FACULTY OF COMPUTER SCIENCE,
"Al. I. Cuza" UNIVERSITY, IAŞI

Operational Semantics of P Systems

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science

by

Oana-Maria Andrei

Advisor: Professor Dorel Lucanu

June, 2005

Contents

The meaning of things lies not in the things themselves,
but in our attitude towards them.

Antoine de Saint-Exupéry

Abstract

Membrane computing is a branch of natural computing which studies computational models abstracted from the structure and the functioning of living cells. Membrane computing is based on *membrane systems* or *P systems*. The computational model is described by the following features: it has a finite tree-like structure, each membrane determines a region where multisets of objects are processed by means of evolution rules, adjacent regions can communicate, and membranes can be dissolved. The P systems provide a nice abstraction for parallel and distributed systems.

The existing results in membrane computing refer mainly to the P systems characterization of Turing computability, and to some polynomial solutions to NP-complete problems.

The main purpose of this paper is to formally describe an operational semantics of P systems. As a first result of the paper, we introduce an abstract syntax for P systems, then we present the three sets of inference rules corresponding to the evolution stages of a P system during a transition step: maximal parallel rewriting, parallel communication through membranes, and parallel membrane dissolving. We provide these definitions with correctness proofs. The above operational semantics is a natural semantics (or big-step like), due to the parallel nature of this model.

The second result of the paper consists in an implementation for P systems in Maude, an algebraic specification language based on rewriting logic. We obtain a simulator for P systems for which we can use an existent LTL model checker implemented in Maude. Moreover, this implementation provides a structural operational semantics (or small-step like).

We end with the proof of the operational correspondence between the two operational semantics given above.

Chapter 1

Introduction

Membrane computing is a branch of natural computing which studies distributed and parallel computing models and computing paradigms abstracted from the structure and the functioning of living cells. Membrane computing is based on *membrane systems* or *P systems*, a new class of computing devices introduced in [40]. The approach is based on hierarchical and compartmentalized systems: finite cell-structures consisting of membranes embedded in a main membrane; the membranes determine regions where objects and evolution rules can be placed. The objects evolve according to the rules associated with each region "in a maximally parallel manner, non-deterministically choosing the rules and the objects" [41], and the regions cooperate in order to maintain the proper behavior of the whole system. A computation starts from an initial configuration of the system, and terminates when no further rule can be applied.

The domain is rather young and very active; new properties are discovered, as well as connections with already known concepts. It is desirable to find new connections with the applied computer science, including implementations and executable specifications. A sequential software simulator of membrane systems is presented in [11], and a parallel simulator is presented in [10], while a flexible Web-based simulator is available at `http://psystems.ieat.ro`.

The P systems are inspired by biological systems, but they are based on the theory of automata and formal languages. The existing results in membrane computing refer mainly to the P systems characterization of Turing computability, and to some polynomial solutions to NP-complete problems. No operational semantics has yet been proposed.

The first result presented in this paper consists of the formal description of an operational semantics of P systems [3]. First we give an abstract syntax of P systems, and then we define three sets of inference rules corresponding to the three stages of an evolution step: maximal parallel rewriting, parallel communication through membranes, and parallel membrane dissolving. Moreover we provide these definitions with proofs for the correctness of each set of inference rules.

Structural operational semantics is an approach originally introduced by Plotkin [42] in which the operational semantics of a programming language or a computational model is specified in a logical way, independent of a machine architecture or implementation details, by means of rules that provide an inductive definition based on the elementary structures of the language or model. A clarifying introduction to this subjects can be found in [26].

Within "structural operational semantics", two main approaches coexist:

- *Big-step semantics* (also called *natural semantics* by Kahn [27], Gunter [25], and Nielson and Nielson [37], and *evaluation semantics* by Hennessy [26]). In this approach, the main inductive predicate describes the overall result or value of executing a computation, ignoring the intermediate steps.

- *Small-step semantics* (also called *structural operational semantics* by Plotkin [42], and Nielson and Nielson [37], *computational semantics* by Hennessy [26], and *transition semantics* by Gunter [25]). In this approach, the main inductive predicate describes in more detail the execution of individual steps in a computation, with the overall computation roughly corresponding to the transitive closure of such small steps.

In general, the small-step style tends to require a greater number of rules that the big-step style, but this is outweighed by the fact that the small-step rules also tend to be simpler. The small-step style facilitates the description of interleaving ([36]).

Structural operational semantics is a specification formalism similar in some ways to rewriting logic, but more limited in its expressive capabilities. The two main differences (as presented in [29]) are the greater expressive power of rewriting logic due to the ability for rewriting modulo user-definable axioms, and the fact that rewriting logic is a full-fledged logic with both a proof and a model theory, whereas structural operational semantics accounts are only proof-theoretic.

Rewriting logic has proven to be a very good semantic framework for concurrency [32]. One important consideration is that, from a computational point of view, rewriting logic deduction is intrinsically concurrent [29]. A wide variety of models of computations, including concurrent ones, can be naturally and directly expressed as rewrite theories in rewriting logic without any artificiality. By combining both equations and rules, rewriting logic semantic definitions unify both the semantic equations of equational semantics and the semantic rules of SOS [33].

Another result of this paper refers to a second style of operational semantics of P systems resulted from translating the previously given operational semantics into a rewriting logic theory specified in Maude. This translation is provided with an operational correspondence proof [2].

Regarding the big-step and small-step semantics, P systems are described in a big-step style, and this is due to the parallel nature of this model. However, each operational step of a P system is based on the contributions of its components: the ingredients of each membrane (objects, messages, dissolving symbols, evolution rules) and the membrane structure of the configurations. Translating the natural semantics of P systems into rewriting logic, we get a certain refinement of the operational description similar to the small-step like.

By using an efficient implementation of rewriting logic as Maude, the formal specification of P systems is automatically transformed into an interpreter (hence, it is executable). Moreover, Maude provides useful formal analysis tools like the `search` command, a semi-decision procedure for finding failures of safety properties, and the Maude's LTL checker, a decision procedure for LTL properties of finite-state programs.

The paper is organized as follows.

Chapter 2 starts with the multiset notations used in this paper, then it reviews the most important features of membrane computing and some types of P systems, with more details on the transition P systems.

In Chapter 3 we introduce a natural semantics for P systems. We begin with an abstract syntax for P systems, then we present the three sets of inference rules corresponding to an evolution step accompanied by the corresponding correctness proofs.

Chapter 4 begins with a survey of the order-sorted equational logic - a sublogic of rewriting logic. We continue with basic features of rewriting logic and its expressivity, and we review three languages based on rewriting logic, with emphasis on Maude. The chapter ends with an introduction of two formal analysis tools for concurrent specification provided for free by Maude: the `search` command and the LTL model checker.

Chapter 5 presents the implementation of P systems in Maude, the algebraic structure, equations and rules, which provides a structural operational semantics for P systems. The implementation is illustrated by two examples of P systems: a deterministic one computing n^2 for a given $n \geq 1$, and a non-deterministic one computing values of the form n^2 for $n = 1, 2, 3, ...$ Besides executing the resulted specifications, we illustrate also the use of the two formal analysis tools presented in Chapter 4.

In Chapter 6 we prove the operational correspondence between the two operational semantics given in Chapter 3 and Chapter 5.

Chapter 7 draws some conclusions and sketches some future work directions.

Chapter 2

Membrane Computing

2.1 Some General Prerequisites

We denote by $\mathbf{N}$ the set of natural numbers, and by $[n]$ the set $\{1, \ldots, n\}$, for every $n \geq 1$.

A *multiset* over an arbitrary set U is a mapping $M : U \longrightarrow \mathbf{N}$ associating to each element a a finite natural number $M(a)$ representing its *multiplicity* in the multiset. The *support* of a multiset M is the set $supp(M) = \{a \in U \mid M(a) > 0\}$. A multiset M is described by the set $\{(a, M(a)) \mid a \in supp(M)\}$. Therefore a multiset is finite when its support is finite, and it is empty when its support is empty; the empty multiset is denoted by $\emptyset$.

Among the operations on multisets we are interested in the *union*. Let $M_1, M_2 : U \to \mathbf{N}$ be two multisets. The union of M_1 and M_2 is the multiset $M_1 \uplus M_2 : U \to \mathbf{N}$ defined by $(M_1 \uplus M_2)(a) = M_1(a) + M_2(a)$, for all $a \in U$, with $supp(M_1 \uplus M_2) = supp(M_1) \cup supp(M_2)$. It is easy to see that the multisets $M_1 \uplus M_2$ and $M_2 \uplus M_1$ coincide on their common domain, and that $M_1 \uplus \emptyset$ and M_1 also coincide. Therefore the union operation on multisets is commutative, and it has as identity element the empty multiset.

An *alphabet* is a finite non-empty set of abstract symbols. Let O be an alphabet of objects. Then we denote by O^* *the free monoid generated by* O with the concatenation operation (denoted by .) obeying the axioms of associativity and identity element (let this be *empty*). Moreover we consider *the free commutative monoid generated by* O denoted by O_c^* with the elements called *strings*.

If we denote by $[O \to \mathbf{N}]_f$ the set of finite multisets on O, then $([O \to \mathbf{N}]_f, \uplus, \emptyset)$ is a free generated commutative monoid. There is an obvious biunivocal correspondence between the two free generated commutative monoids $([O \to \mathbf{N}]_f, \uplus, \emptyset)$ and $(O_c^*, ., empty)$. Consequently, hereinafter we represent a multiset of objects as a string from O_c^*, where O is an alphabet of objects.

2.2 Membrane Computing at a Glance

Membrane computing is an area of computer science which tries to abstract computing models and computing paradigms from the structure and the functioning of living cells. It deals with an abstract model of parallel and distributed computing inspired by cell compartments and molecular membranes - a membrane system (hereinafter called P system). A cell is divided in various compartments, each compartment with its own multisets of objects and a different task, and all of them working simultaneously and cooperating (by means of communication) to accomplish a more general task of the whole system. P systems provide a nice abstraction for parallel systems, and a suitable framework for distributed and parallel algorithms [9].

The essential ingredient of a P system is its membrane structure. This can take two forms: (1) a hierarchical arrangement of membranes, like in a cell (hence described by a tree), (2) a net of membranes, like in a tissue or a neural net (corresponding to a graph-like structure). The rôle of a membrane is to separate two regions and to allow (a certain type of) communication between them.

Membrane computing is rather young research area; the paper that introduced it is [40]. It is important to remark that membrane computing was not conceived to provide models to biology; but even if the current research on this area still does not provide such kinds of models, the recent developments and increasing interest promise good results in this direction.

But although membrane computing does not (yet) provide models to biology, it is considered a part of *natural computing*. The justification consists in its attempt to provide to computer science new useful ideas, models, paradigms from the way living cells "compute". Neural networks, genetic algorithms, and DNA computing are three areas of natural computing already well established (in the first two cases also with proved practical usefulness). Membrane computing can be seen as an extension of DNA (more generally, molecular) computing, from the "one-processor" level to a distributed computing model. However, nature computes not only at the neural or genetic level, but also at the cellular level. None of the domains enumerated above considers the cell itself as its main object of research, nor it pays any attention to membranes and compartmentalization. And this is where membrane computing comes into stage.

2.3 The General Structure and Functioning of a Living Cell

In this section we present the cell from the biologist point of view as in [41], while in the next section we continue with the computer scientist point of view.

Everything alive consists of cells or has to do in a direct way with cells. The cell is the smallest "thing" unanimously considered *alive*. It is very small and very intricate in its structure and functioning, it has an elaborate internal activity and an exquisite interaction with the neighboring cells, and with the environment in general. It is fragile and robust at the same time, with a way to organize and control the bio-chemical and informational processes which was polished during billions of years of evolution.

Any cell means membranes. The cell itself is defined (separated from its environment) by a membrane, the external one. Inside the cell, several membranes enclose "protected reactors", compartments where specific biochemical processes take place. In particular, a membrane encloses the nucleus (of eukaryotic cells), where the genetic material is placed. Through vesicles enclosed by membranes one can transport packages of molecules from a part of the cell to other parts of the cell - in such way that the transported molecules are not "aggressed" during their journey by neighboring chemicals.

Then, the membranes allow a selective passage of substances among the compartments delimited by them. This can be a simple selection by size, in the case of small molecules, or a much more intricate selection, through protein channels, which not only select, but can also move molecules from a low concentration to a higher concentration, perhaps coupling molecules, through so-called symport and antiport processes.

The membranes of a cell do not only delimit compartments where the specific reactions take place in solution, hence *inside* the compartments, but many reactions in a cell develop *on the membranes*, catalyzed by the many proteins bound on them. It is said that when a compartment is too large for the local biochemistry to be efficient, life creates membranes, both in order to create smaller "reactors", and in order to create further "reaction surfaces".

There are cells living alone (unicellular organism, such as ciliates, bacteria, etc.), but in general the cells are organized in tissues, organs, organisms, communities of organisms. All these suppose a specific organization, starting with the direct communication/cooperation among neighboring cells, and ending with the interaction with the environment, at various levels. Together with the internal structure and organization of the cell, all these suggest a lot of ideas, exciting from a mathematical point of view, and potentially useful from a computability point of view. Part of them are already explored in membrane computing.

2.4 The Structure of Cell-Like Systems

A detailed description of the P systems can be found in [39]. A *P system* consists of a hierarchy of membranes that do not intersect, with a distinguishable membrane, called the *skin membrane*, surrounding them all. A membrane without any other membranes inside it is said to be *elementary*, while a non-elementary membrane it is said to be a *composite* membrane. The membranes produce a compartmentalization consisting of *regions*. For each membrane there is a unique associated region: the space delimited from above by it and from below by the membranes placed directly inside, if any exists. The space outside the skin membrane is called the *outer region* (or the *environment*). Because of this one-to-one correspondence we sometimes use membrane instead of region.

The membranes (and the corresponding regions) are uniquely labelled using a given set, usually ranging from 1 (for the skin) to the total number of membranes (with 0 for the outer region); but the labels can be more informative names.

Graphically, a membrane structure is represented by a Venn diagram in which two sets can be either disjoint, or one is a subset of the other. This is illustrated in Figure 2.1:

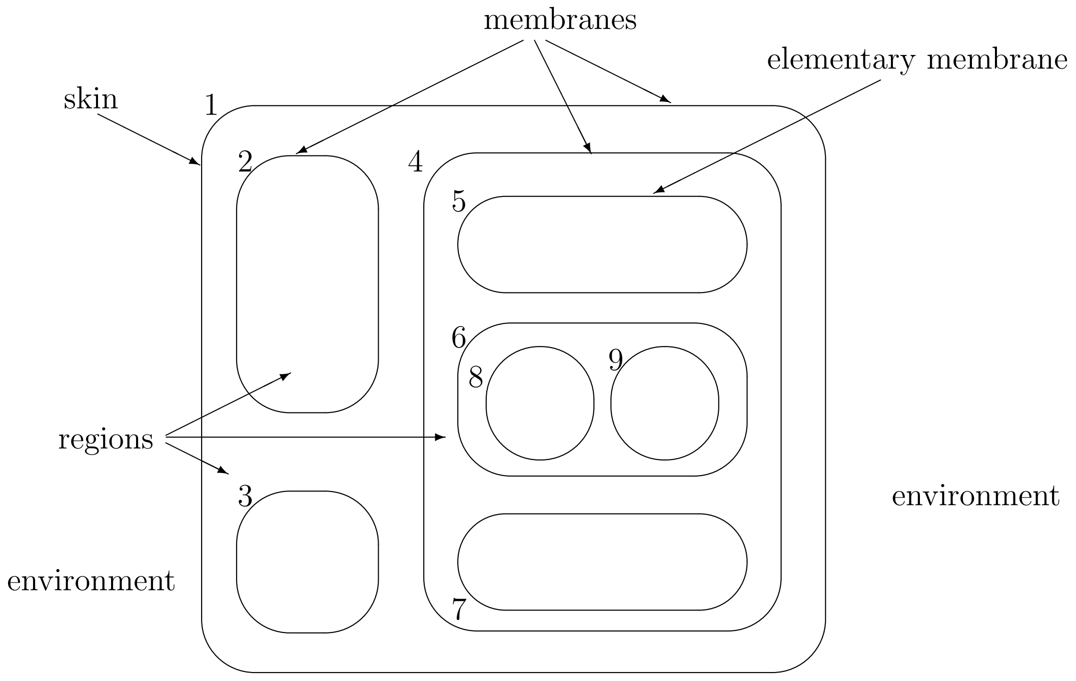

Figure 2.1: A membrane structure

Clearly, a hierarchical structure of membranes can be represented by a tree with skin as its root and elementary membranes as leaves. Figure 2.2 gives a tree which describes the membrane structure from Figure 2.1. This tree-like representation suggests the usage of graph-theoretic notions such as the distance in the tree, the level of a membrane, the height/depth of the membrane structure, as well as terminology, such as parent/child membrane, ancestor, etc.

Another representation of a hierarchical structure of membranes is given by a string of labelled matching parentheses; each pair of matching parentheses corresponds to a membrane. For instance, a string corresponding to the structure from Figure 2.1 is the following one:

$$[_1\ [_2\ ]_2\ [_3\ ]_3\ [_4\ [_5\ ]_5\ [_6\ [_8\ ]_8\ [_9\ ]_9\ ]_6\ [_7\ ]_7\ ]_4\ ]_1.$$

The Venn diagrammatic-like representation makes clear that the order of sibling membranes is irrelevant (as they float around), while, on the contrary, the inclusion relationship (or parent-child relationship in the tree-like representation) between membranes is essential. This is the reason for saying that Figure 2.2 gives “a” tree repre-

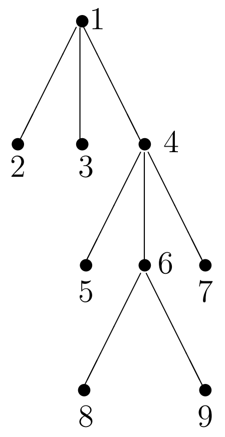

Figure 2.2: A tree describing the membrane structure from Figure 2.1

sentation for the given membrane structure; while for the parenthesized representation, the following string is also correct:

$$[_1\ [_3\ ]_3\ [_4\ [_6\ [_8\ ]_8\ [_9\ ]_9\ ]_6\ [_7\ ]_7\ [_5\ ]_5\ ]_4\ [_2\ ]_2\ ]_1.$$

The compartments of a cell contains substances (ions, small molecules, macromolecules) swimming in an aqueous solution; there is no ordering there, everything is closed to everything, the concentration matters, i.e., the number of copies of each molecule. Hence we work with sets of objects whose multiplicities matters - *multisets*.

A multiset can be represented in many ways; but, as we presented in the beginning of this chapter, we use the string representation.

It is important to emphasize that membrane computing deals with models which are intrinsically *discrete* (hence working with multisets of objects of finite support) and evolve through *rewrite-like* rules.

We have given a description of P systems in the general form, without specifying the type of rules. Depending of the way rules are applied and their form there are three central types of cell-like P systems considered in membrane computing: transition P systems, P systems with symport/antiport rules, and P systems with active membranes.

2.5 Transition P Systems

Only rules in a region delimited by a membrane act on the objects in that region. The multisets of objects from a region correspond to the "chemicals swimming in the solution in the cell compartment", while the rules correspond to the "chemical reactions possible in the same compartment". The rules must contain target indications, specifying the membrane where objects are sent after applying the rule. The objects can remain in the same region when they have a *here* target, or they can pass through membranes, in two directions: they can be sent *out* of the membrane which delimits a region from outside, or can be sent *in* one of the membranes which delimit a region from inside, precisely identified by its label. In a step, the objects can pass only through one membrane. We say that objects are enclosed in *messages* together with the target indication. Therefore we have *here* messages of typical form $(w, here)$ with w a possibly empty multiset of objects, *out* messages of typical form (w, out), and *in* messages of typical form (w, in_L), both with w a non-empty multiset of objects.

For the sake of simplicity, in the following of this paper we consider that the messages with the same target indication merge into one message, such that
$(w_1, here) \dots (w_n, here) = (w, here)$,
$(w_1, in_L) \dots (w_n, in_L) = (w, in_L)$, and
$(w_1, out) \dots (w_n, out) = (w, out)$,
where $w = w_1 \dots w_n$.

A membrane is *dissolved* if the symbol δ resulted from a rule application. When such an action takes place, the membrane disappears, its contents (objects and membranes) remain free in the membrane placed immediately outside, and the evolution rules of the dissolved membranes are lost. The skin membrane is never dissolved. The application of evolution rules is done in parallel, and it is eventually regulated by *priority* relationships between rules.

Formally, a *transition P system* is a structure

$$\Pi = (O, \mu, w_1, \dots, w_m, (R_1, \rho_1), \dots, (R_m, \rho_m), i_o),$$

where:

(i) O is an alphabet of objects;

(ii) μ is a membrane structure;

(iii) w_i are the initial multisets over O associated with the regions defined by μ;

(iv) R_i are finite sets of evolution rules over O associated with the membranes, of typical form $u \to v$, with u a non-empty multiset of objects, and v a non-empty multiset of messages and/or the dissolving symbol δ;

(v) ρ_i is a partial order relation over R_i, specifying a *priority* relation among the rules: $(r_1, r_2) \in \rho_i$ iff $r_1 > r_2$ (i.e., r_1 has a higher priority than r_2);

(vi) i_0 is either a number between 1 and m specifying the *output* membrane of Π, or it is equal to 0 indicating that the output is the outer region.

Observation. Hereinafter when we say *P systems*, we refer to *transition P systems* if no other specific type is specified (like symport/antiport, with active membranes, etc.)

The total number of membranes in a P systems gives its *degree*; hence Π is a P system of degree m.

Since the skin is not allowed to be dissolved, we consider that the rules of the skin do not involve δ.

In order to describe a membrane we consider its label and the current multiset of objects together with its structure. We use the mappings *rules* and *priority* to associate to a membrane label the set of evolution rules and the priority relation : $rules(L_i) = R_i$, $priority(L_i) = \rho_i$, and the projections L and w which return from a membrane its label and its current multiset, respectively.

Notation. If X is a set, then X_c^* denotes the set of the finite multisets defined over X, and X_c^+ denotes X_c^* without the empty multiset. This notations are inspired by the one-to-one correspondence from the set of the finite multisets defined over X onto the free commutative monoid generated by X.

Formally, the set of *membranes for a P system* Π, denoted by $\mathcal{M}(\Pi)$, and *the membrane structure* are inductively defined as follows:

- if L is a label, and w is a multiset over $O \cup (O_c^* \times \{here\}) \cup (O_c^+ \times \{out\}) \cup \{\delta\}$, then $\langle\, L \mid w \,\rangle \in \mathcal{M}(\Pi)$; $\langle\, L \mid w \,\rangle$ is called *simple (or elementary) membrane*, and it has the structure $\langle\rangle$;
- if $M_1, \ldots, M_n \in \mathcal{M}(\Pi)$ with $n \geq 1$, the structure of M_i is μ_i for all $i \in [n]$, L is a label, w is a multiset over $O \cup (O_c^* \times \{here\}) \cup (O_c^+ \times \{out\}) \cup (O_c^+ \times \{in_{L(M_j)} | j \in [n]\}) \cup \{\delta\}$, then $\langle\, L \mid w\,;\, M_1, \ldots, M_n \,\rangle \in \mathcal{M}(\Pi)$; $\langle\, L \mid w\,;\, M_1, \ldots, M_n \,\rangle$ is called *a composite membrane*, and it has the structure $\langle \mu_1, \ldots, \mu_n \rangle$.

A finite multiset of membranes is usually written as $M_1, \ldots, M_n$. We denote by $\mathcal{M}^+(\Pi)$ the set of non-empty finite multisets of membranes. The union of two multisets of membranes $M_+ = M_1, \ldots, M_m$ and $N_+ = N_1, \ldots, N_n$ is written as $M_+, N_+ = M_1, \ldots, M_m, N_1, \ldots, N_n$. An element from $\mathcal{M}^+(\Pi)$ is either a membrane, or a set of sibling membranes.

A *committed configuration* for a P system Π is a skin membrane which has no messages and no dissolving symbol δ, i.e., the multisets of all regions are elements in O_c^*. We denote by $\mathcal{C}(\Pi)$ the set of committed configurations for Π, and it is a proper subset of $\mathcal{M}(\Pi)$. We have $C \in \mathcal{C}(\Pi)$ iff C is the skin membrane of Π, and $w(M)$ is a multiset over O for each membrane M in $\mathcal{C}$.

An *intermediate configuration* is a skin membrane in which we have messages or the dissolving symbol δ. The set of intermediate configurations is denoted by $\mathcal{C}^\#(\Pi)$. We have $C \in \mathcal{C}^\#(\Pi)$ iff C is the skin membrane of Π such that there is a membrane M in C with $w(M) = w'w''$, $w' \in (Msg(O) \cup \{\delta\})_c^+$, and $w'' \in O_c^*$. By $Msg(O)$ we denote the set $(O^* \times \{here\}) \cup (O^+ \times \{out\}) \cup (O^+ \times \{in_{L(M)}\})$, with the mention that a message in_L is allowed to appear in a region only if the membrane labelled by L is included in that region. $\mathcal{C}^\#(\Pi)$ is also a proper subset of $\mathcal{M}(\Pi)$.

A *configuration* is either a committed configuration or an intermediate configuration.

Each P system has an initial committed configuration which is characterized by the initial multiset of objects for each membrane and the initial membrane structure of the system. The membranes preserve the initial labelling, evolution rules and priority relation among them in all subsequent configurations.

For two configurations C_1 and C_2 of Π, we say that there is a *transition* from C_1 to C_2, and write $C_1 \Rightarrow C_2$, if the following *steps* are executed in the given order:

1. *maximal parallel rewriting step*, consisting of non-deterministically assigning objects to evolution rules in every membrane and executing them in a maximal parallel manner;
2. *parallel communication of objects through membranes*, consisting in sending existing messages;
3. *parallel membrane dissolving*, consisting in dissolving the membranes containing δ.

The last two steps take place only if there are messages or δ symbols resulted from the first step, respectively. If the first step is not possible, consequently neither the other two steps, then we say that the system has reached a *halting configuration* if the output region i_0 still exists in this configuration (in the case when i_0 is the label of a membrane, it can be dissolved during the computation). In this case we say that the computation is *successful*.

We can associate a result to a successful computation in various ways. If we have an output region specified, and this is an internal region, then we have an internal output: we count the objects present in the output region in the halting configuration and this number is the result of the computation. When we have $i_0 = 0$, we count the objects which leave the system during the computation, and this is called external output. In both cases the result is a number. If we distinguish among different objects, then we can have as the result a vector of natural numbers. The objects which leave the system can also be arranged in a sequence according to the moments when they exit the skin membrane, and in this case the result is a string. Non-halting computations provide no output, and if the output membrane is dissolved during the computation, then the computation aborts.

In our approach, the output region corresponds to an internal one, and the rules from the output region do not involve the dissolving symbol δ.

In chapter 3 we present an operational semantics of the P systems, considering each of the three steps.

Example 2.1 Consider the P system of degree 4

$$\begin{array}{rcl}
\Pi_1 & = & (O, \mu, w_1, w_2, w_3, w_4, (R_1, \rho_1), (R_2, \rho_2), (R_3, \rho_3), (R_4, \rho_4), 4), \\
O & = & \{a, c, d, e, f\}, \\
\mu & = & [_1[_2[_3]_3[_4]_4]_2]_1 \\
w_1 & = & \lambda, R_1 = \emptyset,\ \rho_1 = \emptyset, \\
w_2 & = & \lambda,\ \rho_2 = \{(r_1, r_2)\}, \\
R_2 & = & \{d \to (c, here), c \to (a, here), a \to (a, here)(e, in_4), \\
 & & r_1 : ff \to (f, here), r_2 : f \to \delta\}, \\
w_3 & = & a^n cf,\ R_3 = \{r_1 : ca \to (cd, here), r_2 : c \to \delta, f \to (ff, here)\},\ \rho_3 = \{(r_1, r_2)\}, \\
w_4 & = & \lambda, R_4 = \emptyset,\ \rho_4 = \emptyset
\end{array}$$

The initial configuration is given in Figure 2.3.

There are no objects in the membranes 1, 2 and 4. The only possibility is to start in membrane 3. Using the rules $ca \to (cd, here)$ and $f \to (ff, here)$, in parallel for all

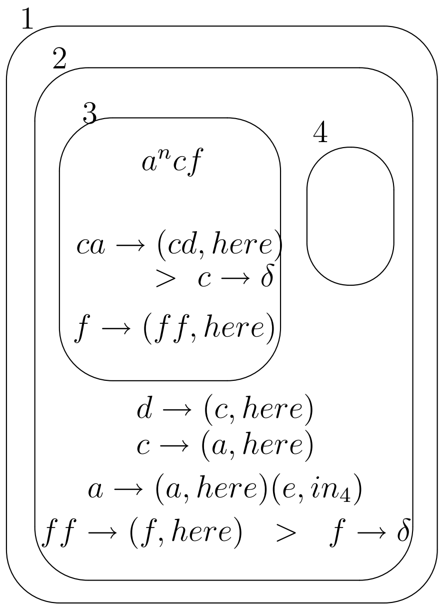

Figure 2.3: The initial configuration of a deterministic P system computing n^2, $n \geq 1$

occurrences of a,c and f currently available, after n steps we get the multiset $cd^n f^{2^n}$. During the next step we can only use the rules $c \to \delta$ and $f \to (ff, here)$: the only occurrence of c is deleted, the number of occurrences of f is doubled, and δ occurs in membrane 3; hence region 3 disappears, its rules are lost, and its objects, $d^n f^{2^{n+1}}$, are moved to region 2; therefore the computation continues in membrane 2. According to the priority relation the rule $ff \to (f, here)$ is used as much as possible. The rules $d \to (c, here)$ and $ff \to (f, here)$ are applied in parallel for n times, and we get the multiset $c^n f^{2^n}$. Then the rules $c \to (a, here)$ and $ff \to (f, here)$ are applied in parallel for n times, resulting the multiset $a^n f^{2^{n-1}}$. Now we can apply in parallel for $(n-1)$ times the rules $a \to (a, here)(e, in_4)$ and $ff \to (f, here)$: during each step the number of a's remains constantly equal to n, the number of f's is halved, and n copies of e are sent in membrane 4. Hence after these $(n-1)$ steps, we get $a^n f$ in membrane 2 and $e^{n(n-1)}$ in membrane 4. During the next step the rule $a \to (a, here)(e, in_4)$ is applied n times (sending n copies of e in membrane 4), and $f \to \delta$ one time (dissolving membrane 2). The obtained configurations corresponds to the skin membrane which contains the multiset a^n, and the output membrane 4 with the multiset e^{n^2}. There are no rules in membranes 1 and 4, hence the system cannot evolve any further. Therefore this is a halting configuration, and the result of the computation is exactly the number of objects from the output membrane 4, that is n^2.

2.6 Other Types of Cell-Like P Systems

There are three central types of cell-like P systems considered in membrane computing, each of them allowing many possible extensions with most spectacular features. Details on these features can be found in [41].

An important part of the cell activity is related to the passage of substances through membranes. This operation is formalized by two types of rules:

- *symport rules* of the form (w, in), (w, out), and
- *antiport rules* of the form $(w_1, out; w_2, in)$,

where w, w_1, w_2 are multisets of arbitrary size.

A natural question rises on the computability power of this type of communication, that is the power of systems with only symport and antiport rules. Here the P systems with symport/antiport rules come into stage. Their definition has many things in common with the definition of transition P systems. In the following we explain the differences.

A *P system with symport/antiport rules* is a construct of the form:

$$\Pi = (O, \mu, w_1, \ldots, w_m, E, R_1, \ldots, R_m, i_o).$$

E is a set of objects ($E \subseteq O$) which are present in the environment in arbitrarily many copies. Because objects are only moved across membranes and a computation starts with a finite multiset of objects present in the system, the number of objects necessary for the computation cannot be increased if there is no way to supply objects; this can be done by considering the set E. The environment is supposed inexhaustible, hence the objects from E are supposed inexhaustible, irrespective how many of them are brought into the system, arbitrarily many remain outside.

The rules from R can be of two types, symport rules and antiport rules, of the forms specified above. As in the case of transition P systems, the rules are used in the non-deterministic maximally parallel manner. But now the rules are associated with the membranes, not with the regions, because each rules governs the communication through a specific membrane.

The P systems with symport/antiport rules have a series of attractive characteristics: they are fully based on biological types of multiset processing rules; the environment takes a direct part into the evolution of the system; the computation is done only by communication, no object is changed, the objects only move across membranes; no object is created or destroyed, hence the conservation law is fulfilled.

The membranes play an important role in the reactions which take place in a cell, and, moreover, they can evolve themselves, either changing their characteristics or even getting divided. This idea motivated the formalization of the third type of cell-like P system, the *P systems with active membranes.* In this case, membranes have electrical polarizations, objects evolve also in the maximally parallel manner as in the other two types of P systems, and the developmental rules are divided in object evolution rules, in and out communication rules, dissolving rules, and division rules for elementary membranes.

An important feature that should be noted for P systems with active membranes is that the membrane structure evolves during the computation not only by decreasing the number of membranes, due to dissolving rules, but also by increasing the number of membranes, by division.

Having a dynamical membrane structure allows not only with membrane dissolving and membrane division (like in the case of P systems with active membranes), but also operations on membranes like creation, merging, endocytosis, exocytosis, or gemmation.

Another interesting feature is provided by the structure of objects which we ignore. This leads for example to *splicing P systems*, systems that evolve by means of rules inspired by DNA computing.

We have presented succinctly some features of cell-like P system classes, many of them constituting very interesting exercises (or challenges) for formalization in the same manner as in the following chapters.

2.7 The Power of P Systems

The variety of suggestions from biology and the range of possibilities to define the architecture and the functioning of a membrane-based-multiset-processing device are practically endless, and already the literature of membrane computing contains a very large number of models. Membrane computing is not a theory related to a specific model, it is a framework for devising compartmentalized models. There are already proposed many types of P systems, and the flexibility and the versatility of P systems seem to be, in principle, unlimited.

The initial goal of membrane computing was to define computability models inspired from the cell biology, and indeed a large part of the investigations in this area was devoted to producing computing devices and examining their computing power, in comparison with the standard models in computability theory, Turing machines and their restricted variants. It has already been proved that most of the considered classes of P systems are equal in power with Turing machines, i.e, they are computationally complete (or Turing complete). Because these proofs are always constructive, universal P systems are easy to be obtained - P systems able to simulate any other P system of the given type, after introducing a "code" of the particular systems as an input in the universal one.

All classes of systems presented in [41] (hence including the ones presented in the previous section) are known to be universal. Thus the cell turns out to be a very powerful "computer", both when standing alone and in tissue.

In general, for P systems working with symbol-objects, these universality results are proved by simulating computing devices which are known to be universal, and which either work with numbers or do not essentially use the positional information from strings. This is possible for register machines, matrix grammars in the binary normal form, programmed grammars, regularly controlled grammars, graph-controlled grammars (but not for arbitrary Chomsky grammars and for Turing machines, which can be used only in the case of string-objects).

In [40] a proof of the computational completeness for transition P systems is presented using matrix grammars with appearance checking, while in [41] the same result is presented for P systems with symport/antiport rules by means of simulating register machines in a surprisingly simple manner.

Chapter 3

P Systems Operational Semantics: A Big-Step Approach

Structural operational semantics descriptions of systems start from abstract syntax. Specifications of abstract syntax introduce symbols for syntactic sets, meta-variables ranging over those sets, and notation for constructor functions. Some of the syntactic sets are usually regarded as basic, and left open or described only informally. The abstract syntax for P systems is given as follows:

Objects	$o \in O$
Multisets of objects	$w \in O_c^*$
Labels	$L \in \{Skin\} \cup \mathcal{L}$
Messages	$(w, here), (w, in_L), (w, out) \in Msg(O)$
Dissolving symbol	δ
Membrane resources	$w \in (O \cup Msg(O) \cup \{\delta\})_c^*$
Membranes	$M \in \mathcal{M}(\Pi),\ M ::= \langle L \mid w \rangle \ \mid\ \langle L \mid w; M_+ \rangle$
Sibling membranes	$M_+ \in \mathcal{M}^+(\Pi) = \mathcal{M}(\Pi)_c^+$
Committed configurations	$C \in \mathcal{C}(\Pi)$
Intermediate configurations	$C \in \mathcal{C}^{\#}(\Pi)$
Configurations	$C \in \mathcal{C}(\Pi) \cup \mathcal{C}^{\#}(\Pi)$

In structural operational semantics, the evolution of systems is modelled by a transition system specified inductively, by rules. The transition system for a P system Π is intuitively defined as follows. For two committed configurations C_1 and C_2 of Π, we say that there is a *transition* from C_1 to C_2, and write $C_1 \Rightarrow C_2$, if the following *steps* are executed in the following given order:

1. *the maximal parallel rewriting step*, consisting of non-deterministically assigning objects to evolution rules in every membrane, and executing them in a maximal parallel manner;

2. *the parallel communication of objects through membranes*, consisting in sending the existing messages;

3. *the parallel membrane dissolving*, consisting in dissolving the membranes which contain the δ symbol.

The last two steps take place only if there are messages or δ symbols resulted from the first step, respectively. If the first step is not possible, consequently neither the other two steps, then we say that the system has reached a *halting configuration.* A halting configuration is always a committed one.

Next we present in detail each of the three steps.

3.1 Maximal Parallel Rewriting Step

We can pass from a configuration to another one by using the evolution rules. This is done in parallel: all objects, from all membranes, which can be the subject of local evolution rules, as prescribed by the priority relation, should evolve simultaneously. The rules of each membrane are using its current objects as much as this is possible in a parallel and non-deterministic way. However, an object introduced by a rule cannot evolve at the same step by means of another rule. The use of a rule $u \to v$ in a region with a multiset w means to subtract the multiset identified by u from w, and then to add the objects of v according to the form of the rule.

We denote the maximal parallel rewriting on membranes by $\stackrel{mpr}{\Longrightarrow}$ and by $\stackrel{mpr}{\Longrightarrow}_L$ the maximal parallel rewriting for a multiset of objects of the membrane labelled by L.

Definition 3.1 *The irreducibility property w.r.t. the maximal parallel rewriting relation for multisets of objects, messages, and δ, for membranes, and for sets of sibling membranes is defined as follows:*

- *a multiset w consisting only of objects is* L-**irreducible** *if there are no rules in $rules(L)$ applicable to w w.r.t. the priority relation $priority(L)$;*
- *a multiset containing at least a message or the dissolving symbol δ is* L-**irreducible***;*
- *a simple membrane $\langle\, L \mid w \,\rangle$ is* **mpr-irreducible** *if w is L-irreducible;*
- *a non-empty set of sibling membranes $M_1, \ldots, M_n$ is* **mpr-irreducible** *if M_i is mpr-irreducible, for every $i \in [n]$;*
- *a composite membrane $\langle L \,|\, w\,;\, M_1, \ldots, M_n \rangle$, with $n \geq 1$, is* **mpr-irreducible** *if w is L-irreducible, and the set of sibling membranes $M_1, \ldots, M_n$ is mpr-irreducible.*

Definition 3.2 *Let M be a membrane labelled by L, and w a multiset of objects. A non-empty multiset $R = (u_1 \to v_1, \ldots, u_n \to v_n)$ of evolution rules is* (L, w)-**consistent** *if:*

- *$R \subseteq rules(L)$,*

- *$w = u_1 \ldots u_n z$, so each rule $r \in R$ is applicable on w,*

- *$(\forall r \in R, \forall r' \in rules(L))$ r' applicable on w implies $(r', r) \notin priority(L)$,*
- *$(\forall r', r'' \in R)$ $(r', r'') \notin priority(L)$,*
- *the dissolving symbol δ has at most one occurrence in the multiset $v_1 \dots v_n$.*

The **maximal parallel rewriting relation** $\overset{mpr}{\Longrightarrow}$ is defined by the following inference rules:

For each $w \in O_c^+$ and L-irreducible $z \in O_c^*$ such that $w = u_1 \dots u_n z$, and $(u_1 \to v_1, \dots, u_n \to v_n)$ (L, w)-consistent,

$$(\mathbf{R_1})\ \frac{}{w \overset{mpr}{\Longrightarrow}_L v_1 \dots v_n z}$$

For each $w \in O_c^+$, $w' \in (O \cup Msg(O) \cup \{\delta\})_c^+$,

$$(\mathbf{R_2})\ \frac{w \overset{mpr}{\Longrightarrow}_L w'}{\langle\, L \mid w \,\rangle \overset{mpr}{\Longrightarrow} \langle\, L \mid w' \,\rangle}$$

For each $w \in O_c^+$, $w' \in (O \cup Msg(O) \cup \{\delta\})_c^+$, $M_+, M'_+ \in \mathcal{M}^+(\Pi)$,

$$(\mathbf{R_3})\ \frac{w \overset{mpr}{\Longrightarrow}_L w',\, M_+ \overset{mpr}{\Longrightarrow} M'_+}{\langle\, L \mid w\, ;\ M_+ \,\rangle \overset{mpr}{\Longrightarrow} \langle\, L \mid w'\, ;\ M'_+ \,\rangle}$$

For each $w \in O_c^+$, $w' \in (O \cup Msg(O) \cup \{\delta\})_c^+$, and mpr-irreducible $M_+ \in \mathcal{M}^+(\Pi)$,

$$(\mathbf{R_4})\ \frac{w \overset{mpr}{\Longrightarrow}_L w'}{\langle\, L \mid w\, ;\ M_+ \,\rangle \overset{mpr}{\Longrightarrow} \langle\, L \mid w'\, ;\ M_+ \,\rangle}$$

For each L-irreducible $w \in O_c^*$, $M_+, M'_+ \in \mathcal{M}^+(\Pi)$,

$$(\mathbf{R_5})\ \frac{M_+ \overset{mpr}{\Longrightarrow} M'_+}{\langle\, L \mid w\, ;\ M_+ \,\rangle \overset{mpr}{\Longrightarrow} \langle\, L \mid w\, ;\ M'_+ \,\rangle}$$

For each $M, M' \in \mathcal{M}(\Pi)$, $M_+, M'_+ \in \mathcal{M}^+(\Pi)$,

$$(\mathbf{R_6})\ \frac{M \overset{mpr}{\Longrightarrow} M',\, M_+ \overset{mpr}{\Longrightarrow} M'_+}{M, M_+ \overset{mpr}{\Longrightarrow} M', M'_+}$$

For each $M, M' \in \mathcal{M}(\Pi)$, and mpr-irreducible $M_+ \in \mathcal{M}^+(\Pi)$,

$$(\mathbf{R_7})\ \frac{M \overset{mpr}{\Longrightarrow} M'}{M, M_+ \overset{mpr}{\Longrightarrow} M', M_+}$$

Example 3.1 *We consider the P system described in example 2.1. Then the inference tree for*

$$\langle\, 1 \,|\, empty \,;\, \langle\, 2 \,|\, aaf \,;\, \langle\, 4 \,|\, ee \,\rangle\,\rangle\,\rangle \overset{mpr}{\Longrightarrow} \langle\, 1 \,|\, empty \,;\, \langle\, 2 \,|\, (aa, here)(ee, in_4)\delta \,;\, \langle\, 4 \,|\, ee \,\rangle\,\rangle\,\rangle$$

is:

$$(R_5)\,\dfrac{(R_4)\,\dfrac{(R_1)\,\dfrac{}{aaf \overset{mpr}{\Longrightarrow}_4 (aa, here)(ee, 4)\delta}}{\langle\, 2 \,|\, aaf \,;\, \langle\, 4 \,|\, ee \,\rangle\,\rangle \overset{mpr}{\Longrightarrow} \langle\, 2 \,|\, (aa, here)(ee, in_4)\delta \,;\, \langle\, 4 \,|\, ee \,\rangle\,\rangle}}{\langle\, 1 \,|\, empty \,;\, \langle\, 2 \,|\, aaf \,;\, \langle\, 4 \,|\, ee \,\rangle\,\rangle\,\rangle \overset{mpr}{\Longrightarrow} \langle\, 1 \,|\, empty \,;\, \langle\, 2 \,|\, (aa, here)(ee, in_4)\delta \,;\, \langle\, 4 \,|\, ee \,\rangle\,\rangle\,\rangle}$$

Lemma 3.1 *If $w \overset{mpr}{\Longrightarrow}_L w'$, then w' is L-irreducible.*

Proof: We get $w \overset{mpr}{\Longrightarrow}_L w'$ only applying (R_1) using an (L, w)-consistent multiset of rules. Then we have $w' = w''z$ such that $w'' \in (Msg(O) \cup \{\delta\})_c^+$, and $z \in O_c^*$ is L-irreducible. Then w' is L-irreducible by definition because it contains messages or δ.

Lemma 3.2 *If $M_+ \overset{mpr}{\Longrightarrow} M'_+$ then M'_+ is mpr-irreducible.*

Proof: Let M_+, M'_+ be two non-empty sets of membranes such that $M_+ \overset{mpr}{\Longrightarrow} M'_+$. We prove that M'_+ is mpr-irreducible by induction on the depth of the associated inference tree. We consider all possible cases for the final step of the inference:

(i) $M_+ \overset{mpr}{\Longrightarrow} M'_+$ is inferred by (R_2). Then $M_+ = \langle\, L \,|\, w \,\rangle$ and $M'_+ = \langle\, L \,|\, w' \,\rangle$ with $w \overset{mpr}{\Longrightarrow} w'$. By Lemma 3.1 w' is L-irreducible, therefore M'_+ is mpr-irreducible by definition.

(ii) $M_+ \overset{mpr}{\Longrightarrow} M'_+$ is inferred by (R_3). Then $M_+ = \langle L \,|\, w \,;\, N_+ \rangle$ and $M'_+ = \langle L \,|\, w' \,;\, N'_+ \rangle$ with $w \overset{mpr}{\Longrightarrow}_L w'$ inferred by (R_1), and $N_+ \overset{mpr}{\Longrightarrow} N'_+$ inferred by a shorter inference tree. By Lemma 3.1 w' is L-irreducible, and by inductive hypothesis, N'_+ is mpr-irreducible. Therefore M'_+ is mpr-irreducible by definition.

(iii) $M_+ \overset{mpr}{\Longrightarrow} M'_+$ is inferred by (R_4). Then $M_+ = \langle\, L \,|\, w \,;\, N_+ \,\rangle$ and $M'_+ = \langle L \,|\, w' \,;\, N_+ \rangle$ with $w \overset{mpr}{\Longrightarrow} w'$ (therefore w' is L-irreducible by Lemma 3.1) and N_+ mpr-irreducible. By definition we obtain that M'_+ is mpr-irreducible.

(iv) $M_+ \overset{mpr}{\Longrightarrow} M'_+$ is inferred by (R_5). Then $M_+ = \langle L \,|\, w \,;\, N_+ \rangle$ and $M'_+ = \langle L \,|\, w \,;\, N'_+ \rangle$ with w L-irreducible and $N_+ \overset{mpr}{\Longrightarrow} N'_+$ inferred by a shorter inference tree. By inductive hypothesis N'_+ is mpr-irreducible, therefore M'_+ is mpr-irreducible.

(v) $M_+ \overset{mpr}{\Longrightarrow} M'_+$ is inferred by (R_6). Then $M_+ = M, N_+$ and $M'_+ = M', N'_+$ where $M \overset{mpr}{\Longrightarrow} M'$ and $N_+ \overset{mpr}{\Longrightarrow} N'_+$ are inferred by shorter inference trees. By inductive hypothesis M' and N'_+ are mpr-irreducible, therefore M', N'_+ is mpr-irreducible.

(vi) $M_+ \overset{mpr}{\Longrightarrow} M'_+$ is inferred by (R_7). Then $M_+ = M, N_+$ and $M'_+ = M', N_+$ where $M \overset{mpr}{\Longrightarrow} M'$ is inferred by a shorter inference tree, and N_+ is mpr-irreducible. By inductive hypothesis M' is mpr-irreducible, therefore M', N_+ is mpr-irreducible by definition.

Theorem 3.1 *Let Π be a P system. If $C \in \mathcal{C}(\Pi)$ and $C \overset{mpr}{\Longrightarrow} C'$, then $C' \in \mathcal{C}^{\#}(\Pi)$ and C' is mpr-irreducible.*

The proof of the theorem follows easily from Lemma 3.2.

3.2 Parallel Communication of Objects

Communication through two membranes M_1 and M_2 can take place only if one is inside the other.

We say that a multiset w is *here-free*/*in$_L$-free*/*out-free* if it does not contain any *here*/*in$_L$*/*out* messages, respectively.

For w a multiset of objects and messages, we introduce the operations *obj*, *here*, *out*, and *in$_L$* as follows:

$$obj(w) \text{ is obtained from } w \text{ by removing all messages,}$$
$$here(w) = \begin{cases} empty & , \text{ if } w \text{ is } here\text{-free} \\ w'' & , \text{ if } w = w'(w'', here) \wedge w' \text{ is } here\text{-free} \end{cases}$$
$$out(w) = \begin{cases} empty & , \text{ if } w \text{ is } out\text{-free} \\ w'' & , \text{ if } w = w'(w'', out) \wedge w' \text{ is } out\text{-free} \end{cases}$$
$$in_L(w) = \begin{cases} empty & , \text{ if } w \text{ is } in_L\text{-free} \\ w'' & , \text{ if } w = w'(w'', in_L) \wedge w' \text{ is } in_L\text{-free} \end{cases}$$

We recall that all the messages with the same target merge in one message.

Definition 3.3 *The* **tar-irreducibility** *property for membranes and for sets of sibling membranes is defined as follows:*

1. *a simple membrane $\langle\, L \mid w \,\rangle$ is* **tar-irreducible** *iff $L \neq Skin \vee (L = Skin \wedge w$ is out-free);*

2. *a non-empty set of sibling membranes $M_1, \ldots, M_n$ is* **tar-irreducible** *iff M_i is tar-irreducible, for every $i \in [n]$;*

3. *a composite membrane $\langle\, L \mid w\,;\, M_1, \ldots, M_n \,\rangle$, $n \geq 1$, is* **tar-irreducible** *iff:*

   (a) $L \neq Skin \vee (L = Skin \wedge w$ is out-free),

   (b) w is $in_{L(M_i)}$-free, for every $i \in [n]$,

   (c) for all $i \in [n]$, $w(M_i)$ is out-free,

   (d) the set of sibling membranes $M_1, \ldots, M_n$ is tar-irreducible.

We consider $Skin$ to refer the label of the skin membrane.

Notation. We treat the messages of form $(w', here)$ as a particular communication inside their membranes consisting in substitution of $(w', here)$ by w'. We denote by $\overline{w}$ the multiset obtained by replacing $(here(w), here)$ by $here(w)$ in w. For instance, if $w = a\,(bc, here)\,(d, out)$ then $\overline{w} = abc\,(d, out)$, where $here(w) = bc$. We note that $in_L(\overline{w}) = in_L(w)$, and $out(\overline{w}) = out(w)$.

The **parallel communication relation** $\overset{tar}{\Longrightarrow}$ is defined by the following five inference rules:

For each tar-irreducible $M_1, \ldots, M_n \in \mathcal{M}^+(\Pi)$ and multiset w such that $here(w) \neq empty$, or $L = Skin \wedge out(w) \neq empty$, or it exists $i \in [n]$ with $in_{L(M_i)}(w) out(w(M_i)) \neq empty$ or $here(w(M_i)) \neq empty$,

$$(\mathbf{C_1})\frac{}{\langle\, L \mid w\, ;\, M_1, \ldots, M_n \,\rangle \overset{tar}{\Longrightarrow} \langle\, L \mid w'\, ;\, M'_1, \ldots, M'_n \,\rangle}$$

where

$$w' = \begin{cases} obj(\overline{w})\, out(w(M_1)) \ldots out(w(M_n)) & \text{, if } L = Skin \\ obj(\overline{w})\, (out(w), out)\, out(w(M_1)) \ldots out(w(M_n)) & \text{, otherwise} \end{cases}$$

and

$$w(M'_i) = obj(\overline{w(M'_i)})\, in_{L(M_i)}(w), \text{ for all } i \in [n]$$

For each $M_1, \ldots, M_n, M'_1, \ldots, M'_n \in \mathcal{M}^+(\Pi)$, and multiset w,

$$(\mathbf{C_2})\frac{M_1, \ldots, M_n \overset{tar}{\Longrightarrow} M'_1, \ldots, M'_n}{\langle\, L \mid w\, ;\, M_1, \ldots, M_n \,\rangle \overset{tar}{\Longrightarrow} \langle\, L \mid w''\, ;\, M''_1, \ldots, M''_n \,\rangle}$$

where

$$w'' = \begin{cases} obj(\overline{w})\, out(w(M'_1)) \ldots out(w(M'_n)) & \text{, if } L = Skin \\ obj(\overline{w})\, (out(w), out)\, out(w(M'_1)) \ldots out(w(M'_n)) & \text{, otherwise} \end{cases}$$

and each M''_i is obtained from M'_i by replacing its resources by

$$w(M''_i) = obj(\overline{w(M'_i)})\, in_{L(M'_i)}(w), \text{ for all } i \in [n]$$

For each multiset w such that $here(w)\, out(w) \neq empty$,

$$(\mathbf{C_3})\frac{}{\langle\, Skin \mid w \,\rangle \overset{tar}{\Longrightarrow} \langle\, Skin \mid obj(\overline{w}) \,\rangle}$$

For each $M, M' \in \mathcal{M}(\Pi)$, and tar-irreducible $M_+ \in \mathcal{M}^+(\Pi)$,

$$(\mathbf{C_4})\frac{M \overset{tar}{\Longrightarrow} M'}{M, M_+ \overset{tar}{\Longrightarrow} M', M_+}$$

For each $M \in \mathcal{M}(\Pi)$, $M_+ \in \mathcal{M}^+(\Pi)$,

$$(\mathbf{C_5})\frac{M \overset{tar}{\Longrightarrow} M', M_+ \overset{tar}{\Longrightarrow} M'_+}{M, M_+ \overset{tar}{\Longrightarrow} M', M'_+}$$

Example 3.2 *The inference tree for*

$\langle\, 1 \,|\, \mathit{empty}\,;\, \langle\, 2 \,|\, (aa, \mathit{here})(ee, \mathit{in}_4)\delta\,;\, \langle\, 4 \,|\, ee \,\rangle\,\rangle\,\rangle \overset{tar}{\Longrightarrow} \langle\, 1 \,|\, \mathit{empty}\,;\, \langle\, 2 \,|\, aa\delta\,;\, \langle\, 4 \,|\, eeee \,\rangle\,\rangle\,\rangle$

is:

$$(C_2)\frac{(C_1)\dfrac{}{\langle\, 2 \,|\, (aa, \mathit{here})(ee, \mathit{in}_4)\delta\,;\, \langle\, 4 \,|\, ee \,\rangle\,\rangle \overset{tar}{\Longrightarrow} \langle\, 2 \,|\, aa\delta\,;\, \langle\, 4 \,|\, eeee \,\rangle\,\rangle}}{\langle\, 1 \,|\, \mathit{empty}\,;\, \langle\, 2 \,|\, (aa, \mathit{here})(ee, \mathit{in}_4)\delta\,;\, \langle\, 4 \,|\, ee \,\rangle\,\rangle\,\rangle \overset{tar}{\Longrightarrow} \langle\, 1 \,|\, \mathit{empty}\,;\, \langle\, 2 \,|\, aa\delta\,;\, \langle\, 4 \,|\, eeee \,\rangle\,\rangle\,\rangle}$$

Lemma 3.3 *If $M_+ \overset{tar}{\Longrightarrow} M'_+$, then M'_+ is tar-irreducible.*

Proof: Let M_+, M'_+ be two non-empty sets of membranes such that $M_+ \overset{tar}{\Longrightarrow} M'_+$. We prove that M'_+ is tar-irreducible by induction on the depth of the associated inference tree. We consider all possible cases for the final step of the inference, i.e., each of the five rules for communication:

(i) $M_+ \overset{tar}{\Longrightarrow} M'_+$ is inferred by (C_1). Then $M_+ = \langle\, L \mid w\,;\, M_1, \ldots, M_n \,\rangle$, $M'_+ = \langle\, L \mid w'\,;\, M'_1, \ldots, M'_n \,\rangle$ where $M_1, \ldots, M_n$ is a tar-irreducible set of membranes, and ($\mathit{here}(w) \neq \mathit{empty}$, or $L = \mathit{Skin} \wedge \mathit{out}(w) \neq \mathit{empty}$, or it exists $i \in [n]$ with $\mathit{in}_{L(M_i)}(w)\mathit{out}(w(M_i)) \neq \mathit{empty}$ or $\mathit{here}(w(M_i)) \neq \mathit{empty}$). Then M'_+ is tar-irreducible by Definition 3.3.3.

(ii) $M_+ \overset{tar}{\Longrightarrow} M'_+$ is inferred by (C_2). Then $M_+ = \langle\, L \mid w\,;\, M_1, \ldots, M_n \,\rangle$, $M'_+ = \langle\, L \mid w''\,;\, M''_1, \ldots, M''_n \,\rangle$ where $M_1, \ldots, M_n \overset{tar}{\Longrightarrow} M'_1, \ldots, M'_n$ is inferred by a shorter inference tree. $M'_1, \ldots, M'_n$ is tar-irreducible by inductive hypothesis. For the membrane M'_+, w'' is *out*-free if $L = \mathit{Skin}$, w'' is $\mathit{in}_{L(M_i)}$-free and $w(M''_i)$ is *out*-free, for all $i \in [n]$. Moreover, the sibling membranes $M''_1, \ldots, M''_n$ are tar-irreducible. Then M'_+ is tar-irreducible by Definition 3.3.3.

(iii) $M_+ \overset{tar}{\Longrightarrow} M'_+$ is inferred by (C_3). Then $M_+ = \langle\, \mathit{Skin} \mid w \,\rangle$, $M'_+ = \langle\, \mathit{Skin} \mid \mathit{obj}(\overline{w}) \,\rangle$, where $\mathit{out}(w) \neq \mathit{empty}$, and $w(M'_+)$ is *out*-free. Then M'_+ is tar-irreducible by Definition 3.3.1.

(iv) $M_+ \overset{tar}{\Longrightarrow} M'_+$ is inferred by (C_4). Then $M_+ = M, N_+$, $M'_+ = M', N_+$ where N_+ is tar-irreducible, and $M \overset{tar}{\Longrightarrow} M'$ is inferred by a shorter inference tree. By inductive hypothesis, M' is tar-irreducible. Therefore M'_+ is tar-irreducible by Definition 3.3.2.

(v) $M_+ \overset{tar}{\Longrightarrow} M'_+$ is inferred by (C_5). Then $M_+ = M, N_+$, $M'_+ = M', N'_+$ where $M \overset{tar}{\Longrightarrow} M'$ and $N_+ \overset{tar}{\Longrightarrow} N'_+$ are inferred by shorter inference trees. M' and N'_+ are tar-irreducible by inductive hypothesis. It follows that M', N'_+ is a tar-irreducible by Definition 3.3.2.

Theorem 3.2 *Let Π be a P system. If $C \in \mathcal{C}^{\#}(\Pi)$ and $C \overset{tar}{\Longrightarrow} C'$, then $C' \in \mathcal{C}(\Pi) \cup \mathcal{C}^{\#}(\Pi)$ and C' is a tar-irreducible.*

The proof follows easily from Lemma 3.3.

3.3 Parallel Membrane Dissolving

If the special symbol δ occurs in the multiset of objects of a membrane labelled by L, the membrane is dissolved producing the following changes in the system:

- its evolution rules and the associated priority relation are lost, and
- its contents (objects and membranes) is added to the contents of the region which was immediately external to the dissolved membrane.

We consider the extension of the operator w (previously defined over membranes) to non-empty sets of sibling membranes by setting $w(M_1, \dots, M_n) = w(M_1) \dots w(M_n)$.

We say that a multiset w is *δ-free* if it does not contain the special symbol δ.

Definition 3.4 *The **δ-irreducibility** property for membranes and for sets of sibling membranes is defined as follows:*

1. *a simple membrane is **δ-irreducible**;*
2. *a non-empty set of sibling membranes $M_1, \dots, M_n$ is **δ-irreducible** iff every membrane M_i is δ-irreducible, for $1 \leq i \leq n$;*
3. *a composite membrane $\langle\, L \,|\, w \,;\, M_+ \,\rangle$ is **δ-irreducible** iff M_+ is δ-irreducible, and $w(M_+)$ is δ-free.*

The **parallel dissolving relation** $\overset{\delta}{\Longrightarrow}$ is defined by the following inference rules:

For each two multisets of objects w_1, w_2, and labels L_1, L_2,

$$(\mathbf{D_1})\frac{}{\langle\, L_1 \,|\, w_1 \,;\, \langle\, L_2 \,|\, w_2\delta \,\rangle \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L_1 \,|\, w_1 w_2 \,\rangle}$$

For each $M_+ \in \mathcal{M}^+(\Pi)$, δ-irreducible $\langle\, L_2 \,|\, w_2\delta \,;\, M_+ \,\rangle$, and label L_1,

$$(\mathbf{D_2})\frac{}{\langle\, L_1 \,|\, w_1 \,;\, \langle\, L_2 \,|\, w_2\delta \,;\, M_+ \,\rangle \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L_1 \,|\, w_1 w_2 \,;\, M_+ \,\rangle}$$

For each $M_+ \in \mathcal{M}^+(\Pi)$, δ-free multiset w_2, and labels L_1, L_2,

$$(\mathbf{D_3})\frac{\langle\, L_2 \mid w_2\, ;\; M_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L_2 \mid w_2' \,\rangle}{\langle\, L_1 \mid w_1\, ;\; \langle\, L_2 \mid w_2\, ;\; M_+ \,\rangle\,\rangle \overset{\delta}{\Longrightarrow} \langle\, L_1 \mid w_1\, ;\; \langle\, L_2 \mid w_2' \,\rangle\,\rangle}$$

For each $M_+, M_+' \in \mathcal{M}^+(\Pi)$, δ-free multiset w_2, multisets w_1, w_2', and labels L_1, L_2

$$(\mathbf{D_4})\frac{\langle\, L_2 \mid w_2\, ;\; M_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L_2 \mid w_2'\, ;\; M_+' \,\rangle}{\langle\, L_1 \mid w_1\, ;\; \langle\, L_2 \mid w_2\, ;\; M_+ \,\rangle\,\rangle \overset{\delta}{\Longrightarrow} \langle\, L_1 \mid w_1\, ;\; \langle\, L_2 \mid w_2'\, ;\; M_+' \,\rangle\,\rangle}$$

For each $M_+ \in \mathcal{M}^+(\Pi)$, multisets w_1, w_2, w_2', and labels L_1, L_2

$$(\mathbf{D_5})\frac{\langle\, L_2 \mid w_2\delta\, ;\; M_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L_2 \mid w_2'\delta \,\rangle}{\langle\, L_1 \mid w_1\, ;\; \langle\, L_2 \mid w_2\delta\, ;\; M_+ \,\rangle\,\rangle \overset{\delta}{\Longrightarrow} \langle\, L_1 \mid w_1 w_2' \,\rangle}$$

For each $M_+ \in \mathcal{M}^+(\Pi)$, multisets w_1, w_2, w_2', and labels L_1, L_2

$$(\mathbf{D_6})\frac{\langle\, L_2 \mid w_2\delta\, ;\; M_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L_2 \mid w_2'\delta\, ;\; M_+' \,\rangle}{\langle\, L_1 \mid w_1\, ;\; \langle\, L_2 \mid w_2\delta\, ;\; M_+ \,\rangle\,\rangle \overset{\delta}{\Longrightarrow} \langle\, L_1 \mid w_1 w_2'\, ;\; M_+' \,\rangle}$$

For each $M_+, N_+ \in \mathcal{M}^+(\Pi)$, δ-irreducible $\langle\, L \mid w\, ;\; N_+ \,\rangle$, and multisets w',

$$(\mathbf{D_7})\frac{\langle\, L \mid w\, ;\; M_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w' \,\rangle}{\langle\, L \mid w\, ;\; M_+, N_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w'\, ;\; N_+ \,\rangle}$$

For each $M_+, M_+', N_+' \in \mathcal{M}^+(\Pi)$, δ-irreducible $\langle\, L \mid w\, ;\; N_+ \,\rangle$, and multisets w', w'',

$$(\mathbf{D_8})\frac{\langle\, L \mid w\, ;\; M_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w'\, ;\; M_+' \,\rangle}{\langle\, L \mid w\, ;\; M_+, N_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w'\, ;\; M_+', N_+ \,\rangle}$$

$$(\mathbf{D_9})\frac{\langle\, L \mid w\, ;\; M_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid ww' \,\rangle \qquad \langle\, L \mid w\, ;\; N_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid ww'' \,\rangle}{\langle\, L \mid w\, ;\; M_+, N_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid ww'w'' \,\rangle}$$

$$(\mathbf{D_{10}})\frac{\langle\, L \mid w\, ;\; M_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid ww' \,\rangle \qquad \langle\, L \mid w\, ;\; N_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid ww''\, ;\; N_+' \,\rangle}{\langle\, L \mid w\, ;\; M_+, N_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid ww'w''\, ;\; N_+' \,\rangle}$$

$$(\mathbf{D_{11}})\frac{\langle\, L \mid w\, ;\; M_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid ww'\, ;\; M_+' \,\rangle \qquad \langle\, L \mid w\, ;\; N_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid ww''\, ;\; N_+' \,\rangle}{\langle\, L \mid w\, ;\; M_+, N_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid ww'w''\, ;\; M_+', N_+' \,\rangle}$$

Example 3.3 *The inference tree for*

$$\langle\, 1 \,|\, \mathit{empty}\, ;\, \langle\, 2 \,|\, \mathit{empty}\, ;\, \langle\, 3 \,|\, dffff\delta \,\rangle, \langle\, 4 \,|\, \mathit{empty} \,\rangle \,\rangle \,\rangle \overset{\delta}{\Longrightarrow} \langle\, 1 \,|\, \mathit{empty}\, ;\, \langle\, 2 \,|\, dffff\, ;\, \langle\, 4 \,|\, \mathit{empty} \,\rangle \,\rangle \,\rangle$$

is:

$$(D_7)\,\dfrac{\langle\, 2 \,|\, \mathit{empty}\, ;\, \langle\, 3 \,|\, dffff\delta \,\rangle \,\rangle \overset{\delta}{\Longrightarrow} \langle\, 2 \,|\, dffff \,\rangle}{(D_4)\,\dfrac{\langle\, 2 \,|\, \mathit{empty}\, ;\, \langle\, 3 \,|\, dffff\delta \,\rangle, \langle\, 4 \,|\, \mathit{empty} \,\rangle \,\rangle \overset{\delta}{\Longrightarrow} \langle\, 2 \,|\, dffff\, ;\, \langle\, 4 \,|\, \mathit{empty} \,\rangle \,\rangle}{\begin{array}{l}\langle\, 1 \,|\, \mathit{empty}\, ;\, \langle\, 2 \,|\, \mathit{empty}\, ;\, \langle\, 3 \,|\, dffff\delta \,\rangle, \langle\, 4 \,|\, \mathit{empty} \,\rangle \,\rangle \,\rangle \overset{\delta}{\Longrightarrow} \\ \langle\, 1 \,|\, \mathit{empty}\, ;\, \langle\, 2 \,|\, dffff\, ;\, \langle\, 4 \,|\, \mathit{empty} \,\rangle \,\rangle \,\rangle\end{array}}}$$

Lemma 3.4 *If $M_+ \overset{\delta}{\Longrightarrow} M'_+$, then M'_+ is δ-irreducible.*

Proof: Let M_+, M'_+ be two non-empty sets of membranes such that $M_+ \overset{\delta}{\Longrightarrow} M'_+$. We prove that M'_+ is δ-irreducible by induction on the depth of the associated inference tree. We consider all possible cases for the final step of the inference:

(i) $M_+ \overset{\delta}{\Longrightarrow} M'_+$ is inferred by (D_1). Since M'_+ is a simple membrane, M'_+ is δ-irreducible by Definition 3.4.1.

(ii) $M_+ \overset{\delta}{\Longrightarrow} M'_+$ is inferred by (D_2). Then $M_+ = \langle\, L_1 \mid w_1 \,;\, \langle\, L_2 \mid w_2\delta \,;\, N_+ \,\rangle \,\rangle$ and $M'_+ = \langle\, L_1 \mid w_1 w_2 \,;\, N_+ \,\rangle$, where $\langle\, L_2 \mid w_2\delta \,;\, N_+ \,\rangle$ is δ-irreducible. It follows that $w(N_+)$ is δ-free, and that M'_+ is δ-irreducible by Definition 3.4.3.

(iii) $M_+ \overset{\delta}{\Longrightarrow} M'_+$ is inferred by (D_3). Then $M_+ = \langle\, L_1 \mid w_1 \,;\, \langle\, L_2 \mid w_2 \,;\, N_+ \,\rangle \,\rangle$ and $M'_+ = \langle\, L_1 \mid w_1 \,;\, \langle\, L_2 \mid w'_2 \,\rangle \,\rangle$, where $\langle\, L_2 \mid w_2 \,;\, N_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L_2 \mid w'_2 \,\rangle$ is inferred by a shorter inference tree, and w_2 is δ-free. $\langle\, L_2 \mid w'_2 \,\rangle$ is δ-irreducible by inductive hypothesis. Since w_2 is δ-free and no dissolving rule preserves δ, it follows w'_2 is δ-free. Then M'_+ is δ-irreducible by Definition 3.4.3.

(iv) $M_+ \overset{\delta}{\Longrightarrow} M'_+$ is inferred by (D_4). Then $M_+ = \langle\, L_1 \mid w_1 \,;\, \langle\, L_2 \mid w_2 \,;\, N_+ \,\rangle \,\rangle$ and $M'_+ = \langle\, L_1 \mid w_1 \,;\, \langle\, L_2 \mid w'_2 \,;\, N'_+ \,\rangle \,\rangle$, where $\langle\, L_2 \mid w_2 \,;\, N_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L_2 \mid w'_2 \,;\, N'_+ \,\rangle$ is inferred by a shorter inference tree, and w_2 is δ-free. $\langle L_2 \,|\, w'_2 ; N'_+ \rangle$ is δ-irreducible by inductive hypothesis. Since w_2 is δ-free and no dissolving rule preserves δ, it follows w'_2 is δ-free. Then M'_+ is δ-irreducible by Definition 3.4.3.

(v) $M_+ \overset{\delta}{\Longrightarrow} M'_+$ is inferred by (D_5). M'_+ is δ-irreducible by Definition 3.4.1.

(vi) $M_+ \overset{\delta}{\Longrightarrow} M'_+$ is inferred by (D_6). Then $M_+ = \langle\, L_1 \mid w_1 \,;\, \langle\, L_2 \mid w_2\delta \,;\, N_+ \,\rangle \,\rangle$ and $M'_+ = \langle\, L_1 \,|\, w_1 w'_2 \,;\, N'_+ \,\rangle$, where $\langle\, L_2 \,|\, w_2\delta \,;\, N_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L_2 \,|\, w'_2\delta \,;\, N'_+ \,\rangle$ is inferred by a shorter inference tree. $\langle\, L_2 \,|\, w'_2\delta \,;\, N'_+ \,\rangle$ is δ-irreducible by inductive hypothesis, and hence N'_+ is δ-irreducible, and $w(N'_+)$ is δ-free. M'_+ is also δ-irreducible by Definition 3.4.3.

(vii) $M_+ \overset{\delta}{\Longrightarrow} M'_+$ is inferred by (D_7). Then $M_+ = \langle\, L \mid w\ ;\ N'_+, N''_+ \,\rangle$ and $M'_+ = \langle\, L \mid w'\ ;\ N''_+ \,\rangle$, where $\langle\, L \mid w\ ;\ N'_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w' \,\rangle$ is inferred by a shorter inference tree, and $\langle\, L \mid w\ ;\ N''_+ \,\rangle$ is δ-irreducible. It follows that $w(N''_+)$ is δ-free, and M'_+ is δ-irreducible by Definition 3.4.3.

(viii) $M_+ \overset{\delta}{\Longrightarrow} M'_+$ is inferred by (D_8). Then $M_+ = \langle\, L \mid w\ ;\ N^1_+, N^2_+ \,\rangle$ and $M'_+ = \langle\, L \mid w'\ ;\ N^3_+, N^2_+ \,\rangle$, where $\langle\, L \mid w\ ;\ N^1_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w'\ ;\ N^3_+ \,\rangle$ is inferred by a shorter inference tree, and $\langle\, L \mid w\ ;\ N^2_+ \,\rangle$ is δ-irreducible. N^3_+ is δ-irreducible and $w(N^3_+)$ is δ-free by inductive hypothesis and Definition 3.4.3. It follows that N^3_+, N^2_+ is δ-irreducible by Definition 3.4.2. Since $w(N^3_+, N^2_+)$ is δ-free, we get that M'_+ is δ-irreducible by Definition 3.4.3.

(ix) M'_+ is δ-irreducible by Definition 3.4.1.

(x) $M_+ \overset{\delta}{\Longrightarrow} M'_+$ is inferred by (D_{10}). Then $M_+ = \langle\, L \mid w\ ;\ N^1_+, N^2_+ \,\rangle$ and $M'_+ = \langle L|ww'w''; N^3_+\rangle$, where $\langle L|w; N^1_+\rangle \overset{\delta}{\Longrightarrow} \langle L|ww'\rangle$ and $\langle L|w; N^2_+\rangle \overset{\delta}{\Longrightarrow} \langle L|ww''; N^3_+\rangle$ are inferred by shorter inference trees. $\langle\, L \,|\, ww''\,;\, N^3_+ \,\rangle$ is δ-irreducible by inductive hypothesis, and hence N^3_+ is δ-irreducible, and $w(N^3_+)$ is δ-free. Then M'_+ is δ-irreducible by Definition 3.4.3.

(xi) $M_+ \overset{\delta}{\Longrightarrow} M'_+$ is inferred by (D_{11}). Then $M_+ = \langle\, L \mid w\ ;\ N^1_+, N^2_+ \,\rangle$ and $M'_+ = \langle\, L \,|\, ww'w''\,;\, N^3_+, N^4_+ \,\rangle$, where $\langle\, L \,|\, w\,;\, N^1_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \,|\, ww'\,;\, N^3_+ \,\rangle$ and $\langle\, L \,|\, w\,;\, N^2_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid ww''\ ;\ N^4_+ \,\rangle$ are inferred by shorter inference trees. Both N^3_+ and N^4_+ are δ-irreducible, and both $w(N^3_+)$ and $w(N^4_+)$ are δ-free by inductive hypothesis and Definition 3.4.3. Therefore N^3_+, N^4_+ is also δ-irreducible, and $w(N^3_+, N^4_+)$ is δ-free. It follows that M'_+ is δ-irreducible by Definition 3.4.3.

Theorem 3.3 *Let Π be a P system. If $C \in \mathcal{C}^{\#}(\Pi)$ is mpr- and tar-irreducible and $C \overset{\delta}{\Longrightarrow} C'$, then $C' \in \mathcal{C}(\Pi)$.*

The proof follows easily from Lemma 3.4.

Proposition 3.1 *Let Π be a P system. If $C \overset{mpr}{\Longrightarrow} C'$ and $C' \overset{tar}{\Longrightarrow} C''$ such that $C \in \mathcal{C}(\Pi)$, $C' \in \mathcal{C}^{\#}(\Pi)$, and C'' is δ-irreducible, then $C'' \in \mathcal{C}(\Pi)$.*

Proof: C' is mpr-irreducible by Theorem 3.1, and C'' is tar-irreducible by Theorem 3.2. Therefore C'' does not contain both messages and δ, and hence it is a committed configuration, i.e., $C'' \in \mathcal{C}(\Pi)$.

Proposition 3.2 *Let Π be a P system. If $C \overset{mpr}{\Longrightarrow} C'$ and $C' \overset{\delta}{\Longrightarrow} C''$ such that $C \in \mathcal{C}(\Pi)$, $C' \in \mathcal{C}^{\#}(\Pi)$, and C' is tar-irreducible, then $C'' \in \mathcal{C}(\Pi)$.*

Proof: According to Theorem 3.1, C' is mpr-irreducible. According to Theorem 3.3, C'' is δ-irreducible. Therefore C'' does not contain both messages and δ, and hence it is a committed configuration, i.e., $C'' \in \mathcal{C}(\Pi)$.

Proposition 3.3 *Let* Π *be a P system. If* $C \overset{mpr}{\Longrightarrow} C'$, $C' \overset{tar}{\Longrightarrow} C''$, *and* $C'' \overset{\delta}{\Longrightarrow} C'''$ *such that* $C \in \mathcal{C}(\Pi)$, *and* $C', C'' \in \mathcal{C}^{\#}(\Pi)$, *then* $C''' \in \mathcal{C}(\Pi)$.

Proof: According to Theorem 3.1, C' is mpr-irreducible. According to Theorem 3.2, C'' is tar-irreducible, and hence C'' does not contain messages. According to Theorem 3.3, C''' is δ-irreducible, and hence it does not contain both messages and δ. Therefore C''' is a committed configuration, i.e., $C''' \in \mathcal{C}(\Pi)$.

Definition 3.5 *Let* Π *be a P system. A* transition step *in* Π *is defined by the following inference rules:*

For each $C, C'' \in \mathcal{C}(\Pi)$, *and* δ*-irreducible* $C' \in \mathcal{C}^{\#}(\Pi)$,

$$\frac{C \overset{mpr}{\Longrightarrow} C', C' \overset{tar}{\Longrightarrow} C''}{C \Rightarrow C''}$$

For each $C, C'' \in \mathcal{C}(\Pi)$, *and tar-irreducible* $C' \in \mathcal{C}^{\#}(\Pi)$,

$$\frac{C \overset{mpr}{\Longrightarrow} C', C' \overset{\delta}{\Longrightarrow} C''}{C \Rightarrow C''}$$

For each $C, C''' \in \mathcal{C}(\Pi)$, *and* $C', C'' \in \mathcal{C}^{\#}(\Pi)$,

$$\frac{C \overset{mpr}{\Longrightarrow} C', C' \overset{tar}{\Longrightarrow} C'', C'' \overset{\delta}{\Longrightarrow} C'''}{C \Rightarrow C'''}$$

The consistency of this definition follows from the previous three propositions.

Chapter 4

Rewriting Logic: A Semantic Framework for Concurrency

In this chapter we explain the basic concepts of order-sorted equational logic [23] and rewriting logic [32], with emphasis on the intrinsic concurrency [33] of the latter logic, and, in the end, we present the Maude system [12] together with the tools provided by Maude for formal analysis of concurrent specifications.

4.1 Order-Sorted Equational Logic

This section presents the most basic definitions and results of Order-Sorted Equational Logic ([23]).

Order-Sorted Algebra

Let S be a partially ordered set (or poset) whose elements are called *sorts*. An *S-sorted set A* is a family of sets A_s for each sort $s \in S$. An *S-sorted function $f : A \to B$* is an S sorted family $f = \{f_s : A_s \to B_s | s \in S\}$.

Let $\leq$ be the partial order relation on S. This relation can be extended over strings of equal length in S^* by $s_1 \ldots s_n \leq s'_1 \ldots s'_n$ iff $s_i \leq s'_i$ for $1 \leq i \leq n$. Similarly, $\leq$ extends to pairs $\langle w, s\rangle$ in $S^* \times S$ by $\langle w, s\rangle \leq \langle w', s'\rangle$ iff $w \leq w'$ and $s \leq s'$.

Definition 4.1 *A* many-sorted signature *is a pair (S, Σ), where S is called the* sort set *and Σ is an $S^* \times S$-sorted family $\{\Sigma_{w,s} | w \in S^*,\ s \in S\}$. Elements of Σ are called operations (or function) symbols, or for short,* operations.

An order-sorted signature *is a triple $(S, \leq, \Sigma)$ such that (S, Σ) is a many-sorted signature, $(S, \leq)$ is a poset, and the operations satisfy the following* monotonicity condition*:*

$$\sigma \in \Sigma_{w_1,s_1} \cap \Sigma_{w_2,s_2} \textit{ and } w_1 \leq w_2 \textit{ imply } s_1 \leq s_2.$$

When the sort set S is clear, we write Σ for (S, Σ), and when the poset $(S, \leq)$ is clear, we write Σ for $(S, \leq, \Sigma)$. When $\sigma \in \Sigma_{w,s}$, we say that σ has *rank* $\langle w, s\rangle$, *arity*

w, and (value, or result, or coarity) *sort* s. We may write $\sigma : w \to s$ for $\sigma \in \Sigma_{w,s}$ to emphasize that σ denotes a function with arity w and sort s. An important special case is $w = \lambda$; then $\sigma \in \Sigma_{\lambda,s}$ denotes a *constant* of sort s. The monotonicity condition excludes overloaded constants, because $w_1 = w_2 = \lambda$ implies $s_1 = s_2$.

Definition 4.2 *An order-sorted signature Σ is* regular *iff given σ in Σ_{w_1,s_1}, and given $w_0 \le w_1$ in S^*, there is a least rank $\langle w, s\rangle \in S^* \times S$ such that $w_0 \le w$ and $\sigma \in \Sigma_{w,s}$.*

Definition 4.3 *Let (S, Σ) be a many-sorted signature. Then an (S, Σ)-*algebra *A is a family $\{A_s | s \in S\}$ of sets called the* carriers *of A, together with a function $A_\sigma : A_w \to A_s$ for each σ in $\Sigma_{w,s}$, where $A_w = A_{s_1} \times \dots A_{s_n}$ when $w = s_1 \dots s_n$, and where A_w is a one point set when $w = \lambda$.*

*Let $(S, \le, \Sigma)$ be an order-sorted signature. Then an $(S, \le, \Sigma)$-*algebra *is an (S, Σ)-algebra A such that the following monotonicity conditions are satisfied:*

1. *$s \le s'$ in S implies $A_s \subseteq A_{s'}$,*
2. *$\sigma \in \Sigma_{w_1,s_1} \cap \Sigma_{w_2,s_2}$ and $w_1 \le w_2$ imply $A_\sigma : A_{w_1} \to A_{s_1}$ equals $A_\sigma : A_{w_2} \to A_{s_2}$ on A_{w_1}.*

When the sort set S is clear, (S, Σ)-algebras may be called *many-sorted Σ-algebras*; similarly, when $(S, \le)$ is clear, $(S, \le, \Sigma)$-algebras may be called *order-sorted Σ-algebras.* We also write $A^{w,s}_\sigma$ for $A_\sigma : A_w \to A_s$.

Definition 4.4 *Let (S, Σ) be a many-sorted signature, and let A and B be (S, Σ)-algebras. Then an (S, Σ)-*homomorphism *$h : A \to B$ is an S-sorted function $h = \{h_s : A_s \to B_s | s \in S\}$ satisfying the following* homomorphism condition*:*

$$h_s(A^{w,s}_\sigma(a)) = B^{w,s}_\sigma(h_w(a)) \text{ for each } \sigma \in \Sigma_{w,s} \text{ and } a \in A_w$$

where $h_w(a) = \langle h_{s_1}(a_1), \dots, h_{s_n}(a_n)\rangle$, when $w = s_1 \dots s_n$ and $a = \langle a_1, \dots, a_n\rangle$, with $a_i \in A_{s_i}$ for $i \in [n]$ when $w \ne \lambda$. If $w = \lambda$, the above condition specializes to $h_s(A^{\lambda,s}_\sigma) = B^{\lambda,s}_\sigma$.

*Let $(S, \le, \Sigma)$ be a order-sorted signature, and let A, B be two order-sorted $(S, \le, \Sigma)$-algebras. Then an $(S, \le, \Sigma)$-*homomorphism *$h : A \to B$ is an (S, Σ)-homomorphism $h = \{h_s : A_s \to B_s | s \in S\}$ satisfying the following* restriction condition*:*

$$s \le s' \text{ and } a \in A_s \text{ imply } h_s(a) = h_{s'}(a).$$

Definition 4.5 *Let (S, Σ) be a many-sorted signature. The many-sorted term algebra T_Σ is inductively constructed as follows:*

- *$\Sigma_{\lambda,s} \subseteq T_{\Sigma,s}$;*
- *if $\sigma \in \Sigma_{w,s}$ and if $t_i \in T_{\Sigma,s_i}$ for $i \in [n]$, where $w = s_1 \dots s_n$ with $n > 0$, then $\sigma(t_1, \dots, t_n) \in T_{\Sigma,s}$.*

For Σ an order-sorted signature, we similarly construct the order-sorted Σ-term algebra $\mathcal{T}_\Sigma$ as the least family $\{\mathcal{T}_{\Sigma,s} | s \in S\}$ of sets satisfying the following conditions:

- $\Sigma_{\lambda,s} \subseteq \mathcal{T}_{\Sigma,s}$, *for* $s \in S$;
- $\mathcal{T}_{\Sigma,s'} \subseteq \mathcal{T}_{\Sigma,s}$ *if* $s' \leq s$;
- *if* $\sigma \in \Sigma_{w,s}$ *and if* $t_i \in \mathcal{T}_{\Sigma,s_i}$, *where* $w = s_1 \ldots s_n \neq \lambda$, *then* $\sigma(t_1, \ldots, t_n) \in \mathcal{T}_{\Sigma,s}$.

A given term t in an order-sorted term algebra can have many different sorts. In particular, if $t \in \mathcal{T}_\Sigma$ has sort s, then it also has sort s' for any $s' \geq s$.

Definition 4.6 *Let Σ be an order-sorted signature. Then an order-sorted Σ-algebra is* initial *in the class of all order-sorted Σ-algebras iff there is a unique order-sorted Σ-homomorphism from it to any other order-sorted Σ-algebra.*

Theorem 4.1 *Let Σ be a regular order-sorted signature. Then $\mathcal{T}_\Sigma$ is an initial order-sorted Σ-algebra.*

Proposition 4.1 *Given a regular order-sorted signature Σ, for every $t \in \mathcal{T}_\Sigma$ there is a least $s \in S$, called the* least sort *of t and denoted $LS(t)$, such that $t \in \mathcal{T}_{\Sigma,s}$.*

The terms considered above are *ground terms*, in the sense that they involve no variables. Terms with variables can be seen as a special case of ground terms, by enlarging the signature with new constant that correspond to the variable symbols. We assume that each variable comes with a given sort, so that we have an S-sorted family $X = \{X_s | s \in S\}$ of disjoint sets. Given an order-sorted signature $(S, \leq, \Sigma)$ and an S-sorted variable set X that is disjoint from Σ, we define the new order-sorted signature $(S, \leq, \Sigma(X))$ by $\Sigma(X)_{\lambda,s} = \Sigma_{\lambda,s} \cup X_s$ and $\Sigma(X)_{w,s} = \Sigma_{w,s}$, for $w \neq \lambda$. It is easy to see that $\Sigma(X)$ is regular if Σ is regular. We can form and view $\mathcal{T}_{\Sigma(X)}$ as an order-sorted Σ-algebra just by forgetting the constants in X. We denote this algebra $\mathcal{T}_\Sigma(X)$.

Theorem 4.2 *Given a regular order-sorted signature $(S, \leq, \Sigma)$, let A be a Σ-algebra and let $a : X \to A$ be an S-sorted function; hereafter we call such a function an* assignment. *Then there is a unique order-sorted Σ-homomorphism $a^* : \mathcal{T}_\Sigma(X) \to A$ such that $a^*(x) = a(x)$ for each $x \in X$.*

Definition 4.7 *For$(S, \leq, \Sigma)$ a regular order-sorted signature, a Σ-*equation *is a triple $\langle X, t, t' \rangle$ (hereafter denoted by $(\forall X)\ t = t'$) where X is a variable set, and $t, t' \in \mathcal{T}_{\Sigma(X)}$ with $LS(t)$ and $LS(t')$ in the same connected component of $(S, \leq)$.*

An order-sorted Σ-algebra A satisfies *a Σ-equation $(\forall X)\ t = t'$ iff $a^*_{LS(t)}(t) = a^*_{LS(t')}(t')$ in A for every assignment $a : X \to A$. Similarly, A satisfies a set Γ of Σ-equations iff it satisfies each member of Γ; in this case, we say that A is a (Σ, Γ)-*algebra.

Order-sorted conditional equations generalize order-sorted equations, i.e., they are expressions of the form $(\forall X)\ t = t'$ if *C, where the condition C is a finite set of* *unquantified Σ-equations involving only variable from X. An order-sorted Σ-algebra A* satisfies *the equation* $(\forall X)\ t = t'$ if C *iff for each assignment $a : X \to A$ such that $a^*_{LS(v)}(v) = a^*_{LS(v')}(v')$ in A for each equation $v = v'$ in C, then also $a^*_{LS(t)}(t) = a^*_{LS(t')}(t')$ in A.*

Order-Sorted Equational Deduction

Given an order-sorted signature Σ and a set Γ of conditional Σ-equations, we consider each unconditional equation in Γ to be *derivable*. Other derivable equations can be obtained by finite application of the following rules:

(1) *Reflexivity.*

$$\frac{}{(\forall X)t = t}$$

(2) *Symmetry.*

$$\frac{(\forall X)t = t'}{(\forall X)t' = t}$$

(3) *Transitivity.*

$$\frac{(\forall X)t_1 = t_2, (\forall X)t_2 = t_3}{(\forall X)t_1 = t_3}$$

(4) *Congruence.* For $\theta, \theta' : X \to \mathcal{T}_\Sigma(X)$ substitutions, $t \in \mathcal{T}_\Sigma(X)$,

$$\frac{(\forall Y)\theta = \theta'}{(\forall Y)\theta(t) = \theta'(t)}$$

(5) *Unconditional Replacement.* For each substitution $\theta : X \to \mathcal{T}_\Sigma(Y)$,

$$\frac{(\forall X)t = t'}{(\forall Y)\theta(t) = \theta(t')}$$

4.2 Rewriting Logic

Since each rewrite theory has an underlying equational theory, different variants of equational logic give rise to corresponding variants of rewriting logic. The expressiveness of the rewrite theories is strongly related to the underlying equational language. The most expressive is the membership equational logic (MEL) [31]: a Horn logic with both equations $t = t'$ and membership equations $t : s$ which generalizes order-sorted equational logic and supports sorts, subsorts, partiality, and sorts defined by equational axioms.

To simplify the exposition we present the order-sorted rewriting logic with unconditional rewrite rules.

A *signature* in (order-sorted) rewriting logic is an (order-sorted) equational theory (Σ, E), where Σ is an equational signature and E is a set of Σ-equations. Rewriting operates on equivalence classes of terms modulo E. In this way, the rewriting is made free from the syntactic constraints of a term representation and it gains more flexibility in deciding what counts as a data structure. For example, string rewriting is obtained by imposing an associative axiom, multiset rewriting by imposing associativity and commutativity, while for the standard term rewriting the set of equations E is empty.

A *sentence* over a given signature (Σ, E) is an expression of the form $(\forall X)[t]_E \rightarrow [t']_E$, where t and t' are $\Sigma(X)$-terms, and $[t]_E$ denotes the equivalence class of t modulo the equations E. A sentence describes the possible transitions from the states described by $[t]$ to the corresponding states described by $[t']$. If $X = \{x_1, \ldots, x_n\}$, then we denote a sentence by $[t(\overline{x})]_E \rightarrow [t'(\overline{x})]_E$, where $\overline{x}$ is the sequence $x_1, \ldots, x_n$. If E and X are understood from the context, we simply write $[t(\overline{x})] \rightarrow [t'(\overline{x})]$ or $[t] \rightarrow [t']$.

A *rewriting specification* $\mathcal{R}$ is a 4-tuple $\mathcal{R}$=(Σ, E, L, R) where (Σ, E) is a rewriting logic signature, L is a set whose elements are called *labels*, and R is a set of labelled rewriting rules (sentences) written as $r : [t(\overline{x})]_E \rightarrow [t'(\overline{x})]_E$.

Given a rewrite theory $\mathcal{R}$, we say that $\mathcal{R}$ *entails the sentence* $[t] \rightarrow [t']$, or that $[t] \rightarrow [t']$ is a *(concurrent)* $\mathcal{R}$*-rewrite*, and write $\mathcal{R} \vdash [t] \rightarrow [t']$ iff $[t] \rightarrow [t']$ can be obtained by finite application of the following *rules of deduction* (where we assume that all the terms are well formed and $t(\overline{w}/\overline{x})$ denotes the simultaneous substitution of w_i for x_i in t):

(1) *Reflexivity.* For each $t \in T_{\Sigma,E}(X)$,

$$\frac{}{[t] \rightarrow [t]}$$

(2) *Congruence.* For each operation symbol $f \in \Sigma_n$, $n \in \mathbf{N}$,

$$\frac{[t_1] \rightarrow [t'_1], \ldots, [t_n] \rightarrow [t'_n]}{[f(t_1, \ldots, t_n)] \rightarrow [f(t'_1, \ldots, t'_n)]}$$

(3) *Unconditional replace.* For each rule $r : [t(\overline{x})] \rightarrow [t'(\overline{x})]$ in R,

$$\frac{[u_1] \rightarrow [v_1], \ldots, [u_n] \rightarrow [v_n]}{[t(\overline{u}/\overline{x})] \rightarrow [t(\overline{v}/\overline{x})]}$$

(4) *Transitivity.*

$$\frac{[t_1] \rightarrow [t_2], \;\; [t_2] \rightarrow [t_3]}{[t_1] \rightarrow [t_3]}$$

The general theory of the rewriting logic allows also *conditional sentences* and *conditional rewriting rules* [13].

Rewriting logic is *reflective*, i.e. there is a (finitely presented) *universal rewriting specification* $\mathcal{U}$ such that for any (finitely presented) rewriting specification $\mathcal{R}$ (including $\mathcal{U}$ itself), we have the following equivalence:

$$\mathcal{R} \vdash [t] \rightarrow [t'] \;\text{ iff }\; \mathcal{U} \vdash \langle \overline{\mathcal{R}}, \overline{t} \rangle \rightarrow \langle \overline{\mathcal{R}}, \overline{t'} \rangle,$$

where $\overline{\mathcal{R}}$ and $\bar{t}$ are terms representing $\mathcal{R}$ and t as data elements of $\mathcal{U}$. Since $\mathcal{U}$ is representable in itself, it is possible to achieve a *reflective tower* with an arbitrary number of reflection levels:

$$\mathcal{R} \vdash [t] \to [t'] \text{ iff } \mathcal{U} \vdash \langle \overline{\mathcal{R}}, \bar{t} \rangle \to \langle \overline{\mathcal{R}}, \overline{t'} \rangle \text{ iff } \mathcal{U} \vdash \langle \overline{\mathcal{U}}, \overline{\langle \overline{\mathcal{R}}, \bar{t} \rangle} \rangle \to \langle \overline{\mathcal{U}}, \overline{\langle \overline{\mathcal{R}}, \overline{t'} \rangle} \rangle \ldots$$

4.3 The Intrinsic Concurrency in Rewriting Logic

Rewriting logic is a logic for reasoning correctly about *concurrent systems* having *states*, and evolving by means of *transitions*. The signature of a rewrite theory describes a particular structure for the states of a system so that its states can be distributed according to such a structure. The rewrite rules in the theory describe which *elementary local transitions* are possible in the distributed state by concurrent local transformations. The rules of rewriting logic allow us to reason correctly about which *general* concurrent transitions are possible in a system satisfying such a description. Thus each rewrite step is a parallel local transition in a concurrent system. Alternatively, the rules of rewriting logic can be considered as *metarules* for a correct deduction in a *logical system*. Logically, each rewriting step is a logical *entailment* in a formal system.

The computational and the logical viewpoints under which rewriting logic can be interpreted can be summarized in the following diagram of correspondences:

State	$\leftrightarrow$	Term	$\leftrightarrow$	Proposition
Transition	$\leftrightarrow$	Rewriting	$\leftrightarrow$	Deduction
Distributed Structure	$\leftrightarrow$	Algebraic Structure	$\leftrightarrow$	Propositional Structure

The last row expresses the fact that a state can be transformed in a concurrently way only if it is nonatomic, that is, if it is composed out of smaller state components that can be changed independently. In rewriting logic this composition of a concurrent state is formalized by the operations of the signature Σ of the rewrite theory $\mathcal{R}$ that axiomatizes the system. From the logical point of view such operations can naturally be regarded as user-definable propositional connectives stating the particular structure that a given state has. It is important to notice that the algebraic structure of a system also involves equations. Such equations describe the system's global state as a concurrent data structure, and they are working as "data solvents" to loosen up the data structures so that more rewrites can be performed in parallel.

Because rewriting logic is neutral about concurrency constructs, it is a general *semantic framework* for concurrency that can express many concurrency models ([33]) such as: equational programming, which is the special case of rewrite theories whose set of rules is empty and whose equations are Church-Rosser, possibly modulo some axioms A; lambda calculi and combinatory reduction systems; labelled transition systems; grammars and string-rewriting systems; Petri nets, including place/transition nets, contextual nets, algebraic nets, colored nets, and timed Petri nets; Gamma and the Chemical Abstract Machine; CCS and LOTOS; the π calculus; concurrent objects and actors; the UNITY language; concurrent graph rewriting; dataflow; neural networks; real-time systems, including timed automata, timed transition systems, hybrid automata, and timed Petri nets; and the tile logic model of synchronized concurrent computation.

Since the above are typically executable, rewriting logic is a flexible *operational* semantic framework to specify such models.

4.4 Rewriting Logic Languages

The idea of using rewrite rules to specify algorithms and compute with them is not new; it is implicit in algebraic simplification techniques that go back to antiquity ([30]). In a formal version it can be traced back to the Herbrand-Gödel-Kleene theory of general recursive functions, and - in a variety of limited forms - it also goes back to Post systems, combinators, the lambda calculus and pure Lisp. In the context of equational logic, early proposal to program with rewrite rules include work by O'Donnell [38], and Goguen and Tardo [24], and a vast body of work has followed since that time ([22] for example). However, the semantics of such rewrite rule languages has usually been based on equational logic, i.e., they have been given a *functional* interpretation.

Several research groups have developed language tools to support formal reasoning and executable specification in rewriting logic.

The ELAN language is a rewriting language developed at LORIA (CNRS, INRIA, and Universities of Nancy) by P. Borovansky, C. Kirchner, H. Kirchner, P.-E. Moreau and M. Vittek [5, 7]. It has as modules *computational systems*, consisting of a rewrite theory and a *strategy* to guide the rewriting process [6]. This group and their collaborators have developed an impressive collection of examples and case studies in areas such as logic programming languages, constraint solving, higher-order substitution, equational theorem-proving and other such computational systems. Besides the ELAN interpreter, there is work in compilation techniques, including compilation of AC-rewriting [34, 35, 28].

The CafeOBJ language implementation, developed at the Japan Advanced Institute of Science and Technology (JAIST) in Kanazawa [19, 16, 17] by K. Futatsugi, R. Diaconescu and T. Sawada, which is also based on rewriting logic, contains OBJ as its functional sublanguage, and supports object-oriented specifications. Furthermore, its semantics is multi-logical and includes hidden-sorted versions of equational and rewriting logic [15, 16, 17]. The CafeOBJ language has been the basis of an ambitious research effort - the Cafe Project - involving several research institutions in Japan, Europe and the US, as well as several Japanese industries, to exploit the promising possibilities of rewriting logic for formal methods applications in software engineering [20]. This project has achieved a distributable version of the language and further work on its semantics, a collection of specification libraries and case studies, and environment, and a collection of theorem proving tools supporting different forms of verification. Furthermore, a compiler has been developed in addition to the Cafe interpreter implementation.

The Maude language implementation was developed at SRI, Menlo Park, California and at University of Illinois at Urbana-Champaign by M. Clavel, F. Duran, S. Eker, P. Lincoln, N. Marti-Oliet, J. Meseguer, and C. Talcott. A quick survey of its features are presented in the next section.

4.5 The Maude System

Maude is a software system developed around the Maude language. Core Maude is the Maude interpreter implemented in C++; it provides the Maude's basic functionality. Full Maude is an extension written in Maude itself, allowing combination of various Maude modules to build more complex modules. Maude can be used for many applications with competitive performance and advantages over the conventional code. The current Maude implementation can execute syntactic rewriting with speeds from half a million to several millions rewrites per second, depending on the particular application and machine. It is able to work well with multisets having millions of elements.

The Maude system is available free of charge, under the terms of the GNU General Public License, at the Maude home page `http://maude.cs.uiuc.edu`. Many useful materials are available, and Maude binaries are provided for selected architectures and operating systems together with installation instructions. For the current work we have used the version 2.1 of Maude, under Linux.

Maude is essentially a mathematical language. It has been influenced in important ways by OBJ3 [22]; in particular, Maude's equational logic sublanguage contains OBJ3 as a sublanguage. The main differences from OBJ3 at the equational level are a much greater performance, and a richer equational logic, namely, membership equational logic [31], that extends OBJ3's order-sorted equational logic [23].

The key novelty of Maude is that, besides efficiently supporting equational computation and algebraic specification in the OBJ style, it also supports *rewriting logic* computation.

The basic programming statements are equations, membership assertions, and rules. Their rewriting semantics is given by the fact that instances of the lefthand side pattern are replaced by corresponding instances of the righthand side.

A Maude program containing only equations and membership assertions is called a *functional module*. Maude's functional modules are theories in membership equational logic [31], a Horn logic whose atomic sentences are equalities $t = t'$ and membership assertions of the form $t : s$, stating that a term t has the sort s. Such a logic extends order-sorted equational logic, and supports sorts, subsort relations, subsort polymorphic overloading of operators, and definition of partial functions with equationally defined domains. Maude's functional modules are assumed to be Church-Rosser and terminating. A Maude program containing also rules is called a *system module*. Maude's system modules specify rewrite theories in rewriting logic. Unconditional rewrite rules have the form $r : t \to t'$. Computationally, they are interpreted as local transition rules in a possible concurrent system. Logically, they are interpreted as inference rules in a logical system [13]. A Maude program can contain as well conditional equations, membership assertions, or rewrite rules.

In system modules computations need not be confluent and need not be terminating, i.e., unlike for equations, there is no assumption that all rewriting sequences will lead to the same final result, and for some systems there may not be any final state. Therefore the issue of executing rewriting logic specifications of system modules in general is considerable more subtle that executing expressions in a functional module, for which the termination and Church-Rosser properties guarantee a unique final result regardless of the order in which equations are applied as simplification rules. Hence, we need to have good ways of controlling the rewriting inference process, which in principle could go in many undesired directions, by means of adequate *strategies*.

The *default strategy* in the Maude interpreter applies the rules in a top-down rule fair way, always reducing to canonical form using the existing equations (modulo all combinations of associativity, commutativity, and identity) before applying any rule. "Top-down" means that each rewrite is attempted beginning at the top of the term, so that any position rewritten does not have a position above it that could also have been rewritten. A limited form of fairness is achieved by keeping the rules in a circular list, and moving a rule to the end after it has been applied [13].

The choice of appropriate strategies to control the rewriting inference process is crucial for executing rewrite theories. In Maude, such strategies are not an extralogical part of the language. They are instead *internal strategies* defined by rewrite theories at the metalevel.

Maude efficiently supports the reflective tower through its `META-LEVEL` module, which makes possible not only the declarative definitions and execution of user-definable rewriting strategies, but also many other metaprogramming applications. In particular, it is possible to define and execute within the logic an extensible module algebra supporting the OBJ style of *parameterized programming* [22], with highly generic and reusable modules. The basic idea is that `META-LEVEL` is extended with new data types for: *parameterized* modules; *theories*, with loose semantics, to state formal requirements in parameters; *views*, to bind parameter theories to their instances; and *module expressions*, instantiating, transforming, and composing parameterized modules. All such new types and operations are defined in Maude itself. Maude also supports *object-oriented* modules, with convenient syntax for object-oriented applications (but their semantics reduces to that of system modules).

4.6 Formal Analysis of Concurrent Specifications in Maude

Specifying in Maude a computational model, we obtain a solid foundational framework for analysis. Besides the executable specification, thanks to generic analysis tools for rewriting logic specifications that are efficiently implemented and currently provided as part of the Maude system, we additionally get the following important analysis tools *for free*:

1. a *semi-decision procedure* to find failures of safety properties in a (possibly infinite-state) concurrent system using Maude's `search` command;

2. an LTL *model checker* for finite-state systems;

3. a *theorem prover* (Maude's ITP [14]) that can be used to semi-automatically prove properties.

We detail the first two items, because they are entirely automatic, and we use them for verifying our implementation of P systems in Maude.

4.6.1 Search

Using Maude's `search` command, one can search a potentially infinite state space for behaviors of interest. Since the search is performed in a breadth-first manner, if any safety violation exists, then it will eventually be found, i.e., this is a semi-decision procedure for finding such errors. Moreover, the command `show search graph` gives the state space generated by the previous `search` command. This command is illustrated in section 5.2.

4.6.2 Model Checking

When the state space of a concurrent system is finite, one can exhaustively analyze all its possible executions and check them against temporal logic properties. Currently, Maude provides a builtin explicit-state model checker for linear temporal logic (LTL) comparable in speed to SPIN [18]. The LTL model checker provides a powerful tool to detect subtle errors and to verify some desired temporal properties.

LTL was designed for expressing the temporal ordering of events. In computer science, temporal logics keep track of the systems states, changes of the variable values, and the order in which they occur. Intuitively, the system state is a snapshot of the system's execution. In this snapshot, every variable has some value. A particular execution of the system is represented by a sequence of system states, and obviously, time progresses during the execution, but there is no keeping track of how long the system is in any particular state. In LTL, formulas are evaluated with respect to a particular execution and a particular state in that execution. Time is totally ordered, usually bounded in the past and unbounded in the future. LTL formulas allow boolean connectives and modalities together with the operators **X** expressing "in the next state", and **U** expressing that one property holds until another holds (Until). These formulas are evaluated on individual executions (computation paths). The modal path operators are $\Diamond$ and $\Box$; $\Diamond$ expressing "at some future time", and $\Box$ expressing "at all future times". They are also found as the letters **F** and **G**.

Chapter 5

P Systems Operational Semantics: A Small-Step Approach

In general rewrite systems have good properties for modelling (dynamical) biological processes [21]. A key result for rewriting systems is that they are *Turing complete*, i.e., every computable process can be described by a rewriting system.

The simulation of any dynamical system - and of dynamical biological systems in particular - by a rewrite system consists in:

- representing the states of the dynamical system by expressions (terms built from constants representing proteins or molecules and operations representing ways in which proteins and molecules can be brought together);
- expressing the evolution rules of the dynamical system as rewrite rules.

Finding appropriate constants, operations and rules is at the heart of building these computational models ([21]).

In [1] we present a Maude specification for P systems using a strategy defined at meta-level. Trying to reducing the complexity of the specification, we came to the conclusion that the need of using the meta-level in specifying the P systems arises only because of the encoding of the evolution rules as Maude rewriting rules. Consequently, using an appropriate new defined data type for evolution rules (pairs of terms, for example), we are able to encode transition P systems in Maude at the object level.

In this chapter we present a Maude specification (or rewrite theory) for P systems. Benefiting from Maude features, these specifications not only permit to describe P systems, but also to explore the models (executability), and, moreover, to reason about them (Maude LTL model checker). These features are illustrated by means of two examples: both of them computes n^2, but one has n as input, while the other generates values of the form n^2 for $1 < n + n^2 < max$, with max as input (we explain later this upper bound).

5.1 Encoding P Systems in Rewriting Logic

5.1.1 Algebraic Structures

We consider the sorts `Obj`, `Soup`, `Membrane`, `CConfiguration` for object names, multisets of ingredients, membranes and commited configurations, respectively. Their definitions and the relations between them are introduced by the following (simplified) functional module:

```
(mod PSCONFIGURATION is
  pr QID .
  sort Label . subsort Label < Qid .
  sorts Obj Soup EmptyMembraneSet Membrane MembraneSet CConfiguration .
  subsort Obj < Soup .
  subsort EmptyMembraneSet Membrane < MembraneSet .

  op empty : -> Soup .
  op __ : Soup Soup -> Soup [assoc comm id: empty] .

  op empty : -> EmptyMembraneSet .

  op <_|_> : Label Soup -> Membrane .
  op _`,_ : MembraneSet MembraneSet -> MembraneSet [assoc comm id: empty] .
  op <_|_;_> : Label Soup MembraneSet -> Membrane .

  op `{_`} : Membrane -> CConfiguration .
endm)
```

The sort `Soup` represents the multiset type with two constructors: `empty` and _ _. The second constructor is required to satisfy the structural laws of associativity, commutativity, and it has the identity `empty`. The subsort relation `Obj < Soup` says that each object defines a particular multiset. The sort `Membrane` has two constructors for the two types of membranes, elementary and composite ones. The operation _‘, _ on a set of membranes is required to satisfy the structural laws of associativity, commutativity and identity because the order of sibling membranes is irrelevant; what matters is the relationship between membranes, if one is inside another one. An expression of the form $\langle L \mid W \rangle$ corresponds to a state of an elementary membrane labelled by L and having the multiset of objects given by W, while an expression of the form $\langle L \mid W \ ; \ M_1, \ldots, M_n \rangle$ corresponds to a composite membrane labelled by L with the current multiset of objects described by W and the state of the i-th component given by M_i. The sort `CConfiguration` corresponds to a P system commited configuration; it has no subsorts and its constructor $\{\_\} : Membrane \rightarrow CConfiguration$ has as the only argument the skin membrane. The intended meaning consists in starting our maximal parallel rewriting strategy only from the skin membrane in order to ensure the processing of all membranes of the P system during a transition step.

For the communications of objects through membranes, we define the sort `Target` together with two operations for sending objects *out* of the membrane or *in* a specified membrane, such that a pair composed of a multiset of objects and a target represents an element of sort `Message`. We omit the target indication *here*, and an objects without an explicit target indication is supposed to remain in the same region where the rule is applied.

In order to deal with the membrane dissolving we add a new sort, `Dissolve`, and a constant operation corresponding to the dissolving element δ.

For the evolution rules we introduce the sort `MRule` together with two operations corresponding to rules with or without priorities:

```
op _:_->_ : Priority Soup Soup -> MRule .
op _->_ : Soup Soup -> MRule .
```

where `Priority` is a subsort of `Qid`, therefore is an identifier. We add a supersort of `MRule`, namely `MRuleSet`, used for giving the evolution rules associated to a membrane, with a concatenation operation $_`,\_$ which is associative, commutative, and has an identity element `none`. For example a set of rules is given through the operator `rules` as follows:

```
eq rules(M1) = (r1 : c -> (c, in(M4))) , (r2 : c -> (b, in(M4))) ,
               (r3 : a -> (a, in(M2)) b) , (d d -> (a, in(M4)))  .
```

In order to extract the components of a rule description we define 3 operators, `prty`, `lhs`, and `rhs`, with the obvious meaning.

The partial order relation over rule priorities in a membrane is given by the operator `rho` through set of pairs of related priorities. For example the following equation describes the priority relation in a membrane identified by the label `M1`:

```
eq rho(M1) = (r1 > r3) , (r2 > r3) .
```

We make intensive use of the order sorted equation logic implemented by Maude. Therefore we consider the sort `Soup` (for multisets of objects) to be a subsort of `DissSoup` (for multisets of objects and δ's), while `DissSoup` is a subsort of `MsgDissSoup` (for multisets of objects, δ's, and messages). All these three sorts are subsorts of the sort `Soup?`. We also define new membrane sorts, `DMembrane`, `Membrane+` and `Membrane?`, corresponding to the three new sorts, `DissSoup`, `MsgDissSoup`, and `Soup?`, respectively, as well as their corresponding sorts for non-empty membrane sets. Moreover, we define a supersort of `CConfiguration`, namely the sort `Configuration?`, extended with intermediate configurations.

5.1.2 Maude Evolution Rules

As we already emphasized, a specific feature of a P system is that it has a tree like structure with the skin as its root, the composite membranes as its internal nodes and the elementary membranes as its leaves. The order of the children of a node is not important due to the associativity and commutativity properties of the concatenation operation for membranes $\_`,\_$.

We consider two colors (operators) *blue* and *green* in order to define the semantics of a P system in Maude:

```
  op blue : Membrane -> Membrane? .
  op blue : MembraneSet -> MembraneSet? .
  op blue : Label Soup Bool -> Soup? .
  op blue : Label MRuleSet Soup Bool -> Soup?.

  op green : Membrane+ -> Membrane? .
  op green : MembraneSet+ -> MembraneSet? .
  op green : MsgDissSoup -> Soup? .
```

The following rule marks the commitment of the the maximal parallel rewriting step:

```
  rl [1] : { M } => { blue(M) } .
```

starting with a term of sort `CConfiguration`.

The operator `blue` traverses in a top-down manner the tree according to its structure, firing the maximal parallel rewriting process in every membrane through the `blue` operator on `Soup` terms. The Boolean argument is used to show if a dissolving rule was chosen during the current maximal parallel step in a membrane with dissolving rules. We need this flag because at most one δ symbol is allowed in a membrane.

```
  crl [3] : blue(< L | S ; neM >) => if PS == noprty
          then
               (if (freeDeltaRules(L) == none)
               then < L | blue(L, S) ; blue(neM) >
               else < L | blue(L, S, false) ; blue(neM) >
               fi)
          else
              (if (freeDeltaRules(L) == none)
              then < L | blue(L, PS, S) ; blue(neM) >
              else < L | blue(L, PS, S, false) ; blue(neM) >
              fi)
          fi
    if PS := priorities(rules(L)) .

  crl [4] : blue(< L | S >) => if PS == noprty
          then
               (if (freeDeltaRules(L) == none)
               then < L | blue(L, S) >
               else < L | blue(L, S, false) >
               fi)
          else
               (if (freeDeltaRules(L) == none)
               then < L | blue(L, PS, S) >
               else < L | blue(L, PS, S, false) >
               fi)
          fi
      if PS := priorities(rules(L)) .
  crl [5] :  blue(neM) => blue(neM1), blue(neM2)
    if neM1, neM2 := neM .
```

where the operator `freeDeltaRules` provides the non-dissolving evolution rules of a membrane.

The multiset of objects is divided into two parts during the maximal parallel rewriting process: *blue* represents the objects available to be "consumed" via evolution rules, while the *green* represents the objects resulted from applying evolution rules over the available objects (therefore the green objects are not available). When no more rule can be applied, the blue part becomes directly green.

```
crl [6] : blue(L, S) =>
             green(rhs(R)) blue(L, S1)
    if R, RS := rules(L) /\ S2 := lhs(R) /\ S2 S1 := S .
crl [7] : blue(L, S) => green(S)
    if irreducible(L, S, rules(L)) .
```

The predicate `irreducible` returns true iff the left-hand side of each evolution rule from the membrane with the label `L` does not match the given soup of objects.

For evolution rules with priorities we have adopted a *stronger* version: if a rule with a higher priority is used, then no rule of a lower priority can be used, even if the two rules do not compete for objects. First we introduces two more operators:

- `rmLower` - given a rule `Rl` and a set of rules `RlS` from a membrane, it removes from `RlS` the rules with a lower priority than of `Rl`'s;
- `allowed` - a matching rule can be applied if either it does not have a priority, or it has a maximal priority, or it has a priority and any rule with a higher priority does not match the current soup of objects.

```
crl [8] : blue(L, PS, S) =>
              green(rhs(R)) blue(L, rmLower(L, prty(R), PS), S1)
    if R, RS1 := rules(L) /\ S2 := lhs(R)
       /\ S2 S1 := S  /\ allowed(L, S, R, RS1, PS) .
crl [9] : blue(L, PS, S) => green(S)
    if irreducible(L, PS, S, rules(L)) .
```

There are four more labelled rules 6'-9' (corresponding to rules 6-9 above) for the case when the membrane where the rewriting takes place has at least a dissolving rule. We consider this case because we do not allow two or more dissolving symbols δ to appear in a membrane.

The operator `green` traverses the tree in a bottom-up manner as follows:

- green multisets from $(O \cup Msg(O) \cup \{\delta\})^*_c$ merge into one green multiset;
- a leaf becomes green if its multiset is green;
- a set of sibling subtrees (with the roots sibling nodes) becomes green if each subtree is green;
- a subtree becomes entirely green if:
    1. the multiset of the root is green;
    2. the subtrees determined by the children of the root form a green set of sibling subtrees;

3. there are no messages to be exchanged between the root node and its children;
4. the multisets corresponding to the children of the root do not contain δ.

```
vars Smd Smd1 : MsgDissSoup . var Sd : DissSoup .
var Sod : OutDissSoup . var So : OutSoup . var S' : Soup? .
var neM' : NeMembraneSet? . vars neM1+ neM2+ : NeMembraneSet+ .
rl [10] : < L | green(Smd) >, neM' => green(< L | Smd >), neM' .
rl [11] : < L1 | S' ; < L | green(Smd) > > =>
          < L1 | S' ; green(< L | Smd >) > .
rl [12] : { < L | green(S) > } => { green(< L | S >) } .
rl [13] : < L | green(Sod) ; green(neM) > => green(< L | Sod ; neM >) .
rl [14] : green(Smd) green(Smd1) => green(Smd Smd1) .
rl [15] : green(neM1+), green(neM2+) => green((neM1+, neM2+)) .
```

In the communication stage of a green subtree, a node can send a message only to its parent or to one of its children. The rules for each direction of communication (*out* and *in*) vary on the structure of the destination membrane.

The rules for *in* messages are the following:

```
crl [16] : < L | green(Smd) ; green(neM+) > =>
           < L | green(Smd2) ; green(< L1 | Smd1 S >, M2+) >
  if Smd2 (S, in(L1)) := Smd /\ < L1 | Smd1 >, M2+ := neM+ .

crl [17] :  < L | green(Smd) ; green(neM+) > =>
            < L | green(Smd2) ; green(< L1 | Smd1 S ; neM1+ >, M2+) >
  if Smd2 (S, in(L1)) := Smd /\ < L1 | Smd1 ; neM1+ >, M2+ := neM+ .
```

An object sent out of the skin membrane is lost. We allow this operation to be executed only if the skin membrane and its internal membranes are green; in this way we follow the same communication policy as for the rest of the membranes. Therefore for *out* messages we have two additional rules for sending messages out of the skin membrane:

```
crl [18] : < L | green(Smd) ; green(neM+) > =>
           < L | green(Smd S) ; green(< L1 | Smd1 >, M2+) >
  if < L1 | Smd1 (S, out) >, M2+ := neM+ .
crl [19] : < L | green(Smd) ; green(neM+) > =>
           < L | green(Smd S) ; green(< L1 | Smd1 ; neM1+ >, M2+) >
  if < L1 | Smd1 (S, out) ; neM1+ >, M2+ := neM+ .
rl [20] : {< L | green(So (S, out)) >} => {< L | green(So) >} .
rl [21] : {< L | green(Smd (S, out)) ; green(neM+) >} =>
          {< L | green(Smd) ; green(neM+) >} .
```

In a P system the dissolving process occurs after the end of the communication process. To fulfill this condition we allow dissolving *only if* there are no messages to be sent (we use only variables of sort `DissSoup` and `DMembrane`). By dissolving a node, all of its resources are transferred to its parent, the rules are lost, and all of its children become children of its parent if it was an internal node. The skin membrane is not allowed to be dissolved. The rules for dissolving are:

```
crl [22] : < L | green(Sd) ; green(neDM) > =>
           < L | green(Sd S1) ; green(neDM1) >
    if < L1 | S1 delta >, neDM1 := neDM .

crl [23] : < L | green(Sd) ; green(neDM) > => < L | green(Sd S1) >
    if < L1 | S1 delta > := neDM .

crl [24] : < L | green(Sd) ; green(neDM) > =>
           < L | green(Sd S1) ; green((neDM1, DM2)) >
    if < L1 | S1 delta ; neDM1 >, DM2 := neDM .
```

When the whole tree becomes green, there are no more messages or nodes to be dissolved, and the following accomplishing rule is applied:

```
rl [2] : { green(M) } => { M } .
```

The resulting term corresponds to a configuration of the P system reachable in one transition step from the given configuration.

In order to apply the model checker, the space of the reachable states of the P system must be finite. Therefore we have to consider a finite subspace of reachable states. An example is the subspace including the states with the size less than, or equal to, a given `maxSize`. The size of a state is computed by the function `#(X)` which counts the number of the objects that occur in `X`. In this sense we modify the "firing" rule [1] replacing it by the following two rules:

```
crl [1] : { Mb } => { blue(Mb) }
    if maxSize =/= 0 /\ (#(Mb) < maxSize) .
crl [1'] : { Mb } => { blue(Mb) }
    if maxSize == 0 .
```

For each P system specification we give a value for `maxSize` as follows:

- if the P system is a generative one, then we consider a "reasonable" positive integer; hence the computation ends when it reaches commited configurations having the size greater than, or equal to `maxSize`;
- otherwise, if the P system is a non-generative one, deterministic, hence the computation will end, then we consider `maxSize` to be 0.

5.2 Examples of P System Specifications

In this section we consider two P systems examples, and then describe and execute their Maude specification. The first one computes n^2 for a given n, while the second one non-deterministically generates numbers of the form n^2, for $n \geq 1$.

Example 5.1 *(cont'd example 2.1)* Let Π_1 be the P system presented in Example 2.1.

The description of this P system in Maude for $n = 2$ is given by the folowing module:

```
(mod NSQUARE is
  inc PSYSTEM .

  ops r1 r2 : -> Priority .
  ops M1 M2 M3 M4 : -> Label .
  ops a c d f e : -> Obj .

  op init : -> CConfiguration .
  eq init = { < M1 | empty ; < M2 | empty ; < M3 | a a c f > ,
                       < M4 | empty > > > } .

  eq rules(M4) = none .
  eq freeDeltaRules(M4) = none .
  eq rho(M4) = empty .

  eq rules(M3) = (r1 : c a -> c d), (r2 : c -> delta), (f -> f f) .
  eq freeDeltaRules(M3) = (r1 : c a -> c d), (f -> f f) .
  eq rho(M3) = (r1 > r2) .

  eq rules(M2) = (d -> c), (c -> a), (a -> a (e, in(M4))),
                 (r1 : f f -> f), (r2 : f -> delta) .
  eq freeDeltaRules(M2) = (d -> c), (c -> a), (a -> a (e, in(M4))),
                          (r1 : f f -> f) .
  eq rho(M2) = (r1 > r2) .

  eq rules(M1) = none .
  eq freeDeltaRules(M1) = none .
  eq rho(M1) = empty .

  eq maxSize = 0 .
endm)
```

We can use Maude commands in order to "play" with the P system specification. For instance, we use the command `rew` to see the result of a given number of term rewriting:

```
Maude> (rew [117] init .)
rewrites: 3138 in 60ms cpu (60ms real) (51501 rewrites/second)
rewrite in NSQUARE :
  init
result Configuration? :
  {< M1 | green(empty); < M2 | green(a(e,in(M4)))blue(M2,r1 r2,a f f,false);
< M4 | blue(M4,prty(none),empty,false)> > >}
```

which is an intermediate configuration occurred during the maximal parallel rewriting step. The halting configuration is the following:

```
Maude> (rew [150] init .)
```

```
rewrites: 3861 in 54ms cpu (54ms real) (70302 rewrites/second)
rewrite in NSQUARE :
  init
result CConfiguration :
  {< M1 | a a ; < M4 | e e e e > >}
```

when the system has only the skin membrane and the output one, and the result is collected from the output membrane, M4: four objects, the square of the number of objects we started with.

A simpler solution is to use the command search which performs a breadth-first search for rewrite proofs starting with init to a final state that matches committed configurations having only skin and output membranes:

```
Maude> (search init =>+ { < M1 | S1:Soup ; < M4 | S2:Soup > > } .)
rewrites: 302521 in 4312ms cpu (4312ms real) (70152 rewrites/second)
search in NSQUARE : init =>+ {< M1 | S1:Soup ; < M4 | S2:Soup > >}.

Solution 1
S1:Soup <- a a ; S2:Soup <- e e e e

No more solutions.
```

The search command is very useful for nondeterministic systems when there might be more than one halting configuration.

Example 5.2 Let Π_2 be the P system Π_2 which non-deterministically generates values of the form n^2, for $n \geq 1$. This example is taken from [39], page 71.

$$\begin{aligned}
\Pi_2 &= (O, \mu, w_1, w_2, w_3, (R_1, \rho_1), (R_2, \rho_2), (R_3, \rho_3), 1), \\
O &= \{a, b, d, e, f\}, \\
\mu &= [_1[_2[_3]_3]_2]_1, \\
w_1 &= \lambda,\ R_1 = \emptyset,\ \rho_1 = \emptyset, \\
w_2 &= \lambda, \rho_2 = \{(r_1, r_2)\}, \\
R_2 &= \{b \to (d, here), d \to (de, here), r_1 : ff \to (f, here), r_2 : f \to \delta\}, \\
w_3 &= af, R_3 = \{a \to (ab, here), a \to (b, here)\delta, f \to (ff, here)\}, \rho_3 = \emptyset
\end{aligned}$$

The initial configuration is given in Figure 5.1.

Since no object is free in membranes 1 and 2, the only possibility to start is by using the rules of membrane 3 together with its free objects a and f. Using the rules $a \to (ab, here)$ and $f \to (ff, here)$ in parallel for the available occurrences of a and f, after $n \geq 1$ steps we get n occurrences of b and 2^n occurrences of f. At any moment we can use $a \to (b, here)\delta$ instead of $a \to ab$, and consequently we get $n+1$ occurrences of b and 2^{n+1} occurrences of f, followed by the dissolving of membrane 3. Region 3 disappears, its rules are lost, and its objects move to region 2. According to the priority relation, the rule $ff \to (f, here)$ is used as much as possible. In one step b^{n+1} are transformed in d^{n+1}, while the number of f occurrences is divided by two. Then,

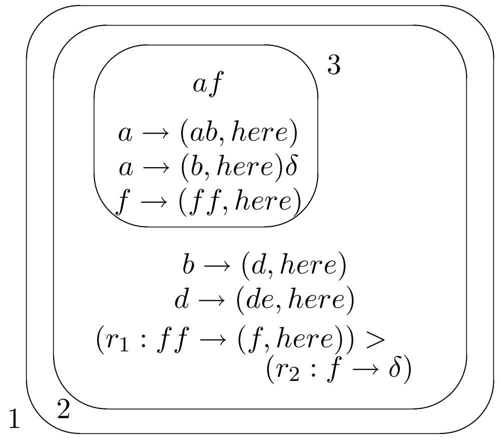

Figure 5.1: The initial configuration of a P system generating n^2

in the next step, $n+1$ occurrences of e are produced, and the number of f occurrences is divided again by two. At each step, further $n+1$ occurrences of e are produced. Finally, after $n+1$ steps (n steps when the rule $ff \to (f, here)$ is used, and one when using the rule $f \to \delta$), membrane 2 is dissolved, its rules are removed, and its objects move to the skin region. The number of the objects e is the square of the number of d. Consequently, if we count the objects e, then Π_2 generates values of the form n^2, for $n \geq 1$. But if we count the total number of objects from the output membrane (which is the skin membrane), then we get $n^2 + n$.

If we add the rule $r_3 : df \to (f, here)$ in membrane 2 such that $r_1 > r_3$ and $r_3 > r_2$, then we obtain in the halting configurations only square numbers, but we cannot anymore determine very easy (as above) that the resulted numbers are indeed square numbers.

The P system Π_2 has the following Maude specification:

```
(mod NSQUARE is
  inc PSYSTEM .

  ops r1 r2 : -> Priority .
  ops M1 M2 M3 : -> Label .
  ops a b d f e : -> Obj .

  op init : -> CConfiguration .
  eq init = { < M1 | empty ; < M2 | empty ; < M3 | a f > > > } .

  eq rules(M3) = (a -> a b), (a -> b delta), (f -> f f) .
  eq freeDeltaRules(M3) = (a -> a b), (f -> f f) .
  eq rho(M3) = empty .

  eq rules(M2) = (b -> d), (d -> d e), (r1 : f f -> f), (r2 : f -> delta) .
  eq freeDeltaRules(M2) = (b -> d), (d -> d e), (r1 : f f -> f) .
  eq rho(M2) = (r1 > r2) .

  eq rules(M1) = none .
  eq freeDeltaRules(M1) = none .
  eq rho(M1) = empty .

  eq maxSize = 15 .
endm)
```

The two intermediated configurations below are resulted after 25 and 45 term rewriting, respectively:

```
Maude> (rew [25] init .)
rewrites: 754 in 5ms cpu (5ms real) (125687 rewrites/second)
rewrite in NSQUARE :
  init
result Configuration? :
  {< M1 | green(empty); < M2 | green(empty); < M3 | green(f f)green(a
    b f f)blue(M3,b,false)> > >}

Maude> (rew [45] init .)
rewrites: 1012 in 6ms cpu (7ms real) (144592 rewrites/second)
rewrite in NSQUARE :
  init
result Configuration? :
  {< M1 | green(empty); < M2 | green(empty); < M3 | green(a b f f f f
    f f) blue(M3,b b f,false)> > >}
```

The following command gives the subspace of reachable committed configurations (containing at most `maxSize` objects) starting from the initial configuration:

```
Maude> (search init =>+ C:CConfiguration .)
```

Due to its large size, the result of this command is presented in Appendix A.

However, we are interested only in the halting configurations, especially in the output membranes from the halting configurations. The command below filters the subspace of reachable committed configurations searching only the halting configurations:

```
Maude> (search init =>+ { < M1 | S:Soup > } .)
rewrites: 228973 in 1656ms cpu (1656ms real) (138206 rewrites/second)
search in NSQUARE : init =>+ {< M1 | S:Soup >}.

Solution 1
S:Soup <- d e

Solution 2
S:Soup <- d d e e e e
```

```
Solution 3
S:Soup <- d d d e e e e e e e e e

No more solutions.
```

Therefore there are only three halting configurations in the subspace considered, and they give the result for the computation of n^2 for $n = 1, 2, 3$.

We illustrate the use of LTL model checker by verifying two atomic propositions:

- `isHalting`, satisfied by a configuration if it is a committed configuration, and it consists of the skin as an elementary membrane (it must be an elementary membrane because the skin is the output membrane);
- `isSquare`, satisfied by a configuration if the number of objects `e` is the square of the number of objects `d`.

```
(mod N2-PREDS is
  including NSQUARE .
  including SATISFACTION .
  subsort Configuration? < State .
  ops isHalting isSquare : -> Prop .
  var C : Configuration? .

  ceq C |= isHalting = true
        if { < M1 | S:Soup > } := C .

  ceq C |= isSquare = true
        if #(C, d) * #(C, d) == #(C, e) .
endm)

(mod PROOF is
  including N2-PREDS .
  including MODEL-CHECKER .
  including LTL-SIMPLIFIER .
endm)
```

We check the temporal formula $\Box$`(isHalting -> isSquare)` by giving the following `red` command:

```
Maude> (red modelCheck(init, [](isHalting -> isSquare)) .)
rewrites: 235072 in 1829ms cpu (1830ms real) (128474 rewrites/second)
reduce in PROOF :
  modelCheck(init,[](isHalting -> isSquare))
result Bool :
  true
```

If we slightly modify the `isSquare` atomic proposition:

```
  ceq C |= isSquare = true
      if #(C, d) * #(C, d) == #(C, e) + 1 .
```

then we get a counter-example. This is also detailed in Appendix A.

Chapter 6

Operational Correspondence

In this chapter we show how the dynamics of the P systems and their corresponding translation into Maude are related. Such a relationship between P systems operational semantics and Maude rewriting relation is given by an operational correspondence result.

Let Π be a P system of degree n, $n \geq 1$,

$$\Pi = (O, \mu, w_1, \dots, w_n, (R_1, \rho_1), \dots, (R_n, \rho_n), i_0)$$

having the initial configuration $\langle\, L_1 \,|\, w_1 \,;\, M_{i_1}, \dots, M_{i_n} \,\rangle$, $\{i_1, \dots, i_n\} \subseteq \{2, \dots, n\}$, with $rules(L_j) = R_j$, $priority(L_j) = \rho_j$, for all membrane labels L_j, and $\Rightarrow$ the transition relation between two configurations.

We associate to Π a rewrite theory $R(\Pi) = (\Sigma, E, L, R)$ in the way we presented in the previous chapter. Σ is the equational signature defining sorts and operation symbols, E is the set of Σ-equations that includes also the appropriate axioms for the associativity, commutativity and identity attributes of operators, L is the set of rule labels, and R is the set of rewriting rules. For two different P systems, their associated rewrite theories differ only in the signature, more precisely in the operators that describe the initial configuration of the system: object and priority names, membrane labels, initial multisets of objects, sets of rules and priority relations on them.

Considering $\rightarrow_{R(\Pi)}$ the rewriting relation, we denote by $\rightarrow^{+}_{R(\Pi)}$ the transitive closure of $\rightarrow_{R(\Pi)}$, and by $\rightarrow^{*}_{R(\Pi)}$ the reflexive and transitive closure.

We define an encoding function $\mathcal{I}_m : \mathcal{M}(\Pi) \rightarrow (T_{\Sigma,E})_{Membrane}$, from the set of membranes of Π to ground terms of sort `Membrane` from the associated rewrite theory, as follows:

- if $M = \langle\, L \,|\, w \,\rangle$, then $\mathcal{I}_m(M) =$ `< L | w >`
- if $M = \langle\, L \,|\, w \,;\, M_1, \dots, M_n \,\rangle$, then
  $\mathcal{I}_m(M) =$ `< L | w ;` $\mathcal{I}_m(M_1)$ `, ... ,` $\mathcal{I}_m(M_n)$ `>`

We extend the encoding function I_m over non-empty sets of sibling membranes by:

$$I_m(M_1, \dots, M_k) = I_m(M_1), \dots, I_m(M_k),\ k \geq 2$$

Restricting the domain of $\mathcal{I}_m$ to $\mathcal{C}(\Pi)$ we obtain an encoding function for configuration as ground terms of sort `CConfiguration`:

$$\mathcal{I} : \mathcal{C}(\Pi) \to (T_{\Sigma,E})_{CConfiguration}$$

such that $I(C) = \{I_m(C)\}$, for every $C \in \mathcal{C}(\Pi)$.

Definition 6.1 *Given two terms $t, t' \in (T_{\Sigma,E})_{CConfiguration}$, t* C-rewrites *to t', and this is denoted by $t \Rightarrow_{R(\Pi)} t'$, if $t \to^{+}_{R(\Pi)} t'$, and any intermediate term has the least sort* `Configuration?`*.*

We denote by $hasPrty(L)$ the truth value of the predicate `priorities(rules(L)) =/= none`, and by $hasDelta(L)$ the truth value of the predicate `freeDeltaRules(L) =/= none`. Therefore if L is the label of a membrane M, the two predicates give information about the existence of a priority relation on the evolution rules of M, and about the existence of evolution rules with δ in their righthand side, respectively.

For the sake of presentation we define a operator $blue^{\#}$ as follows:

$$blue^{\#}(L, w) = \begin{cases} blue(L, w) & , \text{ if } \neg hasPrty(L) \wedge \neg hasDelta(L) \\ blue(L, w, false) & , \text{ if } \neg hasPrty(L) \wedge hasDelta(L) \\ blue(L, priorities(rules(L)), w) & , \text{ if } hasPrty(L) \wedge \neg hasDelta(L) \\ blue(L, priorities(rules(L)), w, false) & , \text{ if } hasPrty(L) \wedge hasDelta(L) \end{cases}$$

We define a sequence of rule applications as follows:

- each label of a Maude rule is a sequence of rule applications;
- $s_1 + s_2$ is the application of one of the sequence s_1 and s_2;
- $s_1|s_2$ is the application of s_1 followed by s_2 or s_2 followed by s_1;
- if s is a sequence of rule applications, then $s^* = \underbrace{s|\ldots|s}_{k}$, with $k \geq 0$, and $s^+ = \underbrace{s|\ldots|s}_{k}$, with $k \geq 1$.

Proposition 6.1 *1. If w is a multiset of L-irreducible objects, then $blue^{\#}(L, w) \longrightarrow^{+} green(w)$ is a rewriting in $R(\Pi)$.*

2. If M_+ is mpr/tar/δ-irreducible, then $blue(I_m(M_+)) \longrightarrow^{+} green(I_m(M_+))$ is a rewriting in $R(\Pi)$.

Proof: For w a L-irreducible multiset of objects we have in $R(\Pi)$ the one-step rewriting $blue^{\#}(L, w) \xrightarrow{7+7'+9+9'} green(w)$. We prove the second property by structural induction:

- $M_+ = \langle\, L \mid w \,\rangle$. By definition, if M_+ is mpr-irreducible, then w is L-irreducible. Therefore the following is a rewriting in $R(\Pi)$:

$$blue(I_m(\langle\, L \mid w \,\rangle)) \xrightarrow{4} \langle\, L \mid blue^{\#}(L, w) \,\rangle \xrightarrow{7+7'+9+9'}$$
$$\langle\, L \mid green(w) \,\rangle \xrightarrow{10+11+12} green(I_m(\langle\, L \mid w \,\rangle))$$

- $M_+ = M_1, \ldots, M_n$, $n \geq 2$. By definition, for every $i \in [n]$ M_i is mpr-irreducible; then $\mathit{blue}(M_i) \longrightarrow^+ \mathit{green}(M_i)$, for every $i \in [n]$. Therefore the following is a rewriting in $R(\Pi)$:

$$\begin{aligned}&\mathit{blue}(I_m(M_+)) = \mathit{blue}(I_m(M_1), \ldots, I_m(M_n)) \xrightarrow{5^+}\\ &\mathit{blue}(I_m(M_1)), \ldots, \mathit{blue}(I_m(M_n)) \longrightarrow^+\\ &\mathit{green}(I_m(M_1)), \ldots, \mathit{green}(I_m(M_n)) \xrightarrow{15^+}\\ &\mathit{green}(I_m(M_1), \ldots, I_m(M_n)) = \mathit{green}(I_m(M_+))\end{aligned}$$

- $M_+ = \langle\, L \mid w \,;\, N_+ \,\rangle$. According to the definition of mpr-irreducibility, w is L-irreducible, and N is mpr-irreducible. From the first part of this proposition, $\mathit{blue}^{\#}(L, w) \xrightarrow{7+7'+9+9'} \mathit{green}(w)$ is a rewriting in $R(\Pi)$, while by inductive hypothesis we have $\mathit{blue}(N_+) \longrightarrow^+ \mathit{green}(N_+)$. Then the following is a rewriting in $R(\Pi)$:

$$\begin{aligned}&\mathit{blue}(I_m(\langle\, L \mid w \,;\, N_+ \,\rangle)) \xrightarrow{3} \langle\, L \mid \mathit{blue}^{\#}(L, w) \,;\, \mathit{blue}(I_m(N_+)) \,\rangle \longrightarrow^+\\ &\langle\, L \mid \mathit{green}(w) \,;\, \mathit{green}(I_m(N_+)) \,\rangle \xrightarrow{13} \mathit{green}(I_m(\langle\, L \mid w \,;\, N_+ \,\rangle))\end{aligned}$$

Proposition 6.2 *If $w \overset{mpr}{\Longrightarrow}_L w'$, then $\mathit{blue}^{\#}(L, w) \longrightarrow^+ \mathit{green}(w')$.*

Proof: If $w \overset{mpr}{\Longrightarrow}_L w'$, then by axiom (R_1) we have $w = u_1 \ldots u_n z$, $(u_1 \longrightarrow v_1, \ldots, u_n \longrightarrow v_n)$ (L, w)-consistent, z L-irreducible, and $w' = v_1 \ldots v_n z$.
(i) If $\neg \mathit{hasPrty}(L) \wedge \neg \mathit{hasDelta}(L)$, then, by choosing the same evolution rules in Maude, we get the following rewrite in $R(\Pi)$:

$$\begin{aligned}&\mathit{blue}(L, w) \xrightarrow{6} \mathit{green}(v_1)\mathit{blue}(L, u_2 \ldots u_n z) \xrightarrow{6} \ldots \xrightarrow{6}\\ &\mathit{green}(v_1) \ldots \mathit{green}(v_n)\mathit{blue}(L, z) \xrightarrow{7} \mathit{green}(v_1) \ldots \mathit{green}(v_n)\mathit{green}(z) \xrightarrow{14^*}\\ &\mathit{green}(v_1 \ldots v_n z) = \mathit{green}(w')\end{aligned}$$

(ii) If $\neg \mathit{hasPrty}(L) \wedge \mathit{hasDelta}(L)$, then the following is a rewrite in $R(\Pi)$:

$$\begin{aligned}&\mathit{blue}(L, w, \mathit{false}) \xrightarrow{6'} \mathit{green}(v_1)\mathit{blue}(L, u_2 \ldots u_n z, \mathit{false\ or\ hasDelta}(v_1)) \xrightarrow{6'} \ldots \xrightarrow{6'}\\ &\mathit{green}(v_1) \ldots \mathit{green}(v_n)\mathit{blue}(L, z, \mathit{false\ or\ hasDelta}(v_1)\ \mathit{or}\ \ldots\ \mathit{or\ hasDelta}(v_n)) \xrightarrow{7'}\\ &\mathit{green}(v_1) \ldots \mathit{green}(v_n)\mathit{green}(z) \xrightarrow{14^*} \mathit{green}(v_1 \ldots v_n z) = \mathit{green}(w')\end{aligned}$$

(iii) If $\mathit{hasPrty}(L) \wedge \neg \mathit{hasDelta}(L)$, then the following is a rewrite in $R(\Pi)$:

$$\begin{aligned}&\mathit{blue}(L, \mathit{priorities}(\mathit{rules}(L)), w) \xrightarrow{8}\\ &\mathit{green}(v_1)\mathit{blue}(L, \mathit{rmLower}(L, \mathit{prty}(u_1 \longrightarrow v_1), \mathit{priorities}(\mathit{rules}(L))), u_2 \ldots u_n z)\\ &\xrightarrow{8} \ldots \xrightarrow{8} \mathit{green}(v_1) \ldots \mathit{green}(v_n)\mathit{blue}(L, PS, z) \xrightarrow{9}\\ &\mathit{green}(v_1) \ldots \mathit{green}(v_n)\mathit{green}(z) \xrightarrow{14^*} \mathit{green}(v_1 \ldots v_n z) = \mathit{green}(w')\end{aligned}$$

For every application of the Maude rule 8, the second argument of the operator *blue*, namely a set of priorities, is changed by removing those priorities which are lower than

the priority of the chosen evolution rule.
(iv) If $hasPrty(L) \wedge hasDelta(L)$, then the following is a rewrite in $R(\Pi)$:

$$blue(L, priorities(rules(L)), w, false) \xrightarrow{8'}$$
$$green(v_1)blue(L, rmLower(L, prty(u_1 \longrightarrow v_1), priorities(rules(L)),$$
$$u_2 \dots u_n z, false\ or\ hasDelta(v_1))) \xrightarrow{8'} \dots \xrightarrow{8'}$$
$$green(v_1) \dots green(v_n)blue(L, PS, z, false\ or\ hasDelta(v_1)\ or\ \dots\ or\ hasDelta(v_n)) \xrightarrow{9'}$$
$$green(v_1) \dots green(v_n)green(z) \xrightarrow{14^*} green(v_1 \dots v_n z) = green(w')$$

where PS represents the set of remaining priorities after successive applications of the operator $rmLower$.

Corollary 6.1 *If $w \overset{mpr}{\Longrightarrow}_L w'$, then $blue^{\#}(L, w) \longrightarrow^+ green(w')$, and it does not exist w'' such that $w'' \neq w'$ and $blue(L, w) \longrightarrow^+ green(w'') \longrightarrow^+ green(w')$.*

Proof: It results from the proof of proposition 6.2.

Proposition 6.3 *If $M_+ \overset{mpr}{\Longrightarrow} M'_+ \overset{tar}{\Longrightarrow} M''_+$ and M''_+ is δ-irreducible, then $blue(I_m(M_+)) \longrightarrow^+ green(I_m(M''_+))$.*

Proof: The inference of $M_+ \overset{mpr}{\Longrightarrow} M'_+ \overset{tar}{\Longrightarrow} M''_+$ can be represented as a tree. We prove $blue(I_m(M_+)) \longrightarrow^+ green(I_m(M''_+))$ by induction on the depth of this inference tree. We consider all possible cases for the final step of the inference $M'_+ \overset{tar}{\Longrightarrow} M''_+$:

(i) $M'_+ \overset{tar}{\Longrightarrow} M''_+$ is inferred by (C_1). Then $M'_+ = \langle\ L \mid w'\ ;\ M'_1, \dots, M'_n\ \rangle$, $M''_+ = \langle\ L \mid w''\ ;\ M''_1, \dots, M''_n\ \rangle$ where $M'_1, \dots, M'_n$ is a tar-irreducible set of membranes, and $(\exists i) in_{L(M'_i)}(w') here(w') here(w(M''_i)) out(w(M''_i)) \neq empty$.
We have the following rewrite in $R(\Pi)$:

$$blue(I_m(M_+)) \longrightarrow^+ \langle\ L \mid green(w'_h)\ ;\ green(M'_1, \dots, M'_n)\ \rangle = t$$

where w'_h is obtained from w' by disclosing the *here* messages. If there are only *here* messages, as we consider this target implicit in the Maude implementation, then $M''_+ = M'_+$, and the above rewriting is continued by:

$$t \xrightarrow{13} green(\langle\ L \mid w'_h\ ;\ M'_1, \dots, M'_n\ \rangle) = green(I_m(M''_+)$$

Next we consider the case when there are *in* or *out* messages. By sending the messages in the corresponding membranes we obtain the following rewrite in $R(\Pi)$:

$$blue(I_m(M_+)) \longrightarrow^+ \langle\ L \mid green(w'_h)\ ;\ green(M'_1, \dots, M'_n)\ \rangle \xrightarrow{16^*|17^*|18^*|19^*}$$
$$\langle\ L \mid green(w'')\ ;\ green(M''_1, \dots, M''_n)\ \rangle \xrightarrow{13} green(\langle\ L \mid w''\ ;\ M''_1, \dots, M''_n\ \rangle) =$$
$$green(I_m(M''_+))$$

(ii) $M'_+ \overset{tar}{\Longrightarrow} M''_+$ is inferred by (C_2). Then $M'_+ = \langle\ L \mid w'\ ;\ M'_1, \ldots, M'_n\ \rangle$ and $M''_+ = \langle\ L \mid w''\ ;\ M'''_1, \ldots, M'''_n\ \rangle$, where $M'_1, \ldots, M'_n \overset{tar}{\Longrightarrow} M''_1, \ldots, M''_n$ is inferred by a shorter inference. Therefore $M''_1, \ldots, M''_n$ is tar-irreducible. By inductive hypothesis the following is a rewrite in $R(\Pi)$:

$$\begin{aligned}&blue(I_m(M_+)) = blue(\langle\ L \mid w\ ;\ M_1, \ldots, M_n\ \rangle) \longrightarrow^+\\ &\langle\ L \mid green(w'_h)\ ;\ green(M''_1, \ldots, M''_n)\ \rangle \xrightarrow{16^*|17^*|18^*|19^*|21^*}\\ &\langle\ L \mid green(w'')\ ;\ green(M'''_1, \ldots, M'''_n)\ \rangle \xrightarrow{13} green(\langle\ L \mid w''\ ;\ M'''_1, \ldots, M'''_n\ \rangle) =\\ &green(I_m(M''_+))\end{aligned}$$

(iii) $M'_+ \overset{tar}{\Longrightarrow} M''_+$ is inferred by (C_3). Then $M'_+ = \langle\ Skin \mid w'\ \rangle$ and $M''_+ = \langle\ Skin \mid obj(w')here(w')\ \rangle$, with $here(w')out(w') \neq empty$. If $out(w') = empty$ then, as the *here* messages were sent during the corresponding maximal parallel step in the Maude implementation, $M''_+ = M'_+$. Now we consider the case when $out(w') \neq empty$. Then the following is a rewrite in $R(\Pi)$:

$$\begin{aligned}&\{blue(I_m(M_+))\} = \{blue(\langle\ Skin \mid w\ \rangle)\} \longrightarrow^+ \{\langle\ Skin \mid green(w'_h)\ \rangle\} \xrightarrow{20^+}\\ &\{\langle\ Skin \mid green(w'')\ \rangle\} \xrightarrow{12} \{green(\langle\ L \mid w''\ \rangle)\} = \{green(I_m(M''_+))\}\end{aligned}$$

(iv) $M'_+ \overset{tar}{\Longrightarrow} M''_+$ is inferred by (C_4). Then $M'_+ = N', N'_+$ and $M''_+ = N'', N'_+$, where $N' \overset{tar}{\Longrightarrow} N''$ is inferred by a shorter inference, and N'_+ is tar-irreducible. Let N be a membrane such that $N \overset{mpr}{\Longrightarrow} N'$. Then by inductive hypothesis the following is a rewrite in $R(\Pi)$:

$$blue(I_m(N)) \longrightarrow^+ green(I_m(N''))$$

We distinguish two subcases:
(a) $M_+ = N, N'_+$ and N'_+ is mpr-irreducible. Then by proposition 6.1 $blue(I_m(N'_+)) \longrightarrow^+ green(I_m(N'_+))$. Therefore the following is also a rewrite in $R(\Pi)$:

$$\begin{aligned}&blue(I_m(N, N'_+)) \xrightarrow{5} blue(I_m(N)), blue(I_m(N'_+)) \longrightarrow^+\\ &green(I_m(N'')), green(I_m(N'_+)) \xrightarrow{15} green(I_m(N'', N'_+))\end{aligned}$$

(b) $M_+ = N, N_+$ and $N_+ \overset{mpr}{\Longrightarrow} N'_+$ such that N'_+ contains no messages, but $\delta \in w(N'_+)$. Then by inductive hypothesis $blue(I_m(N_+)) \longrightarrow^+ green(I_m(N'_+))$. Therefore the following is a rewrite in $R(\Pi)$:

$$\begin{aligned}&blue(I_m(N, N_+)) \xrightarrow{5} blue(I_m(N)), blue(I_m(N_+)) \longrightarrow^+\\ &green(I_m(N'')), green(I_m(N'_+)) \xrightarrow{15} green(I_m(N'', N'_+))\end{aligned}$$

(v) $M'_+ \overset{tar}{\Longrightarrow} M''_+$ is inferred by (C_5). Then $M'_+ = N', N'_+$ and $M''_+ = N'', N''_+$, where $N' \overset{tar}{\Longrightarrow} N''$ and $N'_+ \overset{tar}{\Longrightarrow} N''_+$ are inferred by shorter inferences. Let $M_+ = N, N_+$

such that $N \overset{mpr}{\Longrightarrow} N'$ and $N_+ \overset{mpr}{\Longrightarrow} N'_+$. Then by inductive hypothesis we obtain the following rewrites in $R(\Pi)$:

$$blue(I_m(N)) \longrightarrow^+ green(I_m(N''))$$
$$blue(I_m(N_+)) \longrightarrow^+ green(I_m(N''_+))$$

By composing them, we obtain the following rewrite in $R(\Pi)$:

$$blue(I_m(N, N_+)) \xrightarrow{5} blue(I_m(N)), blue(I_m(N_+)) \longrightarrow^+$$
$$green(I_m(N'')), green(I_m(N''_+)) \xrightarrow{15} green(I_m(N'', N''_+))$$

Corollary 6.2 *If $M_+ \overset{mpr}{\Longrightarrow} M'_+ \overset{tar}{\Longrightarrow} M''_+$ and M''_+ is δ-irreducible then $blue(I_m(M_+)) \longrightarrow^+ green(I_m(M''_+))$, and it does not exist M^0_+ such that $M^0_+ \neq M''_+$ and $blue(M_+) \longrightarrow^+ green(M^0_+) \longrightarrow^+ green(M''_+)$.*

Proposition 6.4 *If $M_+ \overset{mpr}{\Longrightarrow} M'_+ \overset{\delta}{\Longrightarrow} M''_+$, then $blue(I_m(M_+)) \longrightarrow^+ green(I_m(M''_+))$.*

Proof: The inference of $M_+ \overset{mpr}{\Longrightarrow} M'_+ \overset{\delta}{\Longrightarrow} M''_+$ can be represented as a tree. We prove $blue(I_m(M_+)) \longrightarrow^+ green(I_m(M''_+))$ by induction on the depth of this inference tree. We consider all possible cases for the final step of the inference $M'_+ \overset{\delta}{\Longrightarrow} M''_+$:

(i) $M'_+ \overset{\delta}{\Longrightarrow} M''_+$ is inferred by (D_1). Then $M'_+ = \langle\, L_1 \mid w'_1 \;;\; \langle\, L_2 \mid w'_2\delta \,\rangle\,\rangle$ and $M''_+ = \langle\, L_1 \mid w'_1 w'_2 \,\rangle$. Let $M_+ = \langle\, L_1 \mid w_1 \;;\; \langle\, L_2 \mid w_2 \,\rangle\,\rangle$. The following is a rewrite in $R(\Pi)$:

$$blue(I_m(M_+)) \xrightarrow{3} \langle\, L_1 \mid blue^{\#}(L_1, w_1) \;;\; blue(\langle\, L_2 \mid w_2 \,\rangle) \,\rangle \longrightarrow^+$$
$$\langle\, L_1 \mid green(w'_1) \;;\; green(\langle\, L_2 \mid w'_2\delta \,\rangle) \,\rangle \xrightarrow{23} \langle\, L_1 \mid green(w'_1 w'_2) \,\rangle \xrightarrow{10+11+12}\longrightarrow$$
$$green(\langle\, L_1 \mid w'_1 w'_2 \,\rangle) = green(I_m(M''_+))$$

(ii) $M'_+ \overset{\delta}{\Longrightarrow} M''_+$ is inferred by (D_2). Then $M'_+ = \langle\, L_1 \mid w'_1 \;;\; \langle\, L_2 \mid w'_2\delta \;;\; N'_+ \,\rangle\,\rangle$ and $M''_+ = \langle\, L_1 \mid w'_1 w'_2 \;;\; N'_+ \,\rangle$, with $\langle\, L_2 \mid w'_2\delta \;;\; N'_+ \,\rangle$ δ-irreducible. Let $M_+ = \langle\, L_1 \mid w_1 \;;\; \langle\, L_2 \mid w_2 \;;\; N_+ \,\rangle\,\rangle$. As N_+ is tar- and δ-irreducible, it follows that is also mpr-irreducible, then $N'_+ = N_+$; hence, we have by proposition 6.1 the rewrite $blue(N_+) \longrightarrow^+ green(N_+)$. Then the following is a rewrite in $R(\Pi)$:

$$blue(I_m(M_+)) \xrightarrow{3} \langle\, L_1 \mid blue^{\#}(L_1, w_1) \;;\; blue(\langle\, L_2 \mid w_2 \;;\; N_+ \,\rangle) \,\rangle \longrightarrow^+$$
$$\langle\, L_1 \mid green(w'_1) \;;\; blue(\langle\, L_2 \mid w_2 \;;\; N_+ \,\rangle) \,\rangle \xrightarrow{3}$$
$$\langle\, L_1 \mid green(w'_1) \;;\; \langle\, L_2 \mid blue^{\#}(L_2, w_2) \;;\; blue(N_+) \,\rangle\,\rangle \longrightarrow^+$$
$$\langle\, L_1 \mid green(w'_1) \;;\; \langle\, L_2 \mid green(w'_2\delta) \;;\; green(N'_+) \,\rangle\,\rangle \xrightarrow{13}$$
$$\langle\, L_1 \mid green(w'_1) \;;\; green(\langle\, L_2 \mid w'_2\delta \;;\; N'_+ \,\rangle) \,\rangle \xrightarrow{24}$$
$$\langle\, L_1 \mid green(w'_1 w'_2) \;;\; green(N'_+) \,\rangle \xrightarrow{13} green(\langle\, L_1 \mid w'_1 w'_2 \;;\; N'_+ \,\rangle) =$$
$$green(I_m(M''_+))$$

(iii) $M'_+ \overset{\delta}{\Longrightarrow} M''_+$ is inferred by (D_3). Then $M'_+ = \langle\, L_1 \,|\, w'_1\,;\, \langle\, L_2 \,|\, w'_2\,;\, N'_+ \,\rangle\,\rangle$ and $M''_+ = \langle L_1 | w'_1; \langle L_2 | w''_2\rangle\rangle$, where $\langle L_2 | w'_2; N'_+\rangle \overset{\delta}{\Longrightarrow} \langle L_2 | w''_2\rangle$ is inferred by a shorter inference, and w'_2 is δ-free (therefore w''_2 is also δ-free). Let $M_+ = \langle\, L_1 \,|\, w_1\,;\, \langle\, L_2 \,|\, w_2\,;\, N_+ \,\rangle\,\rangle$. Then $\langle\, L_2 \,|\, w_2\,;\, N_+ \,\rangle \overset{mpr}{\Longrightarrow} \langle\, L_2 \,|\, w'_2\,;\, N'_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L_2 \,|\, w''_2 \,\rangle$. By inductive hypothesis $\mathit{blue}(I_m(\langle\, L_2 \,|\, w_2\,;\, N_+ \,\rangle)) \longrightarrow^+ \mathit{green}(I_m(\langle\, L_2 \,|\, w''_2 \,\rangle))$. Therefore the following is a rewrite in $R(\Pi)$:

$$\begin{aligned}&\mathit{blue}(I_m(M_+)) \xrightarrow{3} \langle\, L_1 \,|\, \mathit{blue}^{\#}(L_1, w_1)\,;\, \mathit{blue}(I_m(\langle\, L_2 \,|\, w_2\,;\, N_+ \,\rangle))\,\rangle \longrightarrow^+\\ &\langle\, L_1 \,|\, \mathit{green}(w'_1)\,;\, \mathit{green}(I_m(\langle\, L_2 \,|\, w''_2 \,\rangle))\,\rangle \xrightarrow{13}\\ &\mathit{green}(I_m(\langle\, L_1 \,|\, w'_1\,;\, \langle\, L_2 \,|\, w''_2 \,\rangle\,\rangle)) = \mathit{green}(I_m(M''_+))\end{aligned}$$

(iv) $M'_+ \overset{\delta}{\Longrightarrow} M''_+$ is inferred by (D_4). Then $M'_+ = \langle\, L_1 \,|\, w'_1\,;\, \langle\, L_2 \,|\, w'_2\,;\, N'_+ \,\rangle\,\rangle$ and $M''_+ = \langle L_1 | w'_1; \langle L_2 | w''_2; N''_+\rangle\rangle$, where $\langle L_2 | w'_2; N'_+\rangle \overset{\delta}{\Longrightarrow} \langle L_2 | w''_2; N''_+\rangle$ is inferred by a shorter inference, and w'_2 is δ-free (therefore w''_2 is also δ-irreducible). Let $M_+ = \langle L_1 | w_1; \langle L_2 | w_2; N_+\rangle\rangle$. Then $\langle L_2 | w_2; N_+\rangle \overset{mpr}{\Longrightarrow} \langle\, L_2 \,|\, w'_2\,;\, N'_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle L_2 | w''_2; N''_+\rangle$. By inductive hypothesis $\mathit{blue}(I_m(\langle L_2 \,|\, w_2; N_+\rangle)) \longrightarrow^+ \mathit{green}(I_m(\langle L_2 \,|\, w''_2; N''_+\rangle))$. Therefore the following is a rewrite in $R(\Pi)$:

$$\begin{aligned}&\mathit{blue}(I_m(M_+)) \xrightarrow{3} \langle\, L_1 \,|\, \mathit{blue}^{\#}(L_1, w_1)\,;\, \mathit{blue}(I_m(\langle\, L_2 \,|\, w_2\,;\, N_+ \,\rangle))\,\rangle \longrightarrow^+\\ &\langle\, L_1 \,|\, \mathit{green}(w'_1)\,;\, \mathit{green}(I_m(\langle\, L_2 \,|\, w''_2\,;\, N''_+ \,\rangle))\,\rangle \xrightarrow{13}\\ &\mathit{green}(I_m(\langle\, L_1 \,|\, w'_1\,;\, \langle\, L_2 \,|\, w''_2\,;\, N''_+ \,\rangle\,\rangle)) = \mathit{green}(I_m(M''_+))\end{aligned}$$

(v) $M'_+ \overset{\delta}{\Longrightarrow} M''_+$ is inferred by (D_5). Then $M'_+ = \langle\, L_1 \,|\, w'_1\,;\, \langle\, L_2 \,|\, w'_2\delta\,;\, N'_+ \,\rangle\,\rangle$ and $M''_+ = \langle\, L_1 \,|\, w'_1 w''_2 \,\rangle$, where $\langle\, L_2 \,|\, w'_2\delta\,;\, N'_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L_2 \,|\, w''_2\delta \,\rangle$ is inferred by a shorter inference. Let $M_+ = \langle\, L_1 \,|\, w_1\,;\, \langle\, L_2 \,|\, w_2\,;\, N_+ \,\rangle\,\rangle$. Then by inductive hypothesis $\mathit{blue}(I_m(\langle\, L_2 \,|\, w_2\,;\, N_+ \,\rangle)) \longrightarrow^+ \mathit{green}(I_m(\langle\, L_2 \,|\, w''_2\delta \,\rangle))$. Therefore the following is a rewrite in $R(\Pi)$:

$$\begin{aligned}&\mathit{blue}(I_m(M_+)) \xrightarrow{3} \langle\, L_1 \,|\, \mathit{blue}^{\#}(L_1, w_1)\,;\, \mathit{blue}(I_m(\langle\, L_2 \,|\, w_2\,;\, N_+ \,\rangle))\,\rangle \longrightarrow^+\\ &\langle\, L_1 \,|\, \mathit{green}(w'_1)\,;\, \mathit{green}(I_m(\langle\, L_2 \,|\, w''_2\delta \,\rangle))\,\rangle \xrightarrow{23} \mathit{green}(\langle\, L_1 \,|\, w'_1 w''_2\,;\,)\,\rangle =\\ &\mathit{green}(I_m(M''_+))\end{aligned}$$

(vi) $M'_+ \overset{\delta}{\Longrightarrow} M''_+$ is inferred by (D_6). Then $M'_+ = \langle\, L_1 \,|\, w'_1\,;\, \langle\, L_2 \,|\, w'_2\delta\,;\, N'_+ \,\rangle\,\rangle$ and $M''_+ = \langle\, L_1 \,|\, w'_1 w''_2\,;\, N''_+\rangle$, where $\langle\, L_2 \,|\, w'_2\delta\,;\, N'_+ \rangle \overset{\delta}{\Longrightarrow} \langle\, L_2 \,|\, w''_2\delta\,;\, N''_+\rangle$ is inferred by a shorter inference. Let $M_+ = \langle L_1 | w_1; \langle L_2 | w_2; N_+\rangle\rangle$. Then by inductive hypothesis $\mathit{blue}(I_m(\langle L_2 \,|\, w_2; N_+\rangle)) \longrightarrow^+ \mathit{green}(I_m(\langle L_2 \,|\, w''_2\delta; N''_+\rangle))$. Therefore the following is a rewrite in $R(\Pi)$:

$$\begin{aligned}&\mathit{blue}(I_m(M_+)) \xrightarrow{3} \langle\, L_1 \,|\, \mathit{blue}^{\#}(L_1, w_1)\,;\, \mathit{blue}(I_m(\langle\, L_2 \,|\, w_2\,;\, N_+ \,\rangle))\,\rangle \longrightarrow^+\\ &\langle\, L_1 \,|\, \mathit{green}(w'_1)\,;\, \mathit{green}(I_m(\langle\, L_2 \,|\, w''_2\delta\,;\, N''_+ \,\rangle))\,\rangle \xrightarrow{24}\\ &\langle\, L_1 \,|\, \mathit{green}(w'_1 w''_2)\,;\, \mathit{green}(N''_+)\,\rangle \xrightarrow{13} \mathit{green}(\langle\, L_1 \,|\, w'_1 w''_2\,;\, N''_+ \,\rangle) = \mathit{green}(I_m(M''_+))\end{aligned}$$

(vii) $M'_+ \overset{\delta}{\Longrightarrow} M''_+$ is inferred by (D_7). Then $M'_+ = \langle\, L \mid w' \,;\, N^1_+, N^2_+ \,\rangle$ and $M''_+ = \langle\, L \mid w'' \,;\, N^2_+ \,\rangle$, where $\langle\, L \mid w' \,;\, N^1_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w'' \,\rangle$ is inferred by a shorter inference, and $\langle\, L \mid w' \,;\, N^2_+ \,\rangle$ is δ-irreducible (therefore the dissolving symbol δ does not occur in N^2_+). Let $M_+ = \langle\, L \mid w \,;\, N^1_+, N^2_+ \,\rangle$. By inductive hypothesis:
$\mathit{blue}(I_m(\langle\, L \mid w \,;\, N^3_+ \,\rangle)) \longrightarrow^+ \langle\, L \mid \mathit{green}(w') \,;\, \mathit{green}(N^3_+) \,\rangle \longrightarrow^+ \langle\, L \mid \mathit{green}(w'') \,\rangle$.
Therefore the following is a rewrite in $R(\Pi)$:

$$\mathit{blue}(I_m(M_+)) \xrightarrow{3} \langle\, L \mid \mathit{blue}^{\#}(L,w) \,;\, \mathit{blue}(I_m(N^1_+, N^2_+)) \,\rangle \xrightarrow{5}$$
$$\langle\, L \mid \mathit{blue}^{\#}(L,w) \,;\, \mathit{blue}(I_m(N^1_+)), \mathit{blue}(I_m(N^2_+)) \,\rangle \longrightarrow^+$$
$$\langle\, L \mid \mathit{green}(w') \,;\, \mathit{green}(N^3_+), \mathit{green}(N^4_+) \,\rangle \longrightarrow^+ \langle\, L \mid \mathit{green}(w'') \,;\, \mathit{green}(N^4_+) \,\rangle$$
$$\xrightarrow{13} \mathit{green}(Lw''N^4_+) = \mathit{green}(I_m(M''_+))$$

(viii) $M'_+ \overset{\delta}{\Longrightarrow} M''_+$ is inferred by (D_8). Then $M'_+ = \langle\, L \mid w' \,;\, N^3_+, N^4_+ \,\rangle$ and $M''_+ = \langle\, L \mid w'' \,;\, N^5_+, N^4_+ \,\rangle$, where $\langle\, L \mid w' \,;\, N^3_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w'' \,;\, N^5_+ \,\rangle$ is inferred by a shorter inference, and $\langle\, L \mid w' \,;\, N^4_+ \,\rangle$ is δ-irreducible. Let $M_+ = \langle\, L \mid w \,;\, N^1_+, N^2_+ \,\rangle$. Then $\langle\, L \mid w \,;\, N^1_+ \,\rangle \overset{mpr}{\Longrightarrow} \langle\, L \mid w' \,;\, N^3_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w'' \,;\, N^5_+ \,\rangle$. By inductive hypothesis:
$\mathit{blue}(I_m(\langle L|w;N^1_+\rangle)) \longrightarrow^+ \langle L|\mathit{green}(w');\mathit{green}(N^3_+)\rangle \longrightarrow^+ \langle L|\mathit{green}(w'');\mathit{green}(N^5_+)\rangle$.
Then the following is a rewrite in $R(\Pi)$:

$$\mathit{blue}(I_m(M_+)) \xrightarrow{3} \langle\, L \mid \mathit{blue}^{\#}(L,w) \,;\, \mathit{blue}(I_m(N^1_+, N^2_+)) \,\rangle \xrightarrow{5}$$
$$\langle\, L \mid \mathit{blue}^{\#}(L,w) \,;\, \mathit{blue}(I_m(N^1_+)), \mathit{blue}(I_m(N^2_+)) \,\rangle \longrightarrow^+$$
$$\langle\, L \mid \mathit{green}(w') \,;\, \mathit{green}(N^3_+), \mathit{green}(N^4_+) \,\rangle \longrightarrow^+$$
$$\langle\, L \mid \mathit{green}(w'') \,;\, \mathit{green}(N^5_+), \mathit{green}(N^4_+) \,\rangle \xrightarrow{15}$$
$$\langle\, L \mid \mathit{green}(w'') \,;\, \mathit{green}(N^5_+, N^4_+) \,\rangle \xrightarrow{13} \mathit{green}(\langle\, L \mid w'' \,;\, N^5_+, N^4_+ \,\rangle) =$$
$$\mathit{green}(I_m(M''_+))$$

(ix) $M'_+ \overset{\delta}{\Longrightarrow} M''_+$ is inferred by (D_9). Then $M'_+ = \langle\, L \mid w' \,;\, N^3_+, N^4_+ \,\rangle$ and $M''_+ = \langle\, L \mid w'w''w''' \,\rangle$, where $\langle\, L \mid w' \,;\, N^3_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w'w'' \,\rangle$ and $\langle\, L \mid w' \,;\, N^4_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w'w''' \,\rangle$ are inferred by shorter inferences. Let $M_+ = \langle L|w;N^1_+,N^2_+\rangle$. Then $\langle L|w;N^1_+\rangle \overset{mpr}{\Longrightarrow}$ $\langle\, L \mid w' \,;\, N^3_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w'w'' \,\rangle$ and $\langle\, L \mid w \,;\, N^2_+ \,\rangle \overset{mpr}{\Longrightarrow} \langle\, L \mid w' \,;\, N^4_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w'w''' \,\rangle$.
By inductive hypothesis we have the following two rewrites in $R(\Pi)$:

$$\mathit{blue}(I_m(\langle\, L \mid w \,;\, N^1_+ \,\rangle)) \longrightarrow^+ \langle\, L \mid \mathit{green}(w') \,;\, \mathit{green}(N^3_+) \,\rangle \longrightarrow^+$$
$$\langle\, L \mid \mathit{green}(w'w'') \,\rangle \xrightarrow{10+11+12} \mathit{green}(\langle\, L \mid w'w'' \,\rangle)$$

$$\mathit{blue}(I_m(\langle\, L \mid w \,;\, N^2_+ \,\rangle)) \longrightarrow^+ \langle\, L \mid \mathit{green}(w') \,;\, \mathit{green}(N^4_+) \,\rangle \longrightarrow^+$$
$$\langle\, L \mid \mathit{green}(w'w''') \,\rangle \xrightarrow{10+11+12} \mathit{green}(\langle\, L \mid w'w''' \,\rangle)$$

Then the following is also a rewrite in $R(\Pi)$:

$$\mathit{blue}(I_m(M_+)) \xrightarrow{3|5} \langle\, L \mid \mathit{blue}^{\#}(L,w) \,;\, \mathit{blue}(N^1_+), \mathit{blue}(N^2_+) \,\rangle \longrightarrow^+$$
$$\langle\, L \mid \mathit{green}(w') \,;\, \mathit{green}(N^3_+), \mathit{green}(N^4_+) \,\rangle \longrightarrow^+ \langle\, L \mid \mathit{green}(w'w''w''') \,\rangle \xrightarrow{10+11+12}$$
$$\mathit{green}(\langle\, L \mid w'w''w''' \,\rangle) = \mathit{green}(I_m(M''_+))$$

(x) $M'_+ \overset{\delta}{\Longrightarrow} M''_+$ is inferred by (D_{10}). Then $M'_+ = \langle\, L \mid w' \,;\, N^3_+, N^4_+ \,\rangle$ and $M''_+ = \langle\, L \mid w'w''w''' \,;\, N^5_+ \,\rangle$, where $\langle\, L \mid w' \,;\, N^3_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w'w'' \,\rangle$ and $\langle\, L \mid w' \,;\, N^4_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w'w''' \,;\, N^5_+ \,\rangle$ are inferred by shorter inferences. Let $M_+ = \langle\, L \mid w \,;\, N^1_+, N^2_+ \,\rangle$. Then $\langle\, L \mid w \,;\, N^1_+ \,\rangle \overset{mpr}{\Longrightarrow} \langle\, L \mid w' \,;\, N^3_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w'w'' \,\rangle$ and $\langle\, L \mid w \,;\, N^2_+ \,\rangle \overset{mpr}{\Longrightarrow} \langle\, L \mid w' \,;\, N^4_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w'w''' \,;\, N^5_+ \,\rangle$. By inductive hypothesis we have the following two rewrites in $R(\Pi)$:

$$\begin{aligned}&\mathit{blue}(I_m(\langle\, L \mid w \,;\, N^1_+ \,\rangle)) \longrightarrow^+ \langle\, L \mid \mathit{green}(w') \,;\, \mathit{green}(N^3_+) \,\rangle \longrightarrow^+ \\ &\langle\, L \mid \mathit{green}(w'w'') \,\rangle \xrightarrow{10+11+12} \mathit{green}(\langle\, L \mid w'w'' \,\rangle)\end{aligned}$$

$$\begin{aligned}&\mathit{blue}(I_m(\langle\, L \mid w \,;\, N^2_+ \,\rangle)) \longrightarrow^+ \langle\, L \mid \mathit{green}(w') \,;\, \mathit{green}(N^4_+) \,\rangle \longrightarrow^+ \\ &\langle\, L \mid \mathit{green}(w'w''') \,;\, \mathit{green}(N^5_+) \,\rangle \xrightarrow{13} \mathit{green}(\langle\, L \mid w'w''' \,;\, N^5_+ \,\rangle))\end{aligned}$$

Then the following is also a rewrite in $R(\Pi)$:

$$\begin{aligned}&\mathit{blue}(I_m(M_+)) \xrightarrow{3|5} \langle\, L \mid \mathit{blue}^{\#}(L, w) \,;\, \mathit{blue}(N^1_+), \mathit{blue}(N^2_+) \,\rangle \longrightarrow^+ \\ &\langle\, L \mid \mathit{green}(w') \,;\, \mathit{green}(N^3_+), \mathit{green}(N^4_+) \,\rangle \xrightarrow{15} \\ &\langle\, L \mid \mathit{green}(w') \,;\, \mathit{green}(N^3_+, N^4_+) \,\rangle \longrightarrow^+ \langle\, L \mid \mathit{green}(w'w'') \,;\, \mathit{green}(N^4_+) \,\rangle \longrightarrow^+ \\ &\langle\, L \mid \mathit{green}(w'w''w''') \,;\, \mathit{green}(N^5_+) \,\rangle \xrightarrow{13} \mathit{green}(\langle\, L \mid w'w''w''' \,;\, N^5_+ \,\rangle) = \\ &\mathit{green}(I_m(M''_+))\end{aligned}$$

(xi) $M'_+ \overset{\delta}{\Longrightarrow} M''_+$ is inferred by (D_{11}). Then $M'_+ = \langle\, L \mid w' \,;\, N^3_+, N^4_+ \,\rangle$ and $M''_+ = \langle L|w'w''w''';N^5_+, N^6_+\rangle$, where $\langle L|w';N^3_+\rangle \overset{\delta}{\Longrightarrow} \langle L|w'w'';N^5_+\rangle$ and $\langle L|w';N^4_+\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w'w''' \,;\, N^6_+ \,\rangle$ are inferred by shorter inferences. Let $M_+ = \langle\, L \mid w \,;\, N^1_+, N^2_+ \,\rangle$. Then $\langle\, L \mid w \,;\, N^1_+ \,\rangle \overset{mpr}{\Longrightarrow} \langle\, L \mid w' \,;\, N^3_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w'w'' \,;\, N^5_+ \,\rangle$ and $\langle\, L \mid w \,;\, N^2_+ \,\rangle \overset{mpr}{\Longrightarrow} \langle\, L \mid w' \,;\, N^4_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w'w''' \,;\, N^6_+ \,\rangle$. By inductive hypothesis we have the following two rewrites in $R(\Pi)$:

$$\begin{aligned}&\mathit{blue}(I_m(\langle\, L \mid w \,;\, N^1_+ \,\rangle)) \longrightarrow^+ \langle\, L \mid \mathit{green}(w') \,;\, \mathit{green}(N^3_+) \,\rangle \longrightarrow^+ \\ &\langle\, L \mid \mathit{green}(w'w'') \,;\, \mathit{green}(N^5_+) \,\rangle \xrightarrow{13} \mathit{green}(\langle\, L \mid w'w'' \,;\, N^5_+ \,\rangle)\end{aligned}$$

$$\begin{aligned}&\mathit{blue}(I_m(\langle\, L \mid w \,;\, N^2_+ \,\rangle)) \longrightarrow^+ \langle\, L \mid \mathit{green}(w') \,;\, \mathit{green}(N^4_+) \,\rangle \longrightarrow^+ \\ &\langle\, L \mid \mathit{green}(w'w''') \,;\, \mathit{green}(N^6_+) \,\rangle \xrightarrow{13} \mathit{green}(\langle\, L \mid w'w''' \,;\, N^6_+ \,\rangle))\end{aligned}$$

Then the following is also a rewrite in $R(\Pi)$:

$$\begin{aligned}&\mathit{blue}(I_m(M_+)) \xrightarrow{3|5} \langle\, L \mid \mathit{blue}^{\#}(L, w) \,;\, \mathit{blue}(N^1_+), \mathit{blue}(N^2_+) \,\rangle \longrightarrow^+ \\ &\langle\, L \mid \mathit{green}(w') \,;\, \mathit{green}(N^3_+), \mathit{green}(N^4_+) \,\rangle \xrightarrow{15} \\ &\langle\, L \mid \mathit{green}(w') \,;\, \mathit{green}(N^3_+, N^4_+) \,\rangle \longrightarrow^+ \\ &\langle\, L \mid \mathit{green}(w'w'') \,;\, \mathit{green}(N^5_+, N^4_+) \,\rangle \longrightarrow^+ \\ &\langle\, L \mid \mathit{green}(w'w''w''') \,;\, \mathit{green}(N^5_+, N^6_+) \,\rangle \xrightarrow{13} \mathit{green}(\langle\, L \mid w'w''w''' \,;\, N^5_+, N^6_+ \,\rangle) = \\ &\mathit{green}(I_m(M''_+))\end{aligned}$$

Corollary 6.3 *If $M_+ \overset{mpr}{\Longrightarrow} M'_+ \overset{\delta}{\Longrightarrow} M''_+$, then $blue(I_m(M_+)) \longrightarrow^+ green(I_m(M''_+))$, and it does not exist M^0_+ such that $M^0_+ \neq M''_+$ and $blue(M_+) \longrightarrow^+ green(M^0_+) \longrightarrow^+ green(M''_+)$.*

Proposition 6.5 *If $M_+ \overset{mpr}{\Longrightarrow} M'_+ \overset{tar}{\Longrightarrow} M''_+ \overset{\delta}{\Longrightarrow} M'''_+$, then $blue(I_m(M_+)) \longrightarrow^+ green(I_m(M'''_+))$.*

Proof: The inference of $M_+ \overset{mpr}{\Longrightarrow} M'_+ \overset{tar}{\Longrightarrow} M'' \overset{\delta}{\Longrightarrow} M'''_+$ can be represented as a tree. We prove $blue(I_m(M_+)) \longrightarrow^+ green(I_m(M'''_+))$ by induction on the depth of this inference tree. We consider all possible cases for the final step of the inference $M''_+ \overset{\delta}{\Longrightarrow} M'''_+$:

(i) $M''_+ \overset{\delta}{\Longrightarrow} M'''_+$ is inferred by (D_1). Then $M''_+ = \langle\, L_1 \mid w''_1 \,;\, \langle\, L_2 \mid w''_2\delta \,\rangle\,\rangle$ and $M'''_+ = \langle L_1 | w''_1 w''_2 \rangle$. Let $M_+ = \langle L_1 | w_1 ; \langle L_2 | w_2 \rangle\rangle$ and $M'_+ = \langle L_1 | w'_1 ; \langle L_2 | w'_2\delta \rangle\rangle$. Then we have in $R(\Pi)$:

$$blue(I_m(M_+)) \xrightarrow{3} \langle\, L_1 \mid blue^{\#}(L_1, w_1) \,;\, blue(\langle\, L_2 \mid w_2 \,\rangle) \,\rangle = t.$$

As $w \overset{mpr}{\Longrightarrow}_{L_1} w'$ and $\langle\, L_2 \mid w_2 \,\rangle \overset{mpr}{\Longrightarrow} \langle\, L_2 \mid w'_2\delta \,\rangle$, then the following two rewriting exist in $R(\Pi)$:

$$blue^{\#}(L_1, w_1) \longrightarrow^+ green(w')$$
$$blue(\langle\, L_2 \mid w_2 \,\rangle) \xrightarrow{3} \langle\, L_2 \mid blue^{\#}(w_2) \,\rangle \longrightarrow^+ \langle\, L_2 \mid green(w'_2\delta) \,\rangle$$

Therefore we can continue the first rewriting as follows:

$$t \longrightarrow^+ \langle\, L_1 \mid green(w'_1) \,;\, \langle\, L_2 \mid green(w'_2\delta) \,\rangle\,\rangle \xrightarrow{16^*|18^*|21^*}$$
$$\langle\, L_1 \mid green(w''_1) \,;\, \langle\, L_2 \mid green(w''_2\delta) \,\rangle\,\rangle \xrightarrow{23} \langle\, L_1 \mid green(w''_1 w''_2) \,\rangle \xrightarrow{10+11+12}$$
$$green(\langle\, L_1 \mid w''_1 w''_2 \,\rangle) = green(I_m(M'''_+))$$

(ii) $M''_+ \overset{\delta}{\Longrightarrow} M'''_+$ is inferred by (D_2). Then $M''_+ = \langle\, L_1 \mid w''_1 \,;\, \langle\, L_2 \mid w''_2\delta \,;\, N''_+ \,\rangle\,\rangle$ and $M''' = \langle\, L_1 \mid w''_1 w''_2 \,;\, N''_+ \,\rangle$, with $\langle\, L_2 \mid w''_2\delta \,;\, N''_+ \,\rangle$ δ-irreducible. Let $M_+ = \langle\, L_1 \mid w_1 \,;\, \langle\, L_2 \mid w_2 \,;\, N_+ \,\rangle\,\rangle$ and $M'_+ = \langle\, L_1 \mid w'_1 \,;\, \langle\, L_2 \mid w'_2 \,;\, N'_+ \,\rangle\,\rangle$. Then:

$$blue(I_m(M_+)) \xrightarrow{3} \langle\, L_1 \mid blue^{\#}(L_1, w_1) \,;\, blue(\langle\, L_2 \mid w_2 \,;\, N_+ \,\rangle) \,\rangle = t.$$

As $\langle\, L_2 \mid w_2 \,;\, N_+ \,\rangle \overset{mpr}{\Longrightarrow} ; \overset{tar}{\Longrightarrow} \langle\, L_2 \mid u_2\delta \,;\, N''_+ \,\rangle$ and $\langle\, L_2 \mid u_2\delta \,;\, N''_+ \,\rangle$ is δ-irreducible, by proposition 6.3:

$$blue(\langle\, L_2 \mid w_2 \,;\, N_+ \,\rangle) \longrightarrow^+ green(\langle\, L_2 \mid u_2\delta \,;\, N''_+ \,\rangle).$$

where w''_2 is obtained from u_2 by removing the *out* messages (if any), and adding the objects sent from L_1 (also, if any).

Therefore we continue the first rewrite in $R(\Pi)$ as follows:

$$t \longrightarrow^+ \langle\, L_1 \mid green(w'_1) \,;\, green(\langle\, L_2 \mid u_2\delta \,;\, N''_+ \,\rangle) \,\rangle \xrightarrow{17^*|19^*|21^*}$$
$$\langle\, L_1 \mid green(w''_1) \,;\, green(\langle\, L_2 \mid w''_2\delta \,;\, N''_+ \,\rangle) \,\rangle \xrightarrow{24}$$
$$\langle\, L_1 \mid green(w''_1 w''_2) \,;\, green(N''_+) \,\rangle \xrightarrow{13} green(\langle\, L_1 \mid w''_1 w''_2 \,;\, N''_+ \,\rangle) =$$
$$green(I_m(M'''_+))$$

(iii) $M''_+ \overset{\delta}{\Longrightarrow} M'''_+$ is inferred by (D_3). Then $M''_+ = \langle\, L_1 \mid w''_1 \,;\, \langle\, L_2 \mid w''_2 \,;\, N''_+ \,\rangle\,\rangle$ and $M'''_+ = \langle\, L_1 \mid w''_1 \,;\, \langle\, L_2 \mid w''_2 w_0 \,\rangle\,\rangle$, where $\langle\, L_2 \mid w''_2 \,;\, N''_+ \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L_2 \mid w''_2 w_0 \,\rangle$ is inferred by a shorter inference, and w_2'' is δ-free. Let $M_+ = \langle\, L_1 \mid w_1 \,;\, \langle\, L_2 \mid w_2 \,;\, N_+ \,\rangle\,\rangle$ and $M_+' = \langle\, L_1 \mid w_1' \,;\, \langle\, L_2 \mid w_2' \,;\, N_+' \,\rangle\,\rangle$. Then:

$$blue(I_m(M_+)) \xrightarrow{3} \langle\, L_1 \mid blue^{\#}(L_1, w_1) \,;\, blue(\langle\, L_2 \mid w_2 \,;\, N_+ \,\rangle) \,\rangle = t.$$

As $\langle\, L_2 \mid w_2 \,;\, N_+ \,\rangle \overset{mpr}{\Longrightarrow} \langle\, L_2 \mid w_2' \,;\, N_+ \,\rangle \overset{tar}{\Longrightarrow} \langle\, L_2 \mid u_2 \,;\, N_+'' \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L_2 \mid u_2 w_0 \,\rangle$, where w_2'' is obtained from u_2 by removing the *out* messages (if any), and adding the objects sent from L_1 (also, if any), then by inductive hypothesis:

$$blue(\langle\, L_2 \mid w_2 \,;\, N_+ \,\rangle) \longrightarrow^+ \langle\, L_2 \mid green(w_2') \,;\, green(N_+') \,\rangle \longrightarrow^* \\ \langle\, L_2 \mid green(u_2) \,;\, green(N_+'') \,\rangle \longrightarrow^+ \langle\, L_2 \mid u_2 w_0 \,\rangle$$

We continue the first rewrite above in $R(\Pi)$ as follows:

$$t \longrightarrow^+ \langle\, L_1 \mid green(w_1') \,;\, green(\langle\, L_2 \mid green(u_2 w_0) \,\rangle) \,\rangle \xrightarrow{17^*|19^*|21^*} \\ \langle\, L_1 \mid green(w_1'') \,;\, green(\langle\, L_2 \mid w_2'' w_0 \,\rangle) \,\rangle \xrightarrow{13} \\ green(\langle\, L_1 \mid w_1'' \,;\, \langle\, L_2 \mid w_2'' w_0 \,\rangle\,\rangle) = green(I_m(M_+'''))$$

(iv) $M_+'' \overset{\delta}{\Longrightarrow} M_+'''$ is inferred by (D_4). Then $M_+'' = \langle\, L_1 \mid w_1'' \,;\, \langle\, L_2 \mid w_2'' \,;\, N_+'' \,\rangle\,\rangle$ and $M_+''' = \langle\, L_1 \mid w_1'' \,;\, \langle\, L_2 \mid w_2'' w_0 \,;\, N_+''' \,\rangle\,\rangle$, where $\langle\, L_2 \mid w_2'' \,;\, N_+'' \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L_2 \mid w_2'' w_0 \,;\, N_+''' \,\rangle$ is inferred by a shorter inference, and w_2'' is δ-free. Let $M_+ = \langle L_1 | w_1 ; \langle L_2 | w_2 ; N_+ \rangle\rangle$ and $M_+' = \langle\, L_1 \mid w_1' \,;\, \langle\, L_2 \mid w_2' \,;\, N_+' \,\rangle\,\rangle$. Then:

$$blue(I_m(M_+)) \xrightarrow{3} \langle\, L_1 \mid blue^{\#}(L_1, w_1) \,;\, blue(\langle\, L_2 \mid w_2 \,;\, N_+ \,\rangle) \,\rangle = t.$$

As $\langle\, L_2 \mid w_2 \,;\, N_+ \,\rangle \overset{mpr}{\Longrightarrow} \langle\, L_2 \mid w_2' \,;\, N_+ \,\rangle \overset{tar}{\Longrightarrow} \langle\, L_2 \mid u_2 \,;\, N_+'' \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L_2 \mid u_2 w_0 \,\rangle$, where w_2'' is obtained from u_2 by removing the *out* messages (if any), and adding the objects sent from L_1 (also, if any), then by inductive hypothesis:

$$blue(\langle\, L_2 \mid w_2 \,;\, N_+ \,\rangle) \longrightarrow^+ \langle\, L_2 \mid green(w_2') \,;\, green(N_+') \,\rangle \longrightarrow^* \\ \langle\, L_2 \mid green(u_2) \,;\, green(N_+'') \,\rangle \longrightarrow^+ \langle\, L_2 \mid green(u_2 w_0) \,;\, green(N_+''') \,\rangle \xrightarrow{13} \\ green(\langle\, L \mid u_2 w_0 \,;\, N_+''' \,\rangle$$

The first rewrite above continues in $R(\Pi)$ as follows:

$$t \longrightarrow^+ \langle\, L_1 \mid green(w_1') \,;\, green(\langle\, L_2 \mid green(u_2 w_0) \,;\, N_+''' \,\rangle) \,\rangle \xrightarrow{17^*|19^*|21^*} \\ \langle\, L_1 \mid green(w_1'') \,;\, green(\langle\, L_2 \mid w_2'' w_0 \,;\, N_+''' \,\rangle) \,\rangle \xrightarrow{13} \\ green(\langle\, L_1 \mid w_1'' \,;\, \langle\, L_2 \mid w_2'' w_0 \,;\, N_+''' \,\rangle\,\rangle) = green(I_m(M_+'''))$$

(v) $M_+'' \overset{\delta}{\Longrightarrow} M_+'''$ is inferred by (D_5). Then $M_+'' = \langle\, L_1 \mid w_1'' \,;\, \langle\, L_2 \mid w_2'' \delta \,;\, N_+'' \,\rangle\,\rangle$ and $M_+''' = \langle L_1 | w_1'' w_2'' w_0 \rangle$, where $\langle L_2 | w_2'' \delta ; N_+'' \rangle \overset{\delta}{\Longrightarrow} \langle L_2 | w_2'' w_0 \delta \rangle$ is inferred by a shorter inference. Let $M_+ = \langle L_1 | w_1 ; \langle L_2 | w_2 ; N_+ \rangle\rangle$ and $M_+' = \langle L_1 | w_1' ; \langle L_2 | w_2' ; N_+' \rangle\rangle$. As $\langle\, L_2 \mid w_2 \,;\, N_+ \,\rangle \overset{mpr}{\Longrightarrow} \langle\, L_2 \mid w_2' \delta \,;\, N_+' \,\rangle \overset{tar}{\Longrightarrow} \langle\, L_2 \mid u_2 \delta \,;\, N_+'' \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L_2 \mid u_2 w_0 \delta \,\rangle$, where w_2'' is obtained from u_2 by removing the *out* messages (if any), and adding the objects sent from L_1 (also, if any), then by inductive hypothesis:

$$blue(\langle\, L_2 \mid w_2 \,;\, N_+ \,\rangle) \longrightarrow^+ \langle\, L_2 \mid green(w_2' \delta) \,;\, green(N_+') \,\rangle \longrightarrow^* \\ \langle\, L_2 \mid green(u_2 \delta) \,;\, green(N_+'') \,\rangle \longrightarrow^+ \langle\, L_2 \mid green(u_2 w_0 \delta) \,\rangle \xrightarrow{10+11+12} green(\langle\, L_2 \mid u_2 w_0 \delta \,\rangle)$$

Then the following is also a rewrite in $R(\Pi)$:

$$\begin{aligned}
&\mathit{blue}(I_m(M_+)) \xrightarrow{3} \langle\, L_1 \mid \mathit{blue}^{\#}(L_1, w_1)\ ;\ \mathit{blue}(\langle\, L_2 \mid w_2\ ;\ N_+\,\rangle)\,\rangle \longrightarrow^{+}\\
&\langle\, L_1 \mid \mathit{green}(w_1')\ ;\ \mathit{green}(\langle\, L_2 \mid u_2 w_0 \delta\,\rangle)\,\rangle \xrightarrow{16^*|18^*|21^*}\\
&\langle\, L_1 \mid \mathit{green}(w_1'')\ ;\ \mathit{green}(\langle\, L_2 \mid w_2'' w_0 \delta\,\rangle)\,\rangle \xrightarrow{23}\\
&\langle\, L_1 \mid \mathit{green}(w_1'' w_2'' w_0)\,\rangle \xrightarrow{10+11+12} \mathit{green}(\langle\, L_1 \mid w_1'' w_2'' w_0\,\rangle) = \mathit{green}(I_m(M_+'''))
\end{aligned}$$

(vi) $M_+'' \overset{\delta}{\Longrightarrow} M_+'''$ is inferred by (D_6). Then $M_+'' = \langle\, L_1 \mid w_1''\ ;\ \langle\, L_2 \mid w_2''\delta\ ;\ N_+''\,\rangle\,\rangle$ and $M_+''' = \langle\, L_1 \mid w_1'' w_2'' w_0\ ;\ N_+'''\,\rangle$, where $\langle\, L_2 \mid w_2''\delta\ ;\ N_+''\,\rangle \overset{\delta}{\Longrightarrow} \langle\, L_2 \mid w_2'' w_0 \delta\ ;\ N_+'''\,\rangle$ is inferred by a shorter inference. Let $M_+ = \langle\, L_1 \mid w_1\ ;\ \langle\, L_2 \mid w_2\ ;\ N_+\,\rangle\,\rangle$ and $M_+' = \langle\, L_1 \mid w_1'\ ;\ \langle\, L_2 \mid w_2'\ ;\ N_+'\,\rangle\,\rangle$. As $\langle\, L_2 \mid w_2\ ;\ N_+\,\rangle \overset{mpr}{\Longrightarrow} \langle\, L_2 \mid w_2'\delta\ ;\ N_+'\,\rangle \overset{tar}{\Longrightarrow} \langle\, L_2 \mid u_2\delta\ ;\ N_+''\,\rangle \overset{\delta}{\Longrightarrow} \langle\, L_2 \mid u_2 w_0 \delta\ ;\ N_+'''\,\rangle$, where w_2'' is obtained from u_2 by removing the *out* messages (if any), and adding the objects sent from L_1 (also, if any), then by inductive hypothesis:

$$\begin{aligned}
&\mathit{blue}(\langle\, L_2 \mid w_2\ ;\ N_+\,\rangle) \longrightarrow^{+} \langle\, L_2 \mid \mathit{green}(w_2'\delta)\ ;\ \mathit{green}(N_+')\,\rangle \longrightarrow^{*}\\
&\langle\, L_2 \mid \mathit{green}(u_2\delta)\ ;\ \mathit{green}(N_+'')\,\rangle \longrightarrow^{+} \langle\, L_2 \mid \mathit{green}(u_2 w_0 \delta)\ ;\ \mathit{green}(N_+''')\,\rangle \xrightarrow{13}\\
&\mathit{green}(\langle\, L_2 \mid u_2 w_0 \delta\ ;\ N_+'''\,\rangle)
\end{aligned}$$

Then the following is also a rewrite in $R(\Pi)$:

$$\begin{aligned}
&\mathit{blue}(I_m(M_+)) \xrightarrow{3} \langle\, L_1 \mid \mathit{blue}^{\#}(L_1, w_1)\ ;\ \mathit{blue}(\langle\, L_2 \mid w_2\ ;\ N_+\,\rangle)\,\rangle \longrightarrow^{+}\\
&\langle\, L_1 \mid \mathit{green}(w_1')\ ;\ \mathit{green}(\langle\, L_2 \mid u_2 w_0 \delta\ ;\ N_+'''\,\rangle)\,\rangle \xrightarrow{17^*|19^*|21^*}\\
&\langle\, L_1 \mid \mathit{green}(w_1'')\ ;\ \mathit{green}(\langle\, L_2 \mid w_2'' w_0 \delta\ ;\ N_+'''\,\rangle)\,\rangle \xrightarrow{24}\\
&\langle\, L_1 \mid \mathit{green}(w_1'' w_2'' w_0)\ ;\ \mathit{green}(N_+''')\,\rangle \xrightarrow{13} \mathit{green}(\langle\, L_1 \mid w_1'' w_2'' w_0\ ;\ N_+'''\,\rangle) =\\
&\mathit{green}(I_m(M_+'''))
\end{aligned}$$

(vii) $M_+'' \overset{\delta}{\Longrightarrow} M_+'''$ is inferred by (D_7). Then $M_+'' = \langle\, L \mid w''\ ;\ N_+^5, N_+^6\,\rangle$ and $M_+''' = \langle\, L \mid w'' w_0\ ;\ N_+^6\,\rangle$, where $\langle\, L \mid w''\ ;\ N_+^5\,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w'' w_0\,\rangle$ is inferred by a shorter inference, and $\langle\, L \mid w''\ ;\ N_+^6\,\rangle$ is δ-irreducible (therefore the dissolving symbol δ does not occur in N_+^6). Let $M_+ = \langle\, L \mid w\ ;\ N_+^1, N_+^2\,\rangle$ and $M_+' = \langle\, L \mid w'\ ;\ N_+^3, N_+^4\,\rangle$. We denote by w_1' the messages sent from the membrane labelled by L inside the membranes represented by N_+^3, and by w_2' the objects sent outside from the membranes represented by N_+^3. By inductive hypothesis we obtain the following two rewrites:

$$\mathit{blue}(\langle\, L \mid w\ ;\ N_+^1\,\rangle) \longrightarrow^{+} \langle\, L \mid \mathit{green}(w')\ ;\ \mathit{green}(N_+^3)\,\rangle$$

and

$$\begin{aligned}
&\langle\, L \mid \mathit{green}(w_1')\ ;\ \mathit{green}(N_+^3)\,\rangle \longrightarrow^{+} \langle\, L \mid \mathit{green}(w_2')\ ;\ \mathit{green}(N_+^5)\,\rangle \longrightarrow^{+}\\
&\langle\, L \mid \mathit{green}(w_2')\,\rangle
\end{aligned}$$

Then the following is a rewrite in $R(\Pi)$:

$$\begin{aligned}&\mathit{blue}(I_m(M_+)) \xrightarrow{3|5} \langle\, L \mid \mathit{blue}^{\#}(L,w)\,;\, \mathit{blue}(I_m(N^1_+)), \mathit{blue}(I_m(N^2_+))\,\rangle \longrightarrow^+ \\ &\langle\, L \mid \mathit{green}(w')\,;\, \mathit{green}(N^3_+, N^4_+)\,\rangle \longrightarrow^+ \langle\, L \mid \mathit{green}(w'')\,;\, \mathit{green}(N^5_+, N^6_+)\,\rangle \longrightarrow^+ \\ &\langle\, L \mid \mathit{green}(w''w_0)\,;\, \mathit{green}(N^6_+)\,\rangle \xrightarrow{13} \mathit{green}(\langle\, L \mid w''w_0\,;\, N^6_+\,\rangle) = \mathit{green}(I_m(M'''_+))\end{aligned}$$

(viii) $M''_+ \overset{\delta}{\Longrightarrow} M'''_+$ is inferred by (D_8). Then $M''_+ = \langle\, L \mid w''\,;\, N^5_+, N^6_+\,\rangle$ and $M'''_+ = \langle\, L \mid w''w_0\,;\, N^7_+, N^6_+\,\rangle$, where $\langle\, L \mid w''\,;\, N^5_+\,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w''w_0\,;\, N^7_+\,\rangle$ is inferred by a shorter inference, and $\langle L|w''; N^6_+\rangle$ is δ-irreducible (therefore the dissolving symbol δ does not occur in N^6_+). Let $M_+ = \langle\, L \mid w\,;\, N^1_+, N^2_+\,\rangle$ and $M'_+ = \langle\, L \mid w'\,;\, N^3_+, N^4_+\,\rangle$. We denote by w'_1 the messages sent from the membrane labelled by L inside the membranes represented by N^3_+, and by w'_2 the objects sent outside from the membranes represented by N^3_+. By inductive hypothesis we obtain the following two rewrites:

$$\mathit{blue}(\langle\, L \mid w\,;\, N^1_+\,\rangle) \longrightarrow^+ \langle\, L \mid \mathit{green}(w')\,;\, \mathit{green}(N^3_+)\,\rangle$$

and

$$\begin{aligned}&\langle\, L \mid \mathit{green}(w'_1)\,;\, \mathit{green}(N^3_+)\,\rangle \longrightarrow^+ \langle\, L \mid \mathit{green}(w'_2)\,;\, \mathit{green}(N^5_+)\,\rangle \longrightarrow^+ \\ &\langle\, L \mid \mathit{green}(w'_2)\,;\, \mathit{green}(N^7_+\,\rangle\end{aligned}$$

Then the following is a rewrite in $R(\Pi)$:

$$\begin{aligned}&\mathit{blue}(I_m(M_+)) \xrightarrow{3|5} \langle\, L \mid \mathit{blue}^{\#}(L,w)\,;\, \mathit{blue}(I_m(N^1_+)), \mathit{blue}(I_m(N^2_+))\,\rangle \longrightarrow^+ \\ &\langle\, L \mid \mathit{green}(w')\,;\, \mathit{green}(N^3_+, N^4_+)\,\rangle \longrightarrow^+ \langle\, L \mid \mathit{green}(w'')\,;\, \mathit{green}(N^5_+, N^6_+)\,\rangle \longrightarrow^+ \\ &\langle\, L \mid \mathit{green}(w''w_0)\,;\, \mathit{green}(N^7_+, N^6_+)\,\rangle \xrightarrow{13} \mathit{green}(\langle\, L \mid w''w_0\,;\, N^7_+, N^6_+\,\rangle) = \\ &\mathit{green}(I_m(M'''_+))\end{aligned}$$

(ix) $M''_+ \overset{\delta}{\Longrightarrow} M'''_+$ is inferred by (D_9). Then $M''_+ = \langle\, L \mid w''\,;\, N^5_+, N^6_+\,\rangle$ and $M'''_+ = \langle\, L \mid w''u_1u_2\,\rangle$, where $\langle\, L \mid w''\,;\, N^5_+\,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w''u_1\,\rangle$ and $\langle\, L \mid w''\,;\, N^6_+\,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w''u_2\,\rangle$ are inferred by shorter inferences. Let $M_+ = \langle\, L \mid w\,;\, N^1_+, N^2_+\,\rangle$ and $M'_+ = \langle\, L \mid w'\,;\, N^3_+, N^4_+\,\rangle$. We denote by w'_1 the messages sent from the membrane labelled by L inside the membranes represented by N^3_+, and by w''_1 the objects sent outside from the membranes represented by N^3_+. We do the same for w'_2, w''_2, and N^4. By inductive hypothesis we obtain the following four rewrites:

$$\begin{aligned}&\mathit{blue}(\langle\, L \mid w\,;\, N^1_+\,\rangle) \longrightarrow^+ \langle\, L \mid \mathit{green}(w')\,;\, \mathit{green}(N^3_+)\,\rangle \\ &\langle\, L \mid \mathit{green}(w'_1)\,;\, \mathit{green}(N^3_+)\,\rangle \longrightarrow^+ \langle\, L \mid \mathit{green}(w''_1)\,;\, \mathit{green}(N^5_+)\,\rangle \longrightarrow^+ \\ &\langle\, L \mid \mathit{green}(w''_1u_1)\,\rangle\end{aligned}$$

and

$$\begin{aligned}&\mathit{blue}(\langle\, L \mid w\,;\, N^2_+\,\rangle) \longrightarrow^+ \langle\, L \mid \mathit{green}(w')\,;\, \mathit{green}(N^4_+)\,\rangle \\ &\langle\, L \mid \mathit{green}(w'_2)\,;\, \mathit{green}(N^4_+)\,\rangle \longrightarrow^+ \langle\, L \mid \mathit{green}(w''_2)\,;\, \mathit{green}(N^6_+)\,\rangle \longrightarrow^+ \\ &\langle\, L \mid \mathit{green}(w''_2u_2)\,\rangle\end{aligned}$$

Then the following is a rewrite in $R(\Pi)$:

$$\begin{array}{l} \mathit{blue}(\langle\, L \mid w \,;\, N_+^1, N_+^2 \,\rangle) \xrightarrow{3} \langle\, L \mid \mathit{blue}^{\#}(L, w) \,;\, \mathit{blue}(N_+^1, N_+^2) \,\rangle \longrightarrow^+ \\ \langle\, L \mid \mathit{green}(w') \,;\, \mathit{green}(N_+^3, N_+^4) \,\rangle \longrightarrow^+ \langle\, L \mid \mathit{green}(w'') \,;\, \mathit{green}(N_+^5, N_+^6) \,\rangle \longrightarrow^+ \\ \langle\, L \mid \mathit{green}(w''u_1u_2) \,\rangle \xrightarrow{10+11+12} \mathit{green}(\langle\, L \mid w''u_1u_2 \,\rangle) = \mathit{green}(I_m(M_+''')) \end{array}$$

(x) $M_+'' \overset{\delta}{\Longrightarrow} M_+'''$ is inferred by (D_{10}). Then $M_+'' = \langle\, L \mid w'' \,;\, N_+^5, N_+^6 \,\rangle$ and $M_+''' = \langle L | w''u_1u_2 \,;\, N_+^7 \rangle$, where $\langle L | w'' ;\, N_+^5 \rangle \overset{\delta}{\Longrightarrow} \langle L | w''u_1 \rangle$ and $\langle L | w'' ;\, N_+^6 \rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w''u_2 \,;\, N_+^7 \,\rangle$ are inferred by shorter inferences. Let $M_+ = \langle\, L \mid w \,;\, N_+^1, N_+^2 \,\rangle$ and $M_+' = \langle\, L \mid w' \,;\, N_+^3, N_+^4 \,\rangle$. By inductive hypothesis we obtain the following rewrites:

$$\begin{array}{l} \mathit{blue}(\langle\, L \mid w \,;\, N_+^1 \,\rangle) \longrightarrow^+ \langle\, L \mid \mathit{green}(w') \,;\, \mathit{green}(N_+^3) \,\rangle \\ \langle\, L \mid \mathit{green}(w_1') \,;\, \mathit{green}(N_+^3) \,\rangle \longrightarrow^+ \langle\, L \mid \mathit{green}(w_1'') \,;\, \mathit{green}(N_+^5) \,\rangle \longrightarrow^+ \\ \langle\, L \mid \mathit{green}(w_1''u_1) \,\rangle \end{array}$$

and

$$\begin{array}{l} \mathit{blue}(\langle\, L \mid w \,;\, N_+^2 \,\rangle) \longrightarrow^+ \langle\, L \mid \mathit{green}(w') \,;\, \mathit{green}(N_+^4) \,\rangle \\ \langle\, L \mid \mathit{green}(w_2') \,;\, \mathit{green}(N_+^4) \,\rangle \longrightarrow^+ \langle\, L \mid \mathit{green}(w_2'') \,;\, \mathit{green}(N_+^6) \,\rangle \longrightarrow^+ \\ \langle\, L \mid \mathit{green}(w_2''u_2) \,;\, \mathit{green}(N_+^7) \,\rangle \end{array}$$

Then the following is a rewrite in $R(\Pi)$:

$$\begin{array}{l} \mathit{blue}(\langle\, L \mid w \,;\, N_+^1, N_+^2 \,\rangle) \xrightarrow{3} \langle\, L \mid \mathit{blue}^{\#}(L, w) \,;\, \mathit{blue}(N_+^1, N_+^2) \,\rangle \longrightarrow^+ \\ \langle\, L \mid \mathit{green}(w') \,;\, \mathit{green}(N_+^3, N_+^4) \,\rangle \longrightarrow^+ \langle\, L \mid \mathit{green}(w'') \,;\, \mathit{green}(N_+^5, N_+^6) \,\rangle \longrightarrow^+ \\ \langle\, L \mid \mathit{green}(w''u_1u_2) \,;\, \mathit{green}(N_+^7) \,\rangle \xrightarrow{13} \mathit{green}(\langle\, L \mid w''u_1u_2 \,;\, N_+^7 \,\rangle) = \mathit{green}(I_m(M_+''')) \end{array}$$

(xi) $M_+'' \overset{\delta}{\Longrightarrow} M_+'''$ is inferred by (D_{11}). Then $M_+'' = \langle\, L \mid w'' \,;\, N_+^5, N_+^6 \,\rangle$ and $M_+''' = \langle\, L \mid w''u_1u_2 \,;\, N_+^7, N_+^8 \,\rangle$, where $\langle\, L \mid w'' \,;\, N_+^5 \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w''u_1 \,;\, N_+^7 \,\rangle$ and $\langle\, L \mid w'' \,;\, N_+^6 \,\rangle \overset{\delta}{\Longrightarrow} \langle\, L \mid w''u_2 \,;\, N_+^8 \,\rangle$ are inferred by shorter inferences. Let $M_+ = \langle\, L \mid w \,;\, N_+^1, N_+^2 \,\rangle$ and $M_+' = \langle\, L \mid w' \,;\, N_+^3, N_+^4 \,\rangle$. By inductive hypothesis we obtain the following rewrites:

$$\begin{array}{l} \mathit{blue}(\langle\, L \mid w \,;\, N_+^1 \,\rangle) \longrightarrow^+ \langle\, L \mid \mathit{green}(w') \,;\, \mathit{green}(N_+^3) \,\rangle \\ \langle\, L \mid \mathit{green}(w_1') \,;\, \mathit{green}(N_+^3) \,\rangle \longrightarrow^+ \langle\, L \mid \mathit{green}(w_1'') \,;\, \mathit{green}(N_+^5) \,\rangle \longrightarrow^+ \\ \langle\, L \mid \mathit{green}(w_1''u_1) \,;\, \mathit{green}(N_+^7) \,\rangle \end{array}$$

and

$$\begin{array}{l} \mathit{blue}(\langle\, L \mid w \,;\, N_+^2 \,\rangle) \longrightarrow^+ \langle\, L \mid \mathit{green}(w') \,;\, \mathit{green}(N_+^4) \,\rangle \\ \langle\, L \mid \mathit{green}(w_2') \,;\, \mathit{green}(N_+^4) \,\rangle \longrightarrow^+ \langle\, L \mid \mathit{green}(w_2'') \,;\, \mathit{green}(N_+^6) \,\rangle \longrightarrow^+ \\ \langle\, L \mid \mathit{green}(w_2''u_2) \,;\, \mathit{green}(N_+^8) \,\rangle \end{array}$$

Then the following is also a rewrite in $R(\Pi)$:

$$\begin{aligned}&blue(\langle\, L \mid w \,;\, N_+^1, N_+^2 \,\rangle) \xrightarrow{3} \langle\, L \mid blue^{\#}(L,w) \,;\, blue(N_+^1, N_+^2) \,\rangle \longrightarrow^+ \\ &\langle\, L \mid green(w') \,;\, green(N_+^3, N_+^4) \,\rangle \longrightarrow^+ \langle\, L \mid green(w'') \,;\, green(N_+^5, N_+^6) \,\rangle \longrightarrow^+ \\ &\langle\, L \mid green(w''u_1u_2) \,;\, green(N_+^7, N_+^8) \,\rangle \xrightarrow{13} green(\langle\, L \mid w''u_1u_2 \,;\, N_+^7, N_+^8 \,\rangle) = \\ &green(I_m(M_+'''))\end{aligned}$$

Corollary 6.4 *If* $M_+ \overset{mpr}{\Longrightarrow} M_+' \overset{tar}{\Longrightarrow} M_+'' \overset{\delta}{\Longrightarrow} M_+'''$, *then* $blue(I_m(M_+)) \longrightarrow^+ green(I_m(M_+'''))$, *and it does not exist* M_+^0 *such that* $M_+^0 \neq M_+''''$ *and* $blue(M_+) \longrightarrow^+ green(M_+^0) \longrightarrow^+ green(M_+''')$.

The following results describe the relationship between P systems operational semantics and Maude rewriting relation.

Theorem 6.1 *[Operational Correspondence] Let* Π *be a P system, and* $C, C' \in \mathcal{C}(\Pi)$.
If $C \Longrightarrow C'$, *then there exists a C-rewriting such that* $\mathcal{I}(C) \longrightarrow^+_{R(\Pi)} \mathcal{I}(C')$.

Proof: We have to consider three cases for $C \Longrightarrow C'$:

(i) $C \overset{mpr}{\Longrightarrow};\overset{tar}{\Longrightarrow} C'$. Then the following is a rewriting in $R(\Pi)$:

$$\mathcal{I}(C) = \{I_m(C)\} \xrightarrow{1} \{blue(I_m(C))\} \overset{Cor.6.2}{\longrightarrow} \{green(I_m(C'))\} \xrightarrow{2} \{I_m(C')\} = \mathcal{I}(C')$$

(ii) $C \overset{mpr}{\Longrightarrow};\overset{\delta}{\Longrightarrow} C'$. Then the following is a rewriting in $R(\Pi)$:

$$\mathcal{I}(C) = \{I_m(C)\} \xrightarrow{1} \{blue(I_m(C))\} \overset{Cor.6.3}{\longrightarrow} \{green(I_m(C'))\} \xrightarrow{2} \{I_m(C')\} = \mathcal{I}(C')$$

(iii) $C \overset{mpr}{\Longrightarrow};\overset{tar}{\Longrightarrow};\overset{\delta}{\Longrightarrow} C'$. Then the following is a rewriting in $R(\Pi)$:

$$\mathcal{I}(C) = \{I_m(C)\} \xrightarrow{1} \{blue(I_m(C))\} \overset{Cor.6.4}{\longrightarrow} \{green(I_m(C'))\} \xrightarrow{2} \{I_m(C')\} = \mathcal{I}(C')$$

Theorem 6.2 *Let* Π *be a P system, and* C *an arbitrary configuration. If* $\mathcal{I}(C) \longrightarrow^+_{R(\Pi)} t$ *is a C-rewriting in* $R(\Pi)$ *with* $t \in (T_{\Sigma,E})_{CConfiguration}$, *and this C-rewriting contains at least one of the Maude rules 6, 6', 8, or 8', then there is a configuration* C' *such that* $C \Longrightarrow C'$ *and* $t =_E \mathcal{I}(C')$.

Remark: The condition that the given C-rewrite contains at least one of the Maude rules 6, 6', 8, or 8', is necessary because each of these rules applies an evolution rule in a membrane. This way we are sure that indeed C "evolved".

Chapter 7

Conclusion and Further Research

In this paper we present a quick survey of P systems and rewriting logic, and, as the main result, the operational semantics of (transition) P systems.

We give an abstract syntax of P systems, and we define a transition step by means of three sets of inference rules corresponding to three stages: maximal parallel rewriting, parallel communication, and parallel dissolving. This definition is provided with the appropriate correctness result. At this point we have a natural or big-step like operational semantics due to the parallel nature of the model. Translating the natural semantics of P systems into rewriting logic, we get a certain refinement of the operational description similar to the small-step like. Moreover, by using an efficient implementation of rewriting logic as Maude, we obtain an interpreter for P systems, and we can verify the properties of systems by means of two formal tools: the `search` command (a semi-decision procedure for finding failures of safety properties), and Maude's LTL model checker.

There is much interesting work left. One important feature is the specification of other types of P systems and their operational semantics; a modular approach is desired.

Another interesting feature is to study the relationship with Cardelli's Brane Calculi [8], and the Chemical Abstract Machine style of operational semantics proposed by Berry and Boudol [4].

A practical desired result consists in defining a Maude command for the C-rewrite relation defined in chapter 6, therefore a command for rewriting on subsorts. This can be done in Full Maude at meta-level.

Bibliography

[1] O. Andrei, G. Ciobanu, D. Lucanu. Executable Specifications of the P Systems. In G.Mauri, Gh.Păun, M.J.Perez-Jimenez, G.Rozenberg, A.Salomaa (Eds.): *Membrane Computing WMC5*, Lecture Notes in Computer Science vol.3365, Springer, 127-146, 2005.

[2] O. Andrei, G. Ciobanu, D. Lucanu. Operational Semantics and Rewriting Logic in Membrane Computing. *Proceedings WMC6, Wien, Austria*, July 2005. *Accepted.*

[3] O. Andrei, G. Ciobanu, D. Lucanu. Structural Operational Semantics of P Systems. *Proceedings SOS Workshop of ICALP, Lisbon, Portugal*, July 2005. *Accepted.*

[4] G. Berry, and G. Boudol. The Chemical Abstract Machine. Theoretical Computer Science, 96 (1992), 217-248.

[5] P. Borovanský, C. Kirchner, H. Kirchner, and P. -E. Moreau. ELAN from the rewriting logic point of view. Theoretical Computer Science, June 2001.

[6] P. Borovanský, C. Kirchner, and H. Kirchner. Controlling rewriting by rewriting. In Meseguer, editor. Proceedings First International Workshop on Rewriting Logic and its Applications, WRLA'96, Asilomar, California, September 3-6, 1996, volume 4 of *Electronic Notes in Theoretical Computer Science*, Elsevier, pages 168-188. `http://www.elsevier.nl/locate/entcs/volume4.html`

[7] P. Borovanský, C. Kirchner, H. Kirchner, P. -E. Moreau, and C. Ringeissen. An overview of ELAN. In Kirchner and Kirchner, editors. Proceedings Second International Workshop on Rewriting Logic and its Applications, WRLA'98, Pont-à-Muson, France, September 1-4, 1998, volume 15 of *Electronic Notes in Theoretical Computer Science*, Elsevier, 1998, pages 329-344. `http://www.elsevier.nl/locate/entcs/volume15.html`

[8] L. Cardelli. Brane Calculus. Proc. Computational Methods in Systems Biology '04, Springer, to appear.

[9] G. Ciobanu. Distributed Algorithms over Communicating Membrane Systems. *Biosystems* vol.70(2), 123-133, 2003.

[10] G. Ciobanu, W. Guo. P Systems Running on a Cluster of Computers. In Gh.Păun, G.Rozenberg, A.Salomaa (Eds.): *Membrane Computing*, LNCS vol.2933, Springer, 123-139, 2004.

[11] G. Ciobanu, D. Paraschiv. P System Software Simulator. *Fundamenta Informaticae* vol.49, 61-66, 2002.

[12] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, C .Talcott. *Maude Manual* (Version 2.1). http://maude.cs.uiuc.edu, 2004.

[13] M. Clavel, F. Duran, S. Eker, P. Lincoln, N. Marti-Oliet, J. Meseguer, and J. F. Quesada. *Maude:Specification and Programming in Rewriting Logic*, *Theoretical Computer Science*, vol.285(2), 187-243, 2002.

[14] M. Clavel, F. Duran, S. Eker, and J. Meseguer. Building equational proving tools by reflection in rewriting logic. In *CAFE: An Industrial-Strength Algebraic Formal Method*. Elsevier, 2000

[15] R. Diaconescu. Behavioural rewriting logic: Semantic foundations and proof theory. October 1996.

[16] R. Diaconescu and K. Futatsugi. CafeOBJ Report. The language, Proof Techniques, and Methodologies for Object-Oriented Algebraic Specification, volume 6 of *AMAST Series in Computing*. World Scientific, 1998.

[17] R. Diaconescu and K. Futatsugi. Logical foundations of CafeOBJ. *Theoretical Computer Science*, 2001.

[18] S. Eker, J. Meseguer, and A. Sridharanarayanan. The Maude LTL Model Checker and Its Implementation. In T.Ball, S.K.Rajamani (Eds.): *Model Checking Software: 10th SPIN Workshop*, LNCS vol.2648, Springer, 230-234, 2003.

[19] K. Futatsugi and A. T. Nakagawa. An overview of CAFE specification environment - An algebraic approach fo creating, verifying, and maintaining formal specifications over networks. In First International Conference on Formal Engineering Methods, Hiroshima, Japan, November 1997.

[20] K. Futatsugi, A. T. Nakagawa, and T. Tamai, editors. Cafe: An Industrial-Strength Algebraic Formal Method. Elsevier, 2000.

[21] J.-L. Giavitto, G. Malcom, O. Michel. Rewriting Systems and the Modelling of Biological Systems.

[22] J. Goguen, C. Kirchner, H. Kirchner, A. Megrelis, J. Meseguer and T. Winkler. An introduction to OBJ3, in: J.-P. Jouannaud and S. Kaplan, eds., *Proc. Conf. on Conditional Term Rewriting, Orsay, France, July 8-10, 1987*, Lecture Notes in Computer Science, Vol. 308 (Springer, Berlin, 1988) 258-263.

[23] J. Goguen, J. Meseguer. Order-sorted algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partial operations. *Theoretical Computer Science*, 105:217-273, 1992.

[24] J. Goguen and J. Tardo. OBJ-0 preliminary users manual, Semantics and theory of computation report 10, UCLA, 1977.

[25] C. Gunter. Forms of semantic specification, *Bulletin of the EATCS 45*, October 1991, pages 98-113.

[26] M. Hennessy. *The Semantics of Programming Languages: An Elementary Introduction Using Structural Operational Semantics*, John Wiley and Sons, 1990.

[27] G. Kahn. *Natural semantics*, Technical report 601, INRIA Sophia Antipolis, February 1987

[28] H. Kirchner and P. -E. Moreau. Promoting rewriting to a programming language: A compiler for non-deterministic rewrite programs in associative-commutative theories. Hournal of Functional Programming, 11(2):207-251, 2001.

[29] N. Marti-Oliet and J. Meseguer. Rewriting logic as a logical and semantical framework. In *Handbook of Philosophical Logic, 2nd. Edition*, pages 1-87. Kluwer Academic Publishers, 2002. First published as SRI Tech. Report SRI-CSL-93-05, August 1993.

[30] J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96(1):73-155, 1992.

[31] J. Meseguer. Membership algebra as a logical framework for equational specification. In *Proc. WADT'97*, pages 18-61. Springer LNCS 1376, 1998.

[32] J. Meseguer. Rewriting Logic as a Semantic Framework for Concurrency: a Progress Report. In *CONCUR'96: Concurrency Theory, 7th International Conference, Pisa, Italy, August 26-29, 1996, Proceedings*, volume 1119 of LNCS, pages 331-372, Spinger, 1996.

[33] J. Meseguer, G. Rosu. Rewriting Logic Semantics: From Language Specification to Formal Analysis Tools. *IJCAR 2004*, Lecture Notes in Computer Science vol.3097, Springer, 1-44, 2004.

[34] P. -E. Moreau and H. Kirchner. Compilation of associative-commutative normalisation with strategies in ELAN. Technical Report 97-R-129, LORIA, Nancy, France, 1997.

[35] P. -E. Moreau and H. Kirchner. A compiler for rewrite programs in associative-commutative theories. In C. Palamidessi, H. Glaser, and K. Meinke, editors, Principles of Declarative Programming, 10th International Symposium, PLIP'98 Held Jointly with the 6th Conference ALP'98, Pisa, Italy, September 16-18, 1998, Proceedings, volume 1490 of *Lecture Notes in Computer Science*, pages 230-249. Spinger-Verlag, 1998.

[36] P. Mosses. Modular Structural Operational Semantics. BRICS RS-05-7, February 2005

[37] H. R. Nielson and F. Nielson, *Semantics with Applications: A Formal Introduction*, John Wiley and Sons, 1992.

[38] M. J. O'Donnell. *Computing in Systems Described by Equations.* Lecture Notes in Computer Science, Vol. 58 (Springer, Berlin, 1977).

[39] Gh. Păun. *Computing with Membranes: An Introduction*, Springer, 2002.

[40] Gh. Păun. Computing with Membranes *Journal of Computer and System Science*, 61, 1 (2000), 108-143

[41] Gh. Păun. *Introduction to Membrane Computing*, 2004. http://www.bio.disco.unimib.it/download/MembIntro2004.pdf

[42] G. D. Plotkin. *A structural approach to operational semantics*, Technical report DAIMI FN-19, Computer Science Department, Aarhus University, September 1981.

Appendix A

Maude Output for Example 5.2

```
                     \||||||||||||||||||/
                   --- Welcome to Maude ---
                     /||||||||||||||||||\
             Maude 2.1 built: Mar 18 2004 11:31:58
             Copyright 1997-2004 SRI International
                   Wed Jun  8 15:01:38 2005

               Full Maude 2.1 (March 19th, 2004)

Maude> in pconfiguration.fm
rewrites: 5189 in 119ms cpu (132ms real) (43248 rewrites/second)
Introduced module PSCONFIGURATION

Maude> in psystem.fm
rewrites: 58450 in 443ms cpu (445ms real) (131664 rewrites/second)
Introduced module PSYSTEM

Maude> in nsquare.fm
rewrites: 4300 in 46ms cpu (46ms real) (91502 rewrites/second)
Introduced module NSQUARE

Maude> (search init =>+ C:CConfiguration .)
rewrites: 229241 in 1824ms cpu (1824ms real) (125630 rewrites/second)
search in NSQUARE : init =>+ C:CConfiguration .

Solution 1
C:CConfiguration <- {< M1 | empty ; < M2 | empty ; < M3 | a b f f > > >}

Solution 2
C:CConfiguration <- {< M1 | empty ; < M2 | b f f > >}

Solution 3
C:CConfiguration <- {< M1 | empty ; < M2 | d f > >}

Solution 4
C:CConfiguration <- {< M1 | empty ; < M2 | empty ; < M3 | a b b f f f f > > >}
```

```
Solution 5
C:CConfiguration <- {< M1 | empty ; < M2 | b b f f f f > >}

Solution 6
C:CConfiguration <- {< M1 | d e >}

Solution 7
C:CConfiguration <- {< M1 | empty ; < M2 | d d f f > >}

Solution 8
C:CConfiguration <- {< M1 | empty ; < M2 | empty ; < M3 | a b b b f f f f f f f
    f > > >}

Solution 9
C:CConfiguration <- {< M1 | empty ; < M2 | b b b f f f f f f f f > >}

Solution 10
C:CConfiguration <- {< M1 | empty ; < M2 | d d e e f > >}

Solution 11
C:CConfiguration <- {< M1 | empty ; < M2 | d d d f f f f > >}

Solution 12
C:CConfiguration <- {< M1 | d d e e e e >}

Solution 13
C:CConfiguration <- {< M1 | empty ; < M2 | empty ; < M3 | a b b b b f f f f f f
    f f f f f f f f f f > > >}

Solution 14
C:CConfiguration <- {< M1 | empty ; < M2 | b b b b f f f f f f f f f f f f f
    f f > >}

Solution 15
C:CConfiguration <- {< M1 | empty ; < M2 | d d d e e e f f > >}

Solution 16
C:CConfiguration <- {< M1 | empty ; < M2 | d d d e e e e e e f > >}

Solution 17
C:CConfiguration <- {< M1 | d d d e e e e e e e e e >}

No more solutions.
Maude> (search init =>+ { < M1 | S:Soup > } .)
rewrites: 228973 in 1656ms cpu (1656ms real) (138206 rewrites/second)
search in NSQUARE : init =>+ {< M1 | S:Soup >}.

Solution 1
S:Soup <- d e

Solution 2
S:Soup <- d d e e e e
```

```
Solution 3
S:Soup <- d d d e e e e e e e e e

No more solutions.
Maude> in mc-nsquare.fm
rewrites: 3228 in 53ms cpu (53ms real) (59786 rewrites/second)
Introduced module N2-PREDS

rewrites: 2328 in 14ms cpu (14ms real) (155231 rewrites/second)
Introduced module PROOF

Maude> (red modelCheck(init, [](isHalting -> isSquare)) .)
rewrites: 235072 in 1829ms cpu (1830ms real) (128474 rewrites/second)
reduce in PROOF :
  modelCheck(init,[](isHalting -> isSquare))
result Bool :
  true

Maude> in mc-nsquare-counter.fm
rewrites: 3137 in 32ms cpu (33ms real) (95077 rewrites/second)
Advisory: Module N2-PREDS redefined.
Introduced module N2-PREDS

rewrites: 2320 in 13ms cpu (14ms real) (165737 rewrites/second)
Advisory: Module PROOF redefined.
Introduced module PROOF

Maude> (red modelCheck(init, [](isHalting -> isSquare)) .)
rewrites: 122514 in 1169ms cpu (1169ms real) (104728 rewrites/second)
reduce in PROOF :
  modelCheck(init,[](isHalting -> isSquare))
result ModelCheckResult :
  counterexample({{< M1 | empty ; < M2 | empty ; < M3 | a f > > >},'1}{{blue(<
    M1 | empty ; < M2 | empty ; < M3 | a f > > >)},'3}{{< M1 | blue(M1,empty);
    blue(< M2 | empty ; < M3 | a f > >)>},'7}{{< M1 | green(empty); blue(< M2 |
    empty ; < M3 | a f > >)>},'3}{{< M1 | green(empty); < M2 | blue(M2,r1 r2,
    empty,false); blue(< M3 | a f >)> >},'9'}{{< M1 | green(empty); < M2 |
    green(empty); blue(< M3 | a f >)> >},'4}{{< M1 | green(empty); < M2 |
    green(empty); < M3 | blue(M3,a f,false)> > >},'6'}{{< M1 | green(empty); <
    M2 | green(empty); < M3 | green(a b)blue(M3,f,false)> > >},'6'}{{< M1 |
    green(empty); < M2 | green(empty); < M3 | green(f f)green(a b)blue(M3,(
    empty).Soup,false)> > >},'14}{{< M1 | green(empty); < M2 | green(empty); <
    M3 | green(a b f f)blue(M3,(empty).Soup,false)> > >},'7'}{{< M1 | green(
    empty); < M2 | green(empty); < M3 | green(empty)green(a b f f)> > >},'14}{{
    < M1 | green(empty); < M2 | green(empty); < M3 | green(a b f f)> > >},'11}{
    {< M1 | green(empty); < M2 | green(empty); green(< M3 | a b f f >)> >},
    '13}{{< M1 | green(empty); green(< M2 | empty ; < M3 | a b f f > >)>},'13}{
    {green(< M1 | empty ; < M2 | empty ; < M3 | a b f f > > >)},'2}{{< M1 |
    empty ; < M2 | empty ; < M3 | a b f f > > >},'1}{{blue(< M1 | empty ; < M2
    | empty ; < M3 | a b f f > > >)},'3}{{< M1 | blue(M1,empty); blue(< M2 |
    empty ; < M3 | a b f f > >)>},'7}{{< M1 | green(empty); blue(< M2 | empty ;
    < M3 | a b f f > >)>},'3}{{< M1 | green(empty); < M2 | blue(M2,r1 r2,empty,
    false); blue(< M3 | a b f f >)> >},'9'}{{< M1 | green(empty); < M2 | green(
    empty); blue(< M3 | a b f f >)> >},'4}{{< M1 | green(empty); < M2 | green(
    empty); < M3 | blue(M3,a b f f,false)> > >},'6'}{{< M1 | green(empty); < M2
    | green(empty); < M3 | green(a b)blue(M3,b f f,false)> > >},'6'}{{< M1 |
    green(empty); < M2 | green(empty); < M3 | green(f f)green(a b)blue(M3,b f,
    false)> > >},'14}{{< M1 | green(empty); < M2 | green(empty); < M3 | green(a
    b f f)blue(M3,b f,false)> > >},'6'}{{< M1 | green(empty); < M2 | green(
    empty); < M3 | green(f f)green(a b f f)blue(M3,b,false)> > >},'14}{{< M1 |
    green(empty); < M2 | green(empty); < M3 | green(a b f f f f)blue(M3,b,
    false)> > >},'7'}{{< M1 | green(empty); < M2 | green(empty); < M3 | green(
    b)green(a b f f f f)> > >},'14}{{< M1 | green(empty); < M2 | green(empty);
    < M3 | green(a b b f f f f)> > >},'11}{{< M1 | green(empty); < M2 | green(
    empty); green(< M3 | a b b f f f f >)> >},'13}{{< M1 | green(empty); green(
    < M2 | empty ; < M3 | a b b f f f f > >)>},'13}{{green(< M1 | empty ; < M2
    | empty ; < M3 | a b b f f f f > > >)},'2}{{< M1 | empty ; < M2 | empty ; <
    M3 | a b b f f f f > > >},'1}{{blue(< M1 | empty ; < M2 | empty ; < M3 | a
    b b f f f f > > >)},'3}{{< M1 | blue(M1,empty); blue(< M2 | empty ; < M3 |
    a b b f f f f > >)>},'7}{{< M1 | green(empty); blue(< M2 | empty ; < M3 | a
    b b f f f f > >)>},'3}{{< M1 | green(empty); < M2 | blue(M2,r1 r2,empty,
    false); blue(< M3 | a b b f f f f >)> >},'9'}{{< M1 | green(empty); < M2 |
    green(empty); blue(< M3 | a b b f f f f >)> >},'4}{{< M1 | green(empty); <
    M2 | green(empty); < M3 | blue(M3,a b b f f f f,false)> > >},'6'}{{< M1 |
    green(empty); < M2 | green(empty); < M3 | green(b delta)blue(M3,b b f f f
    f,true)> > >},'6'}{{< M1 | green(empty); < M2 | green(empty); < M3 | green(
    f f)green(b delta)blue(M3,b b f f f,true)> > >},'14}{{< M1 | green(empty);
    < M2 | green(empty); < M3 | green(b delta f f)blue(M3,b b f f f,true)> >
    >},'6'}{{< M1 | green(empty); < M2 | green(empty); < M3 | green(f f)green(b
    delta f f)blue(M3,b b f f,true)> > >},'14}{{< M1 | green(empty); < M2 |
    green(empty); < M3 | green(b delta f f f f)blue(M3,b b f f,true)> > >},
    '6'}{{< M1 | green(empty); < M2 | green(empty); < M3 | green(f f)green(b
    delta f f f f)blue(M3,b b f,true)> > >},'14}{{< M1 | green(empty); < M2 |
    green(empty); < M3 | green(b delta f f f f f f)blue(M3,b b f,true)> > >},
    '6'}{{< M1 | green(empty); < M2 | green(empty); < M3 | green(f f)green(b
    delta f f f f f f)blue(M3,b b,true)> > >},'14}{{< M1 | green(empty); < M2 |
    green(empty); < M3 | green(b delta f f f f f f f f)blue(M3,b b,true)> > >},
    '7'}{{< M1 | green(empty); < M2 | green(empty); < M3 | green(b b)green(b
    delta f f f f f f f f)> > >},'14}{{< M1 | green(empty); < M2 | green(
    empty); < M3 | green(b b b delta f f f f f f f f)> > >},'11}{{< M1 | green(
    empty); < M2 | green(empty); green(< M3 | b b b delta f f f f f f f f >)>
    >},'23}{{< M1 | green(empty); < M2 | green(b b b f f f f f f f f)> >},'11}{
    {< M1 | green(empty); green(< M2 | b b b f f f f f f f f >)>},'13}{{green(<
    M1 | empty ; < M2 | b b b f f f f f f f f > >)},'2}{{< M1 | empty ; < M2 |
    b b b f f f f f f f f > >},'1}{{blue(< M1 | empty ; < M2 | b b b f f f f f
    f f f > >)},'3}{{< M1 | blue(M1,empty); blue(< M2 | b b b f f f f f f f f
    >)>},'7}{{< M1 | green(empty); blue(< M2 | b b b f f f f f f f f >)>},'4}{{
    < M1 | green(empty); < M2 | blue(M2,r1 r2,b b b f f f f f f f f,false)> >},
    '8'}{{< M1 | green(empty); < M2 | green(d)blue(M2,r1 r2,b b f f f f f f f
    f,false)> >},'8'}{{< M1 | green(empty); < M2 | green(d)green(d)blue(M2,r1
    r2,b f f f f f f f f,false)> >},'14}{{< M1 | green(empty); < M2 | green(d
    d)blue(M2,r1 r2,b f f f f f f f f,false)> >},'8'}{{< M1 | green(empty); <
    M2 | green(d)green(d d)blue(M2,r1 r2,f f f f f f f f,false)> >},'14}{{< M1
```

```
| green(empty); < M2 | green(d d d)blue(M2,r1 r2,f f f f f f f f,false)>
>},’8’}{{< M1 | green(empty); < M2 | green(f)green(d d d)blue(M2,r1,f f f f
f f,false)> >},’14}{{< M1 | green(empty); < M2 | green(d d d f)blue(M2,r1,f
f f f f f,false)> >},’8’}{{< M1 | green(empty); < M2 | green(f)green(d d d
f)blue(M2,r1,f f f f,false)> >},’14}{{< M1 | green(empty); < M2 | green(d d
d f f)blue(M2,r1,f f f f,false)> >},’8’}{{< M1 | green(empty); < M2 |
green(f)green(d d d f f)blue(M2,r1,f f,false)> >},’14}{{< M1 | green(
empty); < M2 | green(d d d f f f)blue(M2,r1,f f,false)> >},’8’}{{< M1 |
green(empty); < M2 | green(f)green(d d d f f f)blue(M2,r1,empty,false)> >},
’14}{{< M1 | green(empty); < M2 | green(d d d f f f f)blue(M2,r1,empty,
false)> >},’9’}{{< M1 | green(empty); < M2 | green(empty)green(d d d f f f
f)> >},’14}{{< M1 | green(empty); < M2 | green(d d d f f f f)> >},’11}{{<
M1 | green(empty); green(< M2 | d d d f f f f >))>},’13}{{green(< M1 | empty
; < M2 | d d d f f f f > >)},’2}{{< M1 | empty ; < M2 | d d d f f f f > >},
’1}{{blue(< M1 | empty ; < M2 | d d d f f f f > >)},’3}{{< M1 | blue(M1,
empty); blue(< M2 | d d d f f f f >))>},’7}{{< M1 | green(empty); blue(< M2
| d d d f f f f >))>},’4}{{< M1 | green(empty); < M2 | blue(M2,r1 r2,d d d f
f f f,false)> >},’8’}{{< M1 | green(empty); < M2 | green(d e)blue(M2,r1 r2,
d d f f f f,false)> >},’8’}{{< M1 | green(empty); < M2 | green(d e)green(d
e)blue(M2,r1 r2,d f f f f,false)> >},’14}{{< M1 | green(empty); < M2 |
green(d d e e)blue(M2,r1 r2,d f f f f,false)> >},’8’}{{< M1 | green(empty);
< M2 | green(d e)green(d d e e)blue(M2,r1 r2,f f f f,false)> >},’14}{{< M1
| green(empty); < M2 | green(d d d e e e)blue(M2,r1 r2,f f f f,false)> >},
’8’}{{< M1 | green(empty); < M2 | green(f)green(d d d e e e)blue(M2,r1,f f,
false)> >},’14}{{< M1 | green(empty); < M2 | green(d d d e e e f)blue(M2,
r1,f f,false)> >},’8’}{{< M1 | green(empty); < M2 | green(f)green(d d d e e
e f)blue(M2,r1,empty,false)> >},’14}{{< M1 | green(empty); < M2 | green(d d
d e e e f f)blue(M2,r1,empty,false)> >},’9’}{{< M1 | green(empty); < M2 |
green(empty)green(d d d e e e f f)> >},’14}{{< M1 | green(empty); < M2 |
green(d d d e e e f f)> >},’11}{{< M1 | green(empty); green(< M2 | d d d e
e e f f >))>},’13}{{green(< M1 | empty ; < M2 | d d d e e e f f > >)},’2}{{<
M1 | empty ; < M2 | d d d e e e f f > >},’1}{{blue(< M1 | empty ; < M2 | d
d d e e e f f > >)},’3}{{< M1 | blue(M1,empty); blue(< M2 | d d d e e e f f
>))>},’7}{{< M1 | green(empty); blue(< M2 | d d d e e e f f >))>},’4}{{< M1 |
green(empty); < M2 | blue(M2,r1 r2,d d d e e e f f,false)> >},’8’}{{< M1 |
green(empty); < M2 | green(d e)blue(M2,r1 r2,d d e e e f f,false)> >},’8’}{
{< M1 | green(empty); < M2 | green(d e)green(d e)blue(M2,r1 r2,d e e e f f,
false)> >},’14}{{< M1 | green(empty); < M2 | green(d d e e)blue(M2,r1 r2,d
e e e f f,false)> >},’8’}{{< M1 | green(empty); < M2 | green(d e)green(d d
e e)blue(M2,r1 r2,e e e f f,false)> >},’14}{{< M1 | green(empty); < M2 |
green(d d d e e e)blue(M2,r1 r2,e e e f f,false)> >},’8’}{{< M1 | green(
empty); < M2 | green(f)green(d d d e e e)blue(M2,r1,e e e,false)> >},’14}{{
< M1 | green(empty); < M2 | green(d d d e e e f)blue(M2,r1,e e e,false)>
>},’9’}{{< M1 | green(empty); < M2 | green(e e e)green(d d d e e e f)> >},
’14}{{< M1 | green(empty); < M2 | green(d d d e e e e e e f)> >},’11}{{< M1
| green(empty); green(< M2 | d d d e e e e e e f >))>},’13}{{green(< M1 |
empty ; < M2 | d d d e e e e e e f > >)},’2}{{< M1 | empty ; < M2 | d d d e
e e e e e f > >},’1}{{blue(< M1 | empty ; < M2 | d d d e e e e e e f > >)},
’3}{{< M1 | blue(M1,empty); blue(< M2 | d d d e e e e e e f >))>},’7}{{< M1
| green(empty); blue(< M2 | d d d e e e e e e f >))>},’4}{{< M1 | green(
empty); < M2 | blue(M2,r1 r2,d d d e e e e e e f,false)> >},’8’}{{< M1 |
green(empty); < M2 | green(d e)blue(M2,r1 r2,d d e e e e e e f,false)> >},
    '8'}{{< M1 | green(empty); < M2 | green(d e)green(d e)blue(M2,r1 r2,d e e e
    e e e f,false)> >},'14}{{< M1 | green(empty); < M2 | green(d d e e)blue(M2,
    r1 r2,d e e e e e e f,false)> >},'8'}{{< M1 | green(empty); < M2 | green(d
    e)green(d d e e)blue(M2,r1 r2,e e e e e e f,false)> >},'14}{{< M1 | green(
    empty); < M2 | green(d d d e e e)blue(M2,r1 r2,e e e e e e f,false)> >},
    '8'}{{< M1 | green(empty); < M2 | green(delta)green(d d d e e e)blue(M2,r1
    r2,e e e e e e,true)> >},'14}{{< M1 | green(empty); < M2 | green(d d d
    delta e e e)blue(M2,r1 r2,e e e e e e,true)> >},'9'}{{< M1 | green(empty);
    < M2 | green(e e e e e e)green(d d d delta e e e)> >},'14}{{< M1 | green(
    empty); < M2 | green(d d d delta e e e e e e e e e)> >},'11}{{< M1 | green(
    empty); green(< M2 | d d d delta e e e e e e e e e >)>},'23}{{< M1 | green(
    d d d e e e e e e e e e)>},'12}{{green(< M1 | d d d e e e e e e e e e >)},
    '2},{{< M1 | d d d e e e e e e e e e >},'1}{{blue(< M1 | d d d e e e e e e
    e e e >)},'4}{{< M1 | blue(M1,d d d e e e e e e e e e)>},'7}{{< M1 | green(
    d d d e e e e e e e e e)>},'12}{{green(< M1 | d d d e e e e e e e e e >)},
    '2})

Maude>
```