ and $join : 2 \to 1$, respectively. They are essential to represent orphans and shared places. The entire set of elementary place graphs is summarised in Figure 3.11. Node-free place graphs with sharing, again called placings, are ranged over by $\phi, \psi, \ldots$. We indicate a placing with one site and n roots by $split_n$. Note that $split_0 = 0$, $split_1 = \mathsf{id}_1$ and $split_2 = split$.

In the following, B, G, ω, ψ, etc are treated as expressions for bigraphs, but equality '$=$' is semantical, i.e. $B = B'$ means that B and B' denote the same bigraph and not that they are identical expressions. Formally, this is a shorthand for the more verbose $\models B = B'$.

Proposition 3.5.1. *Every place graph with sharing can be expressed as an expression containing only the elementary place graphs given in Figure 3.11 as constants and composition and tensor product as operators.*

Proof. We prove the proposition by induction on the number of nodes of the place graph with sharing. For the base case we show that node-free place graphs, i.e. placings, satisfy the proposition. A placing $\psi : m \to n$ may be expressed as

$$\psi = (merge_{n_0} \otimes \cdots \otimes merge_{n_{n-1}})\, \pi\, (split_{m_0} \otimes \cdots \otimes split_{m_{m-1}})$$

where $\sum_i m_i = m$, $\sum_i n_i = n'$ and $\pi : n' \to n'$ is a permutation. Since a permutation may be generated by composition and tensor product of symmetries $\gamma_{m,n}$, the proposition holds.

Now, let $B : m \to n$ be a place graph with sharing with $k + 1$ nodes. Then, it has a concretion $\widetilde{B} : m \to n$ with support V. Let $v \in V$ be a node in which none of its children are nodes (we call it a *leaf*). Such a node must exist by acyclicity of $prnt_B$. Note that v can still have $m' \leq m$ sites as children. Formally, $(u, v) \in prnt_B$ if and only if $u \in m'$. Without loss of generality we assume $ctrl(v) = \mathsf{K}$. Let $\widetilde{B}_1 : m + 1 \to n$ be the place graph obtained from $\widetilde{B}$ by removing node v and substituting it with a site. Furthermore, let

$\widetilde{B}_0 : m \to m+1$ be the place graph containing the sites in $\widetilde{B}$ and v. Then we have that $\widetilde{B} = \widetilde{B}_1 \circ \widetilde{B}_0$. By dropping the supports we can write $B = B_1 \circ B_0$. But B_0 can be defined in terms of elementary place graphs as follows:

$$B_0 = \psi \circ (\mathsf{id}_m \otimes \mathsf{K}) \circ \phi \qquad \text{with} \qquad \psi : m+1 \to m+1 \qquad \phi : m \to m+1 \ .$$

Therefore, the statement follows by inductive hypothesis on B_1 since it has k nodes. □

The construction presented in the proof above can be adopted to define a normal form for place graphs with sharing. Intuitively, a place graph with sharing B can be expressed by iteratively applying the procedure for the construction of B_0, until all its nodes are consumed. The only difference is that all the leafs are removed in one go instead of removing only a single leaf at each step. We make this precise by introducing the notion of *normalised levels*.

Definition 3.5.1 (level)**.** A *level* $L : m \to n$ is a place graph with sharing that can be expressed uniquely, up to permutations, as

$$L = (\bigotimes_{i<n-l} \mathsf{K}_i \otimes \mathsf{id}_l) \circ \phi \ ,$$

with placing $\phi : m \to n$.

Definition 3.5.2 (normalised levels)**.** Take two levels $L_1 : m_1 \to m_2$ and $L_0 : m_0 \to m_1$ given by

$$L_1 = (K_1 \otimes \mathsf{K}' \otimes \mathsf{id}_{l_1}) \circ \phi_1 \qquad L_0 = (K_0 \otimes \mathsf{id}_{l_0}) \circ \phi_0$$
$$K_1 = \bigotimes_{i<m_2-l_1-1} \mathsf{K}_i \qquad K_0 = \bigotimes_{i<m_1-l_0} \mathsf{K}_i$$

with $\phi_0 : m_0 \to m_1$ and $\phi_1 : m_1 \to m_2$ placings. We say that L_1 and L_0 are *normalised* if

$$((K_1 \otimes \mathsf{id}_{l_1+1}) \circ \phi_1') \circ ((K_0 \otimes \mathsf{K}' \otimes \mathsf{id}_{l_0}) \circ \phi_0') \neq L_1 \circ L_0 \ ,$$

for any K' and any placings

$$\phi_0' : m_0 \to m_1 + 1 \qquad \phi_1' : m_1 + 1 \to m_2 \ .$$

Put in words, this means that K' is not a leaf. Thus, it cannot be pushed down a level and L_0 is already a maximal set of leafs in $L_1 \circ L_0$. When considering several levels $L_n \circ \cdots \circ L_0$, we say that the levels are normalised if every pair of adjacent levels $L_i \circ L_{i-1}$ is normalised. We can now define a *stratified normal form* (SNF) for place graphs with sharing.

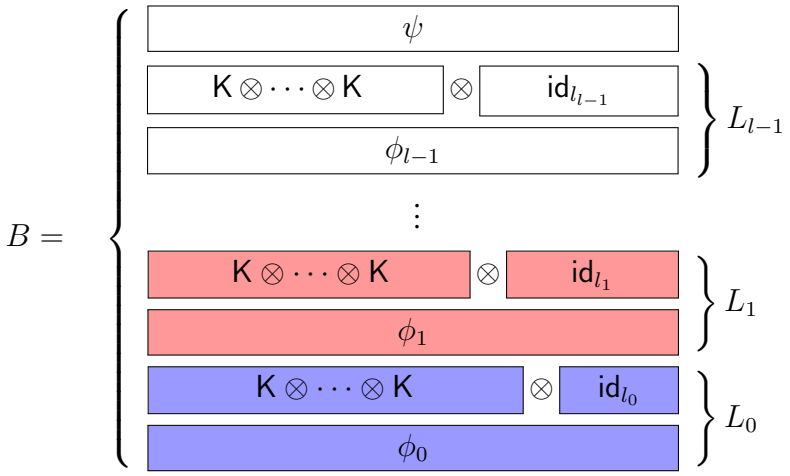


Figure 3.12: Graphical representation of SNF for place graph with sharing B. Levels L_0 and L_1 are drawn in blue and red, respectively.

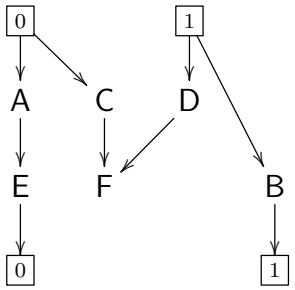
Proposition 3.5.2 (SNF for place graphs with sharing)**.** *Every place graph with sharing $B : m \to n$ can be expressed uniquely as*

$$B = \psi \circ L_{l-1} \circ \cdots \circ L_0$$

where ψ is a placing and each L_i is a normalised level.

A graphical explanation of the definition is shown in Figure 3.12. We now use SNF to express an example bigraph with sharing.

Example 3.5.1. Take $B : 2 \to 2$ given by



It is expressed in SNF as $B = \psi \circ L_1 \circ L_0$ with

$$\begin{aligned} \psi &= (\mathit{join} \otimes \mathit{join}) \\ L_1 &= (\mathsf{A} \otimes \mathsf{C} \otimes \mathsf{D} \otimes \mathsf{id}) \circ (\mathsf{id} \otimes \mathit{split} \otimes \mathsf{id}) \\ L_0 &= (\mathsf{E} \otimes \mathsf{F} \otimes \mathsf{B}) \circ (\mathsf{id} \otimes 1 \otimes \mathsf{id}) \quad . \end{aligned}$$

Note that the two levels are normalised. Now consider expression $B = \psi' \circ L_2 \circ L'_1 \circ L'_0$ with

$$\begin{aligned}
\psi' &= (join \otimes join) \circ (\mathsf{id} \otimes \gamma_{1,1} \otimes \mathsf{id}) \\
L_2 &= (\mathsf{A} \otimes \mathsf{B} \otimes \mathsf{id}_2) \circ \gamma_{2,2} \\
L'_1 &= (\mathsf{C} \otimes \mathsf{D} \otimes \mathsf{id}_2) \circ (split \otimes \mathsf{id}_2) \\
L'_0 &= (\mathsf{F} \otimes \mathsf{E} \otimes \mathsf{id}) \circ (1 \otimes \mathsf{id}_2) \ .
\end{aligned}$$

This is not in SNF because the levels are not normalised. Namely, A and B can be pushed down to levels L'_1 and L'_0, respectively. By doing so, we obtain a valid SNF expression equal, up to permutations, to the previous one. ■

This normal form can also be extended to represent bigraphs with sharing. Since nodes are the only structure in common between link graphs and place graphs, it suffices to replace the node generator described above with the ion $\mathsf{K}_{\vec{x}} : 1 \to \langle 1, \{\vec{x}\}\rangle$ defined for standard bigraphs. Hence, *extended levels* $L : m \to \langle n, X\rangle$ are defined as follows:

$$L = (\bigotimes_{i<n-l} \mathsf{K}_{\{\vec{x_i}\}} \otimes \mathsf{id}_l) \circ \phi \ ,$$

where $X = \biguplus_i \{\vec{x_i}\}$. Since normalisation is defined over place graphs, the definition of normalised levels is not affected by the introduction of ions.

Proposition 3.5.3 (SNF for bigraphs with sharing)**.** *Every bigraph with sharing $G : \langle m, X\rangle \to \langle n, Y\rangle$ may be expressed, up to permutations and renaming, as*

$$\begin{aligned}
G &= (\psi \otimes \omega) \circ S_{l-1} \circ \cdots \circ S_0 \\
S_i &= L_i \otimes \mathsf{id}_{X_i}
\end{aligned}$$

where ψ is a placing with outer interface n, ω is a linking with outer interface Y, and each L_i is a normalised extended level. Also the inner face of L_0 is m and $X_0 = X$.

SNF extends the notion of stratified diagrams for bigraphs with sharing presented in Section 3.3. A summary of the features in common between the two representations is the following:

- linking ω corresponds to the links on the top of the diagram,
- identities id_{X_i} are drawn as links crossing through the layers,
- levels are encoded by layers and the dashed lines connecting them.

Observe that in the SNF of a non-sharing bigraph all the placings ψ, ϕ_i are functions instead of relations.

The algebraic operators defined for standard bigraphs (e.g. merge product, parallel product and nesting) can also be used to express bigraphs with sharing. However, an additional operator capable of expressing sharing is required. We define *share expressions* as

$$\text{share } F \text{ by } \phi \text{ in } G \stackrel{\text{def}}{=} G \circ (\phi \otimes \mathsf{id}_X) \circ F$$

where all the operations are assumed defined. An example for bigraph $B : \epsilon \to \langle 1, w \rangle$ drawn in Figure 3.4 is

$$\begin{aligned} B &= \text{share } F \text{ by } \phi \text{ in } G \\ F &= \mathsf{D}.1 \parallel \mathsf{E}_y.1 \parallel \mathsf{F}.1 \parallel \mathsf{G}_y.1 \\ \phi &= (\mathsf{id} \parallel join \parallel merge_3) \circ (\gamma_{1,1} \parallel \gamma_{1,1} \parallel \mathsf{id}_2) \\ &\quad \circ (\mathsf{id} \parallel \gamma_{1,1} \parallel \mathsf{id}_3) \circ (split \parallel split_3 \parallel \mathsf{id} \parallel 0) \\ G &= \mathsf{A}_y \mid \mathsf{B} \mid \mathsf{C} \end{aligned}$$

Expressions F and G correspond to the bottom and top layers in the stratified diagram, respectively. Placing $\phi : 4 \to 3$ is encoded by the dashed edges connecting the roots and sites of the two layers. It can more conveniently be written as a vector of subsets of ordinal 3:

$$\phi = [\{1, 2\}, \{0, 1, 2\}, \{2\}, \emptyset]$$

3.5.1 Axioms for bigraphical equality

We now introduce a set of axioms that allow to prove every valid equation between bigraphical expressions. We write $\vdash B = B'$ to indicate equality inferred from the axioms. In the previous section, we proved by defining functor $\mathfrak{W} : \mathbf{SPg}(\mathcal{K}) \to \mathbf{Rel}$ that any placing $\phi : m \to n$ is also a relation between finite ordinals m and n. Thus, ϕ can also be viewed as an arrow in category $\mathbf{Rel}$. In [49, Theorem 7], Mimram proves that a complete axiomatisation of $\mathbf{Rel}$ is given by the equational theory of qualitative bicommutative bialgebrae. This allows us to define a complete axiomatisation for placings with sharing as a superset of the axioms for $\mathbf{Pg}(\mathcal{K})$ specified in Chapter 2, namely Axioms (2.1) and (2.6) for symmetrical monoidal categories, and Axioms (2.2) for commutative monoid $(\{0\}, join, 1)$. The

additional axioms that are required to obtain completeness over placings with sharing are

$$\begin{aligned}(split \otimes \mathsf{id}_1) \circ split &= (\mathsf{id}_1 \otimes split) \circ split \\ (0 \otimes \mathsf{id}_1) \circ split &= \mathsf{id}_1 = (\mathsf{id}_1 \otimes 0) \circ split \\ \gamma_{1,1} \circ split &= split\end{aligned} \tag{3.7}$$

specifying the co-monoid $(\{0\}, split, 0)$, and

$$\begin{aligned}split \circ join &= (join \otimes join) \circ (\mathsf{id}_1 \otimes \gamma_{1,1} \otimes \mathsf{id}_1) \circ (split \otimes split) \\ 0 \circ join &= 0 \otimes 0 \\ split \circ 1 &= 1 \otimes 1 \\ 0 \circ 1 &= \mathsf{id}_0 \\ join \circ split &= \mathsf{id}_1\end{aligned} \tag{3.8}$$

for qualitative bialgebra $(\{0\}, join, 1, split, 0, \gamma_{1,1})$. We remind the reader that detailed descriptions of monoids and bialgebrae appear in the Appendix A. In order to achieve completeness over $\mathbf{SPg}(\mathcal{K})$ we also need to include the ion axiom:

$$(\mathsf{id}_1 \otimes \alpha) \circ \mathsf{K}_{\vec{x}} = \mathsf{K}_{\alpha(\vec{x})} \ .$$

We claim without proving that this set of axioms is complete for expressions denoting place graphs with sharing. Formally, $B = B'$ implies $\vdash B = B'$ for all expressions B, B'.

A sketch of the proof is as follows. First, we show that $\vdash B = E$ and $\vdash B' = E'$, where E and E' are SNF expressions. Observe that Proposition 3.5.2 assures that such expressions exist for any place graph with sharing. Second, we show that the two normal forms can be proven equal, i.e. $\vdash E = E'$. This amounts to showing that the two expressions only differ by permutations of the ions in the levels:

$$\vdash \overbrace{\psi \circ L_{l-1} \circ \cdots \circ L_0}^{E} = \overbrace{\psi' \circ L'_{l-1} \circ \cdots \circ L'_0}^{E'}$$

$$\psi \circ (\overbrace{\bigotimes_i \mathsf{K}_i \otimes \mathsf{id}_l}^{L_{l-1}}) \circ \phi \circ \cdots \circ L_0 = \psi' \circ (\overbrace{\bigotimes_i \mathsf{K}_{\pi(i)} \otimes \mathsf{id}_l}^{L'_{l-1}}) \circ \phi' \circ \cdots \circ L'_0$$

$$E = \overbrace{\psi' \circ (\pi \otimes \mathsf{id}_l)}^{\psi} \circ (\bigotimes_i \mathsf{K}_i \otimes \mathsf{id}_l) \circ (\pi \otimes \mathsf{id}_l) \circ \phi' \circ \cdots \circ L'_0$$

$$\vdots$$

The conjecture can be extended to $\mathbf{SBg}(\mathcal{K})$ by including the complete axiomatisation for

expressions over linkings given in the previous chapter.

We remark that this axiomatisation will not be used in the rest of the thesis as our implementation of bigraph equality is based on a reduction to the graph isomorphism problem. We will describe this appraoch in great detail in chapters 4 and 5.

3.6 Discussion

We have seen that bigraphs with sharing are an elegant mathematical theory enjoying familiar properties: they form a wide[1] spm category and possess a normal form. It still remains to analyse the impact of sharing on the other aspects of Milner's bigraphical theory such as the dynamical theory of BRSs, the definition of sorting, and the description on bigraphical terms with the spatial logic **BiLog**.

We begin by considering reactive systems of bigraphs with sharing. In the definition of reaction rules given in Chapter 2, the redex and the reactum are solid. If we inspect Definition 2.6.1, this means that they are epi and mono. Since epis and monos are defined differently in bigraphs with sharing, we need a new definition of solid bigraphs.

Definition 3.6.1 (solid bigraphs with sharing)**.** A bigraph with sharing is *solid* if these conditions[2] hold:

1. it is epi, i.e. no root or outer name is idle and *no two roots are partners*;
2. it is mono, i.e. no two sites or inner names are siblings and *no site is an orphan*;
3. every site is guarding;
4. no outer name is linked to an inner name.

The reaction relation over bigraphs with sharing is computed as in the standard setting by matching a redex and replacing its occurrences with the corresponding reactum. Observe that the introduction of sharing does not entail changes to the standard definition of matching. However, we now have to allow discrete parameter d to have nodes shared also by nodes or roots in context D. In more detail, the occurrence of a redex $R : m \to J$ in an agent g is given by $g = D \circ (R \otimes \mathsf{id}_{\langle m', Y \rangle}) \circ d$ with $d : \epsilon \to \langle m + m', Y \rangle$. Note that this corresponds to Definition 2.6.2 when $m' = 0$. The matching algorithm identifying such occurrences needs be modified in order to support the new DAG structure of place graphs. We will describe a matching algorithm for bigraphs with sharing in the next chapter. Given the similarity with

[1] A category equipped with a functor to $\mathbf{Finord}$.

[2] The new conditions introduced for bigraphs with sharing are written in italics.

the standard definition, we use the acronym BRS also to indicate reactive systems of bigraphs with sharing.

Sorting disciplines can be used for bigraphs with sharing in a straightforward way. We only need extend formation rules for place sortings so that properties involving sharing can be expressed. An example is given by:

an $\widehat{\mathsf{ab}}$-node is an orphan;
all partners of an $\widehat{\mathsf{ab}}$-node have sort $\widehat{\mathsf{ab}}$;
all parents of a c-node have sort $\widehat{\mathsf{ab}}$.

Link sortings are as defined in standard bigraphs.

A similar extension is required to express sharing (i.e. elementary placings $0 : 1 \to 0$ and $split : 1 \to 2$) within the **BiLog** framework. The set of atomic formulae is redefined as follows:

$$\mathbf{\Omega}^s ::= \quad \mathbf{\Omega} \quad \Big| \quad \mathbf{0} \quad \Big| \quad \mathbf{split}$$

We call this extended logic **BiLog**^{s}. For ease of notation, we often drop the superscript when there is no ambiguity.

Finally, we analyse the compatibility of sharing with the other extensions of bigraphs presented in the literature. We begin with bigraphs with sharing.

Observe that stochastic rates can safely be attached to reaction rules as in standard SBRSs. However, the number of distinct concrete occurrences of a redex in an agent can no longer be used to compute the rate of a reaction. This is a direct consequence of the new definition of parameter d introduced above for BRS with sharing. To illustrate why this is the case, consider the following example.

Example 3.6.1. Take a reaction rule R, with redex $R = \mathsf{B}$ and an agent $g : \epsilon \to 1$ defined by

$$g = join \circ ((\mathsf{A} \circ (join \circ (\mathsf{B} \otimes \mathsf{id}) \circ (split \circ (\mathsf{C} \circ 1)))) \otimes (\mathsf{K} \circ 1))$$

Intuitively, R can be applied only once because g contains only a B-node that can be matched by R. However, according to Definition 2.3.9, it is possible to identify two distinct occurrences:

$$g = (D \circ (\mathsf{id}_2 \otimes \delta)) \circ (R \otimes \mathsf{id}) \circ d$$
$$g = D \circ (R \otimes \mathsf{id}_2) \circ (d \otimes \delta)$$

with $D = join \circ ((\mathsf{A} \circ join) \otimes \mathsf{id})$, $\delta = \mathsf{K} \circ 1$ and $d = split(\mathsf{C} \circ 1)$. The corresponding diagrams are given in Figure 3.13. ■

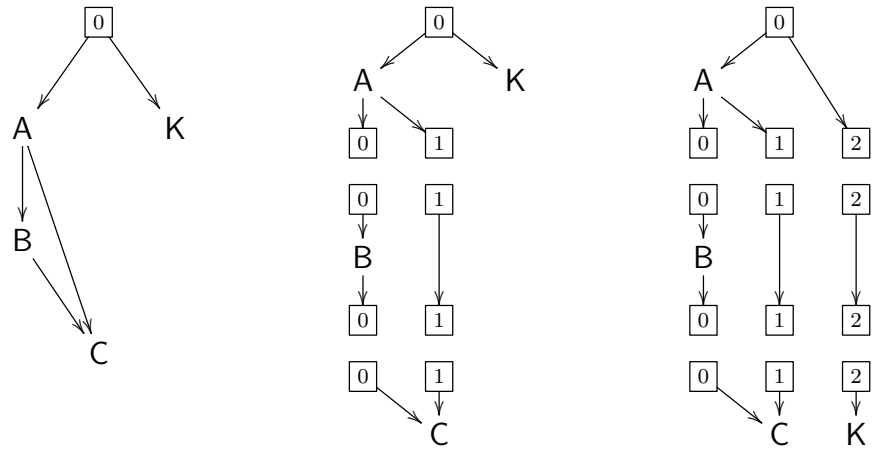


Figure 3.13: Agent g (on the left) and two distinct occurrences of R in g.

As the example above shows, it does not matter whether δ is in the context or in the parameter. This because δ contains places that do not have ancestors in R. Therefore, distinct occurrences differing only in this way need be counted only once when computing the reaction rate. We make this formal for concrete bigraphs as follows:

Definition 3.6.2 (δ-equivalent occurrences)**.** Consider concrete bigraphs G, F and two concrete occurrences $G = C_1 \circ (F' \otimes \mathsf{id}_I) \circ C_0$ and $G = C'_1 \circ (F' \otimes \mathsf{id}_{I'}) \circ C'_0$, with $F \eqsim F'$. They are *δ-equivalent* if one of the following conditions holds

- they differ only by a permutation or a bijective renaming on the mediating interfaces (as in Definition 2.3.10);
- there exists a non node-free bigraph δ and an interface J such that $C_1 \circ (\mathsf{id}_J \otimes \delta) = C'_1$ and $C_0 = C'_0 \otimes \delta$.

They are *δ-distinct* otherwise.

Given a reaction rule $\mathsf{R} = (R, R', \rho)$, we write $\mu_{\mathsf{R}}[g, g']$ to indicate the number of δ-distinct occurrences of $\widetilde{R}$ in $\widetilde{g}$ such that $g \longrightarrow\!\!\!\!\triangleright_{\mathsf{R}} g'$. By overloading this operator, reaction rates can be defined as in the previous chapter. Also in this case a new matching algorithm is needed to identify and count the occurrences of a reaction.

The introduction of sharing in directed bigraphs is easier because the two extensions do not interfere with each other. This is assured by orthogonality of place graphs with sharing and directed link graphs. The impact of allowing sharing places on the theory of binding bigraphs has not been analysed yet. In particular, it remains to establish whether a link bound to a place r can cross the boundaries of any r's partners. In other words, we need

to answer the question: is is possible to share a bound link? Therefore, further research in this direction is needed to decide which approach is more tractable theoretically and more suitable for practical applications.

3.7 Summary

In this chapter we presented bigraphs with sharing, a novel generalisation of bigraphs in which locality is modelled by DAGs. In Section 3.1, we discussed the reasons that led to the introduction of sharing. We gave a list of scenarios demanding sharing for their models to be expressive and, at the same time, straightforward to produce. We also explained why we decided to extend the formalism instead of encoding it within the original definition of bigraphs. A formal introduction to bigraphs with sharing was given in Section 3.2. Stratified and nesting diagrams for sharing bigraphs were described in Section 3.3. We proved that $\mathbf{SBg}(\mathcal{K})$ is a wide spm category in Section 3.4. We introduced an axiomatisation and normal form SNF in Section 3.5. In Section 3.6, we analysed the impact of sharing on other aspects of bigraphs, such as reactive systems and **BiLog**.

In the next chapter we will formalise a new matching algorithm that supports bigraphs with sharing. The algorithm is based on a reduction of the matching problem to the sub-graph isomorphism problem.

Chapter 4

Matching of bigraphs with sharing

This chapter introduces a matching algorithm for bigraphs with sharing. In Section 4.1, we give an overview of the matching problem and briefly describe the matching algorithm for non-sharing bigraphs introduced by Birkedal *et al.* in [12]. We also discuss the relationship with other well-known computational problems, e.g. the sub-graph isomorphism problem, and motivate the need for the definition of a new matching algorithm that supports sharing. Section 4.2 is based on our previous work on matching [57]. We first present the formal definition of a graph theoretic algorithm for the bigraph matching problem. Then, we analyse the algorithm in detail through some examples. Proofs of soundness and completeness are also provided. Finally, some concluding remarks are given in Section 4.3.

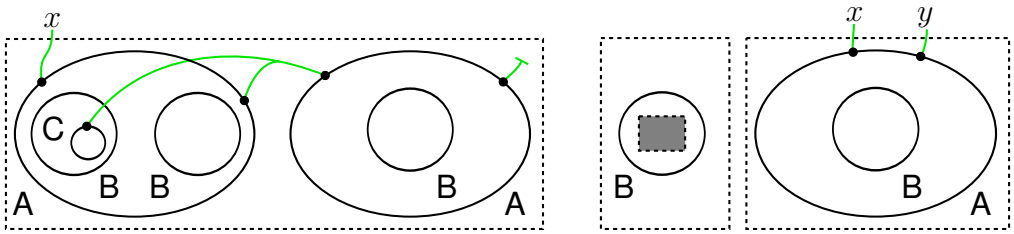
4.1 Introduction

The bigraph matching problem is a computational task in which a bigraph P, called *pattern*, and a bigraph T, called *target*, are given as input, and one must determine whether P occurs in T. A formal definition was given in Chapter 2. We recall it here with an example instance.

Example 4.1.1. Take a target $T : \epsilon \to \langle 1, \{x\}\rangle$ and a pattern $P : \langle 1, \emptyset\rangle \to \langle 2, \{x, y\}\rangle$ as in Figure 4.1. The matching instance consists of identifying the occurrences of P in T, i.e. the decompositions of the target defined as follows:

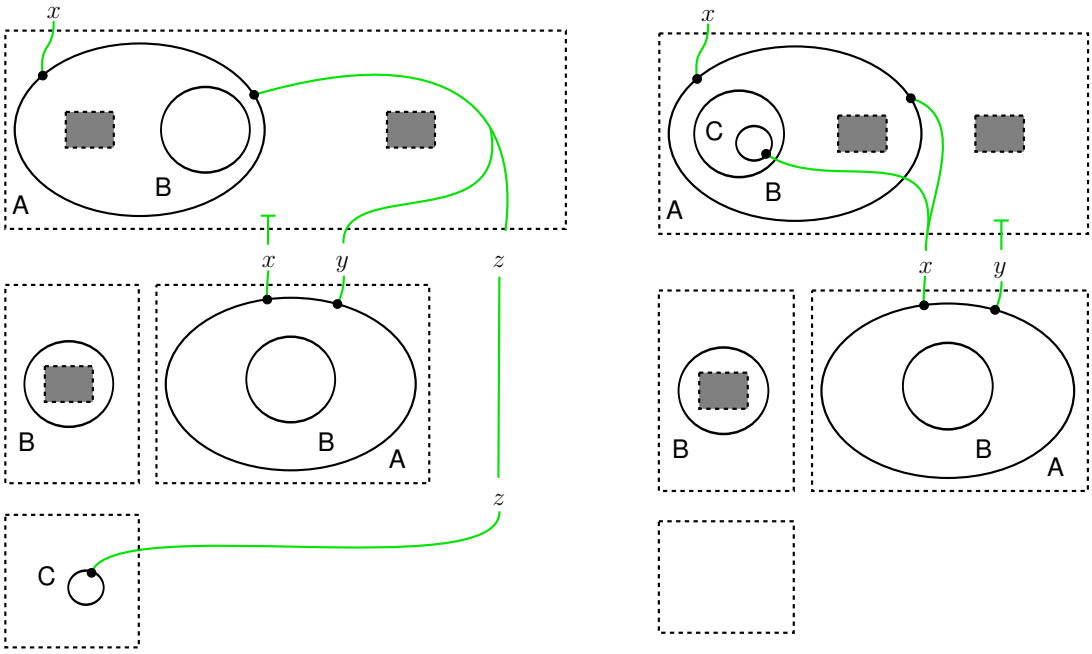
$$T = C_1 \circ (P \otimes \mathsf{id}_I) \circ C_0$$

for some interface I and bigraphs C_0 and C_1. We call C_1 the *context* and C_0 the *parameter*. When the leftmost region in the pattern matches the atomic B-node, a possible decomposition is described by the diagram in Figure 4.2b. On the other hand, if the same region matches the B-node containing the C-node, the decomposition shown in Figure 4.2a can be obtained.



(a) $T : \epsilon \to \langle 1, \{x\}\rangle$ (b) $P : \langle 1, \emptyset\rangle \to \langle 2, \{x, y\}\rangle$

Figure 4.1: Example target T and pattern P.



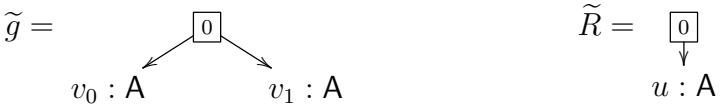
(a) $T = C \circ (P \otimes \mathsf{id}_z) \circ D$ (b) $T = C' \circ (P \otimes \mathsf{id}_\epsilon) \circ 1$

Figure 4.2: Example occurrences of pattern P in target T.

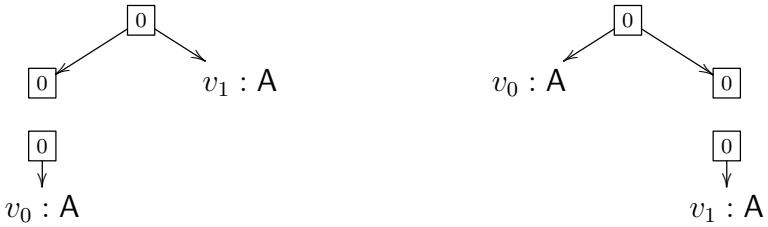
Considering the pattern can only match one of the two siblings of control B in the target, all the other decompositions can be constructed by modifying the ones in Figure 4.2. For instance, C' can be replaced by $C' \circ (\mathsf{id}_2 \otimes x/y \otimes y/x)$ in a different decomposition in which an isomorphism is applied to the mediating interface. ■

The matching problem plays a fundamental rôle in the definition of BRS: as we have seen in the previous chapters that an instance of the problem arises every time the occurrences of a redex in an agent are computed. In a stochastic setting, it is also important to take into account the number of distinct (and δ-distinct in SBRS with sharing) occurrences when computing the rate of a stochastic reaction. In order to distinguish between occurrences, the supports of the target and pattern need be used. This means that the bigraphs in the resulting matching instances are concrete. This was formalised in the definition of operator $\mu_{\mathsf{R}}[g, g']$ and in Definition 2.9.2. We explain why abstract bigraphs are not suitable for occurrence counting with the following example.

Example 4.1.2. Take a concrete agent $\widetilde{g}$ and a concrete redex $\widetilde{R}$ given by



There are two distinct occurrences of $\widetilde{R}$ in $\widetilde{g}$: in one u matches v_0 and in the other one u matches v_1. The corresponding decompositions are



Note that when sharing is allowed, it is possible to construct two further decompositions by placing the node not matched by the pattern in the parameter. However, these occurrences are δ-equivalent to the ones shown. Therefore, they do not contribute to the occurrence count. Now consider the corresponding abstractions g and R. If we count the number of distinct occurrences of R in g we can construct only one decomposition. This can be seen by dropping the supports in the diagrams above and observing that the two abstract decompositions are indistinguishable. ■

The earliest formalisation of a matching algorithm for standard bigraphs was introduced in [12]. In this paper, Birkedal *et al.* provide a characterisation of the occurrences of a pattern in a target by structural induction on their algebraic representation. Valid matches are derived by using an inference system composed of ten rules. The algorithm is proven sound and complete and also supports binding bigraphs. An example (simplified) rule for deriving a match for the tensor product of two place graphs is given by

$$\text{PAR}\ \frac{a, R \hookrightarrow C, d \qquad b, S \hookrightarrow D, e}{a \otimes b, R \otimes S \hookrightarrow C \otimes D, d \otimes e}$$

where R has no inner names (i.e. it is a redex). The notation $g, R \hookrightarrow D, d$ indicates a *matching sentence*. It is valid if and only if $g = D \circ R \circ d$. Therefore, the premises in the rule above are two valid matching sentences. In the bottom-up traversal of an inference tree, target g is decomposed while constructing context D. Since bigraphs may be denoted by several bigraphical terms, the inference system contains a rule STRUCT which allows to rewrite terms by using the axioms for the algebraic form of abstract bigraphs described in Chapter 2. This is also necessary in order to define a complete algorithm, as shown in the example below.

Example 4.1.3. Consider a target $g = (\mathsf{K} \otimes \mathsf{L})$ and a pattern $R = \mathsf{K}$. A valid matching sentence is $g, R \hookrightarrow (\mathsf{id}_1 \otimes \mathsf{L}), \mathsf{id}_1$. However, it cannot be derived without using STRUCT because PAR is not applicable. This is due to the fact that the pattern is not represented as a tensor product. By applying axiom

$$A \otimes \mathsf{id}_\epsilon = A = \mathsf{id}_\epsilon \otimes A$$

the pattern can be rewritten as $\mathsf{K} \otimes \mathsf{id}_0$ and the matching sentence is derived. ■

The matching problem for standard bigraphs is closely related to the sub-graph isomorphism problem. In more detail, the matching of place graphs can be reduced to the sub-forest isomorphism problem. A full match can then be computed by introducing extra constraints expressing the matching of link graphs. Both the sub-graph and sub-forest isomorphism problems[1] are proven to be **NP**-hard [59]. Therefore, algorithms that compute their solutions in polynomial time are not known. Traditionally, these problems are solved by using some type of backtracking search, as shown for instance in [64]. Observe that when the input bigraphs have only one root, the matching instance can be reduced to an instance of the sub-tree isomorphism which can be solved in polynomial time. The matching problem for bigraphs with sharing can also be reduced to the sub-graph isomorphism problem. More precisely, it is reducible to the sub-DAG isomorphism problem, which is also **NP**-complete [66]. Note that the introduction of sharing does not involve an increase in complexity.

We have chosen to follow the sub-DAG isomorphism approach, and not to extend the algorithm based on inference rules described above for several reasons. The first one is that the new DAG structure of the place graphs increases the amount of unnecessary blind search during the inference process, which negatively affects the performance of the algorithm. Intuitively, this is due to the fact that there are even more ways to decompose the pattern and the target at every application of an inference rule. Further, a reduction to a standard problem tends to produce faster algorithms because it fully exploits established solvers which are very efficient due to the use of advanced heuristics. A typical example of this approach is the encoding of hard problems into instances of the boolean satisfiability problem (SAT) [38, 19]. Finally, we want our algorithm to support concrete bigraphs and arbitrary patterns. Support for concrete bigraphs is required to allow the enumeration of all occurrences of the pattern in the target (see Example 4.1.2). This is necessary for any complete implementation of stochastic BRS. Furthermore, only concrete bigraphs can be used in a future implementation of labelled transition systems based on RPO construction. Support for arbitrary patterns (i.e. bigraphs in which inner names are allowed), is necessary because we will use the matching algorithm not only to match a redex in an agent during the computation of the reaction relation in a BRS, but also to encode an algorithm for the verification of **BiLog** formulae.

[1] When expressed as decision problems.

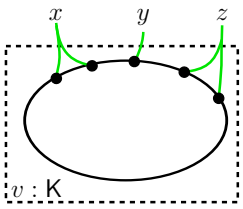


Figure 4.3: Bigraph $B : \epsilon \to \langle 1, \{x, y, z\}\rangle$.

Observe that the very nature of the algorithm described above does not allow for a straightforward extension that supports these two new features. This because the inference tree is built upon algebraic terms for *abstract* bigraphs expressed in DNF.

4.2 Algorithm

In this section we present a graph-theoretic matching algorithm for concrete bigraphs with sharing. Note that our algorithm natively supports standard bigraphs as we will prove in Proposition 4.2.6 at the end of the chapter. First, we introduce some useful definitions that will allow for a more concise specification of the algorithm. The sets of points and the set of ports of a link l are defined as follows:

$$points_F(l) \stackrel{\text{def}}{=} \{p \mid link_F(p) = l\} \qquad ports_F(l) \stackrel{\text{def}}{=} points_F(l) \setminus X \ .$$

We then define an equivalence relation among links with "similar" sets of ports. Intuitively, when comparing the sets of ports of two links, only the number of ports belonging to the same node is taken into account, while the port identifiers are ignored. Before giving the formal definition, we illustrate the idea with an example.

Example 4.2.1. Take concrete bigraph $B : \epsilon \to \langle 1, \{x, y, z\}\rangle$ defined in Figure 4.3. Links x and z are equivalent because they both have two ports belonging to node v. The sets of ports of links x, y and z are

$$ports_B(x) = \{(v, 0), (v, 1)\} \quad ports_B(y) = \{(v, 2)\} \quad ports_B(z) = \{(v, 3), (v, 4)\} \ .$$

If all port identifiers are dropped, these sets can be transformed into multisets. The multisets for x and z are the same. ■

Definition 4.2.1. Take two links l, k in a bigraph $F : \langle m, X\rangle \to \langle n, Y\rangle$. Equivalence relation $\doteq$ is defined as follows

$$l \doteq k \qquad \text{iff} \qquad count(l) = count(k) \text{ and } \tau(l) = \tau(k)$$

where

$$\tau(l) = \begin{cases} \mathsf{name} & \text{if } l \in Y \\ \mathsf{edge} & \text{if } l \in E_F \text{ and } points(l) \subseteq P_F \\ \mathsf{open\text{-}edge} & \text{otherwise} \end{cases}$$

and $count(l) \stackrel{\text{def}}{=} \{\{v \mid (v,i) \in ports(l)\}\}$ is a multiset.

Returning to Example 4.2.1, we can write $count(x) = count(z) = \{\{v, v\}\}$ and therefore $x \doteq z$. We will use $\doteq$-equivalence classes of links in the algorithm when constructing mappings between the links of the pattern and the links of the target. Finally, we introduce an explicit transformation from concrete place graphs to DAGs:

Definition 4.2.2 (underlying graph)**.** Let $F = (V_F, ctrl_F, prnt_F) : m \to n$ be a concrete place graph. Directed graph $\mathcal{G}_F = (V, E)$ is called the *underlying graph* of F. The set of nodes is $V = V_F$ and the set of edges is $E = V_F \lhd prnt_F \rhd V_F$.

The transformation consists of dropping all the roots and sites from bigraph F. An important property is that any control-preserving graph isomorphism ι between two underlying graphs is a support translation ρ_V between the corresponding place graphs when sites and roots are ignored. Formally, $\iota : \mathcal{G}_F \to \mathcal{G}_G$ is a bijection such that $ctrl_F(u) = ctrl_G(\iota(u))$, $ctrl_F(v) = ctrl_G(\iota(v))$ and

$$(u,v) \in prnt_F \qquad \text{iff} \qquad (\iota(u), \iota(v)) \in prnt_G \ .$$

Note that this is equivalent to writing

$$(V_G \lhd prnt_G \rhd V_G) \circ \iota = \iota \circ (V_F \lhd prnt_F \rhd V_F) \ .$$

Hence, ι satisfies Definition 2.2.5 when roots and sites are ignored. This correspondence between ι and ρ_V motivates the reduction of the matching problem to the sub-graph isomorphism problem.

Notation and conventions

We write $prnt(v)$ to indicate set $\{u \mid (v,u) \in prnt\}$ and $prnt^{-1}$ for the inverse of $prnt$. Graph isomorphisms and maps between links are indicated by ι and η, respectively. We write $\widehat{\mathcal{G}_T}$ to denote the sub-graph of $\mathcal{G}_T$ that is isomorphic via ι to $\mathcal{G}_P$. Therefore, $\widehat{V_T} \subseteq V_T$ is the range of ι. A multiset A is included in a multiset B, written $A \subseteq B$, iff every element of A occurs at least the same number of times in B. In the following, we assume the pattern

and the target are $P : \langle m, X\rangle \to \langle n, Y\rangle$ and $T : \langle m', X'\rangle \to \langle n', Y'\rangle$, respectively. Finally, the set of idle links of concrete bigraph P is defined by

$$P^{\mathsf{L}}_{idle} \stackrel{\text{def}}{=} \{l \in E_P \uplus Y \mid points_P(l) = \emptyset\} \ .$$

4.2.1 Definition

We are now ready to introduce our matching algorithm. We begin with an informal outline. The inputs of the algorithm are two concrete place graphs: the first being the target while the second being the pattern. The output is a set of pairs specifying all the occurrences of the pattern in the target. Each pair consists of a graph isomorphism and a mapping between links. The first phase of the algorithm consists of invoking a procedure that solves the sub-graph isomorphism problem. The underlying graphs of the pattern and the target are used as inputs. The output is a set of graph isomorphisms. In the next phase, the algorithm checks that every isomorphism obtained in the previous phase satisfies the following *compatibility* conditions:

- Node controls are preserved.
- The pattern has sites and roots allowing for the construction of a context and a parameter by decomposing the target.
- No node in the context has an ancestor in the pattern.

All the isomorphisms failing to pass any of the checks above are discarded. Finally, output pairs are computed by associating every compatible isomorphism with a mapping from the pattern links to the target links. Each mapping ensures that the link graph of the pattern can be composed with the link graph of a context and the link graph of a parameter, both obtained by decomposing the target. If the procedure fails to build such a mapping, then the corresponding isomorphism is discarded.

The algorithm is formally defined by the pseudocode given in Figure 4.4. It takes the form of a function MATCH$(_, _)$ in which the two arguments are concrete bigraphs. Since node-free patterns are a match for any target, we assume the second argument has a non-empty node set, i.e. $V_P \neq \emptyset$. Note that the pattern's port set can still be empty if all nodes in V_P have 0-arity controls.

We now describe in detail the operations described in the pseudocode. In line 2, the reduction to an instance of the the sub-graph isomorphism problem is performed by invoking sub-routine SUB_ISO$(\mathcal{G}_T, \mathcal{G}_P)$. We use it in a black-box fashion. Therefore, we assume an implementation is provided. The results are stored in set I. Every isomorphism $\iota \in I$ takes the form $\iota : V_P \to \widehat{V_P}$.

```
1   function MATCH(T, P)

2   I := SUB_ISO(G_T, G_P);
3   I' := ∅;
4   M := ∅;
5   forall ι ∈ I do
6       V̂_P := ι(V_P);
7       if CTRL(ι, ctrl_T, ctrl_P) ∧ SITES(ι, T, P) ∧ ROOTS(ι, T, P) ∧ TRANS(T, V̂_P) then
8           I' := {ι} ∪ I';
9       end if;
10  forall ι ∈ I' do
11      η := BUILD_LINK_MAP(∅, (E_P ⊎ Y) \ P^L_idle, ι, T, P);
12      if η ≠ ∅ then
13          M := {(ι, η)} ∪ M;
14      else if P_P = ∅ then
15          M := {(ι, ∅)} ∪ M;
16      end if;
17  return M;
18  end function
```

Figure 4.4: Pseudocode for the matching algorithm.

The loop in lines 5-9 filters set I by checking that every element satisfies the compatibility conditions listed above. Each condition is checked in line 7 by invoking a different sub-routine. The isomorphisms passing all the tests are stored in I'. Pseudocode for function CTRL(_, _, _) is given in Figure 4.5. The inputs are an isomorphism and the control maps of the target and the pattern, in that order. In line 2, there is a check that the input isomorphism preserves controls. Sites and roots are checked by invoking SITES(_, _, _) and ROOTS(_, _, _), respectively. The pseudocode for SITES(_, _, _) is in Figure 4.6. It takes as inputs an isomorphism, the target and the pattern, in that order. The loop in lines 2-5 checks that P's sites allow the composition $(P \otimes \mathsf{id}_I) \circ D$. More precisely, it checks that for every node or site c in D having parents in $\widehat{V_P}$, there exists a set of sites S in P whose parent set is isomorphic

```
1   function CTRL(ι, ctrl_T, ctrl_P)

2   if exists (v, u) ∈ ι such that ctrl_P(v) ≠ ctrl_T(u) then
3       return false;
4   else
5       return true;
6   end if;
7   end function
```

Figure 4.5: Pseudocode for sub-routine CTRL(_, _, _).

1 **function** SITES(ι, T, P)

2 **forall** $c \in \{v \in (m' \uplus V_T) \setminus \widehat{V_P} \mid prnt_T(v) \cap \widehat{V_P} \neq \emptyset\}$ **do**
3 **if** not exists $S \subseteq m$ such that $\iota(\bigcup_{s \in S}(prnt_P \rhd V_P)(s)) = (prnt_T \rhd \widehat{V_P})(c)$ **then**
4 **return false**;
5 **end if**;
6 **return true**;
7 **end function**

Figure 4.6: Pseudocode for sub-routine SITES(_, _, _).

1 **function** ROOTS(ι, T, P)

2 **forall** $p \in \{v \in (V_T \uplus n') \setminus \widehat{V_P} \mid prnt_T^{-1}(v) \cap \widehat{V_P} \neq \emptyset\}$ **do**
3 **if** not exists $R \subseteq n$ such that $\iota(\bigcup_{r \in R}(prnt_P^{-1} \rhd \widehat{V_P})(r)) = (prnt_T^{-1} \rhd \widehat{V_P})(p)$ **then**
4 **return false**;
5 **end if**;
6 **return true**;
7 **end function**

Figure 4.7: Pseudocode for sub-routine ROOTS(_, _, _).

to c's parent set. The pseudocode for ROOTS(_, _, _) is shown in Figure 4.7. Again the inputs are an isomorphism, the target and the pattern. The procedure is the dual of the previous one. It checks that P's roots allow the composition $C \circ (P \otimes \mathsf{id}_I)$. Namely, the loop in lines 2-5 checks that for every node or root p in C having children in $\widehat{V_P}$, there exists a set of roots R in P whose set of children is isomorphic to p's set of children. The final condition on the isomorphisms is checked by invoking TRANS($T, \widehat{V_P}$), pseudocode for this sub-routine is given in Figure 4.8. In line 2, there is a check that a node in $\widehat{V_P}$ does not have a parent in the context that has an ancestor in $\widehat{V_P}$. Observe that each $\iota \in I'$ satisfies all the compatibility conditions. Hence, it is a support translation for P with range $\widehat{V_P}$.

The second loop in MATCH(_, _) (lines 10-16 in Figure 4.4) builds a mapping between P's and T's links. The solutions are stored in set M. If no mapping is returned and there are no ports in the pattern, then only the isomorphisms are returned (lines 14-16). A mapping for every ι found in the previous phases of the algorithm is obtained by invocation BUILD_LINK_MAP($\emptyset, (E_P \uplus Y) \setminus P^{\mathsf{L}}_{idle}, \iota, T, P$) in line 12. Since idle links can always occur in any pattern, only non-idle links of P are considered as shown by the second argument. Pseudocode for the sub-routine is given in Figure 4.9.

The first argument η is a partial mapping. It is used by the procedure to recursively construct a full mapping. The second argument L is a subset of T's link. Its elements are

```
1  function TRANS(T, V̂_P)

2  if exists u ∈ V_T \ V̂_P and v, v' ∈ V̂_P such that u ∈ prnt_T(v) and v' ∈ prnt_T^+(u) then
3        return false;
4  else
5        return true;
6  end if;
7  end function
```

Figure 4.8: Pseudocode for sub-routine TRANS(_, _).

the links not in the range of η. The other three arguments are an isomorphism, the target and the pattern, in this order. When argument L is empty, this means that η contains a mapping for every non-idle links in the pattern. Therefore, η is accepted as a valid result in line 20. On the other hand, when L is not empty, a link l is selected in line 3. Then, the operations in lines 5-12 construct a set K whose elements are all the links in the target that can be mapped to l. There are three cases:

- If l is an outer name (i.e. $\tau(l) = \mathsf{name}$), then l can be mapped to any link $k \in K$ such that l's ports are an isomorphic subset of k's ports. This corresponds to the conditional statement in lines 5-6.

- If l is an edge and all its points are ports (i.e. $\tau(l) = \mathsf{edge}$), then l can be mapped to any edge $k \in K$ such that l's ports are isomorphic to k's ports. This corresponds to the instructions in lines 7-8.

- If l is an edge having an inner name $x \in X$ as a point (i.e. $\tau(l) = \mathsf{open\text{-}edge}$), then l can be mapped to any edge $k \in K$ such that l's ports are an isomorphic subset of k's ports and the other k's ports belong to nodes who are not ancestors of any node in $\widehat{V_P}$. In other words, they must belong to nodes in the parameter. This is described by the pseudocode in lines 9-11.

Observe that port identifiers are ignored because all comparisons are performed on multisets of nodes generated by function $count(\_)$. The loop in lines 13-17 builds res, the output of the sub-routine. This is the result of one of the recursive invocations BUILD_LINK_MAPS$(\eta \cup \{(l,k)\}, L \setminus \{l\}, \iota,$ REM_PORTS$(T,k,S), P)$. By analysing the arguments of the sub-routine, we see that pair (l,k) is added to partial solution η and l is removed from L, the set of links yet to be checked. Moreover, target T is replaced by a bigraph T' resulting from invocation REM_PORTS(T,k,S), with multiset S defined in line 4. Intuitively, T' is defined exactly as T but each port corresponding to a node in S is removed from the set of ports of link k. In this way, we ensure that the k's ports used to match a link in the pattern are not

```
1   function BUILD_LINK_MAP(η, L, ι, T, P)

2   if L ≠ ∅ then
3       choose l ∈ L;
4       S := ι(count_P(l));
5       if τ(l) = name then
6           K := {k | S ⊆ count_T(k)};
7       else if τ(l) = edge then
8           K := {k ∈ E_T | S = count_T(k)};
9       else
10          K := {k ∈ E_T | S ⊆ count_T(k) ∧
11              (not exists v ∈ V̂_P, u ∈ (count_T(k) \ S) s.t. (v, u) ∈ prnt_T^+)};
12      end if;
13      forall k ∈ K do
14          res := BUILD_LINK_MAP(η ∪ {(l, k)}, L \ {l}, ι, REM_PORTS(T, k, S), P);
15          if res ≠ ∅ then
16              return res;
17          end if;
18      return ∅;
19  else
20      return η;
21  end if;
22  end function
```

Figure 4.9: Pseudocode for sub-routine BUILD_LINK_MAP(_, _, _, _, _)

used again to match a different link in a successive recursive invocation. If no mapping can be constructed, either because all recursive invocations return $\emptyset$ or because $K = \emptyset$, then BUILD_LINK_MAP(_, _, _, _, _) returns an empty mapping in line 18 and the partial solution η is discarded. This because only total mappings can be valid solutions.

Pseudocode defining sub-routine REM_PORTS(_, _, _) is given in Figure 4.10. The loop in lines 3-5 iterates over multiset S which contains all the nodes v having ports already been used in a match, thus to be removed. In line 4, a fresh port corresponding to v is chosen and it is added to res in line 5. In line 6, the link map of T is updated by removing all the ports in res from the set of ports of k. Bigraph T' is returned in line 7.

The computational complexity of the algorithm is dominated by the running time of sub-routine SUB_ISO(_, _, _), which is $\mathcal{O}(|V_T|^{|V_P|})$.

4.2.2 Examples

We illustrate the various phases of algorithm MATCH(_, _) through some examples.

Example 4.2.2. Consider concrete bigraphs T and P_0, with link graphs assumed to be $\mathsf{id}_\emptyset$.

1 **function** REM_PORTS(T, k, S)

2 $res := \emptyset$;
3 **forall** $v \in S$ **do**
4 **choose** i such that $(v, i) \in ports_T(k)$ and $(v, i) \notin res$;
5 $res := res \cup \{(v, i)\}$;
6 $link'_T := link_T \setminus \{(p, k) \mid p \in res\}$;
7 **return** $\langle T^{\mathsf{P}}, (V_T, E_T, ctrl_T, link'_T)\rangle$;
8 **end function**

Figure 4.10: Pseudocode for sub-routine REM_PORTS$(_, _, _)$.

The corresponding place graphs are drawn in Figure 4.11. The matching instance is given by MATCH(T, P_0). The first operation performed by the algorithm is the computation of the underlying graphs $\mathcal{G}_{P_0}$ and $\mathcal{G}_T$ and the invocation of SUB_ISO$(\mathcal{G}_T, \mathcal{G}_{P_0})$. The result is the set of all the isomorphisms preserving edge (v_0, v_1) in $\mathcal{G}_{P_0}$. The ones that also preserve the controls are:

$$\iota_0 = \{(v_0, u_0), (v_1, u_1)\} \qquad \iota_1 = \{(v_0, u_0), (v_1, u_2)\} \qquad \iota_2 = \{(v_0, u_3), (v_1, u_6)\}$$
$$\iota_3 = \{(v_0, u_4), (v_1, u_7)\} \qquad \iota_4 = \{(v_0, u_5), (v_1, u_7)\} \ .$$

They are the isomorphisms satisfying the condition in CTRL$(\iota_i, ctrl_T, ctrl_{P_0})$. To check whether a solution has sites allowing for a valid match, sub-routine SITES(ι_i, T, P_0) is invoked. When ι_0 is the input, the loop iterates over $\{u_4, u_2, u_3\}$. Since there are no sites in P_0 having a parent set isomorphic to $prnt_T(u_2) = prnt_T(u_3) = \{u_0\}$, ι_0 is discarded. Same for ι_1. Similarly, ROOTS(ι_i, T, P_0) is invoked to check the roots of P_0. When ι_3 is the input, the loop iterates over $\{u_1, u_5\}$. In this case, there are no roots in P_0 having a set of children isomorphic to $prnt_T^{-1}(u_5) = \{u_7\}$. Therefore, ι_3 is discarded. The same for ι_4. Hence, the only control preserving isomorphism with compatible sites and roots is ι_2. The last check to be performed is TRANS$(T, \widehat{V_{P_0}})$, with $\widehat{V_{P_0}}$ being the range of ι_2. Since node u_0 (i.e. the only parent of $\iota_2(v_0) = u_3$) has no ancestors in $\widehat{V_{P_0}}$, the final output of MATCH(T, P_0) is $\{(\iota_2, \emptyset)\}$. No mapping η is created because there are no ports (check on line 14 in Figure 4.4). ■

Example 4.2.3. Take concrete bigraphs T and P_1 with empty link graphs. Their diagrams are given in Figure 4.12. When sub-routine MATCH(T, P_1) is invoked the following steps can be described. The isomorphisms constructed by SUB_ISO$(\mathcal{G}_T, \mathcal{G}_{P_1})$ that also satisfy the condition in CTRL$(\iota_i, ctrl_T, ctrl_{P_1})$ are

$$\iota_0 = \{(v_0, u_0), (v_1, u_1), (v_2, u_4)\} \qquad \iota_1 = \{(v_0, u_0), (v_1, u_2), (v_2, u_5)\} \ .$$

When SITES(ι_0, T, P_1) is invoked, the loop iterates over set $\{u_2, u_3, u_7\}$. Isomorphism ι_0

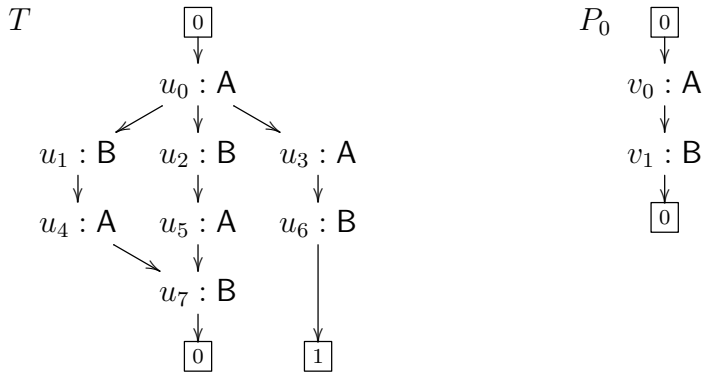


Figure 4.11: Bigraphs $T : 2 \to 1$ and $P_0 : 1 \to 1$. The only compatible isomorphism is $\{(v_0, u_3), (v_1, u_6)\}$.

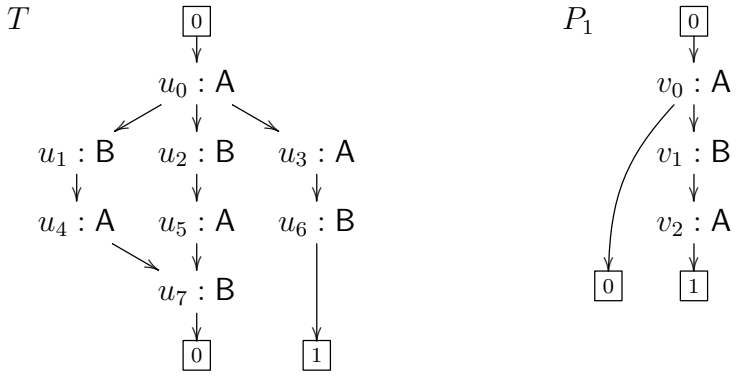


Figure 4.12: Bigraphs $T : 2 \to 1$ and $P_1 : 2 \to 1$. The compatible isomorphisms are $\{(v_0, u_0), (v_1, u_1), (v_2, u_4)\}$ and $\{(v_0, u_0), (v_1, u_2), (v_2, u_5)\}$.

is accepted because there are compatible sites, namely $prnt_{P_1}(0) = \{v_0\}$ and $prnt_{P_1}(1) = \{v_2\}$. When the input is ι_1, the set of children used for the iteration is $\{u_1, u_7, u_3\}$. Also in this case compatible sites exist. Hence, ι_1 is accepted. Since the parent sets to iterate over are empty, ROOTS(ι_i, T, P_1) returns **true**. The same for TRANS$(T, \widehat{V_P})$. No mapping η need be constructed. Therefore, the final output of MATCH(T, P_1) is $\{(\iota_0, \emptyset), (\iota_1, \emptyset)\}$. ■

Example 4.2.4. Take concrete bigraphs T and P_2 as in Figure 4.13. Both the link graphs are $\mathsf{id}_\emptyset$. The results of the invocation of sub-routine SUB_ISO$(\mathcal{G}_T, \mathcal{G}_{P_2})$ are

$$\iota_0 = \{(v_0, u_0), (v_1, u_1), (v_2, u_4), (v_3, u_7)\} \quad \iota_1 = \{(v_0, u_0), (v_1, u_2), (v_2, u_5), (v_3, u_7)\} \ .$$

Both isomorphisms satisfy the condition in CTRL$(\iota_i, ctrl_T, ctrl_{P_2})$. Sites are checked by the iterations over sets of children $\{u_2, u_3\}$ and $\{u_1, u_3\}$ for ι_0 and ι_1, respectively. In both cases, the existence of site 0 in P_2 causes the invocations of SITES(ι_i, T, P_2) to return **true**. The invocations of ROOTS(ι_i, T, P_2) iterate over $\{u_5\}$ and $\{u_4\}$ for ι_0 and ι_1, respectively.

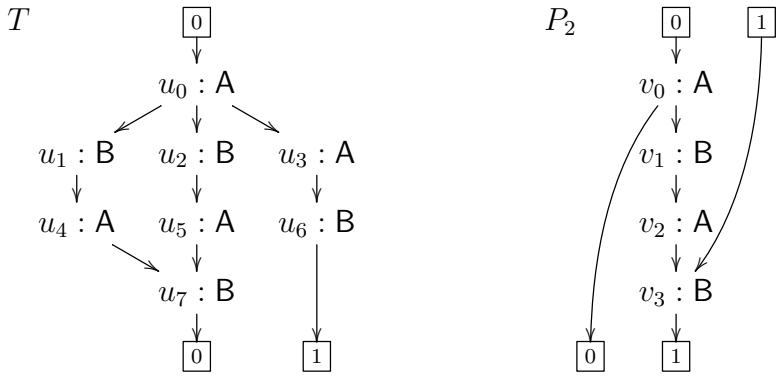


Figure 4.13: Bigraphs $T : 2 \to 1$ and $P_2 : 2 \to 2$. No matches are possible.

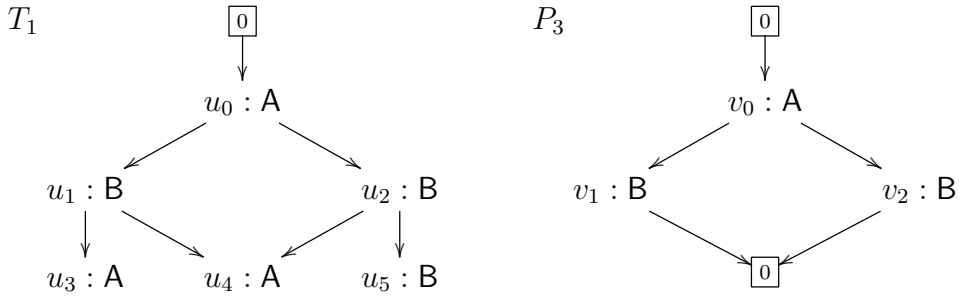


Figure 4.14: Bigraphs $T_1 : 0 \to 1$ and $P_3 : 1 \to 1$. No matches are possible.

Root 1 in P_2 is compatible for both isomorphisms. Finally the transitive closure is checked. When $\textsc{Trans}(T, \widehat{V_{P_0}})$ is invoked for ι_0, node u_5 forces the sub-routine to return **false**. This because $u_0 \in prnt_T^+(u_5)$. Analogously, ι_1 is discarded because $u_0 \in prnt_T^+(u_4)$. Therefore, $\textsc{Match}(T, P_2) = \emptyset$, i.e. P_2 does not occur in T. ■

Example 4.2.5. Consider instance $\textsc{Match}(T_1, P_3)$ in which target T_1 and pattern P_3 are given in Figure 4.14. Sub-routine $\textsc{Sub\_iso}(\mathcal{G}_{T_1}, \mathcal{G}_{P_2})$ produces three isomorphisms. The only one that is accepted by $\textsc{Ctrl}(\iota, ctrl_{T_1}, ctrl_{P_3})$ is $\iota = \{(v_0, u_0), (v_1, u_1), (v_2, u_2)\}$. The loop in $\textsc{Sites}(\iota, T_1, P_3)$ iterates over set $\{u_3, u_4, u_5\}$. The sub-routine returns **false** because P_3 does not have a set of sites capable of providing a parent set to node u_3. More formally, there is no site s such that $\iota(prnt_{P_3}(s)) = prnt_{T_1}(u_3) = u_1$. Note that it is also impossible to find a set of sites for u_5. Therefore, P_3 is not a match in T_1 and the result of the algorithm is $\textsc{Match}(T_1, P_3) = \emptyset$. ■

Example 4.2.6. Take concrete bigraphs T_2 and P_4 specified by the diagrams in Figure 4.15. Invocation $\textsc{Match}(T_2, P_4)$ executes the first loop of the algorithm which returns $I' = \{\iota\}$ with $\iota = \{(v_0, u_0), (v_1, u_1)\}$. In the second loop, one of the following mappings between links can be constructed, depending on the order of the recursive invocations of sub-routine

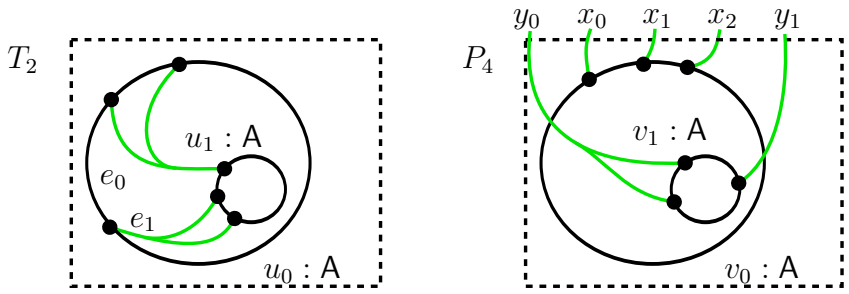


Figure 4.15: Bigraphs T_2 and P_4.

BUILD_LINK_MAP(_, _, _, _, _) in line 14 (see Figure 4.9):

$$\eta_0 = \{(x_0, e_1), (x_1, e_0), (x_2, e_0), (y_0, e_1), (y_1, e_0)\}$$
$$\eta_1 = \{(x_0, e_0), (x_1, e_0), (x_2, e_1), (y_0, e_1), (y_1, e_0)\}$$
$$\eta_2 = \{(x_0, e_0), (x_1, e_1), (x_2, e_0), (y_0, e_1), (y_1, e_0)\} \ .$$

The topmost invocation in MATCH(_, _) is always BUILD_LINK_MAP($\emptyset, X, \iota, T_2, P_4$), with $X = \{x_0, x_1, x_2, y_0, y_1\}$. Let us analyse the instructions performed by the sub-routine in more detail. If x_0 is chosen in line 3, we have $S = \iota(\{\{v_0\}\}) = \{\{u_0\}\}$ and $K = \{e_1, e_0\}$. This set is obtained on lines 5-6 because $S \subseteq count_{T_2}(e_i)$ with

$$count_{T_2}(e_1) = \{\{u_0, u_1, u_1\}\} \qquad count_{T_2}(e_0) = \{\{u_0, u_0, u_1\}\} \ .$$

Assuming e_1 is the first element of K to be picked by the loop in lines 13-17, the next invocation is BUILD_LINK_MAP($\{(x_0, e_1)\}, X \setminus \{x_0\}, \iota, T_2', P_4$) where T_2' is the result of REM_PORTS($T_2, e_1, \{\{u_0\}\}$). The only difference with T_2 is that we now have $count_{T_2'}(e_1) = \{\{u_1, u_1\}\}$. This is the result of removing pair $((u_0, i), e_1)$ from $link_{T_2}$ for some port identifier i. The final output of the algorithm when executing the sequence of invocations in the order described above is $M = \{(\iota, \eta_0)\}$. Observe that $x_0 \doteq x_1 \doteq x_1$ and that any η_i can be obtained by swapping equivalent names in another mapping. For instance, η_2 is the result of swapping x_1 and x_2 in η_1. As a result, the decompositions induced by the three mappings differ only by renamings on the inner names of the context. Therefore, all the mappings describe the same occurrence and the sequence of invocations chosen during the execution of the algorithm is irrelevant. ■

4.2.3 Soundness and completeness

We now prove that the algorithm is sound and complete. Informally, soundness is proven by showing that any solution obtained as output of MATCH(T, P) identifies an occurrence of P in T. This means that a context and a parameter can be obtained by decomposing T

as specified by Definition 2.3.10. Dually, completeness is proven by showing that when P occurs in T, then the algorithm produces a solution that identifies the decompositions of T induced by match P. Before starting, we present some useful properties characterising composition of concrete place graphs. We will use these results in the following proofs.

Proposition 4.2.1. *Take two place graphs $G : m \to n$ and $F : k \to m$ such that*

$$B : k \to n = G \circ F \ .$$

For any node $v \in V_F$, if $(v', v) \in prnt_B^+$, then $v' \in (k \uplus V_F)$.

Informally, the proposition above states that all the descendants of a node in F are also in F. We prove it as follows:

Proof. Assume the hypothesis holds but $v' \notin (k \uplus V_F)$. Then $v' \in V_G$. Since $v \in V_F$ and $(v', v) \in prnt_B^+$, we have

$$(v', v_0), \ldots, (v_i, m_i) \in prnt_G \qquad (m_i, v_{i+1}), \ldots, (v_{i+j}, v) \in prnt_F$$

for some $m_i \in m$. However, by Definition 3.2.1, $prnt_G \subseteq (m \uplus V_G) \times (V_G \uplus n)$. Therefore, $(v_i, m_i) \notin prnt_G$ because $m_i \notin (V_G \uplus n)$. Similarly, $(m_i, v_{i+1}) \notin prnt_F$ because $m_i \notin (k \uplus V_F)$. This contradicts the hypothesis. Hence, $v' \in (k \uplus V_F)$. □

Another property specifies that in a composition $G \circ F$, when v, a site or node in V_F, has some parents which are roots or nodes in V_G, then a set of sites in G has exactly the same parents. Moreover, this set corresponds to the roots in F belonging to the parent set of v. This allows for v's parent set to be constructed in the composition. We formalise the property as follows:

Proposition 4.2.2. *Take two place graphs $G : m \to n$ and $F : k \to m$ such that*

$$B : k \to n = G \circ F \ .$$

For any node or site $v \in (k \uplus V_F)$ such that $(prnt_B \rhd (V_G \uplus n))(v) = P$, with $P \neq \emptyset$, there exists a set $S \subseteq m$ such that

$$(prnt_F \rhd m)(v) = S \qquad \bigcup_{s' \in S} prnt_G(s') = P$$

Proof. Immediate by Definition 3.2.2. Assume S does not exist. Then, it means that $prnt_F(v) \subseteq V_F$. But $(prnt_B \rhd (V_G \uplus n))(v) = \emptyset$, so we have a contradiction. □

The dual property on the children set of a node or root in G also holds:

Proposition 4.2.3. *Take two place graphs $G : m \to n$ and $F : k \to m$ such that*

$$B : k \to n = G \circ F \ .$$

For any node or root $v \in (V_G \uplus n)$ such that $(prnt_B^{-1} \rhd (k \uplus V_F))(v) = C$, with $C \neq \emptyset$, there exists a set $R \subseteq m$ such that

$$(prnt_G^{-1} \rhd m)(v) = R \qquad \bigcup_{r' \in R} prnt_F^{-1}(r') = C$$

Proposition 4.2.4 (soundness)**.** *Let T and P be two concrete bigraphs, with $V_P \neq \emptyset$. If* MATCH(T, P) *returns a solution (ι, η), then (ι, η) identifies an occurrence of P in T.*

Proof. Recall that a decomposition takes the form $T = C \circ (P' \otimes \mathsf{id}_{\langle j,J \rangle}) \circ D$ with $P' = \rho \cdot P$. We begin by proving the decomposition over place graphs.

By construction $\iota = \rho_V$ and $\widehat{V_P} = V_{P'}$. Set V_C is defined to contain all the nodes in V_T having a descendant in $V_{P'}$. It follows that $V_D = V_T \setminus (V_C \uplus V_{P'})$. The interfaces of the context and the parameter are $C : n + j \to n'$ and $D : m' \to m + j$, respectively. Parameter j is the number of nodes or sites in D having a parent in C. By definition of composition, the parent relations are

$$prnt_D \rhd V_D = prnt_T \rhd V_D \qquad V_C \lhd prnt_C = V_C \lhd prnt_T \ .$$

Relations $prnt_D \rhd j$ and $j \lhd prnt_C$ contain the pairs (u, j_i) and (j_i, v), respectively, for every $(u, v) \in prnt_T$, with $u \in (m' \uplus V_D)$ and $v \in (V_C \uplus n')$. Finally, $prnt_D \rhd m$ is the relation containing pairs (c, s) identified by procedure SITES(ι, T, P). Note that $c \in (m' \uplus V_D)$ and $s \in m$. In a similar fashion, relation $n \lhd prnt_C$ is formed by pairs (r, p) detected by ROOTS(ι, T, P), with $r \in n$ and $p \in (V_C \uplus n')$.

We now construct a decomposition for link graphs. By construction $E_P \lhd \eta = \rho_E$ and $E_{P'} = \rho_E(E_P)$. Set E_D contains all the edges in E_T that have a port set entirely over nodes in V_D. The edges of the context are $E_C = E_T \setminus (E_{P'} \uplus E_D)$. The interfaces in the link graphs of the decomposition are $C : Y \uplus J \to Y'$ and $D : X' \to X \uplus J$. Set J contains a name for each point in D being mapped to a link in C. Therefore, identity id_J is obtained with a construction similar to the one described for id_j: pairs (p, w), (w, l), with $w \in J$, are added to $link_D \rhd J$ and $J \lhd link_C$, respectively, for every $(p, l) \in link_T$. Therefore, the complete link map for C is given by

$$link_C = (J \lhd link_C) \uplus (Y \lhd \eta) \uplus (P_C \lhd link_T) \ .$$

Relation $link_D \rhd X$ contains the pairs (p, x), with $p \in (X' \uplus P_D)$ and $x \in X$, for every

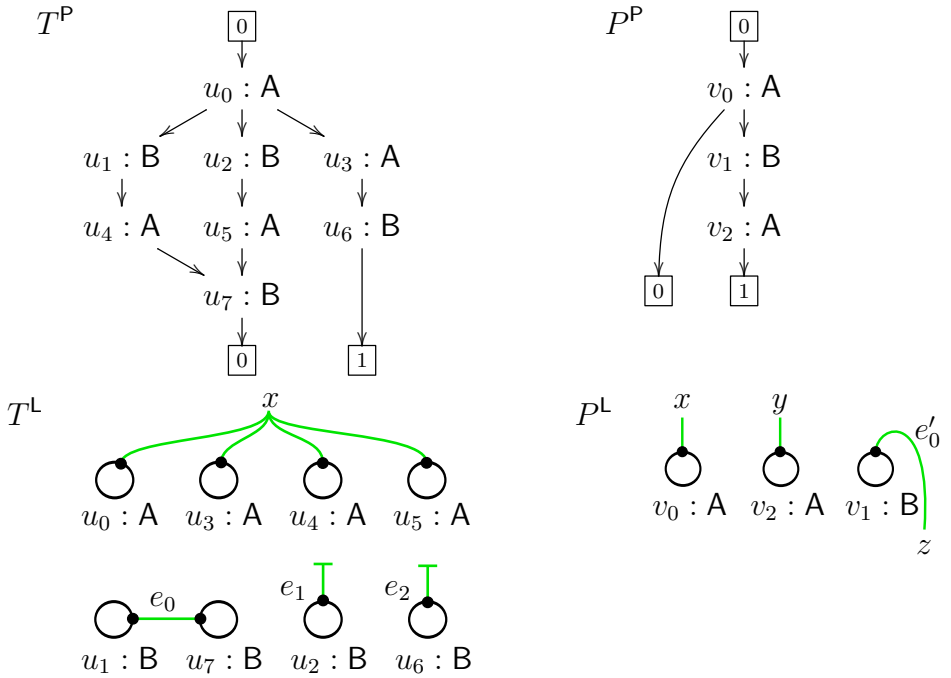


Figure 4.16: Link graphs and place graphs for target $T : 2 \to \langle 1, \{x\}\rangle$ and pattern $P : \langle 2, \{z\}\rangle \to \langle 1, \{x, y\}\rangle$.

$(p, e) \in link_T$ and $(x, e) \in link_{P'}$. Finally, the complete link map for D is

$$link_D = (link_D \rhd J) \uplus (link_D \rhd X) \uplus (link_T \rhd E_D) \ .$$

This concludes the proof. □

The constructive proof described above can be used as a procedure for the decomposition of a target given a pattern and a pair (ι, η). This is explained by the following example.

Example 4.2.7. Consider bigraphs T and P given in Figure 4.16. Note that the place graphs correspond to the bigraphs used in Example 4.2.3. The output of $\textsc{Match}(T, P)$ is

$$(\iota_0, \eta_0) = (\{(v_0, u_0), (v_1, u_1), (v_2, u_4)\}, \{(x, x), (y, x), (e'_0, e_0)\})$$
$$(\iota_1, \eta_1) = (\{(v_0, u_0), (v_1, u_2), (v_2, u_5)\}, \{(x, x), (y, x), (e'_0, e_1)\}) \ .$$

We now construct a decomposition $T = C \circ (P' \otimes \mathsf{id}_I) \circ D$ by using solution (ι_0, η_0). The first step consists in defining $V_{P'}$ as the range of ι_0 and $E_{P'}$ as the range of $E_P \lhd \eta_0$:

$$V_{P'} = \{u_0, u_1, u_4\} \qquad\qquad E_{P'} = \{e_0\} \ .$$

The second step is to define context C. Since there are no nodes in T that are a parent of a node in $V_{P'}$, we set $V_C = \emptyset$. Sub-routine $\textsc{Roots}(\iota_0, T, P)$ can only identify $(0, 0)$. This

Figure 4.17: Place and link graphs for context $C : \langle 1, \{x, y, w, t\}\rangle \to \langle 1, \{x\}\rangle$.

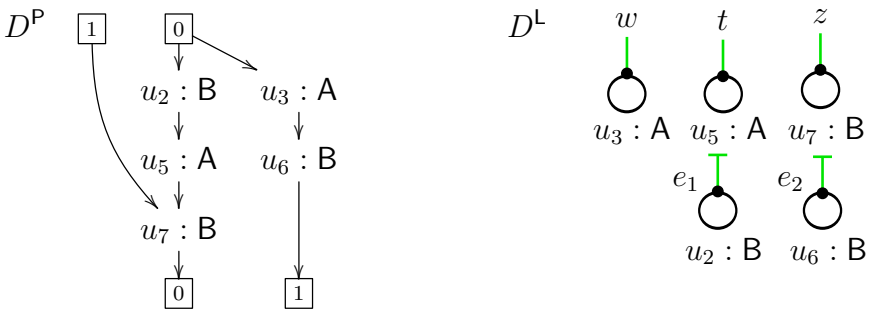


Figure 4.18: Place and link graphs for parameter $D : 2 \to \langle 2, \{z, w, t\}\rangle$.

is also the only pair in $prnt_C$. The edge set of the context is $E_C = \emptyset$ because all the edges in T have port sets entirely over V_D. Finally, two fresh names w, t need be introduced in order to break pairs $((u_3, 0), x)$, $((u_5, 0), x)$ in $link_T$. The full definition of context $C : \langle 1, \{x, y, w, t\}\rangle \to \langle 1, \{x\}\rangle$ is given in Figure 4.17. The final step is to define parameter D. The node and edge sets are

$$V_D = \{u_2, u_3, u_5, u_6, u_7\} \qquad E_D = \{e_1, e_2\} \ .$$

Sub-routine SITES(ι_0, T, P) detects pairs $(u_7, 1)$, $(u_2, 0)$ and $(u_3, 0)$. The other pairs in $prnt_D$ are taken from $prnt_T \rhd V_D$. The link map is constructed by adding the following three pairs to the pairs in $link_T \rhd E_D$:

$$((u_3, 0), w) \qquad ((u_5, 0), t) \qquad ((u_7, 0), z) \ .$$

Diagrams for the place and the link graphs of $D : 2 \to \langle 2, \{z, w, t\}\rangle$ are drawn in Figure 4.18. ■

Proposition 4.2.5 (completeness)**.** *Let T and P be two concrete bigraphs, with $V_P \neq \emptyset$. If P is a match in T, then* MATCH(T, P) *returns a solution (ι, η) for every occurrence of P in T.*

Proof. By Definition 2.3.10, the hypothesis is there exists C, D such that $T = C \circ (P' \otimes \mathsf{id}_I) \circ D$ with $P' = \rho \cdot P$. By Definition 2.2.5 and Definition 4.2.1, ρ_V is a control-preserving

isomorphism from $\mathcal{G}_P$ to $\mathcal{G}_{P'}$. Since $V_{P'} \subseteq V_T$, ρ_V is an output of SUB_ISO$(\mathcal{G}_T, \mathcal{G}_P)$ and CTRL$(\rho_V, ctrl_T, ctrl_P)$ returns **true**. By Proposition 4.2.2 and by hypothesis $(P' \otimes \mathsf{id}_I) \circ D$, SITES$(\rho_V, T, P)$ returns **true**. Similarly, by Proposition 4.2.3 and by hypothesis $C \circ (P' \otimes \mathsf{id}_I)$, ROOTS$(\rho_V, T, P)$ returns **true**. The hypothesis and Proposition 4.2.1 also ensure that TRANS$(T, \widehat{V_P})$ returns **true**. We now show that the algorithm returns a valid link mapping η. By Definition 2.3.2, we know that when $\tau_{P'}(e) =$ edge holds for an edge $e \in E_{P'}$, then $\tau_T(e) =$ edge also holds. Therefore, $\rho_E \subset \eta$ because of lines 7-8 in BUILD_LINK_MAP$(_, _, _, _, _)$. The other pairs in η are found by checks on lines 5-6 and 10-11. The existence of these pairs follows from the hypothesis and by Definition 2.2.3. This proves that (ι, η), with $\iota = \rho_V$, is a solution returned by MATCH(T, P) when the hypothesis holds. This concludes the proof. □

We conclude the chapter by proving the following proposition:

Proposition 4.2.6. *Algorithm* MATCH$(_, _)$ *natively supports non-sharing concrete bigraphs.*

Proof. We need to prove that whenever T and P are non-sharing, then context C and parameter D in *any* decomposition constructed by MATCH(T, P) are also non-sharing. We begin by proving that D is non-sharing.

Assume D is a sharing bigraph. Then by Definition 3.2.1 its place graph can contain i) orphans and ii) shared places. We prove the two cases separately.

i) Let c be an orphan in D. Since c is also a place in T, then by Proposition 4.2.4

$$prnt_D(c) = prnt_T(c) = \emptyset \ .$$

But T is non-sharing, therefore $|prnt_T(c)| = 1$. This is a contradiction.

ii) Without loss of generality, let $prnt_D(c) = \{p, p'\}$ with $\{p, p'\} \subseteq (V_D \uplus j)$. There are six cases:

1) If $\{p, p'\} \subseteq V_D$, then by Proposition 4.2.4 $prnt_D \rhd V_D = prnt_T \rhd V_D$. Therefore, $prnt_T(c) = \{p, p'\}$. But this is a contradiction because T is non-sharing by hypothesis.

2) If $p \in V_D$ and $p' \in m$, then by Proposition 4.2.4 and by the fact that P is non-sharing, there exists one $p'' \in V_{P'}$ such that $(c, p'') \in prnt_T$. However, Proposition 4.2.4 also implies that $(c, p) \in prnt_T$, which contradicts the hypothesis.

3) If $p \in V_D$ and $p' \in j$, then by Proposition 4.2.4 there exists one $p'' \in V_C$ such that $(c, p'') \in prnt_T$. As in the previous case, this contradicts the hypothesis.

4) If $\{p, p'\} \subseteq m$ then by Proposition 4.2.4 and by the fact that P is non-sharing, we have that $|prnt_T(c)| = 2$. But this is a contradiction because T is non-sharing by hypothesis.[2]

5) If $\{p, p'\} \subseteq j$ then c has a parent in C. The construction of id_j described in Proposition 4.2.4 ensures the minimality of j. Therefore, there is only one site in id_j connecting c with its parent in C. This implies that $(c, p') \notin prnt_D$. Contradiction.

6) If $p \in m$ and $p' \in j$, then by Proposition 4.2.4 and by the fact that P does not have any orphan sites, we have that $\{(c, v), (c, p'')\}) \in prnt_T$, with $v \in V_{P'}$ and p'' in C. This is a contradiction because T is non-sharing by hypothesis.

The proof for C is similar and thus omitted. □

4.3 Summary

In this chapter we defined a graph theoretic matching algorithm for bigraphs with sharing. In Section 4.1, we gave an overview of the problem with an example instance. We also briefly described an algorithm for non-sharing bigraphs based on inference rules. Furthermore, we discussed the relationship of matching with related computational problems. The formal definition of the algorithm was introduced in Section 4.2. We showed that it consists of a reduction to the sub-graph isomorphism problem and we proved its soundness and completeness.

This chapter concludes the first part of this thesis that was devoted to the theory of bigraphs with sharing. In the next part the focus will shift to applications of bigraphs. We will describe BigraphER, a software tool based on a SAT[3] encoding of our matching algorithm that allows manipulation and visualisation of bigraphs and efficient computation of BRS and SBRS. Then we will model a communication protocol for wireless networks and describe real-time verification in a home-network environment.

[2] Observe that by definition of procedure SITES(_, _, _), it is impossible to have $\{(p, v), (p', v)\} \subseteq prnt_P$ if $prnt_D(c) = \{p, p'\}$.

[3] Boolean satisfiability problem.

Part II

Applications

Chapter 5

BigraphER: Bigraph Evaluator & Rewriting

In this chapter we describe BigraphER, an implementation of BRS and SBRS that natively supports place graphs with sharing. It is based on the graph theoretic algorithm presented in Chapter 4.

Section 5.1 gives a brief overview of the architecture of the system. In Section 5.2, data types for the representation of bigraphs with sharing and their constituents are defined and then, the details of our approach are explained by analysing the manipulation and visualisation routines, the matching engine based on a SAT encoding of matching and the rewriting engine for the computation of a reaction relation in a BRS. Section 5.3 introduces a method based on matching to check a class of **BiLog** predicates. Finally, a summary of the chapter is provided in Section 5.4.

5.1 Overview

The BigraphER system consists of an OCaml library and a command-line tool that provides efficient manipulation and visualisation of bigraphs and simulation of BRS and stochastic BRS. A prototype implementation can be downloaded from the website `http://dcs.gla.ac.uk/~michele/bigrapher.html`.

We start off by describing the architecture of the command-line tool. The technical details of the implementation are given in Section 5.2. The tool is composed of three distinct modules: the *compiler*, the *matching engine* and the *rewriting engine*. Their interconnections are shown in Figure 5.1. The tool's input is a source file containing the model specification. The language used, called BigraphER *specification language*, closely resembles the algebraic form of bigraphs introduced in Chapter 3. This similarity can be observed in the example in

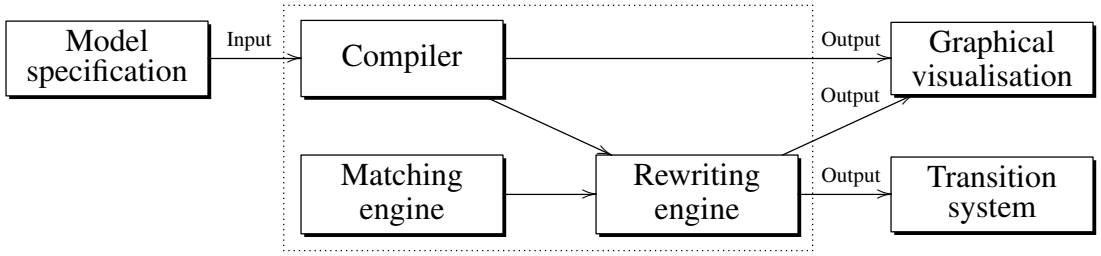


Figure 5.1: Architecture of the command-line tool. The modules are represented by the boxes within the dotted box. Unlabelled arrows show the dependency relation between the modules.

Figure 5.2, where a comparison between the two representations is shown. The BigraphER specification language allows one to define the signature of the model, a set of bigraphs and a set of reaction rules. An overview of the main features of the language are as follows.

A signature is specified by a sequence of control declarations of the following form:

$$\texttt{ctrl}\ ctrl\_name = int\texttt{;}$$

where $ctrl\_name$ is a string identifier. Each declaration introduces a new control and assigns an arity, i.e. integer int, to it. Variables representing bigraphs are declared in a similar fashion:

$$\texttt{big}\ [\texttt{init}]\ name = big\_exp\texttt{;}$$

where identifier $name$ is a string. The optional argument `init` allows one to specify a bigraph as the initial state of the model. Examples of big_exp expressions are the right-hand sides of the `big` declarations in Figure 5.2a. The operators on big_exp expressions are

$$\texttt{\_*\_} \quad \texttt{\_+\_} \quad \texttt{\_||\_} \quad \texttt{\_|\_} \quad \texttt{\_.\_} \quad \texttt{share\_by\_in}$$

They correspond to their algebraic counterparts: composition, tensor product, parallel product, merge product, nesting and share expressions, respectively. Recursive declarations are not allowed. Declarations of reaction rules have the form:

$$\texttt{react}\ name = big\_exp\ \texttt{->}\ big\_exp\texttt{;}$$

Finally, stochastic reaction rules are declared as follows:

$$\texttt{sreact}\ name = big\_exp\ \texttt{->}\ big\_exp\ \texttt{@}\ float\ \texttt{;}$$

where $float$ indicates the reaction rate.

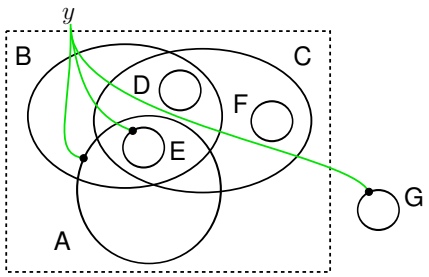
The compiler is the component that translates an input source file into a run-time repres-

```
ctrl A = 1; ctrl E = 1; ctrl G = 1;
ctrl B = 0; ctrl C = 0; ctrl D = 0; ctrl F = 0;
big b = share f by phi in g;
big f = D.1 || E({y}).1 || F.1 || G({y}).1;
big phi = ([{1,2},{0,1,2},{2},{}], 3);
big g = A({y}) | B | C;
```

(a)

$B = \text{share } F \text{ by } \phi \text{ in } G$
$F = \mathsf{D}.1 \parallel \mathsf{E}_y.1 \parallel \mathsf{F}.1 \parallel \mathsf{G}_y.1$
$\phi = [\{1,2\},\{0,1,2\},\{2\},\emptyset]$
$G = \mathsf{A}_y \mid \mathsf{B} \mid \mathsf{C}$

(b)



(c)

Figure 5.2: Comparision between a representation in the BigraphER specification language (a) and the algebric form (b) of bigraph $B : \epsilon \to \langle 1, y \rangle$ (c).

entation of the model. In more detail, each declaration specifies the binding of an identifier to a data type representing either a control, a bigraph, a reaction rule or a stochastic reaction rule. Controls and reaction rules are stored as OCaml integers and records, respectively. The encoding of a bigraph is somewhat more complex and requires two specialised data structures, one for the place graph and one for the link graph. They are described extensively in Section 5.2. Observe that the source file specifies abstract bigraphs. However, a concretion of a given bigraph is actually stored in memory. The compiler is used by the rewriting engine to retrieve the data structures corresponding to the initial state and to the reaction rules of the model. The compiler can also return a graphical representation of every bigraph specified in the input file. This is generated by the automatic graph-layout generator Graphviz [33].

The matching engine implements the matching algorithm for bigraphs with sharing introduced in Chapter 4. It is used by the rewriting engine to apply reaction rules to a state and to check equality of states. The implementation is based on a SAT encoding of the matching algorithm. Solutions are obtained by passing the SAT instance resulting from the encoding to the MiniSat solver [27].

The rewriting engine builds a graph representing the transition system (CTMC) corresponding to a BRS (SBRS). It is constructed by iteratively applying the reaction rules to each state and then storing the resulting states, until a fixed point is reached. This happens when all the bigraphs obtained by the application of the reaction rules are already present in the graph. Note that the model is assumed to have a finite state space. Therefore, no controls are performed to assure that the loop terminates. In order to avoid an abrupt termination when

the program runs out of memory, a switch specifying the number of iterations can be used when the tool is launched. The rewriting engine can return either a textual or a graphical representation of the graph. It can also output a graphical representation of each state.

The BigraphER OCaml library provides programming interfaces for the data structures used internally by the command-line tool. For instance, it is equipped with functions to compute the composition of two bigraphs and the result of a reaction application. A binding to the matching engine is also implemented. More details are given in the following.

5.2 Implementation

In this section, we describe our implementation of the main components of the BigraphER system. Namely, we discuss how bigraphs and functions for their manipulation are expressed in OCaml, we analyse and justify our SAT encoding of $\textsc{Match}(\_, \_)$ in the matching engine, and we explain our approach to the implementation of the rewriting engine and the visualisation functions for bigraphs. Some of these topics are covered in our previous work [57].

5.2.1 Bigraphical structures

As we previously mentioned, a concrete bigraph is represented in memory as two distinct data structures corresponding to its place and link graph. Functions implementing composition and tensor product are also implemented independently on the two structures. Observe that this mirrors the formal definition of bigraphs given in Chapter 3.

We begin by presenting the implementation of concrete place graphs. Our approach consists of representing DAGs with adjacency matrices. These are OCaml values of `bool array array` type. We chose this representation as it allows the implementation of fast in-place modification algorithms. Operations on place graphs are encoded by matrix operations. Formally, given a concrete place graph $F : s \to r$, with $|V_F| = n$, the corresponding $(s+n)$-by-$(n+r)$ boolean matrix is

$$\mathbf{F}_{(s+n)\times(n+r)} = \begin{bmatrix} \mathbf{S}_{s\times n} & \mathbf{N}_{s\times r} \\ \mathbf{V}_{n\times n} & \mathbf{R}_{n\times r} \end{bmatrix}$$

Integers are used as node identifiers, i.e. $V_F = \{0, \ldots, n-1\}$. Therefore, they can be interpreted as row or column indexes. A zero-based indexing of matrices is used. The four sub-matrices encode different subsets of $prnt_F$. Matrix $\mathbf{S}_{s\times n}$ corresponds to $(s \lhd prnt_F \rhd V_F)$. It stores the parents of the sites that are nodes. Matrix $\mathbf{N}_{s\times r}$ represents $(s \lhd prnt_F \rhd r)$, i.e. the parents of the sites that are roots. Similarly, the other two matrices record the parents

of the nodes of the place graph. Matrices $\mathbf{V}_{n\times n}$ and $\mathbf{R}_{n\times r}$ encode relations $(V_F \lhd prnt_F \rhd V_F)$ and $(V_F \lhd prnt_F \rhd r)$, respectively. Their elements are defined as follows:

$$m_{i,j} \stackrel{\text{def}}{=} \begin{cases} \textbf{true} & \text{if } (i,j) \in R \\ \textbf{false} & \text{otherwise} \end{cases}$$

where $m_{i,j}$ and R range over the elements of $\mathbf{S}_{s\times n}$, $\mathbf{N}_{s\times r}$, $\mathbf{V}_{n\times n}$, $\mathbf{R}_{n\times r}$ and the corresponding parent sub-relations, respectively.

Composition of concrete place graphs follows Definition 3.2.2. It is based on boolean matrix multiplication, i.e. row-by-column multiplication in which summation and product are $\vee$ and $\wedge$, respectively. Take two concrete bigraphs $F : s \to r$, $G : r \to r'$, with $|V_F| = n$ and $|V_G| = n'$. Their run-time representation is given below:

$$\mathbf{F}_{(s+n)\times(n+r)} = \begin{bmatrix} \mathbf{S}_{s\times n} & \mathbf{N}_{s\times r} \\ \mathbf{V}_{n\times n} & \mathbf{R}_{n\times r} \end{bmatrix} \qquad \mathbf{G}_{(r+n')\times(n'+r')} = \begin{bmatrix} \mathbf{S'}_{r\times n'} & \mathbf{N'}_{r\times r'} \\ \mathbf{V'}_{n'\times n'} & \mathbf{R'}_{n'\times r'} \end{bmatrix}$$

The result of the function implementing composition is place graph $G \circ F : s \to r'$, with $|V_{G\circ F}| = n + n' = m$. It is encoded by the following matrix:

$$[\mathbf{G} \circ \mathbf{F}]_{(s+m)\times(m+r')} = \begin{bmatrix} \mathbf{D}_{(s+n)\times n} & \mathbf{C}_{(s+n)\times(n'+r')} \\ \mathbf{0}_{n'\times n} & \mathbf{D'}_{n'\times(n'+r')} \end{bmatrix}$$

Sub-matrix $\mathbf{C}_{(s+n)\times(n'+r')}$ is the result of the following multiplication:

$$\mathbf{C}_{(s+n)\times(n'+r')} = \begin{bmatrix} \mathbf{N}_{s\times r} \\ \mathbf{R}_{n\times r} \end{bmatrix} \begin{bmatrix} \mathbf{S'}_{r\times n'} & \mathbf{N}_{r\times r'} \end{bmatrix}$$

The other sub-matrices are

$$\mathbf{D}_{(s+n)\times n} = \begin{bmatrix} \mathbf{S}_{s\times n} \\ \mathbf{V}_{n\times n} \end{bmatrix} \qquad \mathbf{D'}_{n'\times(n'+r')} = \begin{bmatrix} \mathbf{V'}_{n'\times n'} & \mathbf{R'}_{n'\times r'} \end{bmatrix}$$

Matrix $\mathbf{0}_{n'\times n}$ is an $n' \times n$ matrix with all elements set to `false`. It encodes the fact that, by construction, no node in G can have a parent in F. The four sub-matrices encode different subsets of $prnt_{G\circ F}$. Matrix $\mathbf{D}_{(s+n)\times n}$ represents $prnt^{\rhd}_{V_F}$, i.e. the places in F that are unaffected by the composition. Analogously, the places in G that do not change are encoded by $\mathbf{D'}_{n'\times(n'+r')}$. It corresponds to relation $prnt^{\lhd}_{V_G}$. Matrix $\mathbf{C}_{(s+n)\times(n'+r')}$ encodes the edges merged during composition. These are the pairs in relation $prnt_{\circ}$. Finally, $\mathbf{0}_{n'\times n}$ encodes $\emptyset$. Note that the procedure assures the supports are disjoint. This is because G's nodes are always below and to the right of F's nodes in the matrix representation. This corresponds to adding offset n to all the nodes in G. We illustrate the implementation of composition by

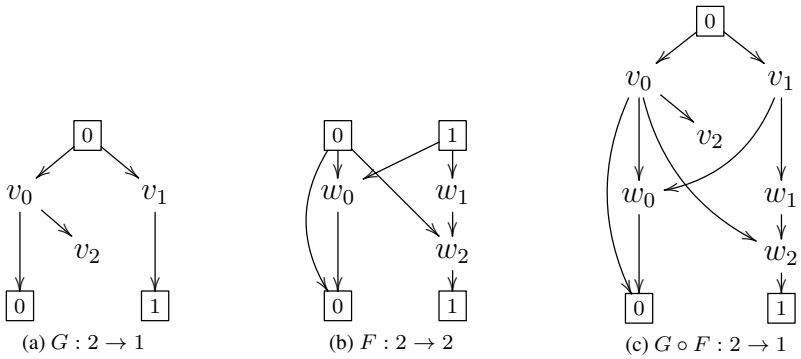


Figure 5.3: Example of composition for concrete place graphs with sharing.

showing how it works on the place graphs introduced in Example 3.2.1.

Example 5.2.1. Let concrete place graphs $G : 2 \to 1$, $F : 2 \to 2$ and their composition $G \circ F : 2 \to 1$ as in Figure 5.3. Their OCaml representation is

$$\mathbf{F}_{(2+3)\times(3+2)} = \begin{array}{c|ccc|cc} & w_0 & w_1 & w_2 & 0 & 1 \\ \hline 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ \hline w_0 & 0 & 0 & 0 & 1 & 1 \\ w_1 & 0 & 0 & 0 & 0 & 1 \\ w_2 & 0 & 1 & 0 & 1 & 0 \end{array} \qquad \mathbf{G}_{(2+3)\times(3+1)} = \begin{array}{c|ccc|c} & v_0 & v_1 & v_2 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ \hline v_0 & 0 & 0 & 0 & 1 \\ v_1 & 0 & 0 & 0 & 1 \\ v_2 & 1 & 0 & 0 & 0 \end{array}$$

where `1` and `0` are short-hands for values `true` and `false`. Row indexes are sites and nodes, while column indexes are nodes and roots. Horizontal and vertical delimiters in the representation help to highlight blocks $\mathbf{S}$, $\mathbf{N}$, $\mathbf{R}$ and $\mathbf{V}$ (clockwise from top left). The representation of composite place graph $G \circ F : 2 \to 1$ is

$$[\mathbf{G} \circ \mathbf{F}]_{(2+3+3)\times(3+3+1)} = \begin{array}{c|ccc|cccc} & w_0 & w_1 & w_2 & v_0 & v_1 & v_2 & 0 \\ \hline 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ w_0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ w_1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ w_2 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ \hline v_0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ v_1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ v_2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}$$

In this case, delimiters are introduced to show sub-matrices $\mathbf{D}$, $\mathbf{C}$, $\mathbf{D}'$ and $\mathbf{0}$ (clockwise from top left). The red and blue boxes aid the visualisation of the structure of blocks $\mathbf{D}$ and $\mathbf{D}'$,

respectively. Their building blocks in $\mathbf{F}$ and $\mathbf{G}$ are highlighted with the same colour. ■

The tensor product of concrete place graphs is computed by interleaving the matrices of the two factors as follows:

$$[\mathbf{F} \otimes \mathbf{G}]_{(s+s'+m)\times(m+r+r')} = \begin{bmatrix} \mathbf{S}_{s\times n} & \mathbf{0}_{s\times n'} & \mathbf{N}_{s\times r} & \mathbf{0}_{s\times r'} \\ \mathbf{0}_{s'\times n} & \mathbf{S'}_{s'\times n'} & \mathbf{0}_{s'\times r} & \mathbf{N'}_{s'\times r'} \\ \mathbf{V}_{n\times n} & \mathbf{0}_{n\times n'} & \mathbf{R}_{n\times r} & \mathbf{0}_{n\times r'} \\ \mathbf{0}_{n'\times n} & \mathbf{V'}_{n'\times n'} & \mathbf{0}_{n'\times r} & \mathbf{R'}_{n'\times r'} \end{bmatrix}$$

This is described in the following example:

Example 5.2.2. Take concrete place graphs $G : 2 \to 1$ and $F : 2 \to 2$ defined in Figure 5.3a and Figure 5.3b, respectively. Their run-time representations are given in Example 5.2.1. The representation of $G \otimes F : 4 \to 3$ is

$[\mathbf{G} \otimes \mathbf{F}]_{(2+2+3+3)\times(3+3+1+2)} =$

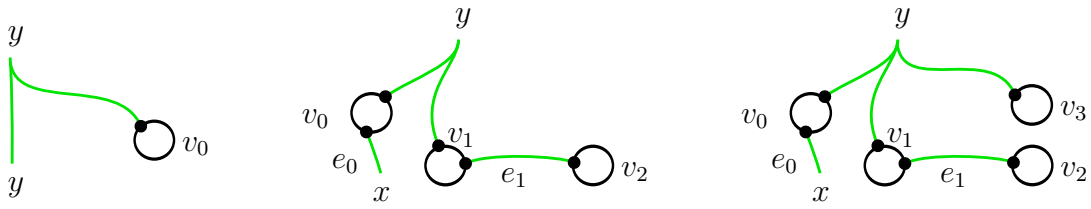
	v_0	v_1	v_2	w_0	w_1	w_2	0	0	1
0	1	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	1	0
1	0	0	0	0	0	1	0	0	0
v_0	0	0	0	0	0	0	1	0	0
v_1	0	0	0	0	0	0	1	0	0
v_2	1	0	0	0	0	0	0	0	0
w_0	0	0	0	0	0	0	0	1	1
w_1	0	0	0	0	0	0	0	0	1
w_2	0	0	0	0	1	0	0	1	0

Blue boxes indicate the sub-matrices taken from $\mathbf{G}$, while red boxes show the blocks taken from matrix $\mathbf{F}$. ■

Support translations are encoded by row and column swapping. For instance, assume $(i, j) \in \rho_V$, with $i \neq j$ and that $G : s \to r$. When computing $\rho_V \cdot G$, columns i and j and rows $(s + i)$ and $(s + j)$ are swapped.

We now present the OCaml implementation of concrete link graphs. Our representation of the hyper-graph structure characterising the link map consists of a set of pairs (type `Lg.t`) in which the first element is a link identifier and the second is the set of its points (type `Points.t`). Module `Set` provided by the OCaml standard library is used to implement all the set structures. Link identifiers have variant type

```
type link = Edg of int | O_nam of string .
```



(a) $G : \{y\} \to \{y\}$ (b) $F : \{x\} \to \{y\}$ (c) $G \circ F : \{x\} \to \{y\}$

Figure 5.4: Example of composition for concrete link graphs.

This reflects the fact that a link can either be an edge or an outer name. Similarly, points can be ports or inner names. They have type

```
type point = Port of (int * int) | I_nam of string .
```

The first element in `Port` is a node index. Note that by Definition 2.2.3 all the point-sets are disjoint.

The tensor product of two concrete link graphs is the union of the two sets of pairs, provided their inner and outer names are disjoint. In order to have disjoint supports, offsets are added to the ports and the edges of the right operand before the union is performed. This is implemented by invoking the library function `fold` twice. The first time it is used to add the number of edges of the left operand (`e_offset`) to every edge index in the set of pairs representing the right operand. The implementation is given by the following code

```
(Edg i, p) -> (Edg (i + e_offset), p) .
```

It is executed when iterating over a set of type `Lg.t`. The second invocation of `fold` adds the number of nodes of the left operand (`v_offset`) to every port in the point-set of each link. The code executed when iterating over a set of type `Points.t` is the following:

```
Port(i, j) -> Port(i + v_offset), j) .
```

Composition of concrete link graphs is implemented according to Definition 2.3.2. The operation can be carried out only when the mediating faces of the two operands in the composition are equal. This equality check is performed by comparing the strings in the `O_nam` and `I_nam` elements. The first step consists of adding offsets to the left operand in order to have disjoint supports. This operation was described above in the implementation for tensor product. In the next step, each `I_nam` in the point-sets of the right operand is substituted by the point-set associated to the corresponding `O_nam` in the left operand. Finally, all the pairs in the right operand having an `Edg` as first element are added to the pair set of the left operand. The implementation is explained in the following example.

Example 5.2.3. Consider concrete link graphs $G : \{y\} \to \{y\}$, $F : \{x\} \to \{y\}$ and their composition $G \circ F : \{x\} \to \{y\}$ as in Figure 5.4. The two operands are represented in OCaml as two sets of type `Lg.t`[1]. The set for link graph G contains only one pair:

```
[(O_nam("y"), [I_nam("y"); Port(0,0)])] .
```

Since $|V_F| = 3$, the set after the application of the offset becomes

```
[(O_nam("y"), [I_nam("y"); Port(3,0)])] .
```

The set for link graph F is

```
[(O_nam("y"), [Port(0,0); Port(1,0)]);
 (Edg(0), [I_nam("x"); Port(0,1)]);
 (Edg(1), [Port(1,1); Port(2,0)])]
```

Composition $G \circ F$ is the result of adding the edges of F to G and of replacing inner name y with the point-set of outer name y in F. This substitution is highlighted by the coloured boxes in the representations above. The OCaml value of $G \circ F$ is given by

```
[(O_nam("y"), [[Port(0,0); Port(1,0); Port(3,0)]);
 (Edg(0), [I_nam("x"); Port(0,1)]);
 (Edg(1), [Port(1,1); Port(2,0)])]
```

■

Support translations are implemented as substitutions of ports and edge-identifiers. An example support translation is the application of an offset we described above.

5.2.2 Matching engine

We now describe the details of the implementation of the matching engine of BigraphER. It is based on a SAT encoding of MATCH(_,_) and the MiniSat solver. In the following, we give insight into our approach by showing the encodings of sub-routines SUB_ISO(_, _) and CTRL(_, _, _). The complete encoding was implemented by Chris Unsworth and is illustrated in [57]. The main reason behind our choice of using a SAT encoding is to easily implement an efficient matching engine. SAT solvers are highly specialised software tools capable of checking satisfiability of propositional formulae. If a formula φ is satisfied, a truth assignment that makes φ evaluate to **true** is returned. The satisfiability problem is **NP**-complete.

[1] To ease the presentation, the sets in the example are displayed as lists. This transformation is the result of applying function `elements` to values of type `Lg.t` and `Points.t`.

A wide range of decision and optimization problems can be encoded into instances of SAT. This is appealing because it is often faster to solve the SAT encoding of a problem with a SAT solver rather than using a specialised algorithm to solve the original problem. This can be particularly true in the presence of **NP**-complete problems. Examples of this approach are presented in [11, 67].

Our SAT encoding is defined by a set of boolean formulae, called *constraints*, expressing an instance of the matching problem. Given a target T and a pattern P, invocation $\text{SUB\_ISO}(\mathcal{G}_T, \mathcal{G}_P)$ is encoded by a matrix $\mathbf{X}$ of boolean variables and by four constraints on its structure. Each element $x_{i,j}$ of $\mathbf{X}$ represents a possible pair $(i, j) \in \iota$, with ι an isomorphism from P to a subset of T. Therefore, $\mathbf{X}$ has n rows and m columns, with $|V_P| = n$ and $|V_T| = m$. The constraints on $\mathbf{X}$ are formulae in conjunctive normal form (i.e. a conjunction of clauses). By definition of isomorphism, ι is total. Hence, every node of the pattern has to be mapped to at least node in the target. This corresponds to having at least one variable on every row of $\mathbf{X}$ that is assigned **true**. Formally, the constraint is defined as follows

$$C_1 = \bigwedge_{0 \leq i < n} \bigvee_{0 \leq j < m} x_{i,j}$$

It contains n clauses. The second constraint specifies that each node in P is mapped to at most one node in T. This corresponds to having at most one variable on every row of the matrix that is assigned **true**. Therefore, the encoding needs to check every pair $(x_{i,j}, x_{i,k})$, with $j < k$ in every row i of $\mathbf{X}$. The definition is given by

$$C_2 = \bigwedge_{0 \leq i < n} \bigwedge_{0 \leq j < m-1} \bigwedge_{j < k < m} (\neg x_{i,j} \vee \neg x_{i,k})$$

The number of clauses is $n\binom{m}{2} = n\frac{m(m-1)}{2} = \mathcal{O}(nm^2)$. Since ι is a bijection, each node in T is mapped by at most one node in P. The corresponding constraint is defined similarly to the previous one but the pairs are taken from the columns of $\mathbf{X}$. The definition is as follows:

$$C_3 = \bigwedge_{0 \leq j < m} \bigwedge_{0 \leq i < n-1} \bigwedge_{i < l < n} (\neg x_{i,j} \vee \neg x_{l,j})$$

It contains $m\binom{n}{2} = \mathcal{O}(mn^2)$ clauses. Finally, the edges of $\mathcal{G}_P$ have to correspond to the edges of $\mathcal{G}_T$ and vice versa. This is expressed by the following constraint:

$$C_4 = \bigwedge_{(i,l,j,k) \in \mathcal{C}} \neg x_{i,j} \vee \neg x_{l,k} \quad \text{with} \quad \mathcal{C} = \{(i, l, j, k) \mid (i, l) \in prnt_P \neq (j, k) \in prnt_T\}$$

In the worst case, the constraint has n^2m^2 clauses. The formula encoding $\text{SUB\_ISO}(\mathcal{G}_T, \mathcal{G}_P)$ is the conjunction of the four constraints. If an isomorphism $\iota : \mathcal{G}_P \to \mathcal{G}_T$ exists, then

MiniSat returns a truth assignment $\mathcal{A}$ such that

$$\mathcal{A} \models C_1 \wedge C_2 \wedge C_3 \wedge C_4 \qquad \text{and} \qquad (i,j) \in \iota \iff \mathcal{A}(x_{i,j}) = \textbf{true} \ .$$

It is possible to encode $\textsc{Ctrl}(\iota, ctrl_T, ctrl_P)$ with a similar constraint on $\mathbf{X}$:

$$C_5 = \bigwedge_{(i,j)\in\mathcal{C}'} \neg x_{i,j} \qquad \text{with} \qquad \mathcal{C}' = \{(i,j) \mid ctrl_P(i) \neq ctrl_T(j)\}$$

It has nm clauses in the worst case. Note that $\bigwedge_{0<i<6} C_i$ can be used to directly obtain a control-preserving isomorphism. In this way, isomorphisms that are not solutions can be detected and discarded earlier in the computation. This is more efficient that computing all the possible isomorphism with $\textsc{Sub\_iso}(\_, \_)$ and filtering them afterwards with $\textsc{Ctrl}(\_, \_, \_)$ as was specified in the definition of $\textsc{Match}(\_, \_)$. The total number of clauses required by the encoding is $\mathcal{O}(n^2m^2)$.

In order to find other isomorphisms, a new SAT instance is generated. A clause corresponding to the negation of a solution $\mathcal{A}$ already found by MiniSat is added to the set of constraints as follows:

$$\mathcal{A}' \models (\bigwedge_{0<i<6} C_i) \wedge (\bigvee_{x\in\mathcal{C}''} \neg x) \qquad \text{with} \qquad \mathcal{C}'' = \{x \mid \mathcal{A}(x) = \textbf{true}\}$$

where $\mathcal{A}'$ corresponds to a different isomorphism. This operation is iterated until the SAT instance we obtain is unsatisfiable. This assures that all the isomorphisms from P to a subset of T are found.

The other phases of the algorithm are expressed as constraints in a similar fashion. The conjunction of all constraints is the SAT instance encoding the matching algorithm. Note that a different matrix is needed for link matching.

The implementation of the matching engine consists of an encoder written in C++ and an OCaml module providing bindings to the other modules of the BigraphER system. The encoder receives as input a textual representation of a target and a pattern, computes the SAT constraints and sends them to MiniSat. The result is a textual representations of the solutions (ι, η). The encoder can also be used as a stand-alone command-line tool. The implementation has been tested with all the examples in Chapter 4. All the correct solutions were returned in less than a millisecond. We also tested the system with large randomly generated bigraphs. It was found that instances with 500 nodes in the target graph and between 10 and 100 nodes in the pattern graph could be solved in less that 20 seconds. Finally, the matching engine was successfully tested on instances arising from the bigraphical models of real applications on wireless network protocols and domestic networks. The results are reported in chapters 6 and 7.

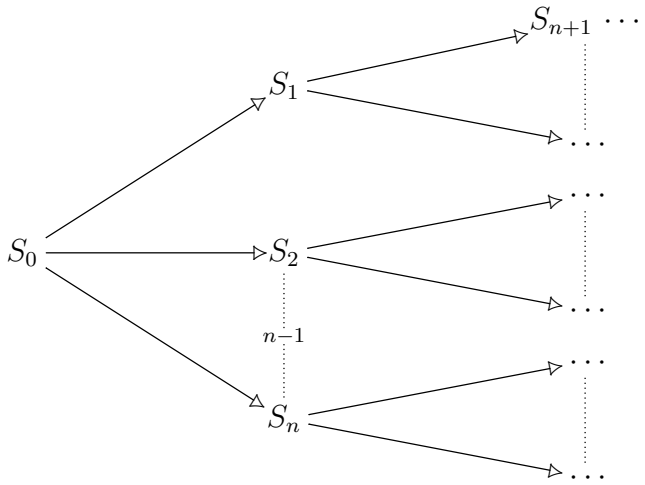


Figure 5.5: Construction of the state space.

5.2.3 Rewriting engine

The rewriting engine is the component of the BigraphER system that computes the dynamic evolution of bigraphs. This is implemented by building the state space of the model specified in the input file. We represent the state space in OCaml as a graph in which the vertices are the states and the edges are their adjacency lists stored in a hash table. Recall that bigraph S_0 is adjacent to bigraph S_1 iff $S_0 \longrightarrow\!\!\triangleright S_1$. Data structures are provided by the standard library in modules `Set` and `Hashtbl`, respectively. Each state is a pair formed by an integer index and an OCaml representation of a concrete bigraph. Each entry in the hash table binds a state index to a set of state indexes. Informally, the rewriting engine incrementally builds the state space in a breadth-first search (BFS) fashion starting from the initial state of the model. The adjacent states of a state are computed on-the-fly by applying the reaction rules to it. This is represented by the diagram in Figure 5.5. Below we describe more precisely our implementation approach.

The rewriting engine is initialised after the input model is successfully parsed by the compiler. Initially, the hash table is empty and the set of vertices contains only state `(0,b)`, where `b` is a concretion of the bigraph declared `init` in the source file. Then, the BFS loop on the graph representing the state space is started. Our implementation is based on the classical algorithm using a queue. At each iteration, the first element, `(i,s)`, is removed from the queue and all its possible reconfigurations are computed. If the states obtained are not already present in the graph, they are added both to the queue and to the set of vertices. Moreover, `(i,s)`'s adjacency list in the hash table is updated by adding the indexes of the reconfigurations. The loop is repeated until the queue is empty. Note that when a new state is added to the graph, its index is the current cardinality of the set of vertices. This assures that states have distinct indexes.

The reconfigurations of `(i,s)` are computed by checking if the redex of each reaction rule in the model occurs in `s`. The representation of each reaction rule is fetched from the compiler, while the occurrences are computed by the matching engine. If redex `r` is a match, state `s` is decomposed by the algorithm defined in the constructive proof of Proposition 4.2.4. The result is a tuple of bigraphs `(c, id, d)`, where `c` is the context, `id` is the identity and `d` is the parameter. Observe that the decomposition is unique because `r` is a solid bigraph (see Section 2.9). The reconfiguration resulting from the application of the reaction rule is obtained by composing the reactum with `c`, `id` and `d`.

Equality checking is also implemented as an instance of matching. Recall a bigraph `f` is $\eqcirc$-equivalent to `g` if it occurs in `g`, the context and parameter in the corresponding decomposition are identities and `id` is the empty bigraph. We note that, the computed state space can be correctly interpreted as an abstract BRS, despite the implementation being based on concrete bigraphs.

When the input model specifies a SBRS, rates are added to every edge in the graph in order to correctly represent a CTMC. This is implemented by a different definition of adjacency list in which each element is a pair formed by a state index and a rate. The BFS loop is also modified to compute the total rate of every reaction by summing the contributions of each stochastic reaction rule.

The efficiency of the rewriting engine heavily relies on the performances of the matching algorithm. As we described above, we have an instance of matching for every reaction rule application and for every equality check. Observe that all the states in the transition system are tested for equality at every iteration of the BFS loop. Therefore, it is essential for the user of the BigraphER system to specify carefully the input models in order to reduce the size of the redexes and to avoid the creation of meaningless states. To this end, we can also support priorities on reaction rules. We will describe this feature in the next chapter.

5.2.4 Visualisation

A graphical representation of bigraphs can be generated automatically by BigraphER. An example output is shown in Figure 5.6. As can be seen, the result closely resembles the stratified diagrams for sharing bigraphs introduced in Chapter 3. Our implementation of the visualisation function uses Graphviz, an open source graph visualization software. In more detail, the compiler computes a textual description of a bigraph and then sends it to the `dot` tool which automatically generates the layout of the graph for the graphical representation. Different styles are used to denote the components of a bigraph. For example, sites and roots are displayed as dashed boxes and links are drawn as solid green lights. Rank constraints on the roots and outer names force them to appear on the top of the diagram. Similarly, sites

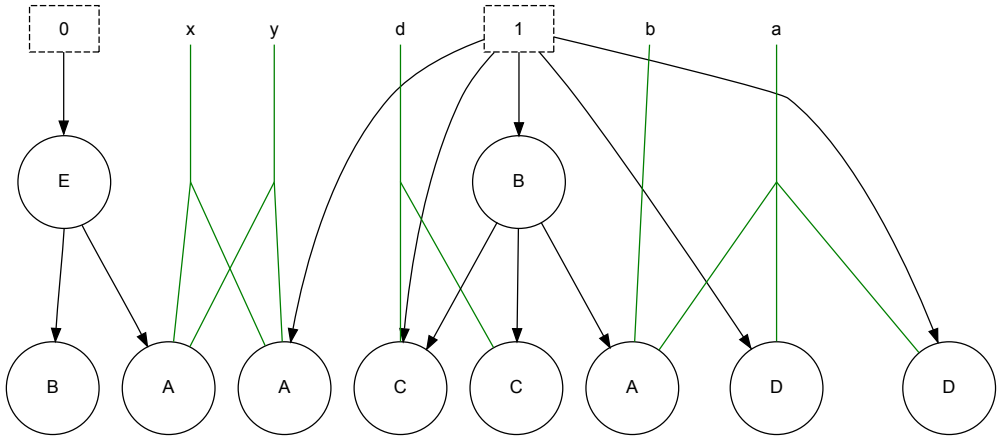


Figure 5.6: Example of automatically-generated visualisation.

and inner names are forced to the bottom. Each hyper-edge in the link graph is encoded by an invisible dummy node to which ports and names are linked.

5.3 Checking predicates

Recall that the structure of bigraphs can be described by **BiLog** as we showed in Chapter 2. Some example formulae expressing properties of an arbitrary bigraph B are:

φ_1: B contains a node of control C, written $B \models \top \circ (\mathbf{C} \otimes \top) \circ \top$.

φ_2: $B \models \psi_1 \circ \top \circ \psi_2$, i.e. B can be decomposed in two bigraphs satisfying formulae ψ_1 and ψ_2, respectively.

φ_3: B contains two A-nodes that are not linked (but may be nested), $B \models ((\top \otimes \mathbf{id}_{x,y}) \circ (\mathbf{A}_x \otimes \top \otimes \mathbf{id}_y) \circ (\mathbf{id}_y \otimes \top) \circ (\mathbf{A}_y \otimes \top) \circ \top$.

Formula $\top$ it the tautology. It is satisfied by any bigraph.

In the following, we present informally how a class of predicates can be checked by reduction to bigraph matching. A similar approach was also adopted in [53].

The **BiLog** fragment we consider contains the formulae solely formed by operators $\circ$ and $\otimes$ and elements of $\mathbf{\Omega} \cup \top$. We write φ to indicate a formula in the fragment. Note that boolean operators and logical adjuncts are not allowed. Intuitively, any formula in this form contains one or more sub-formulae that can be used as a pattern in an instance of bigraph matching.

Take for instance predicate φ_1 specified above. It is possible to check $B \models \varphi_1$ by invoking $\textsc{Match}(\widetilde{B}, \widetilde{P_{\varphi_1}})$, where the pattern, corresponding to sub-formula $\mathbf{C}$, is a single ion,

i.e. $P_{\varphi_1} = \mathsf{C}$. Note that two concretions have to be used as input of the matching algorithm. If the procedure returns a match, then the predicate is satisfied. Observe that the invocation of the matching algorithm is not necessary when **C** is substituted by any node-free formula because the predicate always holds.

Checking $B \models \varphi_2$ is more complex and requires two instances of matching. To simplify the presentation, sub-formulae ψ_1 and ψ_2 are assumed to contain some node constant and $\top$ does not occur in them. This allows for a straightforward construction of patterns P_{ψ_1} and P_{ψ_1}. The first step is to invoke $\textsc{Match}(\widetilde{B}, \widetilde{P_{\psi_1}})$. If the algorithm is unable to identify a match, then predicate φ_2 is not satisfied. On the other hand, when a solution is returned, it is possible to build a decomposition

$$B = C \circ (P_{\psi_1} \otimes \mathsf{id}_I) \circ D \ .$$

Recall that by definition, context C contains all the ancestors of the pattern, while parameter D contains all its descendants. Hence, predicate φ_2 can be satisfied only if $I = \epsilon$ and context C is an identity. The second step consists of checking these constraints on the decomposition. If they are satisfied, the third step is performed. It this phase, the second instance of matching arises. The pattern is specified by ψ_2, while the target is parameter D obtained in the previous steps. If an occurrence is identified, it is possible to build another decomposition:

$$D = C' \circ (P_{\psi_2} \otimes \mathsf{id}_{I'}) \circ D' \ .$$

This time, φ_2 can be satisfied only if $I' = \epsilon$ and parameter D' is an identity. Finally, $B \models \varphi_2$ if the constraints on the new decomposition are satisfied. Note that context C' is the part of B being "matched by" $\top$.

Predicate φ_3 is checked by following a similar procedure. We briefly describe the steps required. The first instance of match checks whether an A-node is present in B. This is necessary to verify sub-formula $(\top \otimes \mathbf{id}_{x,y}) \circ (\mathbf{A}_x \otimes \top \otimes \mathbf{id}_y)$. The invocation is $\textsc{Match}(\widetilde{B}, \widetilde{P})$ where the pattern is ion A_x. The resulting decomposition is

$$B = C \circ (P \otimes \mathsf{id}_I) \circ D \ .$$

Predicate φ_3 can be satisfied only if the context contains $\mathsf{id}_{x,y}$ and $y \in I$. If these constraints are satisfied the procedure continues. In the second step, the existence of the second A-node in D is verified by invoking $\textsc{Match}(\widetilde{D}, \widetilde{P'})$ where the pattern is ion A_y. This checks sub-formula $(\mathbf{id}_y \otimes \top) \circ (\mathbf{A}_y \otimes \top) \circ \top$. Similarly to the previous step, φ_3 is satisfied only when it is possible to build a decomposition

$$D = C' \circ (P' \otimes \mathsf{id}_{I'}) \circ D'$$

such that context C' contains identity id_y.

The previous examples explain how to define inductively an algorithm to check a class of **BiLog** formulae. Summarising, matching instances correspond to the formulae generated by the following grammar:

$$\psi_1, \psi_2 ::= \quad \mathbf{\Omega} \quad \Big| \quad \psi_1 \circ \psi_2 \quad \Big| \quad \psi_1 \otimes \psi_2 \quad .$$

Formulae generated by

$$\begin{aligned} \zeta ::= \quad & \top \circ \varphi \quad \Big| \quad \varphi \circ \top \quad \Big| \quad \top \otimes \varphi \quad \Big| \quad \varphi \otimes \top \\ \varphi ::= \quad & \top \quad \Big| \quad \psi \quad \Big| \quad \zeta \end{aligned}$$

are checked by imposing constraints on the decompositions resulting from the matching instances on ψ sub-formulae. This approach allows to minimise the number of matching instances if compared to an algorithm defined inductively on formulae generated by a grammar with only one non-terminal. Therefore, efficiency is improved. Finally, we remark that boolean operators $\vee$ and $\wedge$ can be supported by modifying procedure $\textsc{Ctrl}(\_, \_, \_)$ and by iterated invocations of the matching algorithm.

We implemented the approach described above in BigraphER. Predicates in the **BiLog** fragment of interest are specified in the input model by the same syntactical constructs used for `big` declarations. Only an additional terminal to encode $\top$ need be introduced. Predicates are checked in the BFS loop in the rewriting engine. Every time a state is added to the graph, all the predicates in the model are checked against it. A labelling function is implemented by a hash table binding a state index to a set whose elements are the identifiers of the predicates that are satisfied by the state. Therefore, the graph together with the hash table can be interpreted as a Kripke structure or a labelled CTMC and temporal properties of the model can be checked.

5.4 Summary

In this chapter we described the implementation of the BigraphER system: a command-line tool and an OCaml library for the manipulation, simulation and visualisation of bigraphs with sharing. In Section 5.1, we discussed the architecture of the system and described the rôle of its components. Section 5.2 was devoted to the analysis of the implementation details. We introduced an OCaml representation of concrete bigraphs, a SAT encoding of the matching algorithm and a rewriting engine capable of computing the dynamics of BRS and SBRS. Finally, a method to check a class of **BiLog** predicate was introduced in Section 5.3.

In the next chapter we will present our first case study: a bigraphical model of the 802.11 CSMA/CA protocol. The command-line tool in the BigraphER system will be used to generate automatically the CTMC capturing the behaviour of an example networks of three machines.

Chapter 6

A bigraphical model of the 802.11 CSMA/CA RTS/CTS protocol

This chapter illustrates the applicability of stochastic bigraphs with sharing in the context of communication protocols for wireless networks. Namely, we present a bigraphical model of the IEEE 802.11 CSMA/CA with RTS/CTS protocol with support for arbitrary network topologies. The model enables for the automatic generation of the CTMC capturing the behaviour of any wireless network governed by this protocol. This in turn allows for the analysis of quantitative properties such as the probability of collision and the average number of transmissions before a data packet is successfully sent.

In Section 6.1 we discuss the suitability of a bigraphical representation for the protocol and we compare with related work on CSMA modelling and analysis. Section 6.2 contains an informal description of the protocol. In particular, we concentrate on the RTS/CTS exchange mechanism. In Section 6.3, the structure of the bigraphs used in the model is defined. We introduce controls and a sorting discipline allowing for a precise representation of wireless networks with arbitrary topology. To facilitate the presentation, different colours and node shapes are used in the graphical notation to indicate nodes with different controls. In Section 6.4 the SBRS model of the protocol is given and we discuss some general problems of specifying behaviour with rewrite rules and our solution using priorities. We present only the graphical form of the stochastic reaction rules. The corresponding algebraic forms are given in Appendix C. Two sample executions (paths) of an example network of three stations are given in Section 6.5. Section 6.6 describes the underlying CTMC resulting from the example, its computation using BigraphER, and some quantitative analysis results. A summary of the chapter is given in Section 6.7.

6.1 Introduction

Wireless local area networks (WLANs) have become hugely popular in recent years and play an increasingly important part in our everyday lives. The international standard IEEE 802.11 [36] was developed to enable the use of heterogeneous communication devices from different vendors within the same network. It specifies the physical layer (PHY) and a MAC (Media Access Control) layer based on CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance). This differs from MAC layers for wired networks in which Collision Detection (CSMA/CD) can be employed to prevent simultaneous transmission on the channel. The reason is that stations of a wireless network cannot listen to their own transmission(s) and are therefore unable to detect collisions. The standard also defines an optional mechanism to reduce collisions called RTS/CTS (Request to Send / Clear to Send). We describe it in the next section.

WLANs exhibit behaviours that depend both on time and space. Hence, the double structure characterising bigraphs and their static-dynamic nature appear to be ideally suited to this domain. Moreover, bigraphs with sharing allow one to naturally model a crucial aspect of wireless networks: *signal interference*. This phenomenon occurs when a device is in the range of more than one signal. In other words, wireless networks are *overlapping*.

Some aspects of the CSMA protocol have been modelled previously: for example collision detection on Ethernet is modelled by a MDP (Markov Decision Process) in [26]. A similar approach was taken in [41] where probabilistic timed automata are used to model the basic two-way handshake mechanism[1] of the 802.11 protocol. The authors assume a fixed network topology consisting of two senders and two receivers. Furthermore, in their model there is exactly one shared signal, and thus each station can sense any other station. Properties of the system are specified in **CSL** (Continuous Stochastic Logic) [2] and automatically verified using probabilistic model checker Prism [40]. The model we present here differs in the following significant ways: support for arbitrary network topologies, and explicit representation of potentially overlapping wireless signals for all the stations in the network. These features are essential to represent networks in which two or more stations transmit to the same receiver and they cannot sense each other, thus causing a transmission collision. This is generally known in the literature as the *hidden node problem*. The topology of an example network suffering from the problem is drawn in Figure 6.3a, where the senders are A and C and the receiver is B.

[1] Note that this protocol is different from RTS/CTS.

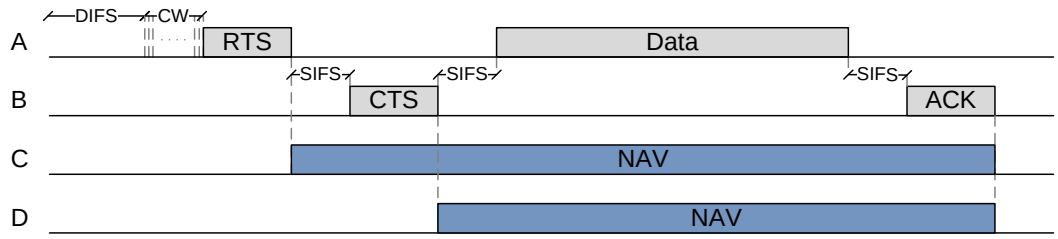


Figure 6.1: The use of virtual channel sensing using CSMA/CA.

6.2 The protocol: 802.11 RTS/CTS handshake

We now describe informally the functioning of the protocol. Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) is the basic access mechanism in the 802.11 protocol [36]. CSMA/CA adopts a slotted binary exponential backoff scheme to reduce collisions due to stations transmitting simultaneously. It defines two access mechanisms: the default, two-way handshaking technique called *basic access* and the optional four-way handshaking RTS/CTS reservation scheme. We focus on the latter here. Observe that this mechanism is more complicated because an extra handshake is introduced to reduce the collisions caused by the hidden node problem.

To illustrate the protocol, assume stations A, B, C, D where B and C are in the range of A, and B is in the range of D. In CSMA/CA a transaction is defined to be an exchange of packets between two stations that terminates with the transmission of an *acknowledgement* ACK. A is willing to transmit a packet to B and so transmits a *request-to-send* RTS, which includes the source, destination, and the duration of the following transaction (i.e. transmission of a *clear-to-send* CTS followed by a the data packet and the corresponding ACK). B responds (if the medium is free) with a CTS. Upon successful receipt of the CTS, A is allowed to transmit the actual data packet. Station B checks whether the received packet is corrupted by performing a cyclic redundancy check (CRC) and sends back an ACK. Receipt of the acknowledgement indicates that no collision occurred. Note that CTS, ACK and the data packet are sent only if the medium is idle for a *short inter-frame space* (SIFS) period. If A does not receive the CTS or the ACK packets, then it will retransmit until it gets acknowledged or reaches a given number of transmissions. All other stations receiving either the RTS and/or the CTS, e.g. C and D, set their virtual carrier sense timer (called NAV) with the duration value contained in those packets, and so will not attempt transmission until the NAV reaches 0. Figure 6.1 illustrates the scenario of A transmitting to B.

The exponential backoff scheme is executed each time the medium is sensed as busy or when a retransmission occurs. The station defers its transmission and the scheme initialises its random backoff timer. This value is obtained by uniformly choosing a real number in the interval $[0, t]$, where $t_{min} \leq t \leq t_{max}$ is the current *contention window* (CW) size, and multiplying it for the duration of a *slot time* denoted by τ. At each timer initialisation, the

Parameter	Symbol	Typical value
DIFS	λ_D	50 μs
SIFS	λ_S	10 μs
Time slot	τ	20 μs
Time-out	λ_T	30 μs
Min CW	t_{min}	15
Max CW	t_{max}	1023
RTS	λ_R	160 μs
CTS	λ_C	112 μs
ACK	λ_A	112 μs

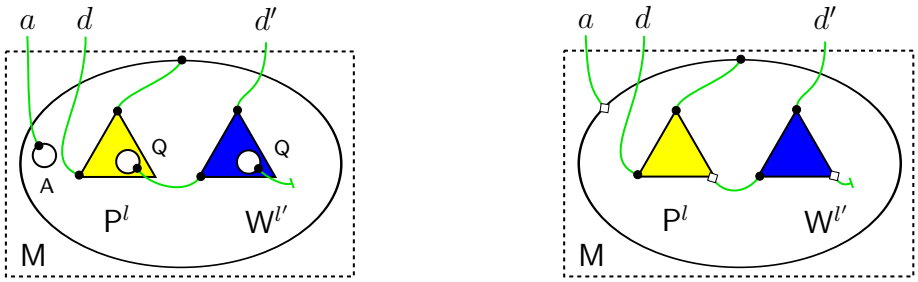
Table 6.1: Parameters of the CSMA/CA protocol.

interval is updated as follows: $[0, t']$, with $t' = 2t + 1$. Upon successful transmission t is reset to t_{min}. The parameters of the protocol specified by the standard are summarised in Table 6.1. The values used correspond to the times defined for a physical layer (PHY) with a data-rate of 1 Mbps. For example, λ_C is the time taken to transmit a CTS packet (with length 14 bytes = 112 bits) at 1 Mbps.

6.3 Bigraphical model of wireless network topology

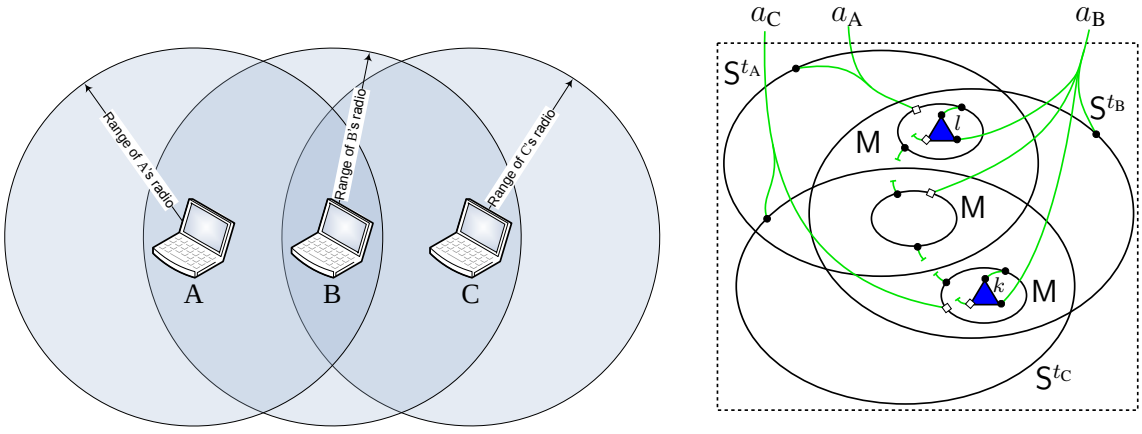
In this section we define the class of bigraphs are used to represent WLANs in our model. Informally, nodes represent the main entities present in the network, namely stations, wireless signals and packets. Their relationships are expressed by links between them and by node sharing. Overlapping signals are modelled by a node representing a station in the intersection of two or more nodes representing the signals. This modelling strategy is explained in greater detail later on.

The graphical form of an example bigraph encoding a station is shown in Figure 6.2a. Node of control M encodes a (mobile) station, while packets to be sent are indicated by triangle-shaped nodes, namely nodes of control P^l and $\mathsf{W}^{l'}$. Each triangle is linked to an outer name (i.e. d, d'') encoding the destination field of a packet. A second link is used to connect a triangle to the node of control Q in the preceding triangle in the *transmission queue*. Observe that the first packet in the queue is always linked with an edge to the parent node of control M. Different colours indicate different controls. They are used to track the status of the sending station during a transmission hand-shake. In this example, triangle P^l encodes the fact that the station has successfully transmitted a data packet and it is now waiting for the ACK packet from the receiver, while $\mathsf{W}^{l'}$ indicates that the station is waiting to transmit a packet. Note that the status of the station is given only by the status of the first packet in the transmission queue. Therefore, all the triangles in the transmission queue but



(a) Standard graphical form (b) Compact representation

Figure 6.2: Bigraphical representation of a station M with two packets P^l and $\mathsf{W}^{l'}$. Station M has sent a data packet P^l and another packet $\mathsf{W}^{l'}$ is stored in its queue (a). Simplified representation in which nodes of control A and Q are encoded by special diamond-shaped ports (b).



(a) WLAN of three stations (b) Bigraphical representation N_0

Figure 6.3: Modelling of network topologies. A network diagram of a WLAN of three stations (a) and the corresponding bigraphical encoding $N_0 : \epsilon \to \langle \mathsf{s}, \{a_\mathrm{A}, a_\mathrm{B}, a_\mathrm{C}, \}\rangle$ (b). Stations A and C are both willing to send to station B.

the first one are always blue (i.e. of control W^l). Each station has a unique MAC address, which is encoded by the link between node of control A and outer name a. In the rest of the chapter, we will adopt a more compact graphical notation in which nodes of control A and Q are replaced by special diamond ports. The compact representation for the bigraph shown in Figure 6.2a is given in Figure 6.2b. Another example is given in Figure 6.3b where the graphical form for the encoding of three stations with overlapping wireless signals is drawn. In the encoding, each node of control M is linked to both its signal and its address. Moreover, the destination field of the packets is also an address. This is shown by the link on outer name a_B. Before proceeding with a formal description of the bigraphical model, we observe that the oval shape used for nodes representing signals does not correspond to the actual shape of an area covered by a signal in a real WLAN. This is because the nesting of nodes represents a topological space and not an Euclidean space. Therefore, distances between places and their shapes are not expressible in place graphs.

Description	Control	Arity	Sort	Notation
Signal idle	S^t	1	s	–
Signal locked	$\mathsf{S}^t_{\mathsf{L}}$	1	s	–
Signal clear	$\mathsf{S}^t_{\mathsf{C}}$	1	s	–
Signal error	$\mathsf{S}^t_{\mathsf{E}}$	1	s	–
Station idle	M	2	m	–
Station locked	M_{L}	2	m	–
Station deferred	M_{D}	2	m	–
Station potential	M_{P}	2	m	–
Station backoff	M_{B}	2	m	–
Address	A	1	a	–
Queue	Q	1	q	–
Packet waiting	W^l	2	p	
RTS packet	RTS^l	2	p	
CTS packet	CTS^l	2	p	
Data packet	P^l	2	p	

Table 6.2: Node controls for the bigraphical model. The parameters are $t_{min} \leq t \leq t_{max}$ and $272\,\mu\text{s} \leq l \leq 18768\,\mu\text{s}$.

The fifteen controls listed in Table 6.2 are used to encode the entities forming a WLAN. These are organised into sorts as follows. Controls describing wireless signals are grouped together into sort $\mathsf{s} = \{\mathsf{S}^t, \mathsf{S}^t_{\mathsf{L}}, \mathsf{S}^t_{\mathsf{C}}, \mathsf{S}^t_{\mathsf{E}}\}$. Parameter t records the contention window size of the station associated to the signal. Its values are drawn from set

$$\{t_{min} = 15, 31, 63, 127, 255, 511, 1023 = t_{max}\} \ ,$$

when parameters are defined as in Table 6.1. This corresponds to allowing seven transmission attempts for every data packet as specified in the protocol's standard. Sort $\mathsf{m} = \{\mathsf{M}, \mathsf{M}_{\mathsf{L}}, \mathsf{M}_{\mathsf{D}}, \mathsf{M}_{\mathsf{P}}, \mathsf{M}_{\mathsf{B}}\}$ is used to represent nodes encoding stations, whilst a packet is indicated with a node of sort $\mathsf{p} = \{\mathsf{W}^l, \mathsf{RTS}^l, \mathsf{CTS}^l, \mathsf{P}^l\}$. Here, parameter l is used to store the time employed to transmit a packet. Observe that the various controls in each sort are required to represent faithfully the WLAN throughout the different stages of the protocol. For instance, a node of control M_{L} indicates a *locked* station while a M_{B}-node expresses a station in *backoff* state. The meaning of each control is explained in Section 6.4, where the reaction rules of the SBRS modelling the protocol are introduced. Two additional sorts are defined: $\mathsf{a} = \{\mathsf{A}\}$ and $\mathsf{q} = \{\mathsf{Q}\}$. They represent the address of a node and a queue of packets respectively: virtual entities introduced only to simplify the modelling process. Specifically, they allow one to distinguish their links from the links of m and s-nodes. The set of sorts is written as $\Theta = \{\mathsf{s}, \mathsf{m}, \mathsf{p}, \mathsf{a}, \mathsf{q}\}$. The signature is given by $\mathcal{K} = \bigcup_{s \in \Theta} s$.

The sorting discipline ensures that only bigraphs with a meaningful structure are con-

Φ_1	all $\widehat{\mathsf{qa}}$-nodes are atomic
Φ_2	all children of a θ-root have sort θ, where $\theta \in \{\mathsf{s}, \mathsf{m}\}$
Φ_3	all children of an s-node have sort m
Φ_4	all children of an m-node have sort $\widehat{\mathsf{pa}}$
Φ_5	an m-node has one a-child
Φ_6	a p-node has one q-child
Φ_7	an a-node is always linked to one grandparent of sort s and a name
Φ_8	in an m-node, one port may be linked to one m-node and the other may be linked to one p-node
Φ_9	in a p-node, one port may be linked to an a-node and the other is always linked to one $\widehat{\mathsf{mq}}$-node (see Figure 6.2a)
Φ_{10}	an a-node may be linked to many p-nodes

Table 6.3: Conditions of formation rule Φ.

structed. For example, it forces packets to be always located inside a machine and a machine to be always surrounded by its signal. This is formalised in formation rule Φ with conditions Φ_i, $1 \leq i \leq 10$, given in Table 6.3. We briefly comment on each condition. Condition Φ_1 states that dummy nodes of control Q and A are atomic (i.e. they contain nothing). Conditions Φ_2–Φ_4 specify a hierarchic structure in the placing of the nodes. In particular, s-nodes are the outermost nodes in the model. They contain m-nodes, which in turn contain $\widehat{\mathsf{pa}}$-nodes. Conditions Φ_5, Φ_6 specify the placing described in the example in Figure 6.2a. Condition Φ_7 ensures that an address A is always connected to a unique name and to one ancestor signal node. This allows the discrimination of signals according to the machine, and thus the address they belong to. Conditions Φ_8, Φ_9 describe the queue-like structure formed by packets which sit within a station as in Figure 6.2a. Condition Φ_{10} states that an address may be linked to several packets. This models the destination field in each packet. Finally, we refer the sorting used in the model as $\Sigma_{802.11} = (\mathcal{K}, \Theta, \Phi)$.

Our approach to modelling network topologies is explained through an example. Consider for instance the WLAN of three stations shown in Figure 6.3a. Assume that both stations A and C are willing to transmit to station B. The bigraphical model needs to record that the system is composed of three stations, each one with its own wireless signal. Then, it also has to satisfy the following requirements:

Req1: A and C sense B,

Req2: B senses A and C,

Req3: A does not sense C and C does not sense A.

This wireless network is faithfully modelled by agent $N_0 : \epsilon \to \langle \mathsf{s}, \{a_{\mathrm{A}}, a_{\mathrm{B}}, a_{\mathrm{C}}, \} \rangle$ given in Figure 6.3b. Recall that for clarity we adopt a simplified graphical notation in which $\widehat{\mathsf{aq}}$-nodes are drawn as diamond ports as shown in Figure 6.2b. Stations A, B, and C are encoded by the three M-nodes linked to names a_{A}, a_{B} and a_{C}, respectively. Their signals are indicated

by nodes $\mathsf{S}^{t_\mathrm{A}}$, $\mathsf{S}^{t_\mathrm{B}}$, $\mathsf{S}^{t_\mathrm{C}}$, respectively, and they are linked to the corresponding name. Observe that each station sits within its own signal node. **Req1** is satisfied by station B being in the intersection of the two signals $\mathsf{S}^{t_\mathrm{A}}$ and $\mathsf{S}^{t_\mathrm{C}}$. **Req2** is satisfied because the two m-nodes for stations A and C are contained by node $\mathsf{S}^{t_\mathrm{B}}$. Finally, **Req3** holds because station A is outside $\mathsf{S}^{t_\mathrm{C}}$ and vice versa. An equivalent algebraic form of N_0 is

$$N_0 = \text{share } (m_\mathrm{A} \parallel m_\mathrm{B} \parallel m_\mathrm{C}) \text{ by } \psi \text{ in } (\mathsf{id}_{a_\mathrm{A} a_\mathrm{B} a_\mathrm{C}} \mid \mathsf{S}^{t_\mathrm{A}}_{a_\mathrm{A}} \mid \mathsf{S}^{t_\mathrm{B}}_{a_\mathrm{B}} \mid \mathsf{S}^{t_\mathrm{C}}_{a_\mathrm{C}}) \ ,$$

where terms

$$\begin{aligned} m_\mathrm{A} &= (\mathsf{id}_{1,a_\mathrm{A} a_\mathrm{B}} \parallel /x \parallel /r \parallel /q\,)(\mathsf{M}_{rx}.(w_\mathrm{A} \mid \mathsf{A}_{a_\mathrm{A}}.1)) \\ m_\mathrm{B} &= (\mathsf{id}_{1,a_\mathrm{B}} \parallel /x \parallel /r\,)(\mathsf{M}_{rx}.\mathsf{A}_{a_\mathrm{B}}.1) \\ m_\mathrm{C} &= (\mathsf{id}_{1,a_\mathrm{C} a_\mathrm{B}} \parallel /x \parallel /r \parallel /q\,)(\mathsf{M}_{rx}.(w_\mathrm{C} \mid \mathsf{A}_{a_\mathrm{C}}.1)) \ , \end{aligned}$$

indicate stations A, B and C, respectively, and terms

$$w_\mathrm{A} = \mathsf{W}^{l}_{x a_\mathrm{B}}.\mathsf{Q}_q.1 \qquad\qquad w_\mathrm{C} = \mathsf{W}^{k}_{x a_\mathrm{B}}.\mathsf{Q}_q.1$$

encode A's and C's packets (i.e. the blue triangles in Figure 6.3b), respectively. The network topology is specified by placing

$$\psi = [\{0,1\},\{0,1,2\},\{1,2\}] \ .$$

This simple example shows how our modelling strategy can provide adequate expressive power for the representation of arbitrary network topologies. Moreover, bigraphs with sharing allow us to express succinctly complex network topologies.

We now list our modelling assumptions:

- All signals have equal power. This implies that if a station A is within the range of another station B, then B is within the range of A.
- The range of broadcast packets is the same of unicast packets.
- The range of small packets is the same of large packets.
- No packet is lost during transmission, i.e. ideal channel conditions.
- Static topology.

We remark that these assumptions derive from those made in the specification of the 802.11 protocol [36].

6.4 Stochastic reaction rules modelling the protocol

Before presenting the bigraphical reactive system, we introduce some notation and conventions. The stochastic reaction rules in our the model have form $\mathsf{R}_{\mathsf{L}}(\pi) = R \xrightarrow{\rho} R'$, where subscript L is a label describing the rule semantics and π is a list of parameters. All the rules respect the sorting $\Sigma_{802.11}$. We call the interface of the rule the interface of bigraphs R and R'. Reaction rates are indicated by $\rho_i(\pi)$ with $i = 1, 2, \ldots$. Names used as addresses range over $a, a', a_{\mathrm{A}}, a_{\mathrm{B}}, \ldots$. Destination addresses are denoted by $d, d', \ldots$. When defining reaction rules, name x is used to denote a link between two m-nodes while name q represents a link between a p-node and a node of control Q. We write $\longrightarrow$ instead of $\xrightarrow{\infty}$ to indicate *instantaneous* reaction rules (i.e. rules with rate ∞).

Additionally, we introduce *rule priorities*, i.e. a partial ordering on the rules of the reactive system. A *Priority BRS* (PBRS) is a BRS with *rule priorities* in the style of [5]. The procedural meaning of such an ordering is that a reaction rule of lower priority can be applied only if no rule of higher priority is applicable. We write $\mathsf{R} < \mathsf{R}'$ to indicate that reaction rule R' has higher priority than reaction rule R. A priority class $\mathfrak{P}$ is a set of reaction rules with the same priority. By an abuse of notation, we write $\mathfrak{P} < \mathfrak{P}'$ when, for any two rules $\mathsf{R} \in \mathfrak{P}$ and $\mathsf{R}' \in \mathfrak{P}'$, we have $\mathsf{R} < \mathsf{R}'$. We also say that class $\mathfrak{P}$ has lower priority than class $\mathfrak{P}'$. Similarly, a *Priority SBRS* (PSBRS) consists of an SBRS augmented with rule priorities. We require all the rules in a priority class to be instantaneous, if it contains an instantaneous rule. This extension allows the specification of concise reaction rules for arbitrary topologies. We discuss the merits of this approach in more detail at the end of this section.

The 802.11 CSMA/CA RTS/CTS protocol is modelled as a PSBRS in which WLAN's configurations are encoded by $\Sigma_{802.11}$ sorted bigraphs and system updates are expressed by reaction rules. These are described in the remainder of this section. In order to obtain a general model of the protocol, each reaction rule is indexed by parameters such as the values of timers (indicated with t and t') and packet sizes (denoted by l). In the following a graphical description of each reaction rule is presented. Equivalent algebraic definitions are reported in Appendix C. A summary of the rules is in Table 6.4 using the following convention: each description takes the form ⟨description of bigraph on lhs⟩ → ⟨description of bigraph on rhs⟩.

The first rule $\mathsf{R}_{\mathsf{RTS}}(t, l)$ models the initial phase of the CSMA/CA protocol: whenever a sender senses the channel free, it is allowed to initiate a communication. A graphical representation of the reaction is given in Figure 6.4. The station willing to transmit is encoded on the left-hand side by a node of control M containing a triangle of control W^l. On the right-hand side, a communication is instantiated. This is shown by the RTS^l triangle, the M_{L} and the $\mathsf{S}^t_{\mathsf{L}}$ nodes. The reaction rate is defined as follows:

$$\rho_1(t) = \left(\lambda_D + \tau\tfrac{t}{2} + \lambda_R\right)^{-1} \quad .$$

	Reaction rule	Rate	Description	Graphical form
1	$\mathsf{R}_{\mathsf{RTS}}(t, l)$	$\rho_1(t)$	machine is idle and willing to send $\rightarrow$ machine is locked and an RTS sent.	Figure 6.4
2	$\mathsf{R}_{\mathsf{CTS}}(t, t', l)$	ρ_2	sender has sent an RTS $\rightarrow$ sender and receiver locked and linked, CTS sent.	Figure 6.5
3	$\mathsf{R}_{\mathsf{DATA}}(l)$	$\rho_3(l)$	sender has sent a CTS $\rightarrow$ sender has sent its data packet.	Figure 6.6
4	$\mathsf{R}_{\mathsf{ACK}}(t, t', l)$	ρ_4	data packet sent $\rightarrow$ ACK sent and locks released.	Figure 6.7
5	$\mathsf{R}_{\mathsf{BACK1}}(t, t', l)$	ρ_5	sender (not the most recent) has to back off and receiver has detected a collision $\rightarrow$ locks are released and the sender is timed out.	Figure 6.9
6	$\mathsf{R}_{\mathsf{BACK2}}(t, t', l)$	ρ_5	most recent sender has to back off and receiver has detected a collision $\rightarrow$ locks are released and the sender is timed out.	Figure 6.10
7	$\mathsf{R}_{\mathsf{D}}(t)$	∞	station receives an RTS or a CTS $\rightarrow$ station defers.	Figure 6.11
8	$\mathsf{R}_{\mathsf{P}}(t, t')$	∞	station receives an RTS from more than one station $\rightarrow$ station defers and detects conflict.	Figure 6.12
9	$\mathsf{R}_{\mathsf{B}}(l)$	∞	sender cannot receive a CTS from the receiver $\rightarrow$ sender has to back off.	Figure 6.13
10	$\mathsf{R}_{\mathsf{UD}}(t)$	∞	the NAV of a deferred station expires $\rightarrow$ lock released.	Figure 6.14
11	$\mathsf{R}_{\mathsf{UP}}(t)$	∞	the NAV of a station in a conflict state expires $\rightarrow$ lock released.	Figure 6.15
12	$\mathsf{R}_{\mathsf{UC}}(t)$	∞	sender or receiver terminated a transmission $\rightarrow$ lock released.	Figure 6.16

Table 6.4: Reaction rules.

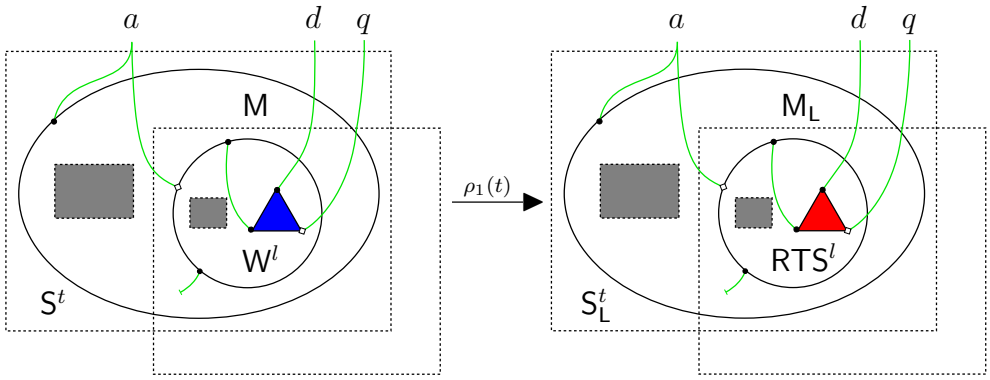


Figure 6.4: $\mathsf{R}_{\mathsf{RTS}}(t,l)$: machine is idle and willing to send $\rightarrow$ machine is locked and an RTS sent.

Note that it depends on t, i.e. the contention window size of the sending station. Term $\tau\frac{t}{2}$ is a constant indicating the average time the sender waits idle before its contention window of size t is fully consumed. In algebraic terms, this reaction rule is defined as

$$\begin{aligned}\mathsf{R}_{\mathsf{RTS}}(t,l) = {} & \text{share } (\mathsf{id} \parallel m_w) \text{ by } \psi_1 \text{ in } (\mathsf{id}_{1,adq} \parallel \mathsf{S}^t_a) \\ & \xrightarrow{\rho_1(t)} \\ & \text{share } (\mathsf{id} \parallel m_l) \text{ by } \psi_1 \text{ in } (\mathsf{id}_{1,adq} \parallel \mathsf{S}^t_{\mathsf{L}a})\end{aligned}$$

where

$$\begin{aligned} m_w &= (\mathsf{id}_{1,aqd} \parallel /x \parallel /r\,)(\mathsf{M}_{rx}.(\mathsf{id} \mid \mathsf{W}^l_{xd}.\mathsf{Q}_q.1 \mid \mathsf{A}_a.1)) \\ m_l &= (\mathsf{id}_{1,aqd} \parallel /x \parallel /r\,)(\mathsf{M}_{\mathsf{L}rx}.(\mathsf{id} \mid \mathsf{RTS}^l_{xd}.\mathsf{Q}_q.1 \mid \mathsf{A}_a.1)) \\ \psi_1 &= [\{1\},\{0,1\}]\ . \end{aligned}$$

The interface of the reaction rule is given by: $\mathsf{R}_{\mathsf{RTS}}(t,l) : \mathsf{m\widehat{p}a} \rightarrow \langle \mathsf{sm}, \{a,d,q\}\rangle$.

The second rule $\mathsf{R}_{\mathsf{CTS}}(t,t',l)$ describes the transmission of a CTS packet from the sender to the receiver. Its graphical representation is given in Figure 6.5. The reaction can be triggered only when the sender and the receiver sense each other and the receiver is available for a transmission. These two preconditions are encoded in the left-hand side by two nodes of control M_{L} and M_{D} being in the same intersection of signals and by the node of control $\mathsf{S}^{t'}$ containing the receiver, respectively. Moreover, M_{D} can be identified as the receiver because it is linked to the sender's RTS^l triangle. Note also that both the receiver and the sender can be in the range of other stations. On the right-hand side the receiver is locked (nodes of control $\mathsf{S}^{t'}_{\mathsf{L}}$ and M_{L}) and a link between the two machines is established. The reaction rate is $\rho_2 = (\lambda_S + \lambda_C)^{-1}$.

The third rule $\mathsf{R}_{\mathsf{DATA}}(l)$ models the transmission of a data packet from the receiver to the sender. Its graphical form is depicted in Figure 6.6. On the left-hand side, the sender's

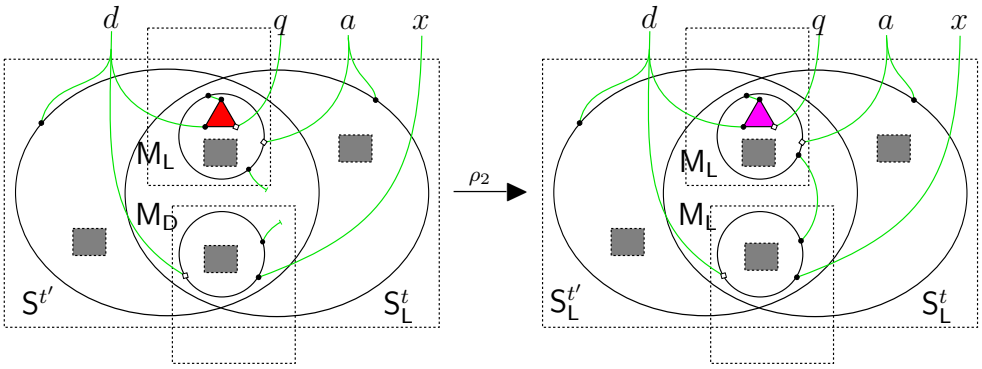


Figure 6.5: $\mathsf{R}_{\mathsf{CTS}}(t, t', l)$: sender has sent an RTS $\rightarrow$ sender and receiver locked and linked, CTS sent.

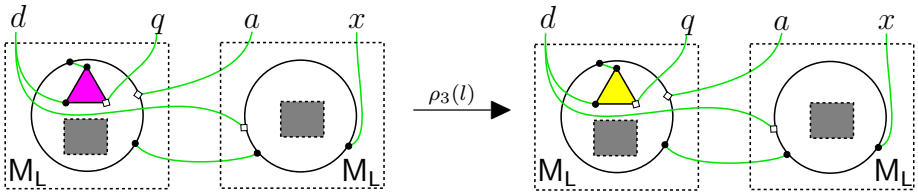


Figure 6.6: $\mathsf{R}_{\mathsf{DATA}}(l)$: sender has sent a CTS $\rightarrow$ sender has sent its data packet.

triangle is of control CTS^l, while on the right-hand side it is of control P^l. The rule can only be applied when a communication between two stations has already been initiated. This is encoded by the edge connecting the nodes of control $\mathsf{M_L}$. Observe that the sender and the receiver can sense two different sets of stations because the $\mathsf{M_L}$ nodes are placed in two different regions. The reaction rate depends on the size of the data packet to be transmitted: $\rho_3(l) = (\lambda_S + l)^{-1}$.

The fourth rule $\mathsf{R}_{\mathsf{ACK}}(t, t', l)$ specifies the last phase of the protocol: upon successfully transmitting a data packet, the receiver sends back to the sender an ACK control packet and stops transmitting. A graphical representation of the rule is given in Figure 6.7. On the left-hand side the two stations are engaged in a communication. This is encoded by the intersecting nodes of control $\mathsf{S_L}$ and by the link connecting the two $\mathsf{M_L}$s. The triangle of the sender is of control P^l, meaning that a data packet has been transmitted. On the right-hand side locks on both the stations are released and the sender's backoff timer is reset to t_{min}. This is expressed by the nodes of control M, $\mathsf{S}_{\mathsf{C}}^{t_{min}}$ and $\mathsf{S}_{\mathsf{C}}^{t'}$. Moreover, the link between the two machines is removed. Also the triangle is removed. The reaction rate is $\rho_4 = (\lambda_S + \lambda_A)^{-1}$.

A pictorial representation of how the previous four stochastic reaction rules: $\mathsf{R}_{\mathsf{RTS}}(t, l)$, $\mathsf{R}_{\mathsf{CTS}}(t, t', l)$, $\mathsf{R}_{\mathsf{DATA}}(l)$, and $\mathsf{R}_{\mathsf{ACK}}(t, t', l)$, are used to encode the packet exchange between two stations as specified by the protocol is given in Figure 6.8. From this figure, it is possible to infer the contribution of the relevant constants to the associated rates (summarised in Table 6.4). For example, rate $\rho_1(t)$, the rate for $\mathsf{R}_{\mathsf{RTS}}(t, l)$ rule, includes the sum of λ_D, the constant for DIFS, λ_R, the constant indicating the transmission time of an RTS packet and $\tau\frac{t}{2}$, the constant denoting the average duration of the sender's contention window of size t.

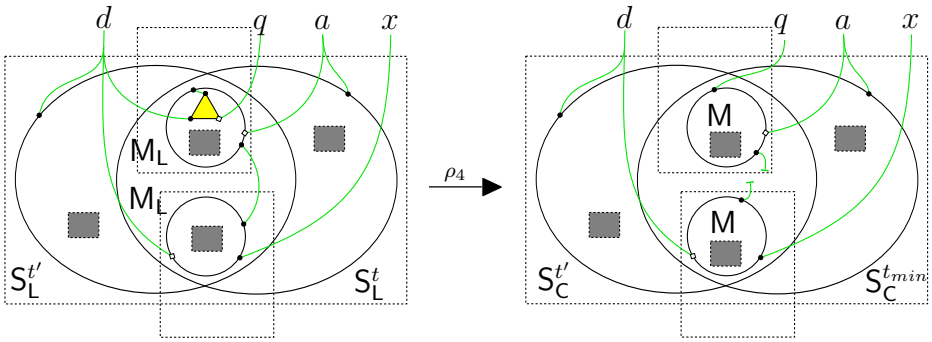


Figure 6.7: $\mathsf{R}_{\mathsf{ACK}}(t, t', l)$: data packet sent → ACK sent and locks released.

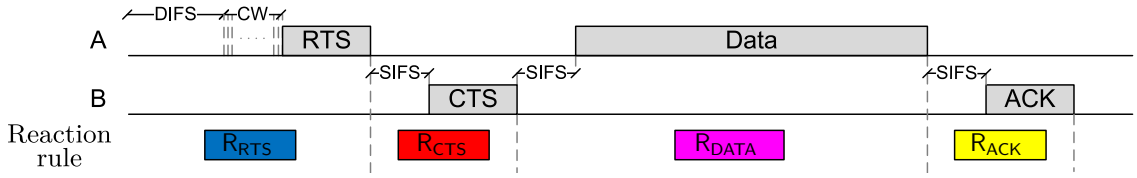


Figure 6.8: Packet exchange between two stations and corresponding reaction rules.

The fifth rule $\mathsf{R}_{\mathsf{BACK1}}(t, t', l)$ models collisions when two stations simultaneously try to transmit an RTS packet. It is expressed graphically in Figure 6.9. On the left-hand side, triangle of control RTS is placed within a node of control M_B. This represents the fact that the sender has to back off. The receiver is in a potential collision state, indicated by control M_P. On the right-hand side locks are released and the sender's contention window is increased. This is encoded by the node of control $\mathsf{S}_\mathsf{C}^{2t+1}$, the triangle of control W^l and nodes of control M. The modification of parameter t models the exponential back-off procedure. The reaction rate is $\rho_5 = \lambda_T^{-1}$, i.e. the sender's time-out expires before a CTS packet is received.

The sixth rule $\mathsf{R}_{\mathsf{BACK2}}(t, t', l)$ is very similar to the previous one. It is applied when the receiver has control M_D instead of M_P. This happens when all the other conflicting stations

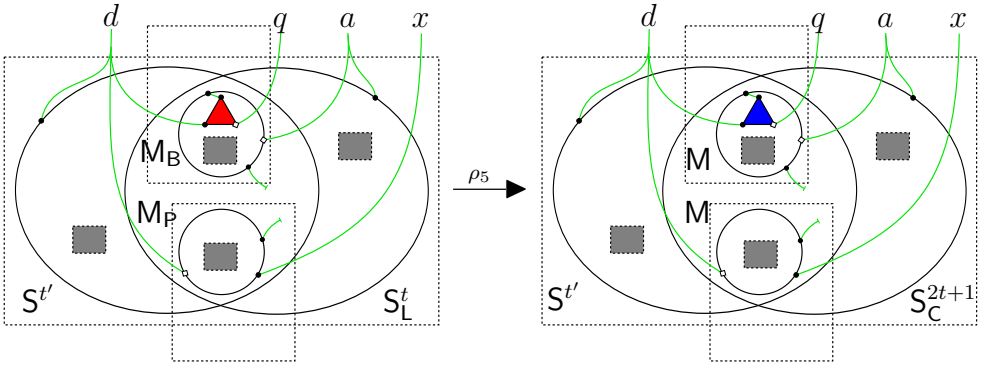


Figure 6.9: $\mathsf{R}_{\mathsf{BACK1}}(t, t', l)$: sender (not the most recent) has to back off and receiver has detected a collision → locks are released and the sender is timed out.

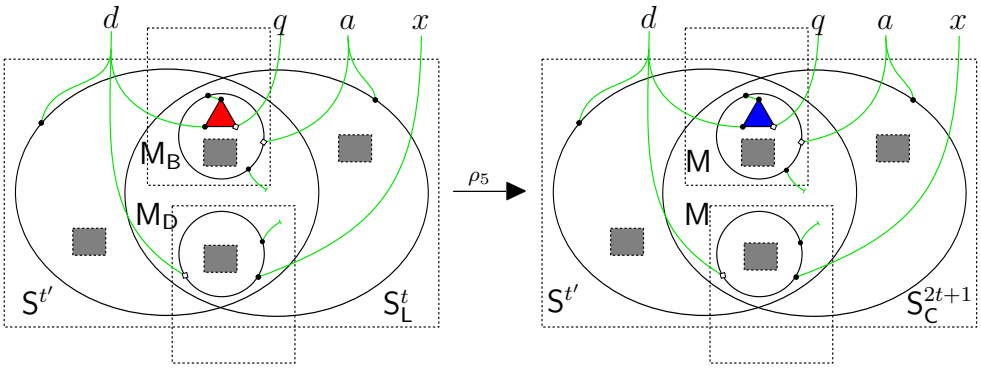


Figure 6.10: $\mathsf{R}_{\mathsf{BACK2}}(t, t', l)$: most recent sender has to back off and receiver has detected a collision $\rightarrow$ locks are released and the sender is timed out.

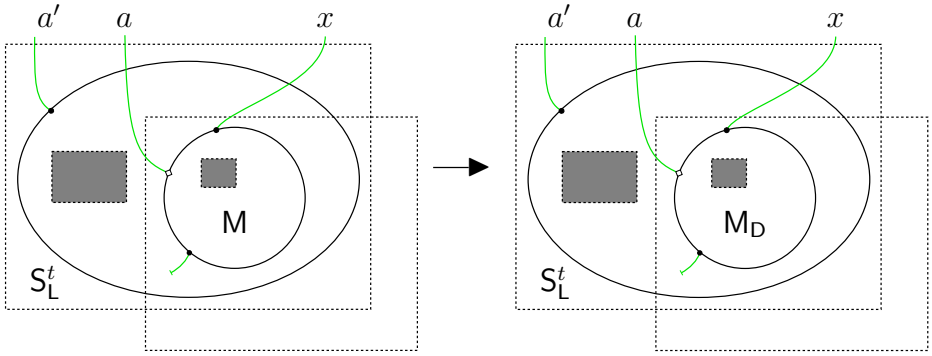


Figure 6.11: $\mathsf{R}_{\mathsf{D}}(t)$: station receives an RTS or a CTS $\rightarrow$ station defers.

have already backed off. A graphical representation of the rule is given in Figure 6.10. In both the previous two rules, nodes of control $\mathsf{S}_{\mathsf{C}}^{2t+1}$ are replaced by S_{E} when $t \geq t_{max}$. This models the protocol when the maximum number of transmission attempts is reached. Rates associated to each of the previous reaction rules are summarised in Table 6.4.

The six stochastic reaction rules described so far encode behaviours of stations as specified by the CSMA/CA RTS/CTS protocol. In the remainder of this section, a set of six instantaneous reaction rules is introduced. These rules are used to enforce conditions on the system that assure the rules presented above can only be fired in a meaningful order. More formally, they force some invariants to be satisfied before one of the stochastic rule is triggered. For example, any station receiving more than one RTS has to be marked as M_{P}. This allows the correct choice of whether rule $\mathsf{R}_{\mathsf{CTS}}(t, t', l)$ or rule $\mathsf{R}_{\mathsf{BACK1}}(t, t', l)$ has to be applied. We explain the interplay between stochastic and instantaneous rules in more detail when rule priorities are assigned (at the end of this section).

An overview of the instantaneous rules is the following. The first three rules deal with the detection and marking of stations involved in a communication. Rule $\mathsf{R}_{\mathsf{D}}(t)$ detects and marks stations that have to defer. This is depicted in Figure 6.11. On the left-hand side, an idle station has received an RTS from some other station. This is modelled by a node of control M placed within a node of control $\mathsf{S}_{\mathsf{L}}^t$. Moreover, links indicating addresses are not

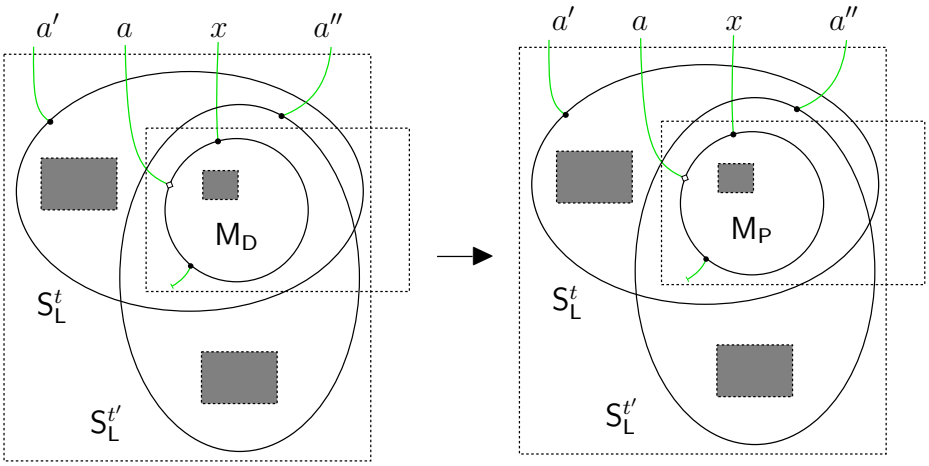


Figure 6.12: $\mathsf{R_P}(t, t')$: station receives an RTS from more than one station $\rightarrow$ station defers and detects conflict.

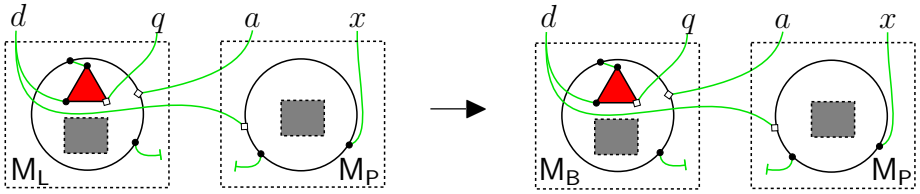


Figure 6.13: $\mathsf{R_B}(l)$: sender cannot receive a CTS from the receiver $\rightarrow$ sender has to back off.

joined together i.e. there are two distinct links to names a and a'. On the right-hand side, the station is marked by assigning control $\mathsf{M_D}$ to it.

Rule $\mathsf{R_P}(t, t')$ detects and marks stations that are in a potential conflict state. The graphical representation is given in Figure 6.12. In the left-hand side, a station receives an RTS from more than one station. This is shown by the node of control $\mathsf{M_D}$ placed within the intersection of two nodes of control $\mathsf{S_L^t}$. Observe that the station can be in the range of other machines. This is modelled by the extra region surrounding $\mathsf{M_D}$. On the right-hand side, the station detecting the conflict is marked by assigning control $\mathsf{M_P}$ to it.

Rule $\mathsf{R_B}(l)$ detects and marks stations that have to backoff when a conflict occurs. This is drawn in Figure 6.13. On the left-hand side, the sender is trying to communicate with a receiver in a conflict state. This is shown by nodes of control $\mathsf{M_L}$, RTS^l and $\mathsf{M_P}$. Note that the triangle and the receiver are linked together on name d. On the right-hand side, the sender is marked by assigning control $\mathsf{M_B}$ to it.

The last three instantaneous reactions are used to release the locks when a communication is terminated. Rule $\mathsf{R_{UD}}(t)$ releases the lock of a deferred station when its NAV reaches 0. This is modelled by the node of control $\mathsf{M_D}$ enclosed by a node of control $\mathsf{S_C^t}$ on the left-hand side. On the right-hand side, the lock is released by assigning control M to the station. A graphical representation of the reaction rule is given in Figure 6.14.

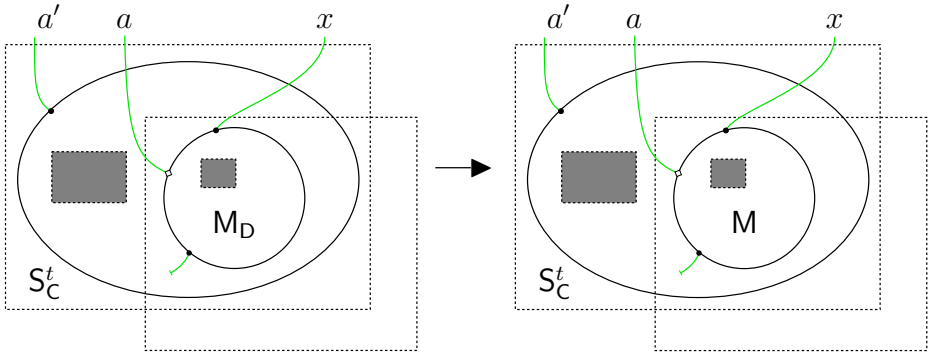


Figure 6.14: $\mathsf{R_{UD}}(t)$: the NAV of a deferred station expires $\rightarrow$ lock released.

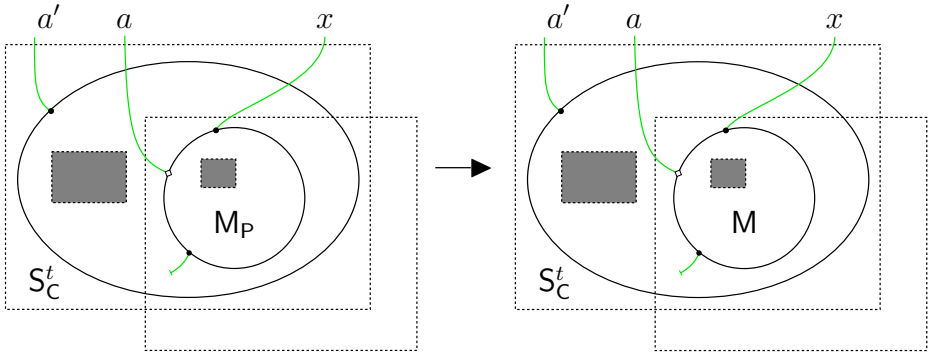


Figure 6.15: $\mathsf{R_{UP}}(t)$: the NAV of a station in a conflict state expires $\rightarrow$ lock released.

Similar to the previous rule $\mathsf{R_{UP}}(t)$ is used to release the lock of a station in a potential conflict state. This is encoded by the node of control $\mathsf{M_P}$ on the left-hand side. The rule is depicted in Figure 6.15.

Finally, rule $\mathsf{R_{UC}}(t)$ models the release of the lock at the end of a transmission. The graphical representation is in Figure 6.15. On the left-hand side, the station is idle and ready to release the lock. This is encoded by the node of control $\mathsf{S_C^t}$ linked to its child node of control M. On the right-hand side, the final stage of the protocol is performed by substituting $\mathsf{S_C^t}$ with S^t. Note that this rule is not in conflict with rules $\mathsf{R_{UD}}(t)$ and $\mathsf{R_{UP}}(t)$ because it can only be applied to a sender or a receiver.

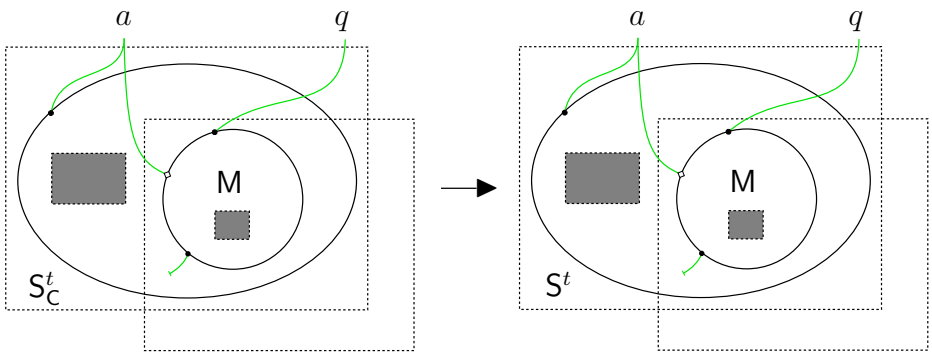


Figure 6.16: $\mathsf{R_{UC}}(t)$: sender or receiver terminated a transmission $\rightarrow$ lock released.

$\mathfrak{P}_5 = \{\mathsf{R_{UD}}(t), \mathsf{R_{UP}}(t)\}$	$\mathfrak{P}_4 = \{\mathsf{R_{UC}}(t)\}$	$\mathfrak{P}_3 = \{\mathsf{R_D}(t)\}$	$\mathfrak{P}_2 = \{\mathsf{R_P}(t, t')\}$
$\mathfrak{P}_1 = \{\mathsf{R_B}(l)\}$	$\mathfrak{P}_0 = \{\mathsf{R_{RTS}}(t, l), \mathsf{R_{CTS}}(t, t', l), \mathsf{R_{DATA}}(l), \mathsf{R_{ACK}}(t, t', l),$ $\mathsf{R_{BACK1}}(t, t', l), \mathsf{R_{BACK2}}(t, t', l)\}$		

Table 6.5: Priority classes. $\mathfrak{P}_i < \mathfrak{P}_j$ when $i < j$.

Reaction rules are organised into priority classes as summarised in Table 6.5. All stochastic rules belong to $\mathfrak{P}_0$, i.e. the class with lowest priority. Therefore, any stochastic rule can be applied only when no instantaneous rule can be applied. A brief description of the priority classes is the following.

$\mathfrak{P}_5$ is the class with highest priority. Rules assigned to this class mark with control M an idle station in a clear signal (i.e. of control $\mathsf{S}^t_{\mathsf{C}}$). Class $\mathfrak{P}_4$ has one rule that marks with control M a sender or receiver inside its clear signal, the signal is marked as S^t. Since $\mathfrak{P}_4 < \mathfrak{P}_5$, all idle stations must be marked as M before the clear signal containing them is marked as S^t. Similarly, rules in priority classes $\mathfrak{P}_3, \mathfrak{P}_2, \mathfrak{P}_1$ mark a station inside a locked signal ($\mathsf{S}^t_{\mathsf{L}}$) as deferred ($\mathsf{M_D}$), in a potential conflict state ($\mathsf{M_P}$) and backing off ($\mathsf{M_B}$), respectively. This is necessary because rules with higher priorities may have marked a station as M even if it is contained in a locked signal. Finally, stochastic rules in $\mathfrak{P}_0$ can be applied safely since all the preconditions are enforced by the prior application of the rules with higher priorities.

Discussion

A difficulty with rewriting rules in general, and therefore a problem for us here, is how to deal with situations where we want to rewrite because a condition does *not* apply. For instance, we want to apply rule $\mathsf{R_{RTS}}(t, l)$ only when the sender is not in a locked signal. In our approach, this issue has been solved by introducing priorities on reaction rules as described above. However, there are two modelling choices other than using PSBRSs.

One option is to parameterise the reactions on the topology of the network. For instance, the first rule (RTS packet transmission) can be modified so all the idle signals sensed by the sender are explicitly listed. Since only idle signals are included and no other signals are allowed via shared regions, then the rule can only be applied when the sender is not within a locked signal. An additional parameter n indicating the number of signals in the sender's range is required, i.e. $\mathsf{R_{RTS}}(t, l, n)$. A major drawback of this approach is that a different rule has to be instantiated for every network topology. Moreover, the specification of the rule becomes cumbersome and difficult to read. An advantage is that less controls are required for the modelling of the protocol's phases and no instantaneous reaction rule is introduced.

The other option involves extending the notion of reaction rule to include what may be called *conditional reaction rules*, having the form (φ, R, R', ρ), where φ is a **BiLog** predicate

expressing a condition imposed upon the context in which redex R is matched. The intended semantics is that a reaction can be applied to a state S only if $C \models \varphi$ (or alternatively $D \models \varphi$), where C (or D) is computed by the usual decomposition induced by matching: $S = C \circ (R \otimes \mathsf{id}_I) \circ D$. Also with this option, less controls are required and instantaneous reaction rules can be avoided. A disadvantage is that it is often necessary to specify predicates expressing universal quantifiers or the absence of a pattern. For example, a predicate for rule $\mathsf{R}_{\mathsf{RTS}}(t, l)$ needs to express that *all* the signals in the context are idle. In general, this kind of predicate cannot be reduced to matching of bigraphs in a straightforward way. A possible work-around consists of iteratively applying an appropriate reaction rule to *tag* the parts of a bigraph that do satisfy the predicate and then checking for the existence of untagged patterns. If one is found, then the predicate is not satisfied. A more detailed description is given in Chapter 7 where we will use this approach extensively to model network policies. Finally, note that the applications of tagging reaction rules introduced by predicates is equivalent to the explicit application of instantaneous reaction rules in our PSBRS.

Our approach allows the specification of concise rules that can be applied to any network topology. This conciseness is important not only because it allows for a clearer description of the model, as we showed in this section, but also because it leads to matching instances that are quicker to solve. The motivation lies in the fact that the computational complexity of the matching problem is $\mathcal{O}(s^r)$ in the general case, where s and r are the number of nodes of a state (used as the target) and a redex (used as the pattern), respectively. However, by avoiding parameterised rules, a redex in our model can have at most 8 nodes (see for example reaction rule $\mathsf{R}_{\mathsf{CTS}}(t, t', l)$). Therefore, the complexity of these instances is always $\mathcal{O}(s^8)$. This means they can easily be solved in polynomial time [31]. On the other hand, the size of a parameterised redex grows with the number of nodes in the system and can be $r \propto s$ in the worst case. This happens, for instance, when in the modified first rule all s signals are idle and in the range of the sending station.

A common issue with instantaneous rules is the generation of intermediate states that may not represent meaningful network configurations. This can be computationally expensive because every time a state is computed, it has to be checked for equality against all the states already visited as we explained for the implementation of the BigraphER rewriting engine in Chapter 5. Therefore, it is crucial to minimise the number of states stored during the execution of the model. This is possible by treating instantaneous rules like rewriting within an equivalence class, and only storing a canonical form, after applying all possible instantaneous rules. We explain this procedure in Section 6.6.

A further benefit of our modelling strategy is that the priority hierarchy helps understanding the details of the protocol.

We note that we have defined exponential rates from time constants, i.e. if we have a

constant k waiting (or transmission, or sum of constants) time, then we use as CTMC rate k^{-1}. Inverse averages are used as rates for random intervals.

Finally, we remark that although the protocol assumes a static topology, we could easily define rules for station movement in and out of a signal range, i.e. protocol behaviour in a dynamic topology.

6.5 Execution of an example network

In this section we describe two example executions of our model starting from initial state N_0, the bigraph encoding of an example WLAN with three stations given in Figure 6.3.

The first execution encodes the successful transmission of a packet from station A to station B, with no collisions occurring. It is the result of the application of reaction rules $\mathsf{R}_{\mathsf{RTS}}(t_{\mathrm{A}}, l)$, $\mathsf{R}_{\mathsf{D}}(t_{\mathrm{B}})$, $\mathsf{R}_{\mathsf{CTS}}(t_{\mathrm{A}}, t_{\mathrm{B}}, l)$, $\mathsf{R}_{\mathsf{D}}(t_{\mathrm{C}})$, $\mathsf{R}_{\mathsf{DATA}}(l)$, $\mathsf{R}_{\mathsf{ACK}}(t_{\mathrm{A}}, t_{\mathrm{B}}, l)$, $\mathsf{R}_{\mathsf{UD}}(t_{\mathrm{C}})$, $\mathsf{R}_{\mathsf{UC}}(t_{\mathrm{A}})$ and $\mathsf{R}_{\mathsf{UC}}(t_{\mathrm{B}})$, in that order. The dynamic evolution of the network is represented pictorially in Figure 6.17. Observe that in state N_6 (second diagram in the third row), the sender's contention windows size is reset: $t'_{\mathrm{A}} = t_{min}$. An alternative execution can be obtained by changing the order in which the locks are released, i.e. by applying $\mathsf{R}_{\mathsf{UC}}(t_{\mathrm{B}})$ before $\mathsf{R}_{\mathsf{UC}}(t_{\mathrm{A}})$. The resulting state is again N_9. Execution $N_0 \xrightarrow{\rho_1(t_{\mathrm{A}})}_{\mathsf{RTS}} \cdots \longrightarrow_{\mathsf{UC}} N_9$ can further proceed in an analogous way by transmitting the remaining packet from C to B. The symmetrical evolution in which station C sends its packet to B can be obtained from initial state N_0 by applying reaction rules $\mathsf{R}_{\mathsf{RTS}}(t_{\mathrm{C}}, k)$, $\mathsf{R}_{\mathsf{D}}(t_{\mathrm{B}})$, $\mathsf{R}_{\mathsf{CTS}}(t_{\mathrm{C}}, t_{\mathrm{B}}, l)$, $\mathsf{R}_{\mathsf{D}}(t_{\mathrm{A}})$, $\mathsf{R}_{\mathsf{DATA}}(k)$, $\mathsf{R}_{\mathsf{ACK}}(t_{\mathrm{C}}, t_{\mathrm{B}}, k)$, $\mathsf{R}_{\mathsf{UD}}(t_{\mathrm{A}})$, $\mathsf{R}_{\mathsf{UC}}(t_{\mathrm{C}})$ and $\mathsf{R}_{\mathsf{UC}}(t_{\mathrm{B}})$, in that order.

The second execution encodes a transmission in which a collision occurs. It is shown in Figure 6.18. The initial state is bigraph N_2 obtained in the trace described above. As can be seen, in this evolution station C sends its RTS before B transmits its CTS. Therefore, stations C and A have to defer and the backoff scheme is executed. This is encoded by the application of rules $\mathsf{R}_{\mathsf{BACK1}}(t_{\mathrm{A}}, t_{\mathrm{B}}, l)$ and $\mathsf{R}_{\mathsf{BACK2}}(t_{\mathrm{C}}, t_{\mathrm{B}}, k)$. Station C can try to retransmit (by applying rule $\mathsf{R}_{\mathsf{RTS}}(t'_{\mathrm{C}}, k)$) only after its timeout has expired. The contention windows of senders A and C are exponentially increased in states N'_4 and N'_7, respectively. Therefore, $t'_i = 2t_i + 1$ with $i \in \{\mathrm{A}, \mathrm{C}\}$. An alternative execution can be computed by inverting the order of application of rules $\mathsf{R}_{\mathsf{B}}(l)$ and $\mathsf{R}_{\mathsf{B}}(k)$. This leads to state N'_3. Another interleaving is obtained when $\mathsf{R}_{\mathsf{BACK1}}(t_{\mathrm{C}}, t_{\mathrm{B}}, k)$ is applied before $\mathsf{R}_{\mathsf{BACK2}}(t_{\mathrm{A}}, t_{\mathrm{B}}, l)$. Also in this case, the final state is still N'_8.

The two evolutions above show how instantaneous rules must be applied before stochastic rules as required by the priority hierarchy of our model. For instance, in the second trace, rules $\mathsf{R}_{\mathsf{P}}(t_{\mathrm{A}}, t_{\mathrm{C}})$, $\mathsf{R}_{\mathsf{B}}(l)$ and $\mathsf{R}_{\mathsf{B}}(k)$ are applied before rule $\mathsf{R}_{\mathsf{BACK1}}(t_{\mathrm{A}}, t_{\mathrm{B}}, l)$.

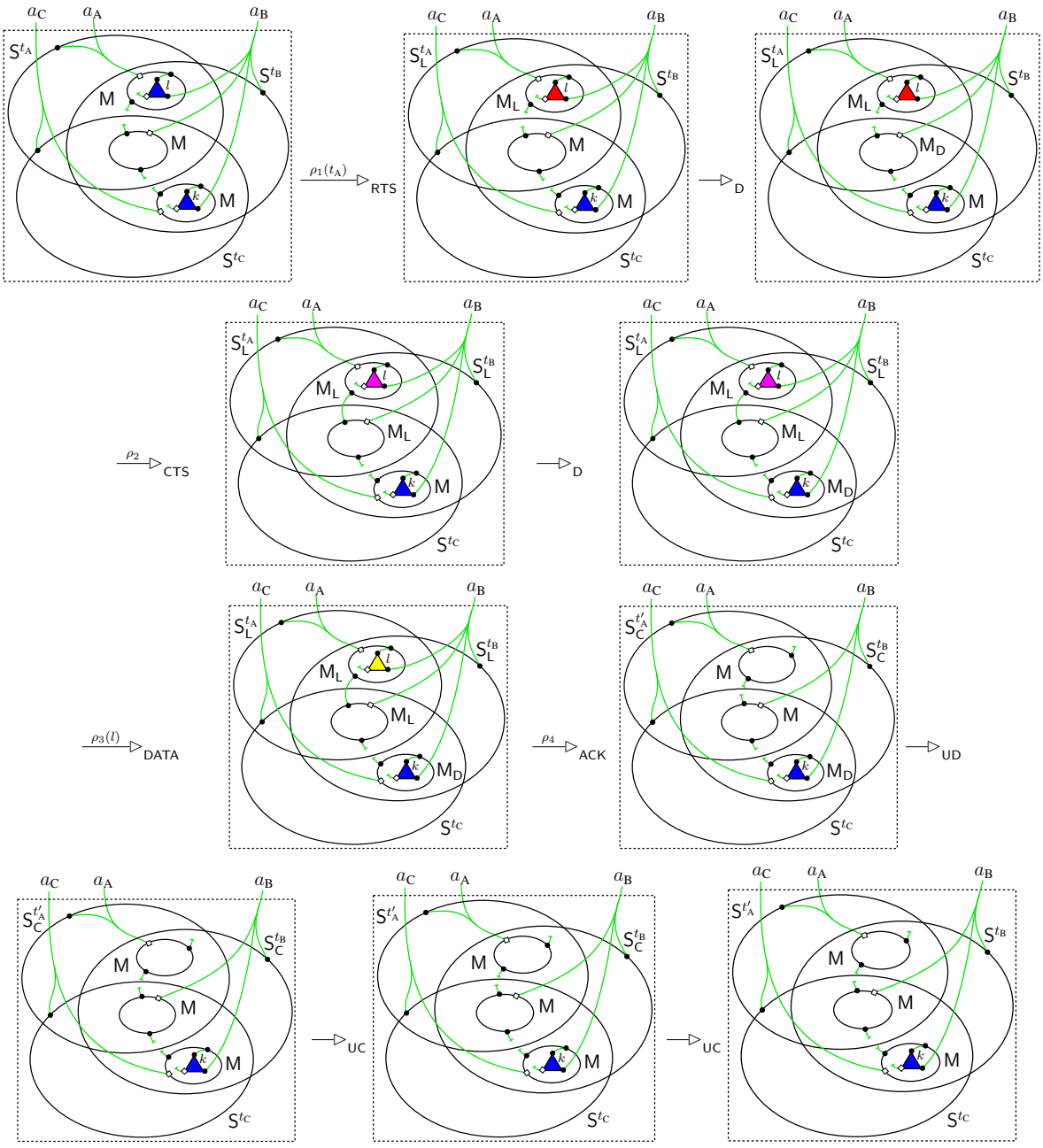


Figure 6.17: Evolution $N_0 \xrightarrow{\rho_1(t_A)}_{\mathsf{RTS}} \cdots \longrightarrow_{\mathsf{UC}} N_9$ showing the successful transmission of a packet from sender A to destination B. Sender's contention window size is reset after the application of rule $\mathsf{R}_{\mathsf{ACK}}(t_A, t_B, l)$ (i.e. $t'_A = t_{min}$).

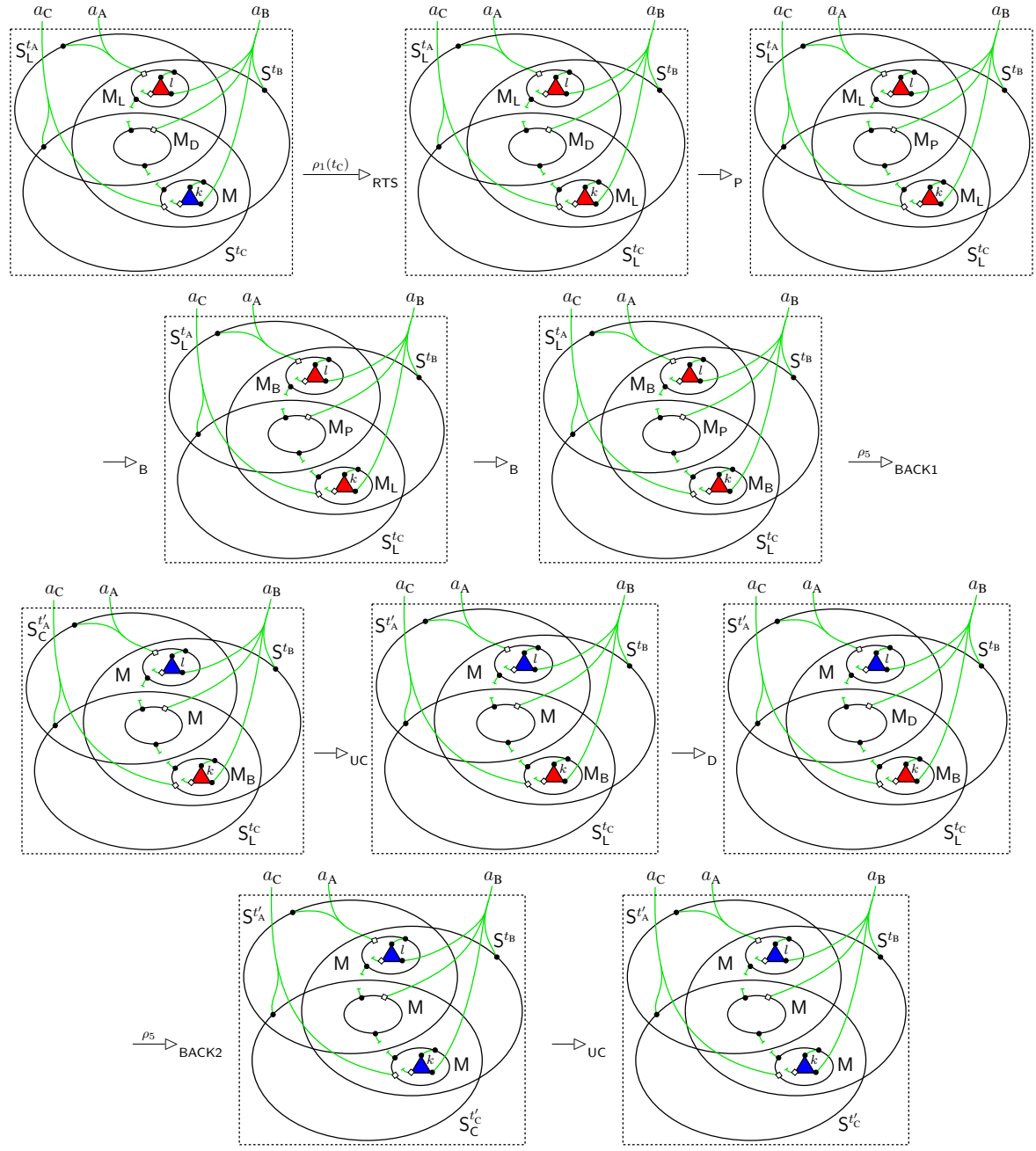


Figure 6.18: Evolution $N_2 \xrightarrow{\rho_1(t_C)}_{\mathsf{RTS}} N'_0 \cdots \longrightarrow_{\mathsf{UC}} N'_8$ showing the collision occurring when senders A and C try to transmit to the same destination B. The contention windows of the senders are exponentially increased: $t'_i = 2t_i + 1$ with $i \in \{\mathrm{A}, \mathrm{C}\}$.

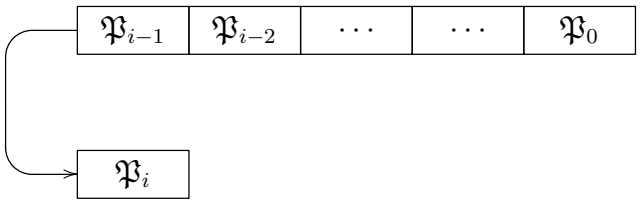


Figure 6.19: Structure of the priority queue. Class $\mathfrak{P}_i$ has the highest priority.

6.6 CTMC analysis

We now explain how the BigraphER rewriting engine computes the CTMC capturing the semantics of a stochastic model with instantaneous reaction rules and priorities. In particular, we describe rewriting with priorities and a method to minimise the number of states stored in memory during execution.

Intuitively, in the BFS loop the rewriting engine applies the reaction rules of the SBRS as described in Chapter 5. However, it now stores only the resulting states that cannot be further rewritten by applications of instantaneous rules. Moreover, priorities are implemented by selecting at every step only the reaction rules belonging to the class currently having the highest priority. This is done by introducing a second queue[2] that stores classes of reaction rules.

First, we give details of the implementation of priorities. At each iteration of the BFS loop, the first elements, S and $\mathfrak{P}_i$, are removed from the state queue and the priority queue, respectively. Note that priority classes are stored in the queue in decreasing order of priority as shown in Figure 6.19. Hence, $\mathfrak{P}_i$ is the class having the highest priority. If some reactions can be applied, all the possible reconfigurations are computed and the new states are stored both in the hash table and in the state queue. Then, the priority queue is reset to its original state and the next iteration of the BFS loop is executed. On the other hand, if no reaction rule in $\mathfrak{P}_i$ can be applied, the next priority in the queue, $\mathfrak{P}_{i-1}$, is selected and the possible reconfigurations are computed using its reaction rules. This iteration on the priority queue is repeated either until a reaction rule is applied or no applicable reaction in $\mathfrak{P}_0$ is found. Observe that this implementation assures that reaction rules of higher priority are always applied before rules of lower priority. Moreover, the implementation can also be used when no rule priorities are specified in the model by introducing a single priority class containing all the reaction rules.

Second, we now describe our approach to reduce the number of intermediate states generated by the application of instantaneous reaction rules. Again, it consists of a modification of the BFS loop in the rewriting engine. More precisely, whenever the loop finds a reconfiguration S' of a state S (i.e. $S \xrightarrow{\rho} \triangleright S'$), the unique fixed point S^* is computed by iteratively

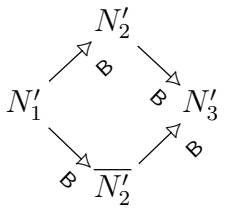
[2]The first queue is used by the BFS loop to store states.

applying to S' all the instantaneous rules. Then, state S^* is stored in memory in place of S' as a reconfiguration of S (i.e. $S \xrightarrow{\rho} \triangleright S^*$) and the BFS loop continues with the next element in the state queue. A formal representation of the procedure is given by

$$S \xrightarrow{\rho} \triangleright S' \overbrace{\longrightarrow \triangleright}^{\mathfrak{P}_i *} S'' \overbrace{\longrightarrow \triangleright}^{\mathfrak{P}_{i-1} *} S''' \cdots \overbrace{\longrightarrow \triangleright}^{\mathfrak{P}_j *} S^*$$

with $j < i - 1$. We write $\longrightarrow \triangleright^*$ to indicate zero or more instantaneous reactions. All the priority classes, $\mathfrak{P}_i < \cdots < \mathfrak{P}_j$, contain only instantaneous reaction rules. Note that all the rule applications respect the priority hierarchy and that all the intermediate states $S', S'', S''', \ldots$ obtained during the computation of S^* are discarded. Furthermore, by definition of fixed point, no instantaneous rule can be applied to S^f. This always forces the first reaction to have rate $\rho \neq \infty$. This approach can only be used if each priority class whose instantaneous reactions are ignored has a unique fixed point (as is the case in our model).

The procedures described above can be explained by showing how the rewriting engine computes the CTMC starting from bigraph N_0, the example WLAN of three stations used in the example in the previous section. The parameters of the model are packet transmission times $l = 8464\mu$s and $k = 4368\mu$s.[3] Contention windows for stations A, B and C are $t_{\mathrm{A}}, t_{\mathrm{B}}, t_{\mathrm{C}} = 15$. In the PSBRS, iterated applications of instantaneous reaction rules belonging to the same priority class always yield a fixed point. Hence, intermediate states generated by $\mathfrak{P}_5, \ldots, \mathfrak{P}_1$ can be ignored. Consider for instance class $\mathfrak{P}_1$ and bigraph N'_1, in the evolution illustrated in Figure 6.18[4]. There are two reaction rules we can apply, leading to different intermediate terms: $\mathsf{R}_{\mathsf{B}}(l)$ marks station A as M_{B} while $\mathsf{R}_{\mathsf{B}}(k)$ does the same for station C. They generate reactions $N'_1 \longrightarrow \triangleright_{\mathsf{B}} N'_2$ and $N'_1 \longrightarrow \triangleright_{\mathsf{B}} \overline{N'_2}$, respectively. The second rule we apply in both cases leads to the same result, i.e. the fixed point. More in detail, reactions $N'_2 \longrightarrow \triangleright_{\mathsf{B}} N'_3$ and $\overline{N'_2} \longrightarrow \triangleright_{\mathsf{B}} N'_3$ are obtained when rules $\mathsf{R}_{\mathsf{B}}(k)$ and $\mathsf{R}_{\mathsf{B}}(l)$ are applied, respectively. This can be shown with a classical confluence diagram thus



Another property of our model is that any reaction $S \xrightarrow{\rho} \triangleright S'$ can only be obtained by a single application of a stochastic reaction rule R. Therefore, $\mu_R[S, S'] = 1$ and $rate[S, S'] = \rho$.

[3] Parameters l and k correspond to the transmission times of data-packets with payloads of 512 bytes and 1024 bytes, respectively. The values are obtained by adding to the payload lengths 30 bytes for the MAC header and 4 bytes for the frame check sequence (FCS).

[4] Third diagram in the first row.

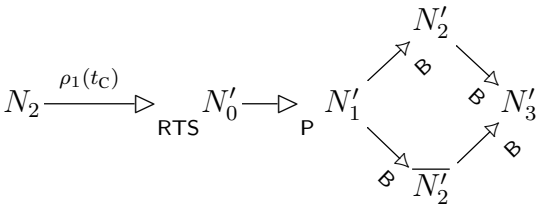
This may not hold with more complex topologies.

The resulting CTMC (indicated by $\mathcal{M}$) is given in Figure 6.20. Reactions corresponding to actions on station C are indicated with left facing arrows and those on station A by right facing arrows. The diagram is organised into seven diamond sub-parts each one representing a different transmission attempt after a collision. Each diamond is indicated by $\mathcal{M}^i$ with $1 \leq i \leq 7$. Note that only the first two diamonds are shown in Figure 6.20. The central node of a diamond (indicated with $\circ$) is the top vertex of the successive one as for instance state N'_8 in $\mathcal{M}^1$ and $\mathcal{M}^2$. The bottom node of $\mathcal{M}^1$ (indicated with $\diamond$) is shared between all the seven diamonds. It represents the state in which both packets are successfully transmitted to B. It is shared because the contention windows of stations A and C are reset to $t_{min} = 15$ in every diamond after every transmission of the ACK control packet. The reactions going to the centre of the diamond encode transmission collisions. A side of a diamond is formed from four reactions, encoding the successful transmission of a packet. For instance, side $N_0 \longrightarrow\!\!\triangleright^* N_9$ encodes transmission of A's packet before C's packet at the first attempt. Dashed arrows correspond to the encoding of the two executions described in Section 6.5. The CTMC has two deadlock states. One is node $\circ$ in $\mathcal{M}^7$ representing stations A and C when the maximum number of transmission attempts is reached, i.e. both signals are $\mathsf{S_E}$. The other is node $\diamond$ described above.

As can be seen in the diagram, the CTMC does not contain the intermediate states generated by the application of instantaneous rules. Take for instance reaction $N_0 \xrightarrow{\rho_1(t_\mathrm{A})} \triangleright N_2$ in $\mathcal{M}^1$. By inspecting the execution in Figure 6.17, we see that it corresponds to reactions

$$N_0 \xrightarrow[\mathsf{RTS}]{\rho_1(t_\mathrm{A})} \triangleright \; N_1 \longrightarrow\!\!\triangleright_{\mathsf{D}} \; N_2 \;\; .$$

In this case, intermediate bigraph N_1 is not used as a CTMC state. Also note that no instantaneous reaction rule can be applied to N_1. Analogously, reaction $N_2 \xrightarrow{\rho_1(t_\mathrm{C})} \triangleright N'_3$ in $\mathcal{M}^1$ corresponds to



as shown in Figure 6.18. Here, intermediate bigraphs N'_0, N'_1, N'_2 and $\overline{N'_2}$ are discarded. Note that only one between N'_2 and $\overline{N'_2}$ is computed by the rewriting engine, because only one path is needed to compute fix point N'_3. In both the reactions considered, the stochastic rate is the rate of the stochastic reaction rule applied at the beginning of the corresponding sequence.

Finally, we note that the CTMCs generated from our model, for any finite topology, will be finite because there are a finite number of transmissions, a finite number of packets, and a

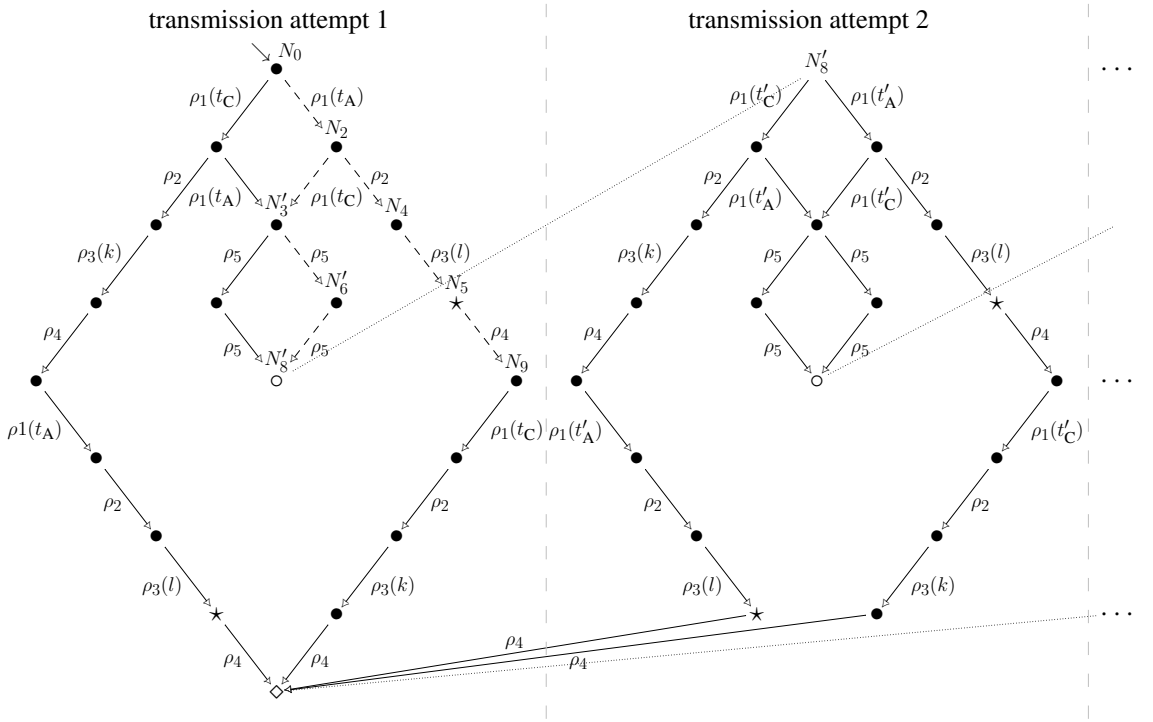


Figure 6.20: Structure of CTMC $\mathcal{M}$ derived from the PSBRS on initial state N_0 (see Figure 6.3b). Dashed arrows correspond to the two executions described in Figure 6.17 and Figure 6.18.

finite number of stations.

6.6.1 Analysis of quantitative properties

It is possible to express quantitative properties about the behaviour of our model by combining **BiLog** and **CSL** (Continuous Stochastic Logic)[5]. We use the fragment of **BiLog** described in Chapter 5 to express predicates that are used as atomic propositions in **CSL**. Recall that this kind of predicate can be reduced to one or more instances of matching of bigraphs. For example, we define pattern $P_{\varphi_{error}} = \mathsf{S_E}$ to encode predicate φ_{error}, which is true whenever a station reaches the maximum number of transmission attempts, namely there is a signal node in error state. This predicate can then be used in any **CSL** formula such as $\psi_1 = \mathbf{P}_{<0.001}(\mathbf{F}\,\varphi_{error})$, i.e. eventually a station is in error state with probability less than 0.001. Note that the only state satisfying φ_{error} is node $\circ$ in $\mathcal{M}$[7].

Quantitative properties can be evaluated with standard tools for CTMC model checking. Here we use the probabilistic model checker Prism [40] on some simple **CSL** formulae expressing properties on $\mathcal{M}$. In particular, we take advantage of the *explicit model import* feature in Prism which allows the importation of a CTMC's rate matrix and its labelling function. These are the textual outputs of the BigraphER rewriting engine. It is also possible to specify properties that evaluate to a numerical value. Let us analyse some examples.

[5] See Appendix B for syntax and semantics.

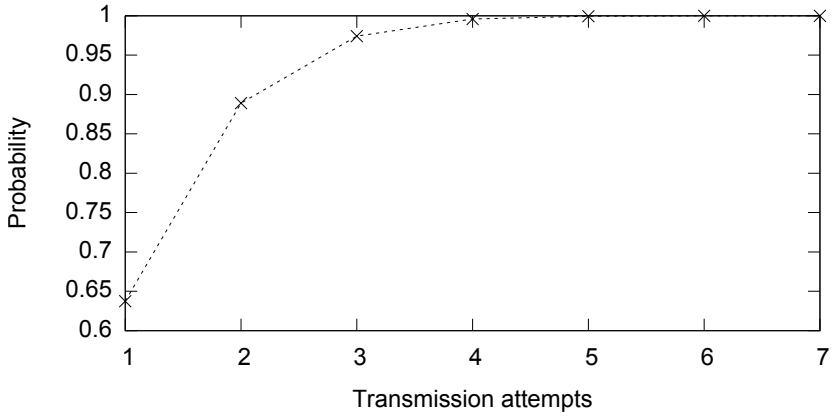


Figure 6.21: Cumulative probability of sending A's packet against the number of transmission attempts.

Example 6.6.1. The probability of a station being in an error state is given by formula $\mathbf{P}_{=?}(\mathbf{F}\,\varphi_{error}) = 6.08 \times 10^{-7}$, which implies that $\mathcal{M} \models \psi_1$. Such a low value is explained by the fact that only two stations compete for transmission in our example topology. ■

Example 6.6.2. The probability that a collision occurs is expressed by $\mathbf{P}_{=?}(\mathbf{F}\,\varphi_{collision}) = 0.36$. Predicate $\varphi_{collision}$ is encoded by the matching of pattern $P_{\varphi_{collision}} = \mathsf{M}_{\mathsf{B}xy} \parallel \mathsf{M}_{\mathsf{B}zw}$. It corresponds to state N_3' in $\mathcal{M}^1$ and the states in the same position in $\mathcal{M}^i$ with $1 < i \leq 7$. ■

Example 6.6.3. The probability of station A successfully transmitting its packet in $1 \leq n \leq 7$ transmission attempts is expressed by $\mathbf{P}_{=?}(\varphi_{\mathrm{A}}^n)$. The cumulative plot is given in Figure 6.21. Predicate φ_{A}^n corresponds to pattern

$$\text{share}\ (\mathsf{id} \parallel /x\, \mathsf{M}_{\mathsf{L}rx}.(\mathsf{id} \mid \mathsf{P}_{xd}^{l} \mid \mathsf{A}_{a_{\mathrm{A}}}.1)\ \text{by}\ [\{0\},\{0,1\}]\ \text{in}\ (\mathsf{S}_{\mathsf{L}a_{\mathrm{A}}}^{t} \parallel \mathsf{id}_{1,dra_{\mathrm{A}}})$$

with $t = 2^{n+3} - 1$. Therefore, the labelling function marks the nodes indicated with $\star$ in $\mathcal{M}$. Observe that the predicate cannot be defined to simply match state $\diamond$ because information about the number of transmission attempts is lost after every application of reaction rule $\xrightarrow{\rho_4}_{\mathsf{ACK}}$ when the contention window is reset. ■

Discussion

Previous studies of the IEEE 802.11 standard have focussed mainly on the quantitative analysis of the basic access mechanism. Two example works that consider the RTS/CTS handshake are [10] and [17]. The results reported by those authors are used to evaluate the performance of the protocol, e.g. system throughput and congestion rate, and are obtained by means of simulation. In both cases, the network topology is not encoded directly. In more detail, a probability of collision p is assigned to each station. Hence, higher values of p allow one to encode "denser" topologies. Another example of analysis is the analytic approach

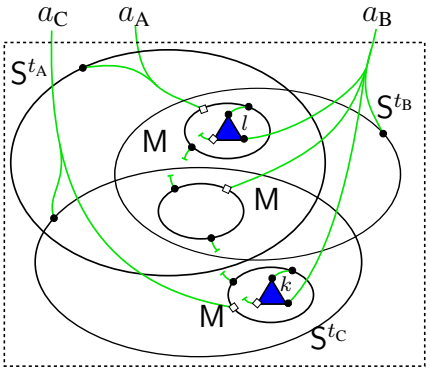


Figure 6.22: Example topology not meeting the assumptions of the protocol: B is in the range of C but C is not in the range of B.

presented in [1], but it is difficult to compare with our analysis, since also in this case the network topology is modelled implicitly by a collision probability.

In general, the model checking approach has some benefits over simulation [20]. Probably, the most relevant advantage in the context of wireless networks modelling is that all possible states are discovered. In particular, model checking can discover those states having a reachability probability that is too small to be discoverable by simulation. This is fundamental to obtain precise and trustworthy results especially when analysing safety-critical systems and border-line or unusual system's behaviours. Consider for instance Example 6.6.1 in which the probability to reach an error state is computed, i.e. $\mathbf{P}_{=?}(\mathbf{F}\,\varphi_{error}) = 6.08 \times 10^{-7}$. In this case, simulation based approaches could erroneously conclude that the system can never be in an an error state (thus giving a wrong answer) because the probability to discover such states is too low to be found in a reasonable number of simulation runs. We remark that our bigraphical model can also be used by BigraphER to simulate a system when there are memory constraints or when only an average-case analysis is required.

The quantitative properties we presented earlier in this section serve mainly to show how SBRS can effectively be analysed and to highlight the expressive power of both bigraphs and **BiLog**. Following these examples, more advanced analysis can be carried out by researchers in wireless networking and practitioners in the field of performance evaluation. For instance, network throughput can be studied by varying the parameters of the protocol and the exponential increase of the contention window size can be compared with different backoff schemes. Another interesting investigation is to analyse the effects of the RTS/CTS mechanism on different network topologies (e.g. sparse or dense) to find out when the mechanism improves network throughput and when it introduces overhead.

Our bigraphical model can easily be extended to analyse more realistic systems, i.e. systems in which the assumptions made in the 802.11 protocol are not met. For example, formation rule Φ of sorting $\Sigma_{802.11}$ given in Table 6.3 allows for the specification of networks in which signals do not have equal power as it is the case in the WLAN encoded by the

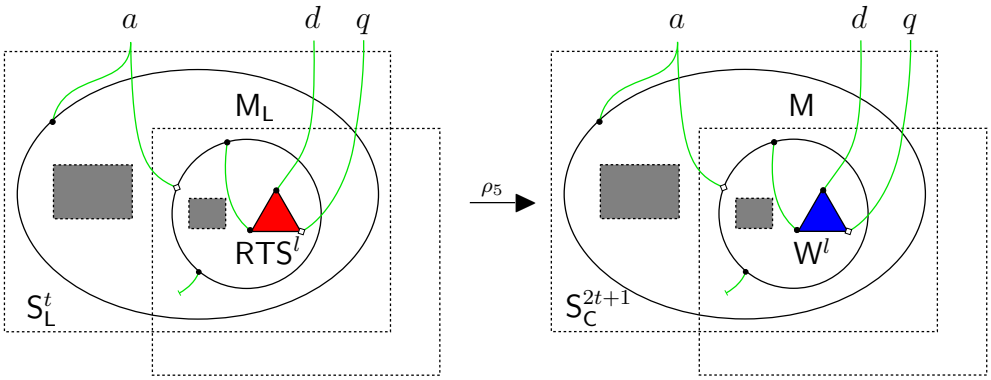


Figure 6.23: $\mathsf{R_T}(t, l)$: machine is locked and an RTS sent $\rightarrow$ lock is released and the machine is timed out.

bigraph in Figure 6.22. The diagram shows that receiver B is within the signal range of station C but C is not in the range of B. Therefore, station C cannot receive a CTS packet from B and reaction rule $\mathsf{R_{CTS}}(t, t', l)$ can never be applied. In order to model this kind of system, stochastic reaction rule $\mathsf{R_T}(t, l)$ given in Figure 6.23 need be introduced. It encodes the behaviour of a sender timing out before a CTS packet is received. It can be thought as the opposite of $\mathsf{R_{RTS}}(t, l)$ described in Section 6.4 where the size of the contention window is increased in the reactum as in reaction rules $\mathsf{R_{BACK1}}(t, t', l)$ and $\mathsf{R_{BACK2}}(t, t', l)$. Also in this case, node of control $\mathsf{S_C^{2t+1}}$ is replaced by $\mathsf{S_E}$ when $t \geq t_{max}$. With the addition of this new reaction rule, the system can evolve to a state in which C's signal has control $\mathsf{S_E}$ after seven transmission attempts.

The model can be further refined by adding reaction rules encoding the movement of stations in and out of signals. They encode both the spatial movement of mobile stations and the variation of signal coverage due to frequency interference. These reaction rules are usually slower than the other stochastic reaction rules.

6.7 Summary

In this chapter, we presented a model of the 802.11 CSMA/CA RTS/CTS based on stochastic bigraphs with sharing. Section 6.1 gave an overview of the application domain and a brief survey of related works on modelling and analysis of wireless protocols. An informal description of the protocol was provided in Section 6.2. In Section 6.3 and Section 6.4, the PSBRS modelling the protocol was formally defined. The use of priorities allows us to specify reaction rules with fixed size regardless of topology. This leads to matching instances that are solvable in polynomial time. Two sample executions of an example network of three stations were described in Section 6.5. Finally, Section 6.6 introduced the CTMC resulting form the previous example and some quantitative analysis results obtained using Prism.

In the next chapter we will use bigraphs to carry out real-time verification in a home-network environment. Briefly, the key aspect of this approach is to establish a one-to-one correspondence between bigraphical reaction rules and events in the network, e.g. a new machine joins the network. The current configuration of a network is captured by a bigraph which is updated by the application of a reaction rule whenever an event in the network occurs. The OCaml library provided by BigraphER will be used to implement the system.

Chapter 7

Real-time verification for home network management

This chapter reports on an application of bigraphs for real-time verification of domestic wireless network management. The contents are based on the presentation given in [15].

Section 7.1 contains an overview of the Homework network management system [61] that permits user-initiated access control policies and an informal description of our addition to the Homework system in which bigraphical models of the network are generated and analysed in real-time. In Section 7.2 we describe in detail how network topologies are modelled as sorted bigraphs and how network events such as moving in and out of the router's range, and granting and revoking of DHCP leases are encoded as reaction rules. We also define **BiLog** predicates to represent the current status of a machine in the network. In the following section, these reaction rules and predicates are used to generate sequences of bigraphical models in real-time. In Section 7.4, we introduce new sorts to represent policies that forbid and allow access behaviour. They are used to specify the reaction rules that encode policy events such as enforce and drop a policy. Section 7.5 formally defines a general procedure for the generation in real-time of sequences of models when policy events occur in the actual home network. In Section 7.6, we discuss the rôle of **BiLog** predicates in the analysis of network configurations and compliance with policies. In the following section, Section 7.7, we give an overview of the implementation and present the results of two trials of the system running on the Homework router with synthetic and captured data. Some final comments and a summary of the chapter are given in Section 7.8.

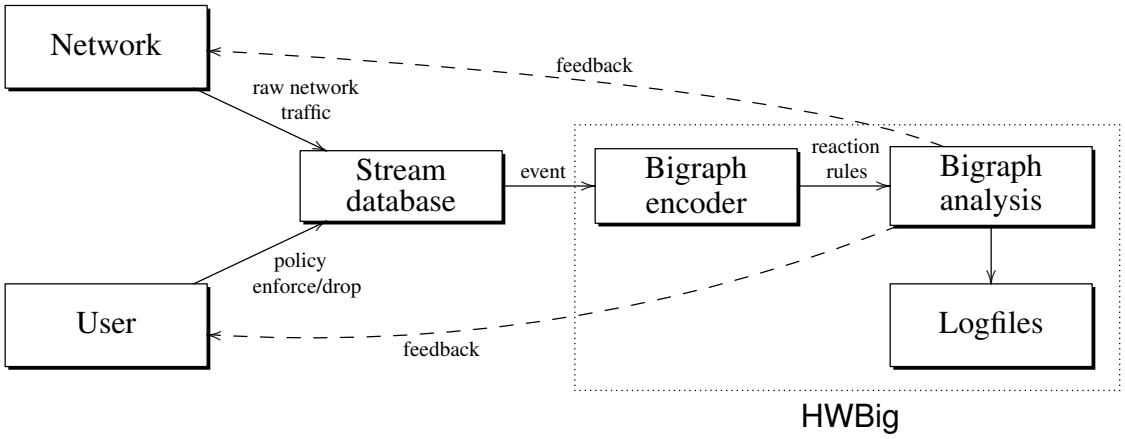


Figure 7.1: Real-time model generation, analysis and feedback. Homework management system is on the left, our extension is on the right.

7.1 Overview

Typical home networks are characterised by a router providing access to the Internet and inter-communication capabilities to all the devices wirelessly connected to it. The Homework network management system [61] has been designed to provide user-oriented support in home wireless network environments. The Homework user interface includes drag and drop, comic-strip style interaction for users; the information plane provides an integrated network monitoring facility consisting of a *stream database* running on the router capable of storing both raw and derived *events*. Raw events record information about individual packets being transmitted over the network. Derived events are constructed by analysing sets of sequential raw events. They provide high-level representation of the current configuration of the network and the status of the traffic between any two machines. Derived events include network behaviours such detecting that a new machine has joined the network and user-initiated behaviours such as enforcing or dropping a *policy*. Policies forbid or allow access control; for example, a policy might block UDP and TCP traffic from a given remote host. From now on, by "event" we mean derived event.

We extended the Homework system with a set of bespoke software components for the generation and analysis in real-time of bigraphical models of the current configuration of the network. We call the set of components the HWBig system. The motivation is to aid users in their understanding of the state of their network, and when and why it is "broken". The architecture of the system is given by the diagram in Figure 7.1. The *Bigraph encoder* component encodes new events as bigraphical reaction rules. The *Bigraph analysis* component has two rôles. First, it generates the bigraphical representation of the current configuration of the network according to sequences of reaction rules received from the bigraph encoder. Second, it analyses the current configuration by checking predicates. Example predicates are "Machine with MAC address 01:23:45:67:89:ab is in the range of the router" and "TCP

Control	Meaning	Sort	Graphical notation
R	Router	r	Circle
S	Wireless signal	s	Oval
M	Wi-Fi enabled machine	m	Circle
Internet	Outside world	j	Box
Properties, ...	Configuration settings	b	Box
W	WLAN	w	Circle
I	Input	i	Small rectangle
O	Output	o	Small arrowhead
MAC, ...	MAC address	p	Rounded box
Hostname, ...	Hostname	p	Rounded box
IP, ...	IP address	p	Rounded box

Table 7.1: Controls and sorts for WLAN.

traffic is blocked for machine with IP address 192.168.0.3". The results are logged and fed back to the system, or to the user, using a graphical notation of bigraphs as explanation, when a verification fails. All this is carried out in real-time.

7.2 Bigraphical model

In this section we outline how a given network topology is represented by a bigraph, and then how network events, such as moving in/out the router's range and granting/revoking leases, are represented by parameterised reaction rules. We give both the graphical and equivalent algebraic forms for the rules. Finally, we define two useful predicates that indicate the state of the network.

7.2.1 Network topology

We use a node to represent each entity present in the network, which can be physical e.g. router, wireless signal, machines, or virtual e.g. configuration properties, the Internet, communication channels. Links connect related entities. For instance, a machine is linked to its signal and to its properties. The sorting discipline ensures that only bigraphs with a meaningful structure are constructed. For example, it enforces that a node representing a machine lies within a node representing its signal. In the graphical notation, different shapes denote nodes of different sorts.

The controls and sorts used to represent the network are listed in Table 7.1. An explanation is as follows. Sort p is assigned to controls indicating MAC addresses, such as control 01:23:45:67:89:ab. We use a special control MAC, to indicate a generic MAC address, con-

Φ_1	all $\widehat{\mathsf{mwiop}}$-nodes are atomic
Φ_2	all children of a θ-root have sort θ, where $\theta \in \{\widehat{\mathsf{sjb}}, \widehat{\mathsf{sj}}, \widehat{\mathsf{sb}}, \mathsf{r}, \mathsf{s}, \mathsf{m}, \mathsf{w}, \mathsf{b}, \mathsf{j}, \widehat{\mathsf{io}}\}$
Φ_3	all children of an s-node have sort $\widehat{\mathsf{rm}}$
Φ_4	an r-node has a w-child
Φ_5	all p-nodes are children of a b-node
Φ_6	all $\widehat{\mathsf{io}}$-nodes are children of a $\widehat{\mathsf{bj}}$-node
Φ_7	all s-nodes are always linked to a $\widehat{\mathsf{rm}}$-child
Φ_8	a b-node is always linked to a m-node
Φ_9	a w-node may only be linked to m-nodes
Φ_{10}	an i-node may only be linked to a o-node
Φ_{11}	an o-node may only be linked to a i-node

Table 7.2: Conditions of formation rule Φ.

trols Hostname and IP, to indicate a generic host-name and IP address, respectively. The set of sorts is written as

$$\Theta = \{\mathsf{r}, \mathsf{s}, \mathsf{m}, \mathsf{j}, \mathsf{b}, \mathsf{w}, \mathsf{i}, \mathsf{o}, \mathsf{p}, \} \ .$$

The signature is given by $\mathcal{K} = \bigcup_{s \in \Theta} s$. Formation rule Φ is given in Table 7.2. Conditions Φ_3–Φ_7 state that each machine is placed inside a signal and is connected to it. Recall that similar conditions were also defined in the formation rule for sorting $\Sigma_{802.11}$ used in Chapter 6. Conditions Φ_4, Φ_7 specify that the router lies within its signal and is linked to it. Conditions Φ_5, Φ_8 ensure that machines are also connected to a property box that contains various configuration details. Condition Φ_9 forces machines that are part of the WLAN to share a link with the w-node inside the router. Condition Φ_6 states that property boxes (and the Internet) are linked to each other via a pair of communication channels. These are represented by an i-node linked to an o-node as specified by conditions Φ_{10}, Φ_{11}. Finally, we write the sorting used in the model as $\Sigma_{\mathsf{HW}} = (\mathcal{K}, \Theta, \Phi)$.

The initial configuration of a WLAN is given by bigraph $S_0 : \epsilon \to \widehat{\mathsf{sjb}}$ in Figure 7.2. It models the scenario in which only the router and the external world are present. The algebraic form is

$$S_0 = /x \, /y \, (\mathsf{S}_x.\mathsf{R}_x.\mathsf{W}_y.1) \mid \mathsf{Internet}.1$$

7.2.2 Network events

Now we turn our attention to the reaction rules that represent the network events, which include moving in and out of the router's range, and the granting and revoking of DHCP leases. We discuss each one of the four events in turn, and we describe them by means of a graphical representation. A summary of all the reaction rules including their algebraic forms

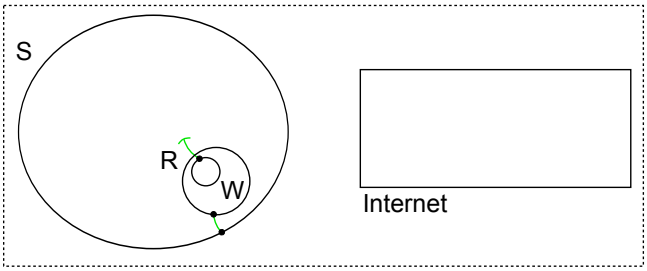


Figure 7.2: Initial configuration $S_0 : \epsilon \to \widehat{\mathsf{sjb}}$.

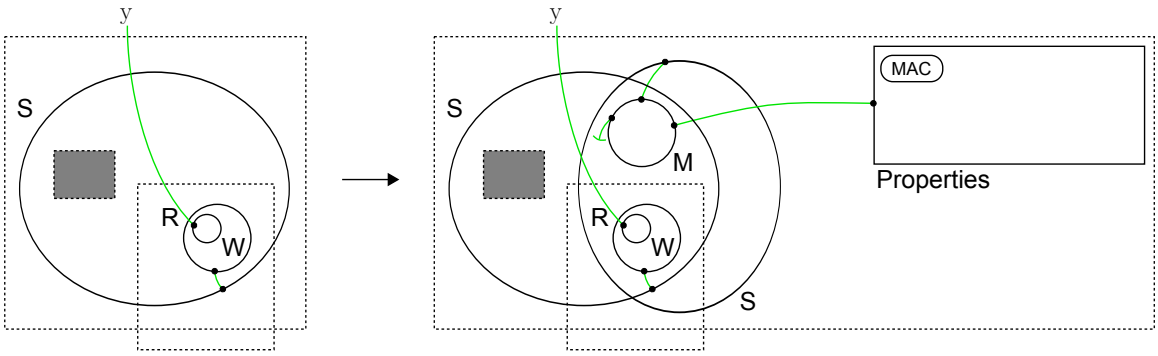


Figure 7.3: Reaction rule $\mathsf{R_A(MAC)}$: a new machine appears in the router's signal range.

is given in Table 7.3. We note that all the reaction rules respect sorting Σ_{HW}.

The first event occurs when machine MAC appears in the signal range of the router. It is encoded by reaction rule $\mathsf{R_A(MAC)} : \widehat{\mathsf{mr}} \to \langle \widehat{\mathsf{sbr}}, \{y\} \rangle$ given in Figure 7.3. On the left-hand side, the router is in the range of its signal and possibly other signals. This is expressed by the region surrounding the r-node. On the right-hand side, a new machine is in the range of the router's signal. The router senses the new machine's signal and possibly other signals. This is expressed by nodes R and M being in the intersection of the two s-nodes and the region surrounding R. A property box (i.e. a b-node) is also linked to M. Note that the only configuration setting specified at this stage is the MAC address of the new machine M. This is witnessed by the p-node placed inside Properties. Observe that this reaction rule forces all m-nodes to be shared by only two s-nodes. This means our model does not capture any interference between the signals of the machines in the system: our model is based solely on information provided by the router. It other words, we can only model what the router senses.

The second event occurs when machine MAC is no longer in the router's signal range. This happens because either a machine switches off its network interface or it moves into a location not reachable by router's signal. It is encoded by the reaction rule $\mathsf{R_R(MAC)}$ given in Figure 7.4. On the left-hand side, a m-node is linked to a b-node and placed within an s-node. These correspond to a machine, its configuration properties and its signal range, respectively. The extra region enclosing M and the site are necessary to allow the machine modelled in the left-hand side to be in the range of the router and possibly other machines. On the right-

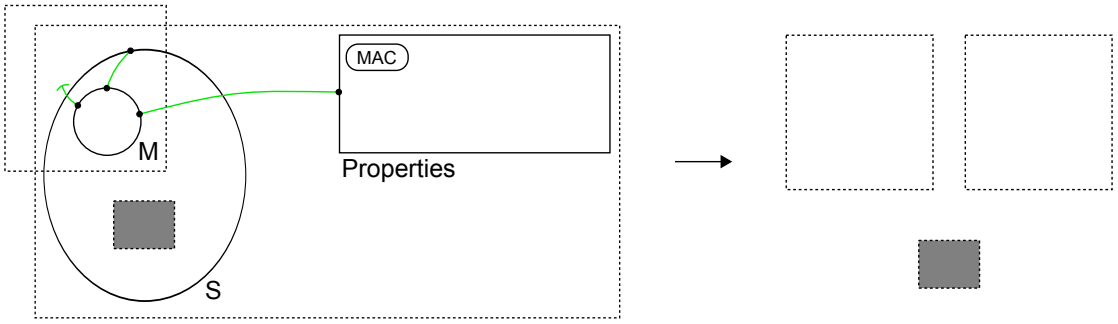


Figure 7.4: Reaction rule $\mathsf{R_R(MAC)}$: a machine is no longer in the router's signal range.

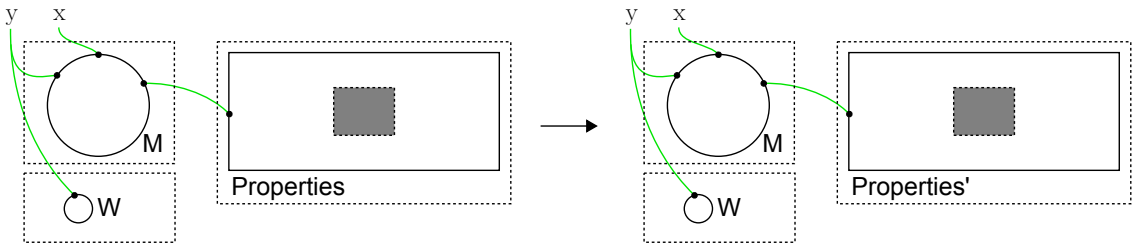


Figure 7.5: Reaction rule $\mathsf{R_{J1}}$. A new machine joins the WLAN (1): all stations already in the WLAN are tagged.

hand side, all the nodes have disappeared. This models the absence of the machine from the system. We note that without sharing, the orphan site does not exist. But in our case, we need it because on the left-hand side, there could be another entity matched by the site (e.g. the router), which would persist even after we remove the parent node S. The interface of the reaction rule is given by: $\mathsf{R_R(MAC)} : \widehat{\mathsf{mr}} \to \langle \widehat{\mathsf{sbm}}, \emptyset \rangle$.

The third event occurs when a machine joins the WLAN and a DHCP lease is granted. We use three reaction rules to describe how the system changes. This requires distinguishing between the new machine and those already in the network. We do so by tagging the latter. The first rule, $\mathsf{R_{J1}}$, implements the tagging, the second rule, $\mathsf{R_{J2}(MAC, IP, Hostname)}$, establishes the network aspects of the untagged machine (i.e. IP address etc.), and the third rule, $\mathsf{R_{J3}(MAC)}$, establishes the communication channels between the new machine and the tagged machines and then it revokes the tags. Reaction $\mathsf{R_{J1}} : \widehat{\mathsf{pio}} \to \langle \mathsf{mwb}, \{x, y\} \rangle$, in Figure 7.5, is used to tag all the machines in the system that are already part of the WLAN. On the left-hand side we have an m-node linked to the w-node. The actual tagging is implemented in the right-hand side, where a node of control Properties′ takes the place of the corresponding node of control Properties in the left-hand side. Reaction rule $\mathsf{R_{J2}(MAC, IP, Hostname)}$ models the DHCP server granting a lease to the machine, as depicted in Figure 7.6. On the left-hand side, a machine is not part of the network and the only configuration property already specified is the MAC address. This is shown by the absence of a link between the m-node and the w node and the absence of a site inside the node of control Properties. On the right-hand side, the machines joins the WLAN, IP address and

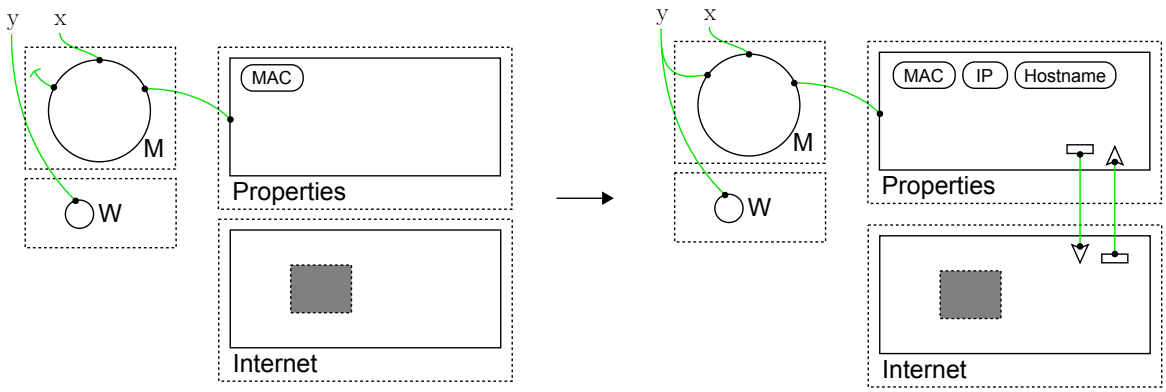


Figure 7.6: Reaction rule $\mathsf{R_{J2}(MAC, IP, Hostname)}$. A new machine joins the WLAN (2): Hostname and IP address are set and communication channels with the Internet are established.

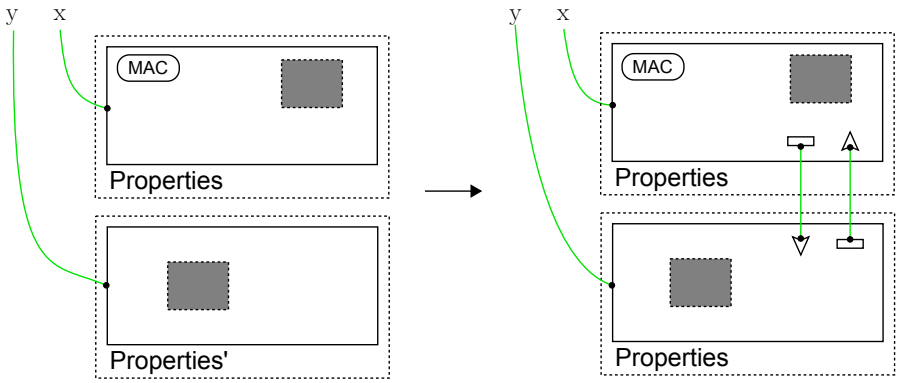


Figure 7.7: Reaction rule $\mathsf{R_{J3}(MAC)}$. A new machine joins the WLAN (3): Communication channels are created between the station and all the machines already present in the WLAN.

hostname are set, and two communication channels with the external world are established. The interface is given by:

$$\mathsf{R_{J2}(MAC, IP, Hostname)} : \widehat{\mathsf{io}} \rightarrow \langle \mathsf{mwbj}, \{x, y\} \rangle \ .$$

Note that the channels are uni-directional and so we have to define a pair (one for incoming traffic, one for outgoing traffic). In reaction rule $\mathsf{R_{J3}(MAC)}$ a pair of communication channels is established between the new machine and the machines already part of the WLAN, see Figure 7.7. On the left-hand side, a node of control Properties and a node of control Properties′ specify the configurations of the new machine and a machine already in the WLAN, respectively. On the right-hand side, a pair of communication channels is established and a node of control Properties replaces the corresponding node of control Properties′ in the left-hand side. The interface of the reaction is $\mathsf{R_{J3}(MAC)} : \widehat{\mathsf{pio}}\,\widehat{\mathsf{pio}} \rightarrow \langle \mathsf{bb}, \{x, y\} \rangle$. We note that this event is encoded by a sequence of application of the three reaction rules described above. Initially, all machines that have already joined the WLAN are tagged, using reaction $\mathsf{R_{J1}}$. This means the reaction is applied n times, where n is the number of machines in the network. The resulting interleaving of applications is confluent, therefore, only one sequence

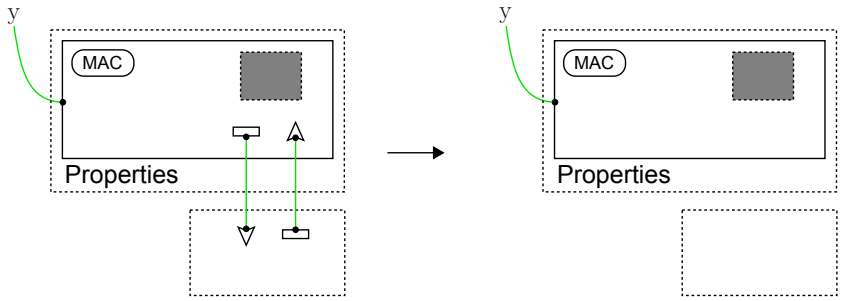


Figure 7.8: Reaction rule $\mathsf{R_{L1}(MAC)}$. A machine leaves the WLAN (1): Pairs of communication channels are removed.

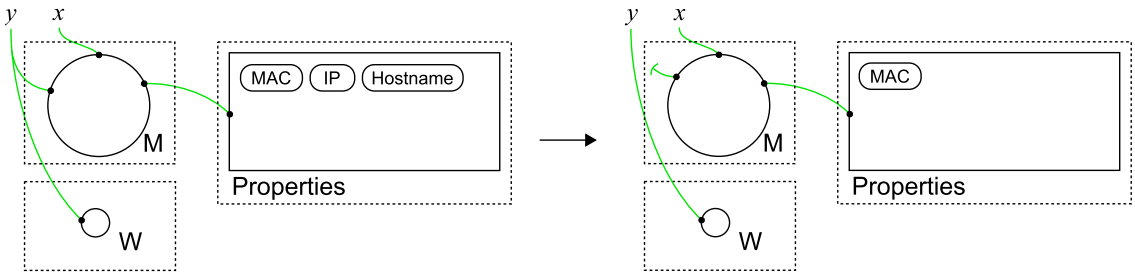


Figure 7.9: Reaction rule $\mathsf{R_{L2}(MAC, IP, Hostname)}$. A machine leaves the WLAN (2): DHCP leases are revoked.

need be considered. Then, reaction $\mathsf{R_{J2}(MAC, IP, Hostname)}$ is applied once. Finally, reaction $\mathsf{R_{J3}(MAC)}$ is applied n times. Again, due to confluence, only one sequence need be considered. We will describe in more detail how sequences of applications are used to generate models of the network in real-time in Section 7.3.

The last event takes place when a machine leaves the WLAN and the lease is revoked. This is represented by two rules. Note, this does not automatically imply that the machine is also leaving the router's signal range. Reaction rule $\mathsf{R_{L1}(MAC)} : \widehat{\mathsf{pio}} \rightarrow \langle \widehat{\mathsf{bio}}, \{x, y\} \rangle$ is given in Figure 7.8. The left-hand side specifies a property box for the machine and a pair of channels. The site also allows the reaction to be applied when other nodes are inside the node of control Properties. On the right-hand side the two communication channels are removed. Reaction rule $\mathsf{R_{L2}(MAC, IP, Hostname)}$ revokes the machine's DHCP lease. This is encoded by the removal of nodes of control Hostname and IP and the breaking of link between M and W, as depicted in Figure 7.9. The interface of the reaction rule is given by

$$\mathsf{R_{L2}(MAC, IP, Hostname)} : \epsilon \rightarrow \langle \mathsf{mwb}, \{x, y\} \rangle \ .$$

Note that reaction $\mathsf{R_{L1}(MAC)}$ is applied first, until no other channels can be removed. Again, the order in which the channels are removed is not important and only one sequence of reactions need be considered. Second, reaction rule $\mathsf{R_{L2}(MAC, IP, Hostname)}$ is applied once.

The algebraic forms for all the seven reaction rules of the model are given in Table 7.3.

$$\mathsf{R_A}(\mathsf{MAC}) = R_1 \longrightarrow\!\blacktriangleright R_1'$$
$$R_1 = /x\,\big(\text{share}\ (\mathsf{R}_x.\mathsf{W}_y.1 \parallel \mathsf{id})\ \text{by}\ [\{1,2\},\{0\}]\ \text{in}\ (\mathsf{S}_x.(\mathsf{id} \mid \mathsf{id}) \parallel \mathsf{id}_{1,yx})\big)$$
$$R_1' = /x\,/z\,/p\,\big(\text{share}\ (\mathsf{R}_x.\mathsf{W}_y.1 \parallel /y\,\mathsf{M}_{yzp}.1 \parallel \mathsf{id})\ \text{by}\ [\{1,2,,3\},\{1,2\},\{0\}]$$
$$\text{in}\ ((\mathsf{S}_x.(\mathsf{id} \mid \mathsf{id}) \mid \mathsf{S}_z \mid \mathsf{Properties}_p.\mathsf{MAC}.1) \parallel \mathsf{id}_{1,yxzp})\big)$$

$$\mathsf{R_R}(\mathsf{MAC}) = R_2 \longrightarrow\!\blacktriangleright 1 \parallel 1 \parallel 0$$
$$R_2 = /x\,/p\,\big(\text{share}\ (/y\,\mathsf{M}_{xyp}.1 \parallel \mathsf{id})\ \text{by}\ [\{1,2\},\{0\}]$$
$$\text{in}\ ((\mathsf{S}_x.(\mathsf{id} \mid \mathsf{id}) \mid \mathsf{Properties}_p.\mathsf{MAC}.1) \parallel \mathsf{id}_{1,xp})\big)$$

$$\mathsf{R_{J1}} = /p\,(\mathsf{M}_{yxp}.1 \parallel \mathsf{W}_y.1 \parallel \mathsf{Properties}_p) \longrightarrow\!\blacktriangleright /p\,(\mathsf{M}_{yxp}.1 \parallel \mathsf{W}_y.1 \parallel \mathsf{Properties'}_p)$$

$$\mathsf{R_{J2}}(\mathsf{MAC},\mathsf{IP},\mathsf{Hostname}) = R_{3b} \longrightarrow\!\blacktriangleright R_{3b}'$$
$$R_{3b} = /p\,(/y\,\mathsf{M}_{xyp}.1 \parallel \mathsf{W}_y.1 \parallel \mathsf{Properties}_p.\mathsf{MAC}.1 \parallel \mathsf{Internet})$$
$$R_{3b}' = /p\,/l\,/h\,\big(\mathsf{M}_{xyp}.1 \parallel \mathsf{W}_y.1$$
$$\parallel \mathsf{Properties}_p.(\mathsf{MAC}.1 \mid \mathsf{Hostname}.1 \mid \mathsf{IP}.1 \mid \mathsf{I}_h.1 \mid \mathsf{O}_l.1)$$
$$\parallel \mathsf{Internet}.(\mathsf{id} \mid \mathsf{I}_l.1 \mid \mathsf{O}_h.1)\big)$$

$$\mathsf{R_{J3}}(\mathsf{MAC}) = R_{3c} \longrightarrow\!\blacktriangleright R_{3c}'$$
$$R_{3c} = \mathsf{Properties}_x.(\mathsf{id} \mid \mathsf{MAC}.1) \parallel \mathsf{Properties'}_y$$
$$R_{3c}' = /l\,/h\,\big(\mathsf{Properties}_x.(\mathsf{id} \mid \mathsf{MAC}.1 \mid \mathsf{I}_l.1 \mid \mathsf{O}_h.1)$$
$$\parallel \mathsf{Properties}_y.(\mathsf{id} \mid \mathsf{I}_h.1 \mid \mathsf{O}_l.1)\big)$$

$$\mathsf{R_{L1}}(\mathsf{MAC}) = R_{4a} \longrightarrow\!\blacktriangleright R_{4a}'$$
$$R_{4a} = /l\,/h\,(\mathsf{Properties}_y.(\mathsf{id} \mid \mathsf{MAC}.1 \mid \mathsf{I}_l.1 \mid \mathsf{O}_h.1) \parallel (\mathsf{I}_h.1 \mid \mathsf{O}_l.1))$$
$$R_{4a}' = \mathsf{Properties}_y.(\mathsf{id} \mid \mathsf{MAC}.1) \parallel 1$$

$$\mathsf{R_{L2}}(\mathsf{MAC},\mathsf{IP},\mathsf{Hostname}) = R_{4b} \longrightarrow\!\blacktriangleright R_{4b}'$$
$$R_{4b} = /p\,(\mathsf{M}_{yxp}.1 \parallel \mathsf{W}_y.1 \parallel \mathsf{Properties}_p.(\mathsf{MAC}.1 \mid \mathsf{Hostname}.1 \mid \mathsf{IP}.1))$$
$$R_{4b}' = /p\,(/y\,\mathsf{M}_{yxp}.1 \parallel \mathsf{W}_y.1 \parallel \mathsf{Properties}_p.\mathsf{MAC}.1)$$

Table 7.3: Reaction rules for network events.

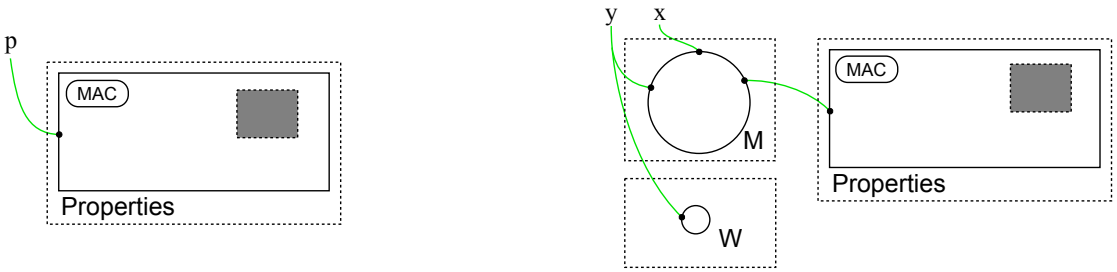


Figure 7.10: Bigraphs $B_{\varphi_{\mathsf{MAC}}}$ (left) and $B_{\psi_{\mathsf{MAC}}}$ (right).

7.2.3 Status predicates

In addition to the reaction rules, we define two predicates, parameterised by a machine MAC address, to represent the current status of a machine in the system:

- φ_{MAC} is true iff the machine MAC is present in the system,
- ψ_{MAC} is true iff the machine MAC is part of the WLAN.

The predicates are encoded by bigraphs $B_{\varphi_{\mathsf{MAC}}}$ and $B_{\psi_{\mathsf{MAC}}}$, which are depicted in Figure 7.10. The corresponding algebraic forms are

$$B_{\varphi_{\mathsf{MAC}}} = \mathsf{Properties}_p.(\mathsf{id} \mid \mathsf{MAC}.1)$$
$$B_{\psi_{\mathsf{MAC}}} = /p\,(\mathsf{M}_{yxp}.1 \parallel \mathsf{W}_y.1 \parallel \mathsf{Properties}_p.(\mathsf{id} \mid \mathsf{MAC}.1))$$

We define $S \models \varphi_{\mathsf{MAC}}$ iff $B_{\varphi_{\mathsf{MAC}}}$ is a match in bigraph S and $S \models \psi_{\mathsf{MAC}}$ iff $B_{\psi_{\mathsf{MAC}}}$ is a match in bigraph S. All the other predicates defined in the following sections are encoded similarly in terms of bigraphical matching.

7.3 Generation of models in real-time

We now describe how the reaction rules and predicates defined in the previous section are used to generate sequences of models of the current network configuration in real-time. For a given current model S_t, we generate a successor model S_{t+1}, such that $S_t \longrightarrow\!\!\triangleright^* S_{t+1}$. The new model S_{t+1} represents the network after an event takes place. Strictly, any model S such that $S_t \longrightarrow\!\!\triangleright^* S$ is a successor model, however, often we store only the model obtained after several rewriting steps, for example when tagging and untagging is required. The model of the current network configuration is stored in the bigraph analysis component of the HWBig system. Note, we generate and store the algebraic form, whereas we use the graphical form for feedback. An example illustrates the generation process.

Event	Encoding	Short form
Add machine	$\longrightarrow\!\triangleright_{\mathsf{A}}$	—
Machine joins	$\longrightarrow\!\triangleright^{*}_{\mathsf{J1}} \longrightarrow\!\triangleright_{\mathsf{J2}} \longrightarrow\!\triangleright^{*}_{\mathsf{J3}}$	$\longrightarrow\!\triangleright^{*}_{\mathsf{J}}$
Machine leaves	$\longrightarrow\!\triangleright^{*}_{\mathsf{L1}} \longrightarrow\!\triangleright_{\mathsf{L2}}$	$\longrightarrow\!\triangleright^{*}_{\mathsf{L}}$
Remove machine	$\longrightarrow\!\triangleright_{\mathsf{R}}$	—

Table 7.4: Encodings for network events.

Example 7.3.1. Assume at time t the stream database generates a network event specifying that machine MAC is present in the system and a DHCP lease has been granted. The current model is denoted by S_t, and the generated event has been sent to the bigraph encoder component. The sequence of reaction rules to be applied to S_t is determined by whether or not machine MAC is already present in the system and if it has joined the WLAN. Therefore, the bigraph analysis component is queried to check if $S_t \models \varphi_{\mathsf{MAC}}$ and $S_t \models \psi_{\mathsf{MAC}}$. The results are sent back to the bigraph encoder component. We then have three cases of model generation, summarised as follows:

- If $S_t \models \varphi_{\mathsf{MAC}}$ and $S_t \models \psi_{\mathsf{MAC}}$, then the system remains unchanged and no reaction rule is applied.
- If $S_t \models \varphi_{\mathsf{MAC}}$ but $S_t \not\models \psi_{\mathsf{MAC}}$, then machine MAC has to join the WLAN. The generated sequence of reactions is: R_{J1}, $\mathsf{R}_{\mathsf{J1}}(\mathsf{MAC}, \mathsf{IP}, \mathsf{Hostname})$, $\mathsf{R}_{\mathsf{J1}}(\mathsf{MAC})$, which is sent to the bigraph analysis component to update the model:

  $$S_t \longrightarrow\!\triangleright^{*}_{\mathsf{J1}} \longrightarrow\!\triangleright_{\mathsf{J2}} \longrightarrow\!\triangleright^{*}_{\mathsf{J3}} S_{t+1} \ .$$

  For brevity, we denote this sequence of applications as $S_t \longrightarrow\!\triangleright^{*}_{\mathsf{J}} S_{t+1}$. Observe that IP and Hostname are part of the event obtained from the stream database.
- If $S_t \not\models \varphi_{\mathsf{MAC}}$, then machine MAC has to appear in the range of the router and then to join the WLAN. The generated sequence is: $\mathsf{R}_{\mathsf{A}}(\mathsf{MAC})$, R_{J1}, $\mathsf{R}_{\mathsf{J1}}(\mathsf{MAC}, \mathsf{IP}, \mathsf{Hostname})$, $\mathsf{R}_{\mathsf{J1}}(\mathsf{MAC})$, which is sent to the bigraph analysis component to update the model:

  $$S_t \longrightarrow\!\triangleright_{\mathsf{A}} \longrightarrow\!\triangleright^{*}_{\mathsf{J}} S_{t+1} \ .$$ ■

Encodings for the four network events are summarised in Table 7.4. Predicates φ_{MAC} and ψ_{MAC} are used as in Example 7.3.1 to select the appropriate sequence of reaction rules that should be applied.

Control	Meaning	Sort	Graphical notation
Port, ...	Port number	p	Bold rounded box
WWW, ...	External host	p	Bold rounded box
P, ...	Protocol	p	Bold rounded box
BLOCKED	All traffic forbidden	p	Bold rounded box

Table 7.5: New controls and sorts for modelling policies.

7.4 Bigraphical models of policies

Now we turn our attention to the representation of access control policies by reaction rules. Access control policies constrain behaviours, for example they can constrain traffic between machines, or types of traffic. New entities are therefore required. For example, new controls are needed to express the ban of a given port or a communication protocol. The additional controls are listed in Table 7.5, which we call *constraints*. Formation rule Φ defined in Table 7.2 is also modified by allowing $\widehat{\mathsf{io}}$-nodes to be linked to p-nodes.

Policies are categorised as *forbid* policies or *allow* policies. The latter are relatively simple to represent because matching can detect the existence of a constraint that requires to be removed. However, the representation of forbid policies is a little more complex.

The key idea of representing a forbid policy is to link chains of p-nodes to communication channels. A chain of constraints represents a conjunction of constraints, and several chains linked to a channel represent a disjunction of constraints. Some policies can be represented by a single reaction rule, whereas others require several when a form of tagging is needed in the representation. We illustrate the possible forms of representation with three example forbid policies. A summary of the algebraic forms of the reaction rules for these policies is given in Table 7.6.

Example 7.4.1 (policy P1)**.** Consider a policy, denoted by P1, that forbids the machine named Laptop from receiving incoming traffic from remote host WWW, defined by reaction rule R_{P1} in Figure 7.11. The left-hand side can match only Laptop's Properties box, its out-going channel to the external world and Internet box. In the right-hand side, constraint WWW is attached to the channel's link. Note that constraints like WWW are always placed within the sender's $\widehat{\mathsf{bj}}$-box[1].

The inverse reaction rule[2] models the policy being dropped. While this policy (P1) is represented by a single reaction rule, we note that we must apply it carefully, to avoid multiple or inconsistent applications. An example illustrates the problem. Consider a network modelled by bigraph S in which machine Laptop is already forbidden from receiving traffic from

[1]This is an arbitrary choice that simplifies the specification of policy predicates. The symmetric encoding where the constraints are always placed within the receiver's $\widehat{\mathsf{bj}}$-box is also possible.

[2]The reaction rule in which the left-hand side and the right hand side are swapped.

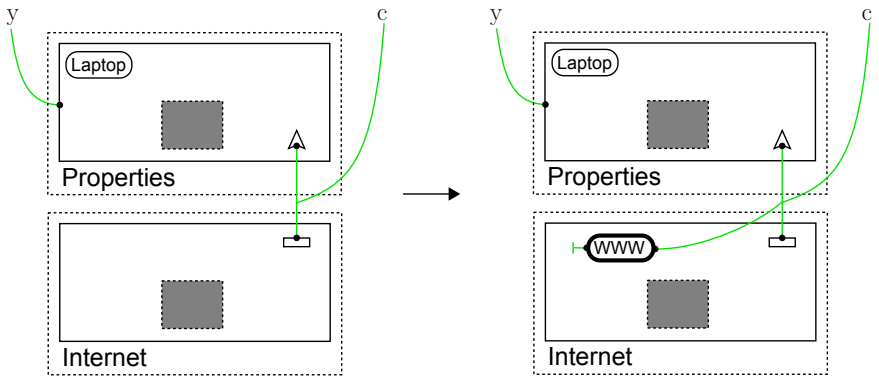


Figure 7.11: Reaction rule R_{P1}. All incoming traffic from WWW to Laptop is blocked.

WWW, i.e. a WWW-node is already linked to the channel from Laptop to Internet. Reaction rule R_{P1} could be applied to this bigraph, and as a result of the rule application, we would obtain a bigraph in which *two* copies of the same constraint are linked to the channel. To avoid this, we must check, before any rule applications for the policy, whether traffic from WWW to Laptop is forbidden. Specifically, the bigraph analysis component is queried to check whether $S \models \varphi_{\mathsf{P1}}$, where predicate φ_{P1} corresponds to the right-hand side of reaction rule R_{P1}. The reaction rule for the policy is applied only if the predicate is false. Since the predicate holds for model S, reaction rule $\mathsf{R}_{\mathsf{P1}}(\mathsf{Laptop}, \mathsf{WWW})$ would not be applied in this case. ■

Example 7.4.2 (policy P2)**.** A more complex model arises when TCP connections with any host using destination ports 8080 or 6881 and source port 6882 are forbidden. We call this policy P2. First, rule $\mathsf{R}_{\mathsf{P2_T}}$ is applied once to all the channels in the system. This results in a bigraph in which all $\widehat{\mathsf{io}}$-nodes are tagged, which is necessary in order to ensure that rule $\mathsf{R}_{\mathsf{P2_E}}$ is applied only once. The algebraic form is

$$\mathsf{R}_{\mathsf{P2_T}} = \mathsf{O}_c.1 \parallel \mathsf{I}_c.1 \quad \longrightarrow \quad \mathsf{O'}_c.1 \parallel \mathsf{I'}_c.1$$

Second, rule $\mathsf{R}_{\mathsf{P2_E}}$ is applied to all the tagged channels, this is depicted in Figure 7.12. The left-hand side matches any tagged channel. On the right-hand side, the constraints are placed by linking them to the channels and $\widehat{\mathsf{io}}$-nodes are untagged. Constraints on source ports are placed inside the box containing node O (i.e. sender's Properties box), while constraints on destination ports are inside the box containing node I (i.e. receiver's Properties box). The order in which channels are tagged and untagged is irrelevant because reaction rule $\mathsf{R}_{\mathsf{P2_E}}$ is applied only when all the channels are tagged. Thus, only one interleaving need be considered. As in the previous example, the bigraph analysis component is queried prior to the application of the application of the reaction rules modelling this policy, in order to avoid double entries and inconsistent constraints. ■

Example 7.4.3 (policy P3)**.** Finally, consider a policy that forbids traffic from the machine

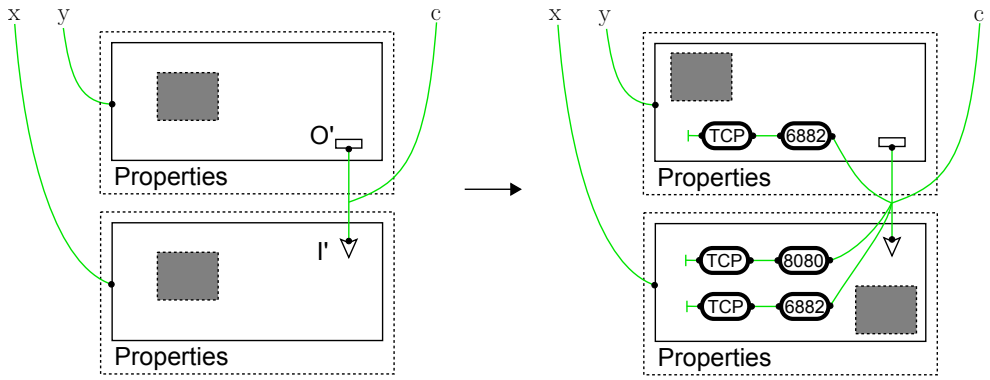


Figure 7.12: Policy reaction rule R_{P2_E}. TCP connections with any host using destination ports 8080 and 6881 and source port 6882 are blocked.

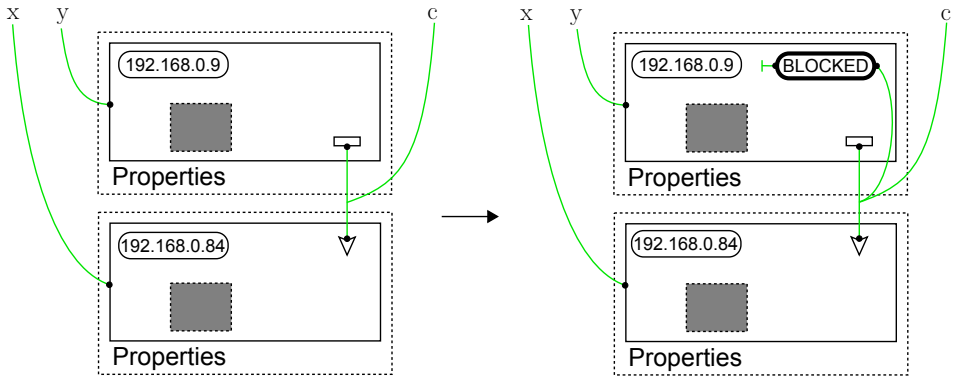


Figure 7.13: Policy reaction rule R_{P3}. Traffic form 192.168.0.9to 192.168.0.84 is forbidden.

whose IP address is 192.168.0.9 to the machine whose IP address is 192.168.0.84. The policy is encoded as reaction rule R_{P3} given in Figure 7.13. The left-hand side matches the channel blocked by the policy. On the right-hand side, special constraint BLOCKED is linked to the channel. ■

7.5 Generating models of policy events in real-time

In Section 7.3 we described how the current model S_t of a network is updated in real-time upon an occurrence of a network event. We now specify a similar procedure to update the model when policy events take place. Reaction rules encoding the policy events generated by the stream database are used by the bigraph analysis component to generate sequences in the form $S_t \longrightarrow\!\triangleright^* S_{t+1}$. There are three possible policy events: *enforce*, *drop* and *check* policy compliance for both forbid and allow policies. Forbid policy events are more difficult to encode than allow policy events, and so we consider these first.

$\mathsf{R_{P1}} = P_1 \longrightarrow\!\blacktriangleright P_1'$ $P_1 = \mathsf{Properties}_y.(\mathsf{id} \mid \mathsf{Laptop}.1 \mid \mathsf{I}_c.1) \parallel \mathsf{Internet}.(\mathsf{id} \mid \mathsf{O}_c.1)$ $P_1' = \mathsf{Properties}_y.(\mathsf{id} \mid \mathsf{Laptop}.1 \mid \mathsf{I}_c.1) \parallel \mathsf{Internet}.(\mathsf{id} \mid /h\, \mathsf{WWW}_{ch}.1 \mid \mathsf{O}_c.1)$
$\mathsf{R_{P2a}} = \mathsf{O}_c.1 \parallel \mathsf{I}_c.1 \quad \longrightarrow\!\blacktriangleright \quad \mathsf{O'}_c.1 \parallel \mathsf{I'}_c.1$
$\mathsf{R_{P2b}} = P_2 \longrightarrow\!\blacktriangleright P_2'$ $P_2 = \mathsf{Properties}_y.(\mathsf{id} \mid \mathsf{O'}_c.1) \parallel \mathsf{Properties}_x.(\mathsf{id} \mid \mathsf{I'}_c.1)$ $P_2' = \mathsf{Properties}_y.(\mathsf{id} \mid \mathsf{O'}_c.1 \mid /q\,(6882_{cq}.1 \mid /r\,\mathsf{TCP}_{qr}.1))$ $\parallel \mathsf{Properties}_x.\big(\mathsf{id} \mid \mathsf{I'}_c.1 \mid /q_1\,(8080_{cq_1}.1 \mid /r\,\mathsf{TCP}_{q_1r}.1)$ $\mid /q_2\,(6881_{cq_2}.1 \mid /r\,\mathsf{TCP}_{q_2r}.1)\big)$
$\mathsf{R_{P3}} = P_3 \longrightarrow\!\blacktriangleright P_3'$ $P_3 = \mathsf{Properties}_y.(\mathsf{id} \mid 192.168.0.9.1 \mid \mathsf{O}_c.1)$ $\parallel \mathsf{Properties}_x.(\mathsf{id} \mid 192.168.0.84.1 \mid \mathsf{I}_c.1)$ $P_3' = \mathsf{Properties}_y.(\mathsf{id} \mid 192.168.0.9.1 \mid /e\,\mathsf{BLOCKED}_{ce}.1 \mid \mathsf{O}_c.1)$ $\parallel \mathsf{Properties}_x.(\mathsf{id} \mid 192.168.0.84.1 \mid \mathsf{I}_c.1)$

Table 7.6: Reaction rules for example policies.

7.5.1 Encoding forbid policy events

A forbid policy is represented by linking constraints (p-nodes) to channels. Again, we employ tagging to indicate when rules may or may not be applicable. In the case of enforce, we employ tagging to ensure that constraints are only added once. In the case of checking policy compliance, the use of tagging is more subtle. The problem we need to overcome is how to check for the non-existence of a pattern in a bigraph, namely, we require to check that we cannot match the left hand-side of a policy enforcement reaction rule. So, we tag channels that comply with the policy. If all the channels are tagged, then a match is not possible, and we can conclude the entire model complies with the policy. Thus, for a policy P, we denote by φ_{P} the predicate for compliance with policy P and $B_{\varphi_{\mathsf{P}}}$ the corresponding bigraph for matching.

Recall that when an event occurs, it is captured by the stream database and encoded into a sequence of reaction rules by the bigraph encoding component of HWBig. The sequence is then applied to S (i.e. the bigraph modelling the current configuration of the network) by the bigraph analysis component. An explanation of the sequence of reaction rules that encode each of the three events for a forbid policy is given below. The corresponding reaction rules are summarised in Table 7.7. The rules are grouped according to the three functions: **tag**, **enforce**, **untag**.

Event	Encoding	Short form
Enforce policy P	$\overset{\text{tag}}{\overbrace{\longrightarrow\!\!\triangleright^{*}}}\ \overset{\text{enforce}}{\overbrace{\longrightarrow\!\!\triangleright^{*}}}\ \overset{\text{untag}}{\overbrace{\longrightarrow\!\!\triangleright^{*}}}$	$\longrightarrow\!\!\triangleright^{*}_{\mathsf{P}}$
Drop policy P	$\overset{\text{remove}}{\overbrace{\longrightarrow\!\!\triangleright^{*}}}$	$\longrightarrow\!\!\triangleright^{*}_{\not{\mathsf{P}}}$
Check policy P	$S \overset{\text{tag}}{\overbrace{\longrightarrow\!\!\triangleright^{*}}} T$ $B_{\varphi_{\mathsf{P}}}$ is a match in $T \Longrightarrow S \not\models \varphi_{\mathsf{P}}$ $B_{\varphi_{\mathsf{P}}}$ is not a match in $T \Longrightarrow S \models \varphi_{\mathsf{P}}$ $T \overset{\text{untag}}{\overbrace{\longrightarrow\!\!\triangleright^{*}}} S$	—

Table 7.7: Encodings for policy events (forbid).

Enforce a forbid policy

1. **(tag)** a sequence of rules that tag channels in the model that comply with the policy,
2. **(enforce)** a sequence of rules that link the constraint specified by the policy to the untagged channels, and then tag these channels so they are not considered again,
3. **(untag)** a sequence of rules that removes the tags applied in steps 1 and 2.

Drop a forbid policy

1. **(remove)** a sequence of rules that removes the policy constraints from channels.

Check a forbid policy

1. **(tag)** a sequence of rules that tag channels in the model that comply with the policy,
2. **(check)** check whether the predicate φ_P holds for the tagged model (from step 1), by attempting to match $B_{\varphi_{\mathsf{P}}}$. If a match is possible, then conclude $S \not\models \varphi_{\mathsf{P}}$, otherwise conclude $S \models \varphi_{\mathsf{P}}$,
3. **(untag)** a sequence of rules that removes the tags applied in step 1.

7.5.2 Encoding allow policy events

Allow policies are much easier to encode because constraints are removed, instead of being added to the model. Thus, we can take advantage of the fact that bigraph matching is a test for

Event	Encoding	Short form
Enforce policy P	$\overset{\text{enforce}}{\overbrace{\longrightarrow\!\!\triangleright^{*}}}$	$\longrightarrow\!\!\triangleright^{*}_{\mathsf{P}}$
Check policy P	$B_{\varphi_{\mathsf{P}}}$ is a match in $S \Longrightarrow S \not\models \varphi_{\mathsf{P}}$ $B_{\varphi_{\mathsf{P}}}$ is not a match in $S \Longrightarrow S \models \varphi_{\mathsf{P}}$	—

Table 7.8: Encodings for policy events (allow).

the existence of a pattern. An overview of allow policy enforce/check is the following, which is also summarised in Table 7.8. We note that it is not possible to drop an allow policy. If the user wishes to stop allowing some behaviour, that behaviour has to be specified explicitly as a forbid policy. Again, assume the current configuration of the network is modelled by bigraph S.

Enforce an allow policy

1. **(enforce)** a sequence of rules that removes the policy constraints from channels. There is no need for a tagging step.

Check an allow policy

1. **(check)** attempt to match $B_{\varphi_{\mathsf{P}}}$. If a match is possible, then conclude $S \not\models \varphi_{\mathsf{P}}$, otherwise conclude $S \models \varphi_{\mathsf{P}}$.

7.5.3 Interplay between network and policy events

When a network event occurs, the bigraph analysis component applies a sequence of reaction rules as described in Section 7.3. However, this may lead to a system in which some policies are not enforced. For example, assume a current model, S_t, of a WLAN where every machine is forbidden to receive data from remote host WWW. Further, assume a new machine joins the WLAN. As a result, the bigraph analysis component updates S_t to S_{t+1} thus: $S_t \longrightarrow\!\!\triangleright_{\mathsf{A}} \longrightarrow\!\!\triangleright^{*}_{\mathsf{J}} S_{t+1}$. But in model S_{t+1} the new machine is not forbidden from receiving data from WWW, thus the policy has to be re-enforced.

In general, after every join network event, the bigraph analysis component applies the sequence of reactions for an enforce policy event. The sequence for a drop policy event is applied before a leave network event, which is in turn followed by the sequence of applications for an enforce policy event.

We illustrate the process, in detail, with the example given in Appendix D.

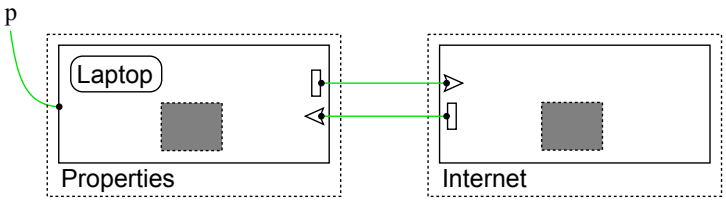


Figure 7.14: Bigraph B_{ς_4} encoding predicate "Host Laptop has unrestricted Internet connection".

7.6 Model analysis

So far we have described how to model home networks with bigraphs and how to update a model in real-time to represent network and policy events. In this section we present an approach that allows us to carry out formal reasoning on the generated models within the HWBig system.

At any point in the model generation process, the bigraph analysis component can check whether the bigraphical representation of the current system satisfies compliance with a policy, or an invariant, or indeed any given property. Some example properties of interest for a WLAN are:

ς_1: "Machine 01:23:45:67:89:ab can receive from host Laptop",

ς_2: "TCP traffic is blocked for machine with IP address 192.168.0.3",

ς_3: "Machine 01:23:45:67:89:ab is in the range of the router's signal and a DHCP lease has been granted",

ς_4: "Host Laptop has unrestricted access to the Internet".

The verification of $S_t \models \varsigma_i$ is an instance of model checking, where S_t is the bigraph representing the current state of the WLAN. Note that all the predicates ς_i above express spatial (i.e. static) properties of the system. Hence, they can be regarded as atomic propositions in a temporal logic and expressed in a straightforward way as a single instance of matching. For example, bigraph B_{ς_4} encoding property ς_4 is represented as shown in Figure 7.14.

We can check any invariants by model checking whether they hold after every update of the system, logging any violations and reporting them, as required, to the system and/or user. Note that conflicting policies can also be detected this way, whenever the application of a new policy invalidates an existing one. The idea is to consider the predicates for compliance with a given policy P (i.e. φ_{P}), as invariants. Therefore, a new policy is conflicting with one of the old ones whenever its application invalidates an invariant. An implementation of the system, can either signal this to the user, deny the enforcement of the second policy or just keep track of conflicts in a logfile.

It is also possible to reason about the dynamical evolution of the system expressed as *temporal* properties. Some examples are:

τ_1: "Machine 01:23:45:67:89:ab *eventually* has to be connected to Laptop",

τ_2: "TCP traffic is *always* blocked for machine with IP address 192.168.0.3",

τ_3: "A lease is granted to machine 01:23:45:67:89:ab *until* it is not in the range of the router's signal".

These properties can be expressed in a temporal logic like **LTL** [51], with temporal modalities such as *until* ($\_\mathbf{U}\_$), *next* ($\mathbf{X}\_$), *finally* ($\mathbf{F}\_$) and *globally* ($\mathbf{G}\_$). The set of atomic propositions is given by the formulae in the fragment of **BiLog** that can be reduced to matching. As discussed above, these are the predicates in the form of ς_i. Observe that a similar approach was adopted in Chapter 6 to express in **CSL** properties of a bigraphical reactive system. The interesting question here is what is the underlying transitions system? We propose that it is the one that can be generated by modelling all the possible sequences of actions, from the current model of the system. In order to generate a finite structure, a fixed set of machines and policies has to be specified by the user. Then, the corresponding reaction rules, as described in sections 7.3 and 7.5 are applied to generate the transition system. For example, a set of rules modelling a station joining the WLAN is generated for every specified MAC address. Figure 7.15 illustrates the generation of the transition system from current model S_t, where S_t is generated from initial state S_0, according to the sequence of real-time events. All states in the generated transition system are reachable from S_t. We note that in general, from any state S_t, we can return to S_0, if we do not constrain the allowable events. Therefore, this approach is only useful if we make some assumptions about what events cannot occur. For example, we might wish to reason about future behaviour, based on the assumption that no machines leave the network, or no new policies are enforced. From a theoretical point of view, the generation process described above is equivalent to the computation of the reaction relation of a PBRS in which the initial state is S_t and the interleavings resulting from the application of reaction rules with lower priorities are discarded. For example, consider the generation of a successor state S_{t+1} obtained by the following sequence of applications

$$S_t \longrightarrow\!\!\!\!\triangleright^{*}_{\mathsf{L1}} \longrightarrow\!\!\!\!\triangleright_{\mathsf{L2}} S_{t+1} \ .$$

All possible intermediate states resulting from an application of reaction rule $\mathsf{R}_{\mathsf{L1}}(\mathsf{MAC})$ are discarded by specifying that $\mathsf{R}_{\mathsf{L1}}(\mathsf{MAC}) < \mathsf{R}_{\mathsf{L2}}(\mathsf{MAC}, \mathsf{IP}, \mathsf{Hostname})$. Note that this approach was also adopted in Chapter 6 for the generation of the underlying CTMC.

In some situations it may be useful to reason about the *past* behaviour of the network. A common example is when one wishes to "debug" the network, i.e. identify the events that

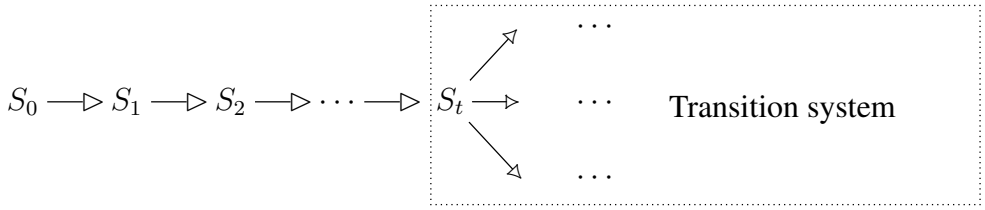


Figure 7.15: Generating all the possible evolutions from current model S_t.

led to a problem in the current configuration of the network. Some temporal properties of this form are:

τ_4: "Machine 01:23:45:67:89:ab has *never* joined the network",

τ_5: "TCP traffic has *always* been blocked for machine with IP address 192.168.0.3 *since* policy P was enforced",

τ_6: "Laptop *was* connected to the Internet".

These properties can be expressed, for instance, in **PtLTL** (**P**ast **t**ime **LTL**) [30, 43], a temporal logic equipped with past-time modalities such as *since* ($\_\mathbf{S}\_$), *previous* ($\mathbf{X}^{-1}\_$), *at some point in the past* ($\mathbf{F}^{-1}\_$) and *globally in the past* ($\mathbf{G}^{-1}\_$). As for the temporal properties about the future we described above, the set of atomic propositions is given by the **BiLog** formulae that are reducible to bigraph matching. In this case, the underlying transition system is simply $S_0 \longrightarrow\!\!\triangleright\, S_1 \longrightarrow\!\!\triangleright \cdots \longrightarrow\!\!\triangleright\, S_t$.

Checking a temporal property involves bigraph matching for atomic propositions and standard model checking techniques for the temporal modalities. We note that the latter is computationally expensive and may not be tractable in real-time, depending on the number of machines and policies and on the temporal modalities of the logic. However, we have as yet found no need for temporal modalities: atomic propositions are currently sufficient for all verification needs expressed by the Homework system users.

Technical details: tracking

Temporal properties (both in the past and in the future) about specific entities of a bigraph can be verified only if they can be *tracked* through a reaction [47, p. 123]. This is in general not possible in abstract BRS because nodes and edges lack identifiers. Consider, for instance, reaction rule

$$\mathsf{A} \mid \mathsf{A} \longrightarrow\!\!\blacktriangleright\, \mathsf{A}$$

that matches two nodes of control A and discards one. When it is applied to a state $S \longrightarrow\!\!\triangleright\, S'$, it is impossible to identify (i.e. track) which one of the two matched A-nodes is still present

in S'. Observe that our bigraphical model suffers from this problem. In particular, M-nodes cannot be tracked. Our workaround consists of allowing only one node with a given MAC control in the model. This is enforced by checking predicates φ_{MAC} and ψ_{MAC} given in Section 7.2 before any topology update. Therefore, unique controls for MAC addresses can be interpreted as node identifiers and employed for node tracking.

7.7 Implementation

A prototype of the HWBig system is fully implemented on the Homework router, which is hosted on a variety of small form-factor PCs. The bigraph encoder and bigraph analysis components are implemented in OCaml by using the library provided by BigraphER. The software runs on a standard Linux Ubuntu distribution. Access control is enforced via NOX (which implements the custom DHCP server) and Open vSwitch, as dictated by the Ponder2 policy engine [63], based on events recorded in the homework stream database.

We trialled the system with both synthetic and experimental data using a router hosted on an Asus Eee PC laptop with the following specification: 1.2GHz Intel Atom CPU, 2 GB RAM, 200 GB SATA HDD, 802.11b/g, 1 Gbps Ethernet, and a USB-to-Ethernet adapter.

For the synthetic data, we added 30 stations to the initial configuration, applying reaction rule $\mathsf{R_A}(\mathsf{MAC}_i)$ 30 times starting from bigraph S_0. The final state is a bigraph with 123 nodes. The update times we recorded also include the delays introduced by the verification of predicates φ_{MAC_i} and ψ_{MAC_i} prior to each reaction rule application. The update times increase with the number of nodes, as indicated in the plot of update times averaged over 100 runs, in Figure 7.16. The non-monotonicity of the plot is due to the random choices in the heuristic of the SAT solver used for matching of bigraphs. Note the slowest update requires just under 0.10 s.

Experimental data was taken from actual network trials. For example, the router sensed the signals of 6 stations, then 4 new devices joined the WLAN and were connected to the Internet. The final state was a bigraph with 71 nodes. The update times were similar to those described above. Evidence from network trials suggests there are rarely more than 20 signals present in a home network and the rate of topology change is much slower than the times used in our (synthetic) experiment.Moreover, our times include a system overhead to generate and store on disk a graphical representation of each bigraph (involving an external invocation of the graph layout generator `dot`). While we expect that considerable speed ups and optimisations are possible to the verification system, we conclude the prototypical system can update and analyse the bigraphical representation of actual home networks in real-time.

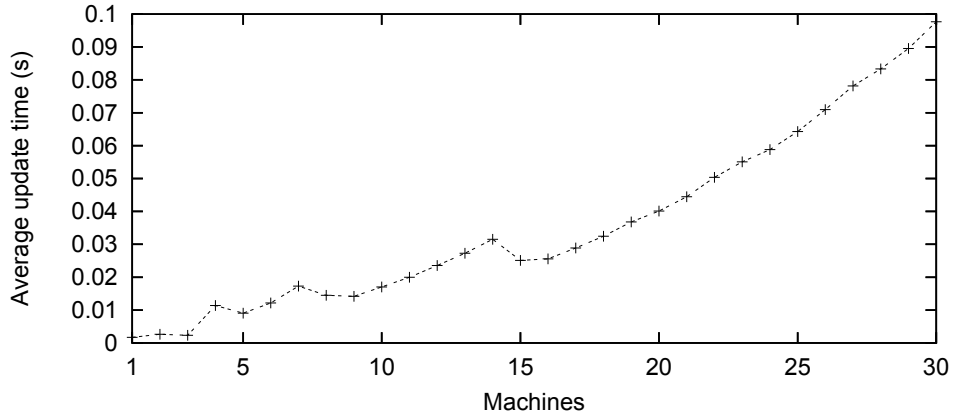


Figure 7.16: Average time to perform $S_n \longrightarrow\!\!\triangleright\, S_{n+1}$, where n is the number of machines in the network (x axis). Each update was performed 100 times and the average time is reported on the y axis.

7.8 Summary

In this chapter we described HWBig, an addition to the Homework information plane that generates and analyses in real-time bigraphical models of the current configuration of the network. The generation process is event-driven: we apply in real-time the reaction rules according to events captured in the stream database. Analysis consists of verification of system properties, including detecting configurations that violate user-invoked access control policies.

In Section 7.1 we gave an overview of the Homework information plane together with the HWBig system. Section 7.2 defined the bigraphs and the reaction rules used to encode network topologies and network events. The sequences of reaction rule applications required to generate the bigraphical models at each network event were described in Section 7.3. Another set of reaction rules was introduced in Section 7.4 to represent policy events. The corresponding generation process was given in Section 7.5. Our bigraphical model defined a general method to check for the non-existence of patterns in reaction rules by tagging and untagging entities. It also introduced a procedure that allows tracking of machines through reaction by using some controls as node identifiers. In Section 7.6, we proposed three approaches to reason on the generated bigraphs. The first was to use **BiLog** predicates to verify static properties of the current model by reduction to bigraph matching. The second consisted of "unfolding" the reaction relation starting from the current model to generate all possible evolutions of the network, which can then be analysed with a temporal logic. The third used predicates expressed in a past-time temporal logic to debug the network. We discussed the implementation details of HWBig in Section 7.7.

Chapter 8

Conclusion and future work

In this thesis, we have introduced bigraphs with sharing, a novel generalisation of Milner's bigraphs specifically conceived to enable a direct representation of spatial locations shared among several entities. The main idea of our formalism is to adopt a definition of place graphs based on DAGs instead of forests. This in turn requires a substantial extension and adaptation of the previous theory of BRS and the development of a new matching algorithm and supporting tools. We showed how sharing facilitates the specification process by modelling two real-world scenarios involving overlapping wireless signals.

In this concluding chapter we summarise the main contributions of this thesis, in Section 8.1, present a critical reflection on what has been achieved and on the experience of using bigraphs for the modelling of wireless networks, in Section 8.2, and outline directions for future research, in Section 8.3.

8.1 Thesis summary

We organised this thesis in two parts: Part I is devoted to the theoretical treatment of bigraphs with sharing; Part II focusses on their applications.

In Part I, Chapter 2 gives a thorough introduction to the standard definition of bigraphs. Our main reference was Milner's book [47]. In more detail, the chapter begins by recalling the fundamental definitions of the constituents of bigraphs (place and link graphs) and their operations (composition and tensor product). It also describes two different ways of expressing bigraphs. The first is an elegant graphical notation in which the nesting of nodes and the edges between them encode locality and connectivity, respectively. The second is an algebraic form that allows us to represent bigraphical terms structurally as in process algebra. The chapter continues with the definition of sorting, a typing discipline that is required in most applications in order to restrict the set of admissible bigraphs. Furthermore, it

presents bigraphical reactive systems (BRS), a dynamical theory of bigraphs based on reaction rules specifying how bigraphs may reconfigure their own placings and linkings. Finally, the categorical interpretation of bigraphs, the spatial logic **BiLog** and several extensions (e.g. stochastic bigraphs) are considered.

In Chapter 3, bigraphs with sharing are defined formally. We justified the need for an extension of the basic formalism enabling a representation of space shared among several entities by exposing the disadvantages of two possible encodings of sharing within the standard definition of bigraphs. The core of the chapter consists of the definition of a new locality model for place graphs with sharing in terms of DAGs instead of forests. We implemented this by replacing the parent map in the standard definition with a binary relation between places. Operations on bigraphs with sharing were also modified in order to support the new formulation. The second part of the chapter describes a stratified graphical notation that overcomes the shortcomings of the Venn diagram style notation by explicitly representing the relation between places. The new formalism is shown to be coherent with Milner's categorical interpretation of bigraphs by introducing $\mathbf{SPg}(\mathcal{K})$ and $\mathbf{SBg}(\mathcal{K})$, the categories of abstract place graphs with sharing and abstract bigraphs with sharing, respectively. Functors to relate them to standard bigraphical categories are also defined. In particular, we constructed functor $\mathsf{width} : \mathbf{SPg}(\mathcal{K}) \to \mathbf{Finord}$. Additionally, we proved self-duality of place graphs with sharing and introduced a characterisation of epimorphisms and monomorphisms. We also showed with an example that precategory $\widetilde{\mathbf{SPg}}(\mathcal{K})$ of concrete place graphs with sharing lacks RPOs in the general case. The chapter concludes by presenting an axiomatisation of expressions for bigraphical terms derived from the equational theory of a bialgebra on finite ordinals. This includes the definition of a stratified normal form.

Chapter 4 introduces a graph theoretic algorithm to solve the bigraph matching problem. We discussed two key features of the algorithm: native support for both standard bigraphs and bigraphs with sharing; capability to enumerate all the distinct occurrences of a pattern in a target. The latter is essential in a stochastic setting for the computation of reaction rates. The algorithm is defined as a reduction to the sub-graph isomorphism problem. In more detail, the first phase of the algorithm finds the isomorphisms between the pattern's underlying DAG and sub-graphs of the target's underlying DAG. In the next phase, the algorithm discards the isomorphisms obtained in the previous phase that do not satisfy the following compatibility conditions: node controls are preserved, pattern's sites and roots allow for a valid decomposition of the target, and no node in the context has an ancestor in the pattern. In the final phase, a mapping between the pattern's and target's link graphs is constructed for every compatible isomorphism. Proofs of soundness and completeness are also provided. This chapter concludes the first part of the thesis on the theory of bigraphs with sharing.

In Part II, Chapter 5 describes the BigraphER system: an OCaml library and a command-line tool for the manipulation, simulation and visualisation of bigraphs with sharing. The

architecture of the command-line tool consists of three interacting components: a compiler, a matching engine, and a rewriting engine. The compiler translates the input specification into a run-time representation of the bigraphical model. We implemented the matching engine by encoding in SAT the matching algorithm defined in Chapter 4. The solutions are then obtained by passing the resulting SAT instance to the MiniSat solver. The rewriting engine builds the reaction relation of the input BRS by iteratively applying the reaction rules to each state. The occurrences of a reaction rule are computed by invoking the matching engine. The visualisation function produces a graphical representation of a bigraph. We based its implementation on the automatic graph layout generation software Graphviz. The BigraphER OCaml library provides programming interfaces for the data structures used internally by the components of the command-line tool. The chapter also introduces a reasoning technique for a class of **BiLog** predicates based on bigraph matching.

In Chapter 6, we presented the first application of bigraphs with sharing to a real-world scenario: a model of 802.11 RTS/CTS that supports overlapping signals and arbitrary network topologies. The model consists of a sorting that specifies the kinds of bigraphs used to encode wireless networks and a set of stochastic reaction rules describing how a network evolves during the execution of the protocol at transmission time. A distinguishing characteristic of this approach is that the network topology is expressed explicitly by representing overlapping wireless signals with sharing nodes. Therefore, locality determines collision probability. In order to avoid the definition of parameterised reaction rules, we organised the reaction rules of our model into priority classes. This not only leads to a more compact and readable specification of the model but also allows BigraphER to build more quickly the reaction relation because only instances of matching that are solvable in polynomial time are generated in the process. In the chapter, we also described a general method to reduce significantly the size of the state space by employing instantaneous reaction rules and discarding the intermediate interleavings obtained by their applications (if confluent). Quantitative analysis is carried out by using the probabilistic model checker Prism and **CSL** predicates expressing properties of a network.

Chapter 7 contains a detailed description of the second application of bigraphs with sharing we considered in this thesis: real-time generation and analysis of bigraphical models of domestic wireless networks. The approach we adopted in this scenario is to define a sorting similar to the one used in Chapter 6 that specifies the structure of the admissible bigraphs used to represent wireless networks and a set of reaction rules to encode network and policy events. However, the fundamental difference here is that the model generation process is event-driven, i.e. instead of computing the whole state space of the BRS, sequences of reaction rules are applied in real-time to update only the current model of the network according to the events captured by the Homework router. The analysis process consists of verification of system properties expressed as **BiLog** predicates. These include detecting

configurations that violate user-invoked access control policies. We implemented the system on the Homework router by using the OCaml library provided by BigraphER. The chapter also introduces a method based on tagging/untagging reaction rules that allows us to verify predicates involving universal quantifiers or negated existential quantifiers by reduction to bigraph matching. Additionally, this method permits to track nodes through a reaction by using a class of controls as unique node identifiers.

8.2 Conclusion

In this thesis, we demonstrated bigraphs with sharing can be formally defined within the general framework of Milner's bigraphical theories, and sharing allows for a native representation of overlapping localities. This feature of our extended formalism is particularly important because it greatly facilitates the modelling of complex wireless networks with non-centralised structure such as sensor networks, wireless ad-hoc networks and ubiquitous systems in general, as proven by the two example applications given in Part II. Furthermore, our research showed that our implementation of BRS (i.e. the BigraphER system) allows bigraphs with sharing to be fruitfully adopted to model and reason about complex applications at industrial level. In particular, the efficiency of our implementation of the matching algorithm was found to be adequate for run-time verification despite being only at an early developing stage.

We now briefly comment on our personal experience of using bigraphs to model wireless networks and spatial-aware systems. When we first encountered bigraphs, our first impression was that the formalism was rather difficult to learn and that it involved too many theoretical notions about process calculi, term rewriting, stochastic modelling and categories. This overwhelming amount of complex mathematical machinery led us to think the learning curve was pretty steep and modelling was going to be a hard task. Not surprisingly, we still face similar reactions when we present the research described in this thesis to the academic community. However, when we began to model the first systems, we soon found out that this complexity could be greatly alleviated by using only the graphical form for the specification of our BRSs. This allowed us to gradually get to know the details of the formalism and to further study its theoretical aspects when they were necessary for a correct understanding of the model. The graphical form proved to be extremely valuable also when our bigraphical models had to be understood and manipulated by non-expert users and by researches without a profound knowledge of concurrency theory and formal modelling.

Other strengths of bigraphs are the ability to easily specify systems with a hierarchical structure and reaction rules that produce complex state modifications when applied. This contrasts with the need to define non-trivial protocols to encode this kind of state modi-

fication we often meet when using process calculi. Despite all the benefits introduced by the adoption of bigraphs, we believe the modelling of realistic systems frequently presents some intrinsic complexity. Therefore, expertise in formal methods is still a modeller's requirement. Two features of BigraphER that speed up and simplify the specification process are parameterised reaction rules, (including parameterised stochastic reaction rates) and rule priorities. In particular, the latter one allowed us to overcome the difficulty in specifying bigraphical reaction rules that can be applied when a condition does not hold as discussed at the end of Section 6.4. Additionally, the automatic generation of a graphical representation of bigraphs was extremely helpful during the model debugging phase. On the other hand, the lack of a visual editor for bigraphs and for reaction rules does not allow us to take full advantage of the graphical form.

8.2.1 Discussion

We now briefly explain the decisions we made in this thesis. Our efforts were driven by the goal of developing a formalism to express in a natural way overlapping topologies. We adopted a problem-oriented approach in which the emphasis was on practical use of bigraphs with sharing for the effective modelling and analysis of real-world scenarios. Therefore, an essential part of our work was to test with examples the adequacy of a locality concept based on DAGs. This is witnessed by the two applications presented in chapters 6 and 7.

The theoretical results presented in Part I are instrumental to the goal of the thesis. For example, we decided to focus on the definition of a normal form for bigraphical expressions in order to derive a specification language that could act as input for a software tool implementing BRSs. Similarly, all the aspects required for the definition of a stochastic semantics and a matching algorithm were investigated in depth. Since our applications did not demand the use of labelled transition systems, the study of label construction through IPOs and bisimulation was left to future research.

A principle we tried to follow throughout this thesis was to concentrate on novel aspects of our research and integrate them with existing theories and tools whenever it was possible. Hence, the following topics were not covered:

- Development of static analysis techniques for reaction rules to prove confluence and termination (refer to the theory of term rewriting [3]).
- Implementation of a model checker for temporal and stochastic predicates. We reduced the verification of **BiLog** predicates to the bigraph matching problem and then used Prism.

- Implementation of a specialised sub-graph isomorphism algorithm. We encoded the problem in SAT and then used MiniSat,
- Implementation of a graph layout generator. We used Graphviz.

8.3 Future work

The work of this thesis suggests some directions for future development of bigraphs with sharing. Some theoretical aspects for further investigation are the following:

- Extension of the definition of the place graph to represent Euclidean space. This may involve the introduction of distances between nodes and node shapes.
- Definition of different BRS semantics. Examples would include BRS based on discrete time Markov Chains, Markov decision processes, timed automata, etc.
- Extension of the definition of the link graph to allow *name aliasing*, i.e. points may have more than one outer name. This generalisation makes link graphs self-dual. It is likely that a complete axiomatisation could be derived from the equational theory of a Frobenius algebra.
- Definition of a procedure for the construction of labelled transition systems for BRS with sharing. This includes the definition of an IPO construction procedure and the study of bisimulation.
- Investigation of whether it is possible to have binding in bigraphs with sharing.

Some possible future developments regarding practical aspects of bigraphs are the following:

- Improvement of the matching algorithm efficiency by reducing the size of the underlying SAT instances (i.e. number and length of clauses).
- Definition of a partial-matching algorithm. This may be suitable as a basis for an RPO construction algorithm and a more efficient verification of **BiLog** predicates.
- Implementation of an automatic procedure to check whether a bigraph respects a given sorting. This simply amounts to interpreting the conditions in a formation rule Φ as **BiLog** predicates.
- Modelling of new phenomena and evaluation of bigraphs with sharing in real deployments – especially in the field of ubiquitous computing and wireless sensor networks.

Appendices

Appendix A

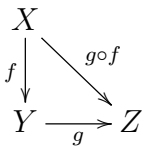
Category theory

This chapter summarises the main ideas of category theory used in this thesis. Our references are the classic work by Mac Lane [45] for general category theory, and Milner's book [47] for RPO and IPO results.

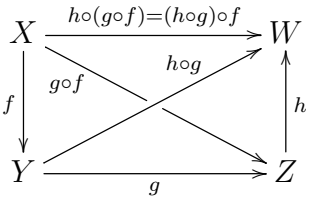
Definition A.1 (category)**.** A *category* $\mathbf{C}$ is given by a collection of *objects* $I, J, K, \ldots$ and a collection of *arrows* (or *morphisms*) $f, g, h, \ldots$ which have the following structure.

1. Each arrow f has a *domain* and *codomain*, both objects. If these are X and Y then we write $f : X \to Y$. We also write $\mathbf{C}(X \to Y)$, for the *homset* of X and Y, i.e. the set of arrows in $\mathbf{C}$ in the form $f : X \to Y$.

2. Given two arrows $f : X \to Y$ and $g : Y \to Z$, the *composition* of f and g is the arrow $g \circ f : X \to Z$. This operation may be pictured by the diagram

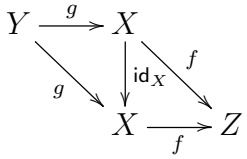


   Composition is *associative*, that is: given $f : X \to Y$, $g : Y \to Z$ and $h : Z \to W$, $h \circ (g \circ f) = (h \circ g) \circ f$. This equation is represented pictorially by the statement that the following diagram commutes[1]



[1] A diagram commutes if every path with common start and end is equal.

3. For every object X there is a unique *identity* arrow $\mathsf{id}_X : X \to X$, satisfying the *unit laws*: $\mathsf{id}_X \circ g = g$ for every $g : Y \to X$ and $f \circ \mathsf{id}_X = f$ for every $f : X \to Z$.



Definition A.2 (functor)**.** A *functor* $\mathfrak{F} : \mathbf{C} \to \mathbf{D}$ between two categories is a function taking objects to objects and arrows to arrows, such that the following properties hold:

1. $\mathfrak{F}(f) : \mathfrak{F}(X) \to \mathfrak{F}(Y)$ for each $f : X \to Y$,
2. $\mathfrak{F}(\mathsf{id}_X) = \mathsf{id}_{\mathfrak{F}(X)}$ for each object X,
3. $\mathfrak{F}(g \circ f) = \mathfrak{F}(g) \circ \mathfrak{F}(f)$ for all arrows $f : X \to Y$ and $g : Y \to Z$.

A functor $\mathfrak{F} : \mathbf{C} \to \mathbf{D}$ is said to be *faithful* if it is injective when restricted to each homset, and it is *full* if it is surjective on each homset. $\mathfrak{F}$ *preserves* a property p that an arrow may have if $\mathfrak{F}(f)$ has property p whenever f has. It *reflects* property p if f has the property whenever $\mathfrak{F}(f)$ has.

Definition A.3 (precategory)**.** A *precategory* $\widetilde{\mathbf{C}}$ is like a category except that the composition $g \circ f$ of $f : X \to Y$ with $g : Y \to Z$ may not always be defined. A functor between precategories is exactly as a functor between categories.

Definition A.4 (subcategory)**.** A *subcategory* $\mathbf{S}$ of a category $\mathbf{C}$ is a collection of some of the objects and some of the arrows of $\mathbf{C}$, which includes with each arrow $f : X \to Y$ both the domain X and the codomain Y, with each object Z its identity arrow id_Z and with each pair of composable arrows $f : X \to Y$, $g : Y \to Z$ their composite $g \circ f : X \to Z$.

Definition A.5 (quotient category)**.** Given a category $\mathbf{C}$, a *congruence relation* on $\mathbf{C}$ specifies, for each pair of objects X, Y, an equivalence relation $\sim_{X,Y}$ on the class of arrows $\mathbf{C}(X, Y)$ which have domain X and codomain Y, such that

1. for $f, g : X \to Y$ and $h : Y \to Z$, if $f \sim_{X,Y} g$ then $h \circ f \sim_{X,Z} h \circ g$;
2. for $f : X \to Y$ and $g, h : Y \to Z$, if $g \sim_{Y,Z} h$ then $g \circ f \sim_{X,Z} h \circ f$.

Given such a congruence relation $\sim$ on $\mathbf{C}$, one can form the *quotient category* $\mathbf{C}/\!\sim$ which has the same objects as $\mathbf{C}$, and arrows $X \to Y$ are $\sim_{X,Y}$-equivalence classes of arrows $X \to Y$ in $\mathbf{C}$.

Definition A.6 (opposite category)**.** The *opposite category* (or *dual category*) $\mathbf{C}^{\mathsf{op}}$ of a given category $\mathbf{C}$ has for objects the objects of $\mathbf{C}$, and its arrows are $f^{\mathsf{op}} : Y \to X$, for each arrow $f : X \to Y$ in $\mathbf{C}$. A category $\mathbf{C}$ is *self-dual* if $\mathbf{C} = \mathbf{C}^{\mathsf{op}}$.

Definition A.7 (isomorphism)**.** An arrow $f : X \to Y$ is an *isomorphism* (iso) if there is $f^{-1} : Y \to X$ such that $f \circ f^{-1} = \mathsf{id}_Y$ and $f^{-1} \circ f = \mathsf{id}_X$. We call f^{-1} the *inverse* of f.

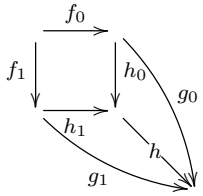
Definition A.8 (monomorphism)**.** An arrow $f : X \to Y$ is a *monomorphism* (mono) if $f \circ g_0 = f \circ g_1$ implies $g_0 = g_1$.

Definition A.9 (epimorphism)**.** An arrow $f : X \to Y$ is an *epimorphism* (epi) if $g_0 \circ f = g_1 \circ f$ implies $g_0 = g_1$.

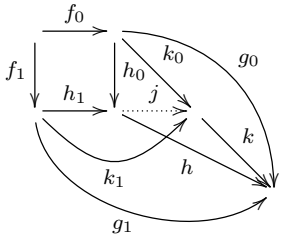
Definition A.10 (span, cospan)**.** A *span* is a pair of arrows $\vec{f} = (f_0, f_1)$ with the same domain. A *cospan* is a pair of arrows $\vec{g} = (g_0, g_1)$ with the same codomain.

Definition A.11 (bound, consistent)**.** If $\vec{f}$ is a span and $\vec{g}$ a cospan such that $g_0 \circ f_0 = g_1 \circ f_1$, then we call $\vec{g}$ a *bound* for $\vec{f}$. If $\vec{f}$ has a bound it is said to be *consistent*.

Definition A.12 (pushout)**.** A *pushout* for a span $\vec{f}$ is a bound $\vec{h}$ for $\vec{f}$ such that, for any bound $\vec{g}$, there is a unique arrow h such that $h \circ h_0 = g_0$ and $h \circ h_1 = g_1$.



Definition A.13 (relative pushout)**.** Let $\vec{g}$ be a bound for $\vec{f}$. A *bound for $\vec{f}$ relative to $\vec{g}$* is a triple $(\vec{h}, h)$ of arrows such that $\vec{h}$ is a bound for $\vec{f}$ and $h \circ h_0 = g_0$ and $h \circ h_1 = g_1$. A *relative pushout (RPO) for $\vec{f}$ relative to $\vec{g}$* is a relative bound $(\vec{h}, h)$ such that for any relative bound $(\vec{k}, k)$ (sometimes called *candidate triple*) there is a unique arrow j for which $j \circ h_0 = k_0$, $j \circ h_1 = k_1$ and $k \circ j = h$. We say that a category *has RPOs* if, whenever a span has a bound, it also has an RPO relative to that bound.



Definition A.14 (idem pushout)**.** If $\vec{f} : X \to \vec{Y}$ is a span, then a cospan $\vec{g} : \vec{Y} \to Z$ is an *idem pushout (IPO)* for $\vec{f}$ if $(\vec{g}, \mathsf{id}_Z)$ is an RPO for $\vec{f}$ to $\vec{g}$.

Definition A.15 (partial monoidal category)**.** A category is *partial monoidal* when it has a partial *tensor product* $\otimes$ both on objects and on arrows satisfying the following conditions.

On objects, $X \otimes Y$ and $Y \otimes X$ are either both defined or both undefined. The same holds for $X \otimes (Y \otimes Z)$ and $(X \otimes Y) \otimes Z$; moreover, they are equal when defined. There is a *unit* object ϵ, for which $\epsilon \otimes X = X \otimes \epsilon$ is always defined.

On arrows, the tensor product of $f : X_0 \to Y_0$ and $g : X_1 \to Y_1$ is defined if and only if $X_0 \otimes X_1$ and $Y_0 \otimes Y_1$ are both defined. The following must hold when both sides are defined:

1. $f \otimes (g \otimes h) = (f \otimes g) \otimes h$
2. $\mathsf{id}_\epsilon \otimes f = f \otimes \mathsf{id}_\epsilon = f$
3. $(f_0 \otimes g_0) \circ (f_1 \otimes g_1) = (f_0 \circ f_1) \otimes (g_0 \circ g_1)$

A functor of partial monoidal categories preserves unit and tensor product.

Definition A.16 (spm category)**.** A partial monoidal category is *symmetric (spm)* if, whenever $X \otimes Y$ is defined, there is an arrow $\gamma_{X,Y} : X \otimes Y \to Y \otimes X$ called a *symmetry*, satisfying the following equations when the compositions and products are defined:

1. $\gamma_{X,\epsilon} = \mathsf{id}_X$
2. $\gamma_{Y,X} \circ \gamma_{X,Y} = \mathsf{id}_{X \otimes Y}$
3. $\gamma_{X_1,Y_1} \circ (f \otimes g) = (g \otimes f) \circ \gamma_{X_0,Y_0}$ for $f : X_0 \to X_1$ and $g : Y_0 \to Y_1$
4. $\gamma_{X \otimes Y,Z} = (\gamma_{X,Z} \otimes \mathsf{id}_Y) \circ (\mathsf{id}_X \otimes \gamma_{Y,Z})$

A functor between spm categories preserves unit, product and symmetries.

Definition A.17 (s-category)**.** An *s-category* $\widetilde{\mathbf{C}}$ is a precategory in which each arrow f is assigned a finite *support* $|f| \subset \mathcal{S}$. Further, $\widetilde{\mathbf{C}}$ possesses a partial tensor product, unit and symmetries, as in an spm category. The identities id_I and symmetries $\gamma_{I,J}$ are assigned empty support. In addition:

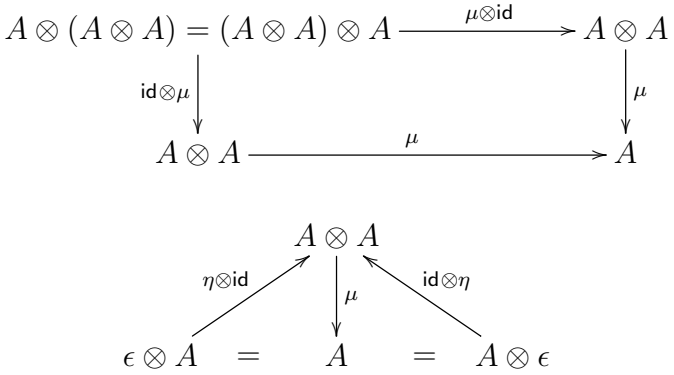
- For $f : I \to J$ and $g : J' \to K$, the composition $g \circ f$ is defined iff $J = J'$ and $|f| \cap |g| = \emptyset$; then $|g \circ f| = |f| \uplus |f|$.
- For $f : I_0 \to I_1$ and $g : J_0 \to J_1$, the tensor product $f \otimes g$ is defined iff $I_i \otimes J_i$ is defined ($i = 0, 1$) and $|f| \cap |g| = \emptyset$; then $|f \otimes g| = |f| \uplus |g|$.

The four kinds of category defined above can be arranged in a hierarchy as shown in Figure A.1.

	Partial composition	Total composition
Symmetric monoidal	s-category	spm category
—	precategory	category

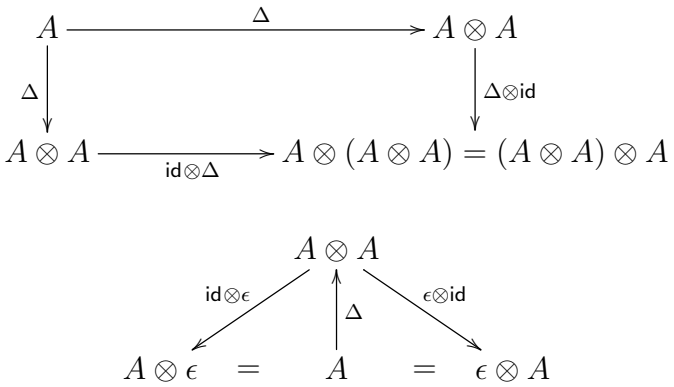
Figure A.1: Hierarchy of categories used in this thesis.

Definition A.18 (monoid)**.** A *monoid* (or *monoid object*) (A, μ, η) in a monoidal category $\mathbf{C}$ is an object A equipped with arrows $\mu : A \otimes A \to A$, called the *multiplication*, and $\eta : \epsilon \to A$, called the *unit*, such that the following diagrams



are commutative. When $\mathbf{C}$ is symmetric and $\mu \circ \gamma = \mu$, we say A is *commutative*.

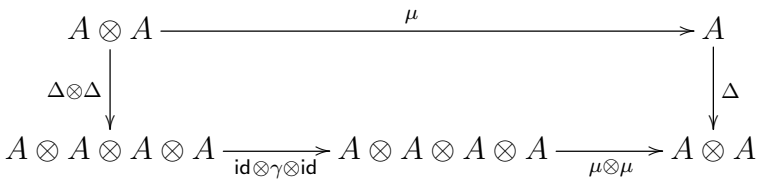
Definition A.19 (co-monoid)**.** A *co-monoid* (A, Δ, ϱ) in a monoidal category $\mathbf{C}$ is an object A equipped with morphisms $\Delta : A \to A \otimes A$, called the *co-multiplication*, and $\varrho : A \to \epsilon$, called the *co-unit*, satisfying

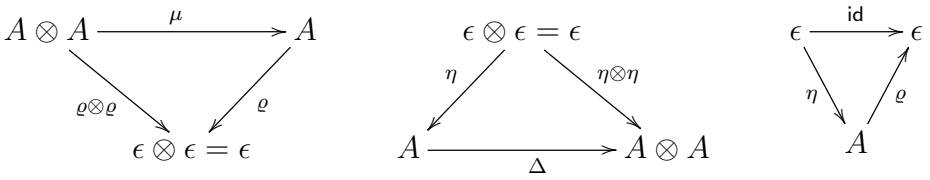


When $\mathbf{C}$ is symmetric and $\gamma \circ \Delta = \Delta$, we say A is *co-commutative*.

Definition A.20 (bialgebra)**.** A *bialgebra* in a symmetric monoidal category $\mathbf{C}$ is given by a tuple $A = (A, \mu, \eta, \Delta, \varrho, \gamma)$ where A is an object of $\mathbf{C}$, γ is a symmetry, (A, μ, η) is a monoid

and (A, Δ, ϱ) is a co-monoid, satisfying





We say A is *commutative* (resp. *co-commutative*) when it is commutative (resp. co-commutative) as a monoid. It is *bicommutative* when it is both commutative and co-commutative. A bialgebra is *qualitative* when the following equality holds:

$$\mu \circ \Delta = \mathsf{id} \ .$$

Appendix B

Continuous Time Markov Chains and the logic CSL

In this chapter we briefly recall the basic concepts of CTMCs and the logic **CSL** (Continuous Stochastic Logic). We follow the presentation given in [6].

Let AP be a fixed, finite set of atomic propositions.

Definition B.1 (labelled CTMC)**.** A *(labelled) CTMC* is a tuple $\mathcal{M} = (S, \mathbf{Q}, L)$ where S is a finite set of *states*, $\mathbf{Q} : S \times S \to \mathbb{R}_{\geq 0}$ is the *rate matrix*, and $L : S \to 2^{AP}$ is a *labelling function* which assigns to each state $s \in S$ the set $L(s)$ of atomic propositions $a \in AP$ that are valid in s.

Intuitively, each $q_{s,s'}$ in $\mathbf{Q}$ specifies that the probability of moving from state s to s' within t time units (for positive t) is $1 - e^{-q_{s,s'}t}$, an exponential distribution with rate $q_{s,s'}$. Hence, the average speed of going from s to s' is ${q_{s,s'}}^{-1}$. If $q_{s,s'} > 0$ for more than one state s', a competition between the transitions is assumed to exist, known as the *race condition*. For any state $s \in S$, the probability of leaving state s within t time units is given by $1 - e^{t\,E(s)}$ where $E(s) = \sum_{s' \in S} q_{s,s'}$. A *path* through a CTMC is an alternating sequence $\sigma = s_0 t_0 s_1 t_1 s_2 \ldots$ such that $q_{s_i,s_{i+1}} > 0$ and $t_i \in \mathbb{R}_{\geq 0}$ for all $i \geq 0$. The time stamps t_i denote the amount of time spent in state s_i. Let $Paths^{\mathcal{M}}(s)$ denote the set of paths of $\mathcal{M}$ that start in state s; $\sigma[i] = s_i$, the i-th state of σ; $\sigma@t$ denote the state of σ occupied at time t; and Pr_s denote the unique probability measure on sets of paths that start in s. We now recall the logic **CSL**.

Definition B.2 (syntax of **CSL**)**.** For $a \in AP$, $p \in [0,1]$, $t \in \mathbb{R}_{\geq 0}$ and $\bowtie \in \{\leq, <, \geq, >\}$, the state formulae of **CSL** are defined by the grammar

$$\Phi ::= \quad \top \quad \Big| \quad a \quad \Big| \quad \Phi \wedge \Phi \quad \Big| \quad \neg\Phi \quad \Big| \quad \mathbf{S}_{\bowtie p}(\Phi) \quad \Big| \quad \mathbf{P}_{\bowtie p}(\varphi)$$

where path formulae are defined by

$$\varphi ::= \quad \mathbf{X}\Phi \quad \Big| \quad \Phi\mathbf{U}\Phi \quad \Big| \quad \Phi\mathbf{U}^{\leq t}\Phi \ .$$

Definition B.3 (semantics of **CSL**)**.** The state formulae are interpreted over the states of a CTMC. Let $\mathcal{M} = (S, \mathbf{R}, L)$ with proposition labels in AP. the definition of the satisfaction relation is as follows.

$$\begin{array}{lcl}
s \models \top & \Leftrightarrow & \text{for all } s \in S \\
s \models a & \Leftrightarrow & a \in L(s) \\
s \models \neg\Phi & \Leftrightarrow & s \not\models \Phi \\
s \models \Phi_1 \wedge \Phi_2 & \Leftrightarrow & s \models \Phi_1 \text{ and } s \models \Phi_2 \\
s \models \mathbf{S}_{\bowtie p}(\Phi) & \Leftrightarrow & \lim_{t\to\infty} Pr_s\{\sigma \in Paths^{\mathcal{M}}(s) \mid \sigma@t \models \Phi\} \bowtie p \\
s \models \mathbf{P}_{\bowtie p}(\varphi) & \Leftrightarrow & Pr_s\{\sigma \in Paths(s) \mid \sigma \models \varphi\} \bowtie p \ .
\end{array}$$

Semantics of path formulae is defined by:

$$\begin{array}{lcl}
\sigma \models \mathbf{X}\Phi & \Leftrightarrow & \sigma[1] \text{ is defined and } \sigma[1] \models \Phi \\
\sigma \models \Phi_1\mathbf{U}\Phi_2 & \Leftrightarrow & \exists k \geq 0.(\sigma[k] \models \Phi_2 \wedge \forall 0 \leq i \leq k.\sigma[i] \models \Phi_1) \\
\sigma \models \Phi_1\mathbf{U}^{\leq t}\Phi_2 & \Leftrightarrow & \exists x \leq t.(\sigma@x \models \Phi_2 \wedge \forall y < x.\sigma@y \models \Phi_1) \ .
\end{array}$$

Appendix C

Reaction rules for the 802.11 CSMA/CA RTS/CTS protocol

We now define an algebraic form for the bigraphs used in Chapter 6 to model the 802.11 CSMA/CA RTS/CTS protocol. Bigraph $N_0 : \epsilon \to \langle \mathsf{s}, \{a_\mathrm{A}, a_\mathrm{B}, a_\mathrm{C}\}\rangle$ represents a wireless network of three stations. It is drawn in Figure 6.3b. Its algebraic definition is given by

$$\begin{aligned} N_0 &= \text{share } (m_\mathrm{A} \parallel m_\mathrm{B} \parallel m_\mathrm{C}) \text{ by } \psi \text{ in } (\mathsf{id}_{a_\mathrm{A} a_\mathrm{B} a_\mathrm{C}} \mid \mathsf{S}^{t_\mathrm{A}}_{a_\mathrm{A}} \mid \mathsf{S}^{t_\mathrm{B}}_{a_\mathrm{B}} \mid \mathsf{S}^{t_\mathrm{C}}_{a_\mathrm{C}}) \\ \psi &= [\{0,1\},\{0,1,2\},\{1,2\}] \ , \end{aligned}$$

where terms

$$\begin{aligned} m_\mathrm{A} &= (\mathsf{id}_{1,a_\mathrm{A} a_\mathrm{B}} \parallel /x \ \parallel /r \ \parallel /q\,)(\mathsf{M}_{rx}.(w_\mathrm{A} \mid \mathsf{A}_{a_\mathrm{A}}.1)) \\ m_\mathrm{B} &= (\mathsf{id}_{1,a_\mathrm{B}} \parallel /x \ \parallel /r\,)(\mathsf{M}_{rx}.\mathsf{A}_{a_\mathrm{B}}.1) \\ m_\mathrm{C} &= (\mathsf{id}_{1,a_\mathrm{C} a_\mathrm{B}} \parallel /x \ \parallel /r \ \parallel /q\,)(\mathsf{M}_{rx}.(w_\mathrm{C} \mid \mathsf{A}_{a_\mathrm{C}}.1)) \ , \end{aligned}$$

indicate stations A, B and C, respectively, and terms

$$w_\mathrm{A} = \mathsf{W}^{l}_{x a_\mathrm{B}}.\mathsf{Q}_q.1 \qquad\qquad w_\mathrm{C} = \mathsf{W}^{k}_{x a_\mathrm{B}}.\mathsf{Q}_q.1$$

encode A's and C's packets, respectively. Graphical representations of placing $\psi : 3 \to 3$ and the other placing we will use in the model are reported in Table C.1.

The following algebraic terms are used to concisely indicate packets:

$$\begin{aligned} w &= \mathsf{W}^{l}_{xd}.\mathsf{Q}_q.1 & w' &= \mathsf{W}^{l'}_{xd}.\mathsf{Q}_q.1 & rts &= \mathsf{RTS}^{l}_{xd}.\mathsf{Q}_q.1 \\ rts' &= \mathsf{RTS}^{l'}_{xd}.\mathsf{Q}_q.1 & cts &= \mathsf{CTS}^{l}_{xd}.\mathsf{Q}_q.1 & p &= \mathsf{P}^{l}_{xd}.\mathsf{Q}_q.1 \ . \end{aligned}$$

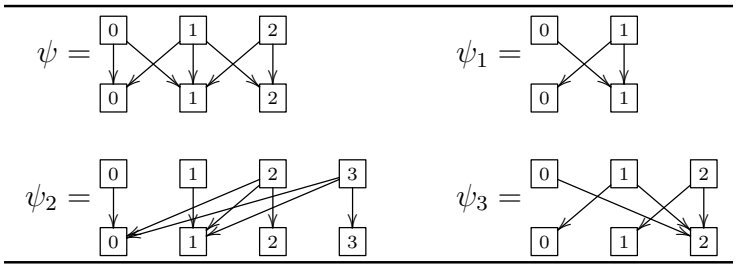


Table C.1: Placings.

Bigraphs indicating machine nodes at various stages of the protocol are given by:

$$\begin{aligned} m_w &= (\mathsf{id}_{1,aqd} \parallel /x \parallel /r\,)(\mathsf{M}_{rx}.(\mathsf{id} \mid w \mid \mathsf{A}_a.1)) \\ m_l &= (\mathsf{id}_{1,aqd} \parallel /x \parallel /r\,)(\mathsf{M}_{\mathsf{L}rx}.(\mathsf{id} \mid rts \mid \mathsf{A}_a.1)) \\ m_b &= (\mathsf{id}_{1,aqd} \parallel /x \parallel /r\,)(\mathsf{M}_{\mathsf{B}rx}.(\mathsf{id} \mid rts \mid \mathsf{A}_a.1)) \ . \end{aligned}$$

We begin with the algebraic description of the stochastic reaction rules. Reaction rule $\mathsf{R}_{\mathsf{RTS}}(t,l) : \mathsf{m}\widehat{\mathsf{pa}} \to \langle \mathsf{sm}, \{a,d,q\}\rangle$ is defined as

$$\begin{aligned} \mathsf{R}_{\mathsf{RTS}}(t,l) = {} & \text{share } (\mathsf{id} \parallel m_w) \text{ by } \psi_1 \text{ in } (\mathsf{id}_{1,adq} \parallel \mathsf{S}^t_a) \\ & \xrightarrow{\rho_1(t)} \\ & \text{share } (\mathsf{id} \parallel m_l) \text{ by } \psi_1 \text{ in } (\mathsf{id}_{1,adq} \parallel \mathsf{S}^t_{\mathsf{L}a}) \\ \psi_1 = {} & [\{1\},\{0,1\}] \ . \end{aligned}$$

Reaction rule $\mathsf{R}_{\mathsf{CTS}}(t,t',l) : \widehat{\mathsf{pa}}\widehat{\mathsf{pa}}\mathsf{mm} \to \langle \mathsf{ssmm}, \{x,a,d,q\}\rangle$ is given by

$$\begin{aligned} \mathsf{R}_{\mathsf{CTS}}(t,t',l) = {} & \text{share } (m_l \parallel \mathsf{M}_{\mathsf{D}rx}.(\mathsf{id} \mid \mathsf{A}_d.1) \parallel \mathsf{id}_2) \text{ by } \psi_2 \text{ in } (\mathsf{id}_{2,xadq} \parallel (\mathsf{S}^t_{\mathsf{L}a} \mid \mathsf{S}^{t'}_d) \parallel /r\,) \\ & \xrightarrow{\rho_2} \\ & \text{share } (send \parallel rec \parallel \mathsf{id}_2) \text{ by } \psi_2 \text{ in } (\mathsf{id}_{2,xadq} \parallel (\mathsf{S}^t_{\mathsf{L}a} \mid \mathsf{S}^{t'}_{\mathsf{L}d}) \parallel /r\,) \ , \end{aligned}$$

with

$$\begin{aligned} send &= (\mathsf{id}_{1,aqdr} \parallel /x\,)(\mathsf{M}_{\mathsf{L}rx}.(\mathsf{id} \mid cts \mid \mathsf{A}_a.1)) \qquad & rec &= \mathsf{M}_{\mathsf{L}rx}.(\mathsf{id} \mid \mathsf{A}_d.1) \\ \psi_2 &= [\{0,2,3\},\{1,2,3\},\{2\},\{3\}] & &. \end{aligned}$$

An algebraic definition of reaction rule $\mathsf{R}_{\mathsf{DATA}}(l) : \widehat{\mathsf{pa}}\widehat{\mathsf{pa}} \to \langle \mathsf{mm}, \{x,a,d,q\}\rangle$ is

$$\begin{aligned} \mathsf{R}_{\mathsf{DATA}}(l) &= (\mathsf{id}_{2,xdaq} \parallel /r\,)(send \parallel rec) \xrightarrow{\rho_3(l)} (\mathsf{id}_{2,xdaq} \parallel /r\,)(send' \parallel rec) \\ send' &= (\mathsf{id}_{1,aqdr} \parallel /x\,)(\mathsf{M}_{\mathsf{L}rx}.(\mathsf{id} \mid p \mid \mathsf{A}_a.1)) \ . \end{aligned}$$

Reaction rule $\mathsf{R}_{\mathsf{ACK}}(t,t',l) : \widehat{\mathsf{pa}}\widehat{\mathsf{pa}}\mathsf{mm} \to \langle \mathsf{ssmm}, \{x,a,d,q\}\rangle$ can be defined as follows

$$\begin{aligned}\mathsf{R}_{\mathsf{ACK}}(t,t',l) = {} & \text{share } (send' \parallel rec \parallel \mathsf{id}_2) \text{ by } \psi_2 \text{ in } (\mathsf{id}_{2,xaqd} \parallel (\mathsf{S}^{t}_{\mathsf{L}a} \mid \mathsf{S}^{t'}_{\mathsf{L}d}) \parallel /r\,) \\ & \xrightarrow{\rho_4} \\ & \text{share } (send'' \parallel rec' \parallel \mathsf{id}_2) \text{ by } \psi_2 \text{ in } (\mathsf{id}_{2,xaqd} \parallel (\mathsf{S}^{t_{min}}_{\mathsf{C}a} \mid \mathsf{S}^{t'}_{\mathsf{C}d}))\ ,\end{aligned}$$

with

$$send'' = (\mathsf{id}_{1,aq} \parallel /r\,)(\mathsf{M}_{rq}.(\mathsf{id} \mid \mathsf{A}_a.1)) \qquad rec' = (\mathsf{id}_{1,dx} \parallel /r\,)(\mathsf{M}_{rx}.(\mathsf{id} \mid \mathsf{A}_d.1))\ .$$

Reaction rule $\mathsf{R}_{\mathsf{BACK1}} : \widehat{\mathsf{pa}}\widehat{\mathsf{pa}}\mathsf{mm} \to \langle \mathsf{ssmm}, \{x,a,d,q\}\rangle$ is given by

$$\mathsf{R}_{\mathsf{BACK1}}(t,t',l) = lhs_1 \xrightarrow{\rho_5} rhs_1\ ,$$

with

$$\begin{aligned} lhs_1 &= \text{share } (m_b \parallel \mathsf{M}_{\mathsf{P}rx}.(\mathsf{id} \mid \mathsf{A}_d.1) \parallel \mathsf{id}_2) \text{ by } \psi_2 \text{ in } (\mathsf{id}_{2,xaqd} \parallel (\mathsf{S}^{t}_{\mathsf{L}a} \mid \mathsf{S}^{t'}_{d}) \parallel /r\,) \\ rhs_1 &= \text{share } (m_w \parallel \mathsf{M}_{rx}.(\mathsf{id} \mid \mathsf{A}_d.1) \parallel \mathsf{id}_2) \text{ by } \psi_2 \text{ in } (\mathsf{id}_{2,xaqd} \parallel (\mathsf{S}^{2t+1}_{\mathsf{C}a} \mid \mathsf{S}^{t'}_{d}) \parallel /r\,)\ .\end{aligned}$$

Similarly, reaction rule $\mathsf{R}_{\mathsf{BACK2}}(t,t',l) : \widehat{\mathsf{pa}}\widehat{\mathsf{pa}}\mathsf{mm} \to \langle \mathsf{ssmm}, \{x,a,d,q\}\rangle$ is

$$\mathsf{R}_{\mathsf{BACK2}}(t,t',l) = lhs_2 \xrightarrow{\rho_5} rhs_2\ ,$$

where

$$\begin{aligned} lhs_2 &= \text{share } (m_b \parallel \mathsf{M}_{\mathsf{D}rx}.(\mathsf{id} \mid \mathsf{A}_d.1) \parallel \mathsf{id}_2) \text{ by } \psi_2 \text{ in } (\mathsf{id}_{2,xaqd} \parallel (\mathsf{S}^{t}_{\mathsf{L}a} \mid \mathsf{S}^{t'}_{d}) \parallel /r\,) \\ rhs_2 &= rhs_1\ .\end{aligned}$$

Finally, we define instantaneous reaction rules. Rule $\mathsf{R}_{\mathsf{D}}(t) : \mathsf{m}\widehat{\mathsf{pa}} \to \langle \mathsf{ms}, \{a',a,x\}\rangle$ is given by

$$\begin{aligned}\mathsf{R}_{\mathsf{D}}(t) = {} & \text{share } (\mathsf{id} \parallel \mathsf{M}_{rx}.(\mathsf{id} \mid \mathsf{A}_a.1)) \text{ by } \psi_1 \text{ in } (\mathsf{id}_{1,ax} \parallel \mathsf{S}^{t}_{\mathsf{L}a'} \parallel /r\,) \\ & \longrightarrow \\ & \text{share } (\mathsf{id} \parallel \mathsf{M}_{\mathsf{D}rx}.(\mathsf{id} \mid \mathsf{A}_a.1)) \text{ by } \psi_1 \text{ in } (\mathsf{id}_{1,ax} \parallel \mathsf{S}^{t}_{\mathsf{L}a'} \parallel /r\,)\ .\end{aligned}$$

Reaction rule $\mathsf{R}_{\mathsf{P}}(t,t') : \mathsf{mm}\widehat{\mathsf{pa}} \to \langle \mathsf{ms}, \{a',a'',a,x\}\rangle$ is defined as

$$\mathsf{R}_{\mathsf{P}}(t,t') = lhs_3 \longrightarrow rhs_3\ ,$$

where terms lhs_3, rhs_3 and placing ψ_3 are given by

$$\begin{aligned} lhs_3 &= \text{share } (\mathsf{id}_2 \parallel \mathsf{M}_{\mathsf{D}rx}.(\mathsf{id} \mid \mathsf{A}_a.1)) \text{ by } \psi_3 \text{ in } (\mathsf{id}_{1,ax} \parallel (\mathsf{S}^t_{\mathsf{L}a'} \mid \mathsf{S}^{t'}_{\mathsf{L}a''}) \parallel /r\,) \\ rhs_3 &= \text{share } (\mathsf{id}_2 \parallel \mathsf{M}_{\mathsf{P}rx}.(\mathsf{id} \mid \mathsf{A}_a.1)) \text{ by } \psi_3 \text{ in } (\mathsf{id}_{1,ax} \parallel (\mathsf{S}^t_{\mathsf{L}a'} \mid \mathsf{S}^{t'}_{\mathsf{L}a''}) \parallel /r\,) \\ \psi_3 &= [\{1\}, \{2\}, \{0,1,2\}] \ . \end{aligned}$$

Reaction rule $\mathsf{R}_{\mathsf{B}}(l) : \widehat{\mathsf{pa}}\widehat{\mathsf{pa}} \to \langle \mathsf{mm}, \{xaqd\} \rangle$ is

$$\mathsf{R}_{\mathsf{B}}(l) = m_l \parallel (\mathsf{id}_{1,xd} \parallel /r\,)(\mathsf{M}_{\mathsf{P}rx}.(\mathsf{id} \mid \mathsf{A}_d.1)) \longrightarrow m_b \parallel (\mathsf{id}_{1,xd} \parallel /r\,)(\mathsf{M}_{\mathsf{P}rx}.(\mathsf{id} \mid \mathsf{A}_d.1)) \ .$$

An algebraic expression for reaction rule $\mathsf{R}_{\mathsf{UD}}(t) : \mathsf{m}\widehat{\mathsf{pa}} \to \langle \mathsf{ms}, \{a', a, x\} \rangle$ is

$$\mathsf{R}_{\mathsf{UD}}(t) = lhs_4 \longrightarrow rhs_4 \ ,$$

with

$$\begin{aligned} lhs_4 &= \text{share } (\mathsf{id} \parallel \mathsf{M}_{\mathsf{D}rx}.(\mathsf{id} \mid \mathsf{A}_a.1)) \text{ by } \psi_1 \text{ in } (\mathsf{id}_{1,ax} \parallel \mathsf{S}^t_{\mathsf{C}a'} \parallel /r\,) \\ rhs_4 &= \text{share } (\mathsf{id} \parallel \mathsf{M}_{rx}.(\mathsf{id} \mid \mathsf{A}_a.1)) \text{ by } \psi_1 \text{ in } (\mathsf{id}_{1,ax} \parallel \mathsf{S}^t_{\mathsf{C}a'} \parallel /r\,) \ . \end{aligned}$$

Reaction rule $\mathsf{R}_{\mathsf{UP}}(t) : \mathsf{m}\widehat{\mathsf{pa}} \to \langle \mathsf{ms}, \{a', a, x\} \rangle$ is defined as

$$\mathsf{R}_{\mathsf{UP}}(t) = lhs_5 \longrightarrow rhs_5 \ ,$$

where

$$\begin{aligned} lhs_5 &= \text{share } (\mathsf{id} \parallel \mathsf{M}_{\mathsf{P}rx}.(\mathsf{id} \mid \mathsf{A}_a.1)) \text{ by } \psi_1 \text{ in } (\mathsf{id}_{1,ax} \parallel \mathsf{S}^t_{\mathsf{C}a'} \parallel /r\,) \\ rhs_5 &= \text{share } (\mathsf{id} \parallel \mathsf{M}_{rx}.(\mathsf{id} \mid \mathsf{A}_a.1)) \text{ by } \psi_1 \text{ in } (\mathsf{id}_{1,ax} \parallel \mathsf{S}^t_{\mathsf{C}a'} \parallel /r\,) \ . \end{aligned}$$

Reaction rule $\mathsf{R}_{\mathsf{UC}}(t) : \mathsf{m}\widehat{\mathsf{pa}} \to \langle \mathsf{ms}, \{a', a, x\} \rangle$ is given by

$$\mathsf{R}_{\mathsf{UC}}(t) = lhs_6 \longrightarrow rhs_6 \ ,$$

with

$$\begin{aligned} lhs_6 &= \text{share } (\mathsf{id} \parallel \mathsf{M}_{rx}.(\mathsf{id} \mid \mathsf{A}_a.1)) \text{ by } \psi_1 \text{ in } (\mathsf{id}_{1,ax} \parallel \mathsf{S}^t_{\mathsf{C}a} \parallel /r\,) \\ rhs_6 &= \text{share } (\mathsf{id} \parallel \mathsf{M}_{rx}.(\mathsf{id} \mid \mathsf{A}_a.1)) \text{ by } \psi_1 \text{ in } (\mathsf{id}_{1,ax} \parallel \mathsf{S}^t_{a} \parallel /r\,) \ . \end{aligned}$$

Appendix D

Interplay between network events and policy events

In this chapter we show step-by-step how the bigraphical model of the current network configuration is updated in real-time, according to different kind of events from the stream database component. We indicate models by $S_0, S_1, \ldots$. When a sequence of reaction rules is applied, due to confluence only one interleaving is considered. Refer to Chapter 7 for a complete description of the HWBig system and the formal definition of the bigraphical representation of WLANs.

Initially, no stations are present in the network, as given by bigraph S_0 in Figure 7.2. Now we consider the following scenario consisting of eight events. A summary of the policy events of the example, and their encodings, is given in Table D.1, and a summary of the sequence of reactions is given in Tables D.2 and D.3.

1. *The user specifies and enforces a new policy that all out-going TCP traffic for any machine is forbidden.*
   This user-action corresponds to a policy event in the stream database, which triggers the generation (by the bigraph encoder component) of the reaction rules for a forbid policy. We denote the policy by P1 and give the corresponding reaction rules in Figure D.1. Tagging reaction rule $\mathsf{R}_{\mathsf{P1}_\mathsf{T}}$ matches any out-going channel of any machine that is part of the WLAN[1] and complies with P1. On the right hand-side, the matched channel is tagged. Enforcing reaction rule $\mathsf{R}_{\mathsf{P1}_\mathsf{E}}$ matches any untagged channel. On the right hand-side, a TCP-node (i.e. the policy constraint) is linked to the matched channel. Moreover, the channel is tagged to avoid the introduction of duplicate constraints. Untagging reaction rule $\mathsf{R}_{\mathsf{P1}_\mathsf{U}}$ removes the tags placed by the applications of the previous reaction rules. Finally, the bigraph encoder component generates bigraph $B_{\varphi_{\mathsf{P1}}}$,

[1] An $\widehat{\text{io}}$-channel is present thus a DHCP lease has already been granted.

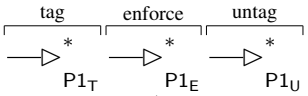
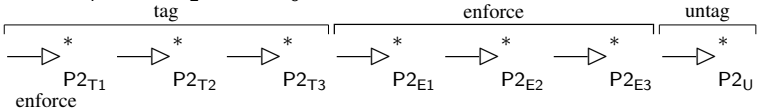
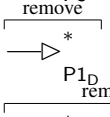
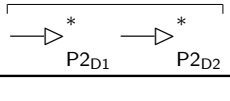
Event	Encoding	Short form
Enforce P1		
Enforce P2		
Enforce P3		
Drop P1		
Drop P2		

Table D.1: Summary of the policy event encodings used in the example.

as given in Figure D.1e. It matches any untagged out-going channel. As described in Section 7.5, P1 is violated when $B_{\varphi_{\mathsf{P1}}}$ is a match in the temporary model in which all blocked out-going channels are tagged. At this point, the bigraph analysis component enforces P1 on S_0. Formally, $S_0 \longrightarrow\!\!\triangleright^*_{\mathsf{P1}} S_0$, i.e. no reaction rule is applicable because no machines are present in S_0. Policy P1 is also checked. Since $B_{\varphi_{\mathsf{P1}}}$ is not a match, then P1 is holds.

2. *Machine* MAC1 *enters a location covered by the router's signal.*
   Since $S_0 \not\models \varphi_{\mathsf{MAC1}}$, the bigraph encoder component generates rule $\mathsf{R_A}(\mathsf{MAC1})$ and the bigraph analysis component updates the system: $S_0 \longrightarrow\!\!\triangleright_{\mathsf{A}} S_1$. The updated model is shown in Figure D.2. After this step, the bigraph analysis component checks whether P1 is violated. In this case, reaction rule $\mathsf{R_{P1_T}}$ is not applicable and $B_{\varphi_{\mathsf{P1}}}$ is not a match in S_1. Therefore, the policy is not violated.

3. *A DHCP lease is granted to machine* MAC1.
   In the current state, we have $S_1 \models \varphi_{\mathsf{MAC1}}$ and $S_1 \not\models \psi_{\mathsf{MAC1}}$ Therefore, the bigraph analysis component updates the system by applying rules $\mathsf{R_{J1}}$, $\mathsf{R_{J2}}(\mathsf{MAC1}, \mathsf{IP1}, \mathsf{N1})$, and $\mathsf{R_{J3}}(\mathsf{MAC1})$: $S_1 \longrightarrow\!\!\triangleright^*_J S_2$. The resulting state is shown in Figure D.3. After the topology update, the bigraph analysis component enforces P1 in S_2. The following updates are performed: $S_2 \longrightarrow\!\!\triangleright^*_{\mathsf{P1_T}} \longrightarrow\!\!\triangleright^*_{\mathsf{P1_E}} \longrightarrow\!\!\triangleright^*_{\mathsf{P1_U}} S_3$[2]. In the following, we indicate this sequence of applications as $\longrightarrow\!\!\triangleright^*_{\mathsf{P1}}$. The graphical form of bigraph S_3 is given in Figure D.4.

4. *Machine* MAC2 *enters the router's signal range.*
   Since $S_3 \not\models \varphi_{\mathsf{MAC2}}$, the bigraph analysis component performs the steps described above for machine MAC1. Initially, the topology is updated with $S_3 \longrightarrow\!\!\triangleright_{\mathsf{A}} S_4$. Then,

[2]In this case $\longrightarrow\!\!\triangleright^*$ is $\longrightarrow\!\!\triangleright$ because only one machine is part of the network in S_2.

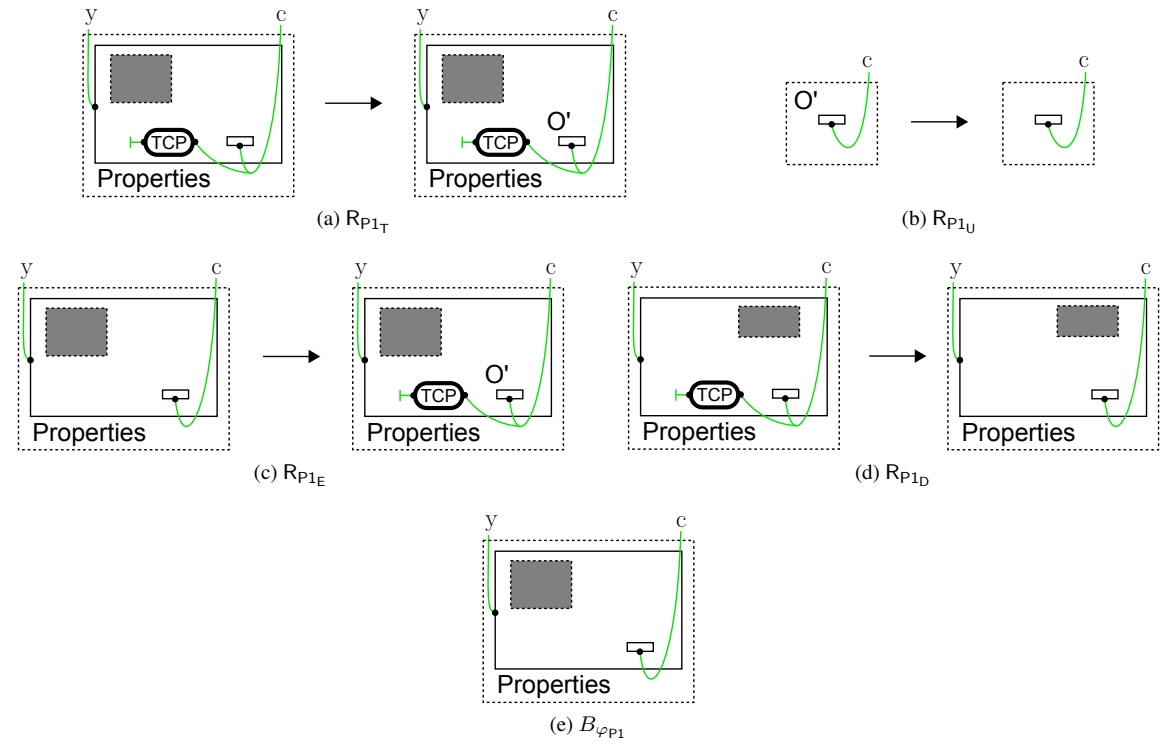


Figure D.1: Policy P1: forbid all out-going TCP traffic for any machine. Tag sequence is $\longrightarrow\!\triangleright^*_{P1_T}$, enforce sequence is $\longrightarrow\!\triangleright^*_{P1_E}$, untag sequence is $\longrightarrow\!\triangleright^*_{P1_U}$, and drop sequence is $\longrightarrow\!\triangleright^*_{P1_D}$.

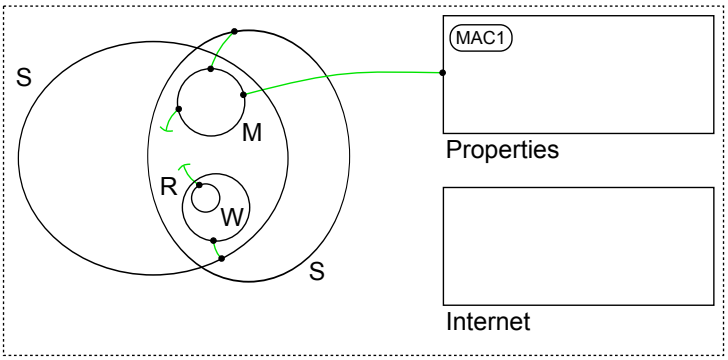


Figure D.2: State S_1: machine MAC1 enters the router's signal range.

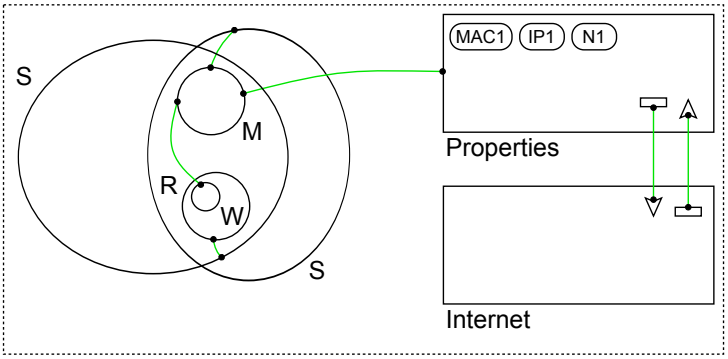


Figure D.3: State S_2: a DHCP lease is granted to machine MAC1.

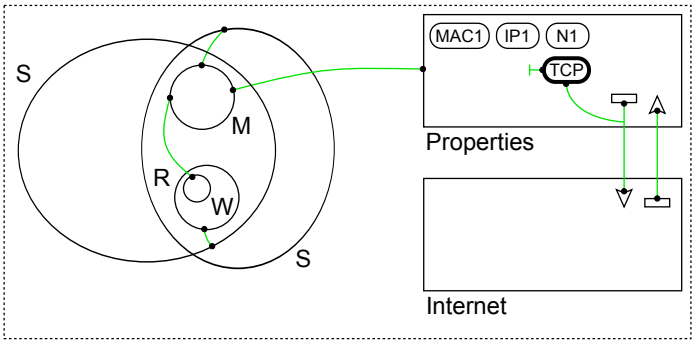


Figure D.4: State S_3: MAC1 is part of the network and P1 is enforced.

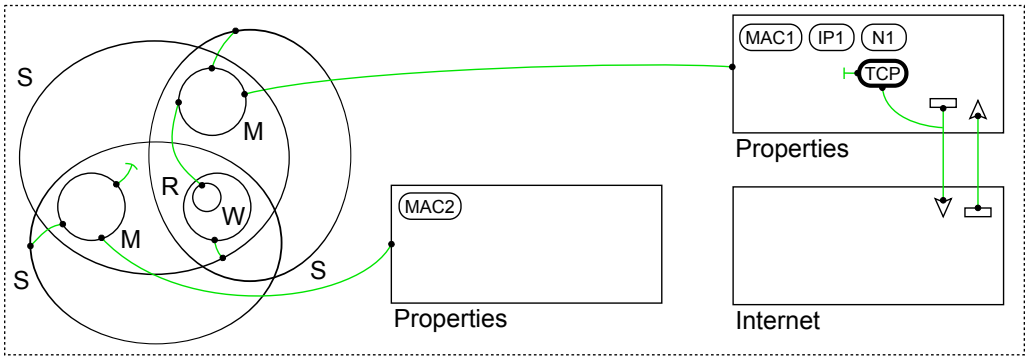


Figure D.5: State S_4: MAC2 enters the router's signal range, MAC1 is part of the WLAN and P1 is enforced.

P1 is enforced with the usual tagging, matching, untagging sequence: $S_4 \longrightarrow\!\triangleright^*_{\mathsf{P1}} S_4$. Observe that no update is performed because reaction rule $\mathsf{R}_{\mathsf{P1_E}}$ is not applicable. This is because there are no out-going channels requiring to be blocked in machine MAC2. Finally φ_{P1} is checked. Since $B_{\varphi_{\mathsf{P1}}}$ is not a match in T_4 (i.e. the bigraph obtained by tagging S_4), then P1 is not violated. The bigraph for updated model S_4 is given in Figure D.5.

5. *Machine* MAC2 *joins the WLAN.*
   The status of the current configuration is $S_4 \models \varphi_{\mathsf{MAC2}}$ and $S_4 \not\models \psi_{\mathsf{MAC2}}$. Hence, the bigraph encoder components generates the reaction rule encoding a join event. Those are then applied by the bigraph analysis component to update the model: $S_4 \longrightarrow\!\triangleright^*_{\mathsf{J}} S'_4$. Afterwards, policy P1 is enforced with $S'_4 \longrightarrow\!\triangleright^*_{\mathsf{P1}} S_5$. Finally, the bigraph analysis component checks whether φ_{P1} holds. Since $B_{\varphi_{\mathsf{P1}}}$ does not occur in temporary model T_5, we have $S_5 \models \varphi_{\mathsf{P1}}$. Updated model S_5 is given in Figure D.6.

The sequence of events and the corresponding model updates described so far are summarised in Table D.2, and we continue with further events as follows.

6. *The user specifies and enforces a new policy that blocks TCP and UDP traffic for machine* MAC2.

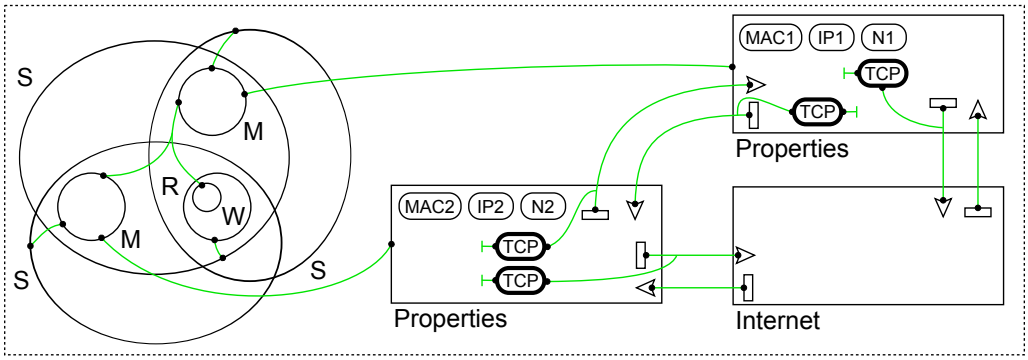


Figure D.6: State S_5: MAC1 and MAC2 joined the WLAN and P1 is enforced.

	Event	Actions	WLAN model	Status
	Initial state	—	S_0	—
1.	P1 enforced	$S_0 \longrightarrow\!\triangleright^*_{\mathsf{P1}} S_0$ $S_0 \longrightarrow\!\triangleright^*_{\mathsf{P1_T}} T_0$ $B_{\varphi_{\mathsf{P1}}}$ ~~match~~ T_0 $T_0 \longrightarrow\!\triangleright^*_{\mathsf{P1_U}} S_0$ check P1	S_0	$S_0 \models \varphi_{\mathsf{P1}}$
2.	MAC1 appears	$S_0 \not\models \varphi_{\mathsf{MAC1}}$ $S_0 \longrightarrow\!\triangleright_{\mathsf{A}} S_1 \longrightarrow\!\triangleright^*_{\mathsf{P1}} S_1$ $S_1 \longrightarrow\!\triangleright^*_{\mathsf{P1_T}} T_1$ $B_{\varphi_{\mathsf{P1}}}$ ~~match~~ T_1 $T_1 \longrightarrow\!\triangleright^*_{\mathsf{P1_U}} S_1$ check P1	S_1	$S_1 \models \varphi_{\mathsf{P1}}$
3.	MAC1 joins the WLAN	$S_1 \models \varphi_{\mathsf{MAC1}}$ and $S_1 \not\models \psi_{\mathsf{MAC1}}$ $S_1 \longrightarrow\!\triangleright^*_{\mathsf{J}} S_2 \longrightarrow\!\triangleright^*_{\mathsf{P1}} S_3$ $S_3 \longrightarrow\!\triangleright^*_{\mathsf{P1_T}} T_3$ $B_{\varphi_{\mathsf{P1}}}$ ~~match~~ T_3 $T_3 \longrightarrow\!\triangleright^*_{\mathsf{P1_U}} S_3$ check P1	S_3	$S_3 \models \varphi_{\mathsf{P1}}$
4.	MAC2 appears	$S_3 \not\models \varphi_{\mathsf{MAC2}}$ $S_3 \longrightarrow\!\triangleright_{\mathsf{A}} \longrightarrow\!\triangleright^*_{\mathsf{P1}} S_4$ $S_4 \longrightarrow\!\triangleright^*_{\mathsf{P1_T}} T_4$ $B_{\varphi_{\mathsf{P1}}}$ ~~match~~ T_4 $T_4 \longrightarrow\!\triangleright^*_{\mathsf{P1_U}} S_4$ check P1	S_4	$S_4 \models \varphi_{\mathsf{P1}}$
5.	MAC2 joins the WLAN	$S_4 \models \varphi_{\mathsf{MAC2}}$ and $S_4 \not\models \psi_{\mathsf{MAC2}}$ $S_4 \longrightarrow\!\triangleright^*_{\mathsf{J}} \longrightarrow\!\triangleright^*_{\mathsf{P1}} S_5$ $S_5 \longrightarrow\!\triangleright^*_{\mathsf{P1_T}} T_5$ $B_{\varphi_{\mathsf{P1}}}$ ~~match~~ T_5 $T_5 \longrightarrow\!\triangleright^*_{\mathsf{P1_U}} S_5$ check P1	S_5	$S_5 \models \varphi_{\mathsf{P1}}$

Table D.2: Generation of models $S_0 \longrightarrow\!\triangleright \cdots \longrightarrow\!\triangleright S_5$.

To model this forbid policy, which we denote by P2, the bigraph encoder component generates the reaction rules and bigraphs given in Figure D.7. Reaction rules $\mathsf{R}_{\mathsf{P2}_{\mathsf{T1}}}$, $\mathsf{R}_{\mathsf{P2}_{\mathsf{T2}}}$, and $\mathsf{R}_{\mathsf{P2}_{\mathsf{T3}}}$ are used to tag MAC2's out-going channels already linked to a constraint. In particular, the first rule tags channels linked to a TCP-node *and* an UDP-node, the second tags channels linked only to a TCP-node, and the third tags channels linked only to an UDP-node. Reaction rules $\mathsf{R}_{\mathsf{P2}_{\mathsf{E1}}}$, $\mathsf{R}_{\mathsf{P2}_{\mathsf{E2}}}$, and $\mathsf{R}_{\mathsf{P2}_{\mathsf{E3}}}$ are used enforce the policy. They link the appropriate policy constraint to a tagged channel. Reaction rule $\mathsf{R}_{\mathsf{P2}_{\mathsf{U}}}$ removes the tag from a channel, while reaction rules $\mathsf{R}_{\mathsf{P2}_{\mathsf{D1}}}$ and $\mathsf{R}_{\mathsf{P2}_{\mathsf{D2}}}$ are applied when P2 is dropped. Bigraph $B_{\varphi_{\mathsf{P2}}}$ encodes policy predicate φ_{P2}. After the generation phase, the bigraph analysis component enforces P2 on model S_5. The sequence of applications is $S_5 \longrightarrow\!\!\triangleright^*_{\mathsf{P2}} S_6$. The updated model is given in Figure D.8. Then, P1 and P2 are checked. Observe that both $B_{\varphi_{\mathsf{P1}}}$ and $B_{\varphi_{\mathsf{P2}}}$ do not match T_6, because all channels in the model can be tagged. Therefore, we conclude compliance with both policies in state S_6.

7. *The user specifies and enforces a new policy that allows out-going TCP traffic for machine* MAC2.

   We denote the new allow policy by P3. Again two steps are performed in the update process. The first consists of the generation by the bigraph encoder component of reaction rule R_{P3} and bigraph $B_{\varphi_{\mathsf{P3}}}$ (see Figure D.9). The left hand-side matches any out-going channel in machine MAC2 that is linked to a TCP-node. On the right hand-side, the constraint is removed. Bigraph $B_{\varphi_{\mathsf{P3}}}$ is used to encode predicate φ_{P3}. It is defined as the left hand-side of enforcing reaction rule R_{P3}. The second step is enforcing of policy P3 on the system by the bigraph analysis component: $S_6 \longrightarrow\!\!\triangleright^*_{\mathsf{P3}} S_7$. The updated model is shown in Figure D.10. Observe there is compliance with P3, but not with P1 and P2.

8. MAC2*'s signal disappears.*

   This is recorded on the stream database ad a network event. Since $S_7 \models \varphi_{\mathsf{MAC2}}$ and $S_7 \models \psi_{\mathsf{MAC2}}$, the machine needs to be removed from the model. First, the bigraph analysis component drops all the policies. Second, sequences for a machine leaving the router's signal range ($\mathsf{R}_{\mathsf{R}}(\mathsf{MAC2})$) and removing a machine ($\mathsf{R}_{\mathsf{L1}}(\mathsf{MAC2})$, $\mathsf{R}_{\mathsf{L2}}(\mathsf{MAC2}, \mathsf{IP2}, \mathsf{N2})$) are applied. Formally, $S_7 \longrightarrow\!\!\triangleright^*_{\cancel{\mathsf{P2}}} \longrightarrow\!\!\triangleright^*_{\cancel{\mathsf{P1}}} \longrightarrow\!\!\triangleright^*_{\mathsf{L}} \longrightarrow\!\!\triangleright_{\mathsf{R}} S'_7$. Finally, the policies are enforced again on model S'_7: $S'_7 \longrightarrow\!\!\triangleright^*_{\mathsf{P1}} \longrightarrow\!\!\triangleright^*_{\mathsf{P2}} \longrightarrow\!\!\triangleright^*_{\mathsf{P3}} S_8$. Observe that $S_8 = S_3$, which is given in Figure D.4. There is compliance with policy P1 because all the out-going channels, namely MAC1's channels, are blocked, and with policies P2 and P3 because there are no MAC2-nodes in S_8. Hence, no matches are possible with $B_{\varphi_{\mathsf{P2}}}$ and $B_{\varphi_{\mathsf{P3}}}$.

A summary of the model generation sequence $S_5 \longrightarrow\!\!\triangleright \cdots \longrightarrow\!\!\triangleright S_8$ is given in Table D.3.

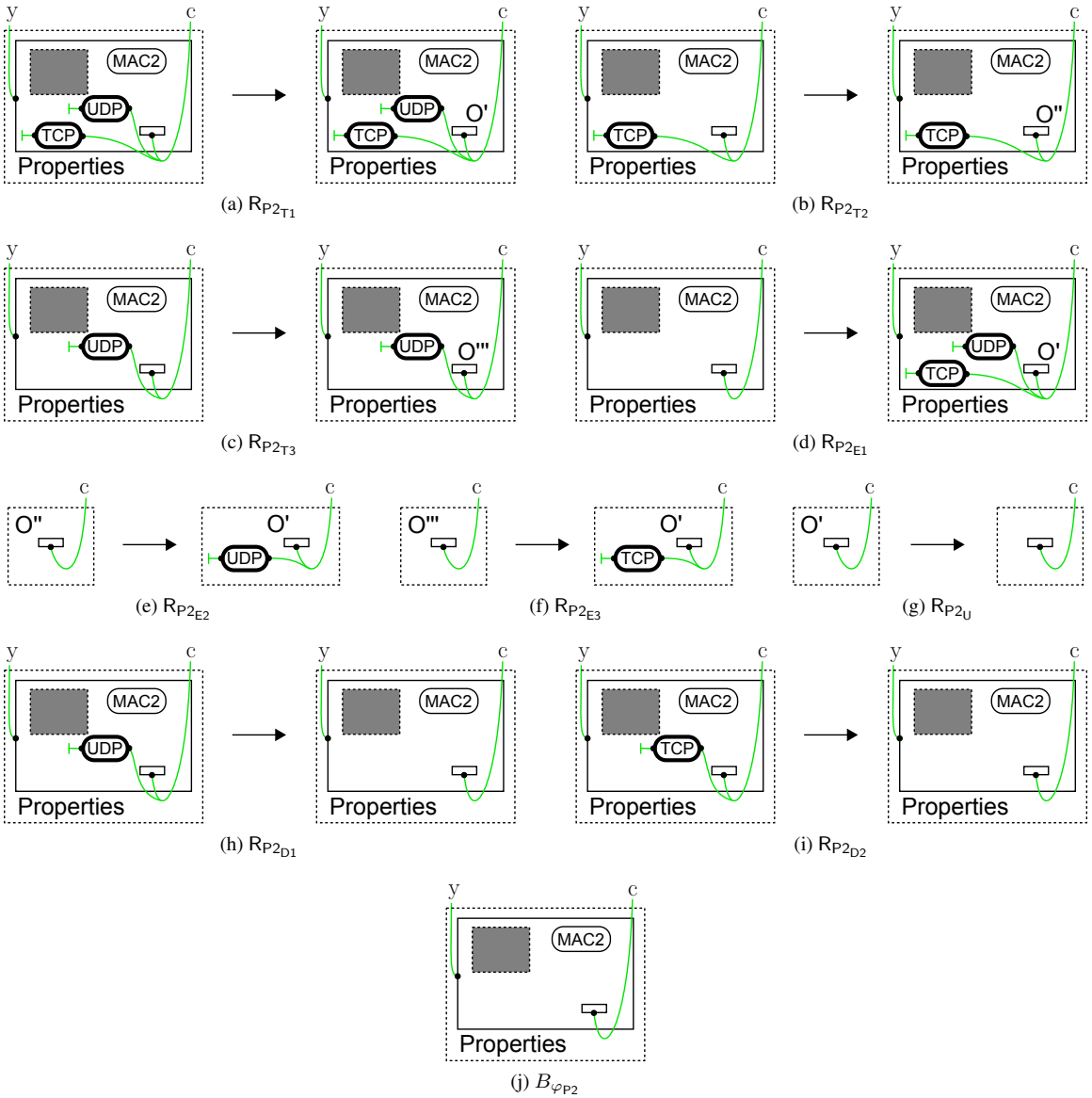


Figure D.7: Policy P2: forbid out-going TCP and UDP traffic for machine MAC2. Tag sequence is $\longrightarrow\!\triangleright^{*}_{P2_{T1}} \longrightarrow\!\triangleright^{*}_{P2_{T2}} \longrightarrow\!\triangleright^{*}_{P2_{T3}}$, enforce sequence is $\longrightarrow\!\triangleright^{*}_{P2_{E1}} \longrightarrow\!\triangleright^{*}_{P2_{E2}} \longrightarrow\!\triangleright^{*}_{P2_{E3}}$, untag sequence is $\longrightarrow\!\triangleright^{*}_{P2_{U}}$, and drop sequence is $\longrightarrow\!\triangleright^{*}_{P2_{D1}} \longrightarrow\!\triangleright^{*}_{P2_{D2}}$.

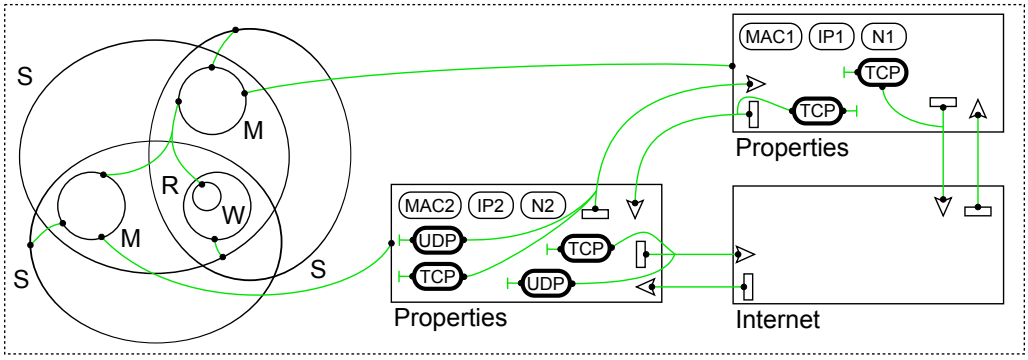
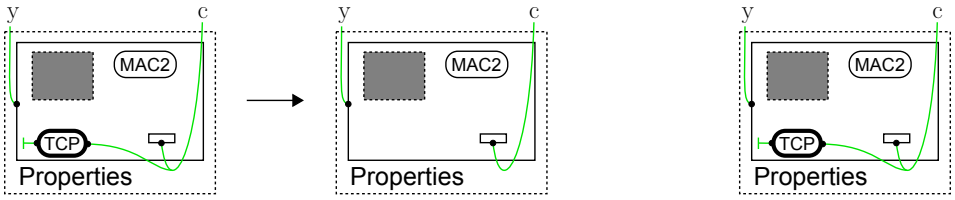


Figure D.8: State S_6: P2 is enforced.



(a) R_{P3} (b) $B_{\varphi_{\mathsf{P3}}}$

Figure D.9: P3: Allow out-going TCP traffic for machine MAC2. Enforce sequence is $\longrightarrow\!\triangleright^*_{\mathsf{P3}}$.

	Event	Actions	WLAN model	Status
6.	P2 enforced	$S_5 \longrightarrow\!\triangleright^*_{\mathsf{P2}} S_6$ $S_6 \longrightarrow\!\triangleright^*_{\mathsf{P1_T}} T_6$ $B_{\varphi_{\mathsf{P1}}}$ ~~match~~ T_6 — check P1 $T_6 \longrightarrow\!\triangleright^*_{\mathsf{P1_U}} S_6$ $S_6 \longrightarrow\!\triangleright^*_{\mathsf{P2_{T1}}} T_6$ $B_{\varphi_{\mathsf{P2}}}$ ~~match~~ T_6 — check P2 $T_6 \longrightarrow\!\triangleright^*_{\mathsf{P2_U}} S_6$	S_6	$S_6 \models \varphi_{\mathsf{P1}}$ $S_6 \models \varphi_{\mathsf{P2}}$
7.	P3 enforced	$S_6 \longrightarrow\!\triangleright^*_{\mathsf{P3}} S_7$ $S_7 \longrightarrow\!\triangleright^*_{\mathsf{P1_T}} T_7$ $B_{\varphi_{\mathsf{P1}}}$ match T_7 — check P1 $T_7 \longrightarrow\!\triangleright^*_{\mathsf{P1_U}} S_7$ $S_7 \longrightarrow\!\triangleright^*_{\mathsf{P2_{T1}}} T_7$ $B_{\varphi_{\mathsf{P2}}}$ match T_7 — check P2 $T_7 \longrightarrow\!\triangleright^*_{\mathsf{P2_U}} S_7$ $B_{\varphi_{\mathsf{P3}}}$ ~~match~~ S_7 — check P3	S_7	$S_7 \not\models \varphi_{\mathsf{P1}}$ $S_7 \not\models \varphi_{\mathsf{P2}}$ $S_7 \models \varphi_{\mathsf{P3}}$
8.	MAC2 disappears	$S_7 \models \varphi_{\mathsf{MAC2}}$ and $S_7 \models \psi_{\mathsf{MAC2}}$ $S_7 \longrightarrow\!\triangleright^*_{\not\mathsf{P2}} \longrightarrow\!\triangleright^*_{\not\mathsf{P1}} \longrightarrow\!\triangleright^*_{\mathsf{L}} \longrightarrow\!\triangleright_{\mathsf{R}} S'_7$ $S'_7 \longrightarrow\!\triangleright^*_{\mathsf{P1}} \longrightarrow\!\triangleright^*_{\mathsf{P2}} \longrightarrow\!\triangleright^*_{\mathsf{P3}} S_8$ $S_8 \longrightarrow\!\triangleright^*_{\mathsf{P1_T}} T_8$ $B_{\varphi_{\mathsf{P1}}}$ ~~match~~ T_8 — check P1 $T_8 \longrightarrow\!\triangleright^*_{\mathsf{P1_U}} S_8$ $S_8 \longrightarrow\!\triangleright^*_{\mathsf{P2_{T1}}} T_8$ $B_{\varphi_{\mathsf{P2}}}$ ~~match~~ T_8 — check P2 $T_8 \longrightarrow\!\triangleright^*_{\mathsf{P2_U}} S_8$ $B_{\varphi_{\mathsf{P3}}}$ ~~match~~ S_8 — check P3	S_8	$S_8 \models \varphi_{\mathsf{P1}}$ $S_8 \models \varphi_{\mathsf{P2}}$ $S_8 \models \varphi_{\mathsf{P3}}$

Table D.3: Generation of models $S_5 \longrightarrow\!\triangleright \cdots \longrightarrow\!\triangleright S_8$.

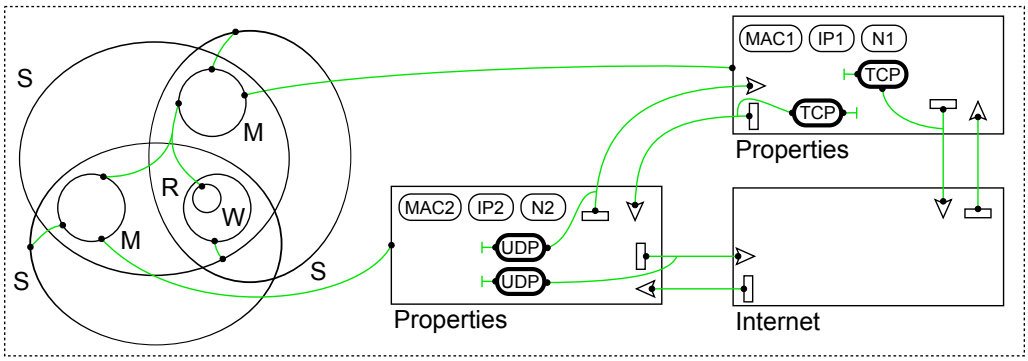


Figure D.10: State S_7: P3 is enforced.

References

[1] A. Alshanyour and A. Agarwal. Performance of IEEE 802.11 RTS/CTS with finite buffer and load in imperfect channels: Modeling and analysis. In *GLOBECOM*, pages 1–6, 2010.

[2] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Verifying continuous time Markov chains. In R. Alur and T. Henzinger, editors, *Computer Aided Verification*, volume 1102 of *Lecture Notes in Computer Science*, pages 269–276. Springer Berlin / Heidelberg, 1996.

[3] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1999.

[4] G. Bacci, D. Grohmann, and M. Miculan. DBtk: a toolkit for directed bigraphs. In *Proceedings of the 3rd international conference on Algebra and coalgebra in computer science*, CALCO'09, pages 413–422, Berlin, Heidelberg, 2009. Springer-Verlag.

[5] J.C.M. Baeten, J.A. Bergstra, J.W. Klop, and W.P. Weijland. Term-rewriting systems with rule priorities. *Theoretical Computer Science*, 67:283–301, October 1989.

[6] C. Baier, J.-P. Katoen, and H. Hermanns. Approximate symbolic model checking of continuous-time Markov chains. In *Proceedings of the 10th International Conference on Concurrency Theory*, CONCUR '99, pages 146–161, London, UK, 1999. Springer-Verlag.

[7] C. Becker and F. Dürr. On location models for ubiquitous computing. *Personal Ubiquitous Computing*, 9:20–31, January 2005.

[8] D. Benyon. The new HCI? navigation of information space. *Knowledge-Based Systems*, 14(8):425–430, 2001.

[9] G. Berry and G. Boudol. The chemical abstract machine. *Theoretical Computer Science*, 96(1):217–248, 1992.

[10] G. Bianchi. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE Journal on Selected Areas in Communications*, 18(3):535–547, 2000.

[11] A. Biere, A. Cimatti, E.M. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, TACAS '99, pages 193–207, London, UK, 1999. Springer-Verlag.

[12] L. Birkedal, T.C. Damgaard, A.J. Glenstrup, and R. Milner. Matching of bigraphs. *Electronic Notes in Theoretical Computer Science*, 175:3–19, July 2007.

[13] L. Birkedal, S. Debois, and T. Hildebrandt. On the construction of sorted reactive systems. In F. van Breugel and M. Chechik, editors, *Proceedings of the 19th International Conference on Concurrency Theory 2008*, volume 5201 of *Lecture Notes in Computer Science*, pages 218–232. Springer-Verlag, August 2008.

[14] R.E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys (CSUR)*, 24(3):293–318, 1992.

[15] M. Calder and M. Sevegnani. Process algebra for event-driven runtime verification: a case study of wireless network management. *Proceedings of IFM2012, Lecture Notes in Computer Science*, pages 21–23, 2012.

[16] L. Cardelli and A. Gordon. Mobile ambients. In M. Nivat, editor, *Foundations of Software Science and Computation Structures*, volume 1378 of *Lecture Notes in Computer Science*, pages 140–155. Springer Berlin / Heidelberg, 1998. 10.1007/BFb0053547.

[17] H.S. Chhaya and S. Gupta. Performance modeling of asynchronous data transfer methods of IEEE 802.11 MAC protocol. *Wireless Networks*, 3(3):217–234, August 1997.

[18] A. Church. A set of postulates for the foundation of logic. *The Annals of Mathematics*, 33(2):346–366, 1932.

[19] K. Claessen, N. Een, M. Sheeran, and N. Sorensson. Sat-solving in practice. In *9th International Workshop on Discrete Event Systems (WODES), 2008.*, pages 61–67. IEEE, 2008.

[20] E. Clarke. The birth of model checking. In O. Grumberg and H. Veith, editors, *25 Years of Model Checking*, volume 5000 of *Lecture Notes in Computer Science*, pages 1–26. Springer Berlin / Heidelberg, 2008.

[21] G. Conforti, D. Macedonio, and V. Sassone. Spatial logics for bigraphs. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *Proceedings*

of the 32th International Colloquium on Automata, Languages and Programming (ICALP'05), volume 3580 of *Lecture Notes in Computer Science*, pages 766–778. Springer-Verlag, 2005.

[22] A. Corradini and F. Gadducci. An algebraic presentation of term graphs, via GS-monoidal categories. *Applied Categorical Structures*, 7:7–299, 1999.

[23] A. Crabtree, T. Rodden, T. Hemmings, and S. Benford. Finding a place for ubicomp in the home. In *Proceedings of Ubicomp*, pages 208–226. Springer, 2003.

[24] J Crowcroft. The privacy and safety impact of technology choices for command, communications and control of the public highway. *SIGCOMM Comput. Commun. Rev.*, 36(1):53–58, January 2006.

[25] T.C. Damgaard and L. Birkedal. Axiomatizing binding bigraphs. *Nordic Journal of Computing*, 13(1–2):58–77, 2006.

[26] M. Duflot, L. Fribourg, T. Herault, R. Lassaigne, F. Magniette, S. Messika, S. Peyronnet, and C. Picaronny. Probabilistic model checking of the CSMA/CD protocol using prism and APMC. *Electronic Notes in Theoretical Computer Science*, 128(6):195–214, 2005.

[27] N. Eén and N. Sörensson. MiniSat: A minimalistic and high-performance SAT solver. `http://minisat.se/`, 2012.

[28] H. Ehrig. Introduction to the algebraic theory of graph grammars (a survey). In *Proceedings of the International Workshop on Graph-Grammars and Their Application to Computer Science and Biology*, pages 1–69, London, UK, 1979. Springer-Verlag.

[29] H. Ehrig and H.-J. Kreowski. Applications of graph grammar theory to consistency, synchronization and scheduling in data base systems. *Information Systems*, 5(3):225 – 238, 1980.

[30] E.A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics*, pages 995–1072. MIT Press, Cambridge, MA, USA, 1990.

[31] D. Eppstein. Subgraph isomorphism in planar graphs and related problems. In *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, SODA '95, pages 632–640, Philadelphia, PA, USA, 1995. Society for Industrial and Applied Mathematics.

[32] A.J. Glenstrup, T.C. Damgaard, L. Birkedal, and E. Hï¿½jsgaard. An implementation of bigraph matching, 2008.

[33] Graphviz - Graph Visualization Software. `http://www.graphviz.org/`, 2012.

[34] D. Grohmann and M. Miculan. Directed bigraphs. *Electronic Notes in Theoretical Computer Science*, 173:121–137, April 2007.

[35] D. Grohmann and M. Miculan. Reactive systems over directed bigraphs. In L. Caires and V.T. Vasconcelos, editors, *Proceedings of the 18th International Conference on Concurrency Theory (CONCUR'07)*, volume 4703 of *Lecture Notes in Computer Science*, pages 380–394. Springer-Verlag, 2007.

[36] IEEE. Std. 802.11e-2005, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, Amendment 8: Medium Access Control (MAC) quality of service enhancements, 2005.

[37] O. Jensen and R. Milner. Bigraphs and transitions. In *Proceedings of the 30th ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages (POPL'03)*, pages 38–49. ACM Press, 2003.

[38] H. Kautz, D. McAllester, and B. Selman. Encoding plans in propositional logic. In *Proc. International Conference on Principles of Knowledge Representation and Reasoning*, pages 374–385. Morgan Kaufmann Publishers, 1996.

[39] J. Krivine, R. Milner, and A. Troina. Stochastic bigraphs. *Electronic Notes in Theoretical Computer Science*, 218:73–96, 2008.

[40] M. Kwiatkowska, G. Norman, and D. Parker. Prism 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *Lecture Notes in Computer Science*, pages 585–591. Springer, 2011.

[41] M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model checking of the IEEE 802.11 wireless local area network protocol. In H. Hermanns and R. Segala, editors, *Proc. 2nd Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification (PAPM/PROBMIV'02)*, volume 2399 of *Lecture Notes in Computer Science*, pages 169–187. Springer, 2002.

[42] J.J. Leifer. *Operational congruences for reactive systems*. PhD thesis, Computer Laboratory, University of Cambridge, 2001.

[43] O. Lichtenstein, A. Pnueli, and L. Zuck. The glory of the past. In Rohit Parikh, editor, *Logics of Programs*, volume 193 of *Lecture Notes in Computer Science*, pages 196–218. Springer Berlin / Heidelberg, 1985.

[44] E. Lupu, N. Dulay, M. Sloman, J. Sventek, S. Heeps, S. Strowes, K. Twidle, S.-L. Keoh, and A. Schaeffer-Filho. Amuse: autonomic management of ubiquitous e-health systems. *Concurrency and Computation: Practice and Experience*, 20(3):277–295, 2008.

[45] S. Mac Lane. *Categories for the Working Mathematician (Graduate Texts in Mathematics)*. Springer, 2nd edition, September 1998.

[46] R. Milner. *A calculus of communicating systems*. Springer-Verlag New York, Inc., 1982.

[47] R. Milner. *The Space and Motion of Communicating Agents*. Cambridge University Press, 2009.

[48] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, Parts I and II. *Information and Computation*, 100(1):1–40, 1992.

[49] S. Mimram. The structure of first-order causality. In *LICS '09: Proceedings of the 2009 24th Annual IEEE Symposium on Logic In Computer Science*, pages 212–221, Washington, DC, USA, 2009. IEEE Computer Society.

[50] T. O'Connell, P. Jensen, A. Dey, and G. Abowd. Location in the aware home. In *Location Modeling for Ubiquitous Computing-Ubicomp 2001 Workshop Proceedings*, pages 41–44, 2001.

[51] Amir P. The temporal logic of programs. *Foundations of Computer Science, IEEE Annual Symposium on*, 0:46–57, 1977.

[52] D. Park. Concurrency and automata on infinite sequences. *Theoretical Computer Science*, pages 167–183, 1981.

[53] G. Perrone, S. Debois, and T.T. Hildebrandt. A model checker for bigraphs. In *Proceedings of ACM Symposium of Applied Computing SVT*, Trento, Italy, March 2012.

[54] C.A. Petri. Kommunikation mit Automaten. Technical report, Technical Report 2 (Schriften des IIM), Institut fur Instrumentelle Mathematik, Bonn, Germany, 1962.

[55] G.D. Plotkin. A structural approach to operational semantics. Technical Report Report DAIMI FN-19, Aarhus University, 1981.

[56] M. Sevegnani and M. Calder. Bigraphs with sharing. Technical Report TR-2010-310, University of Glasgow, 2010.

[57] M. Sevegnani, C. Unsworth, and M. Calder. A SAT based algorithm for the matching problem in bigraphs with sharing. Technical Report TR-2010-311, University of Glasgow, 2010.

[58] P. Sewell. Global/local subtyping and capability inference for a distributed π-calculus. *Automata, Languages and Programming*, pages 695–706, 1998.

[59] R. Shamir and D. Tsur. Faster subtree isomorphism. *Journal of Algorithms*, 33(2):267 – 280, 1999.

[60] J.G. Stell. *Categorical aspects of unification and rewriting*. PhD thesis, University of Manchester, 1992.

[61] J. Sventek, A. Koliousis, O. Sharma, N. Dulay, D. Pediaditakis, M. Sloman, T. Rodden, T. Lodge, B. Bedwell, K. Glover, and R. Mortier. Proceedings of the 12th IFIP/IEEE international symposium on integrated network management, IM 2011. In N. Agoulmine, C. Bartolini, T. Pfeifer, and D. O'Sullivan, editors, *Integrated Network Management*, Dublin, Ireland, May 2011. IEEE.

[62] A.M. Turing. On computable numbers, with an application to the Entscheidungsproblem. A correction. *Proceedings of the London Mathematical Society*, 2(1):544, 1938.

[63] K. Twidle, N. Dulay, E. Lupu, and M. Sloman. Ponder2: A Policy System for Autonomous Pervasive Environments. In *The Fifth International Conference on Autonomic and Autonomous Systems*, April 2009.

[64] J.R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM*, 23(1):31–42, 1976.

[65] M. Weiser, R. Gold, and J.S. Brown. The origins of ubiquitous computing research at PARC in the late 1980s. *IBM systems journal*, 38(4):693–696, 1999.

[66] T. Werth. Design and Implementation of a DAG-Miner. Diploma thesis, Friedrich-Alexander-Universität, Erlangen-Nürnberg, 2007.

[67] P.H. Yuh, C.C.Y. Lin, T.W. Huang, T.Y. Ho, C.L. Yang, and Y.W. Chang. A SAT-based routing algorithm for cross-referencing biochips. In *Proceedings of the System Level Interconnect Prediction Workshop*, SLIP '11, pages 6:1–6:7, Piscataway, NJ, USA, 2011. IEEE Press.

Index