Relating Two Metric Semantics for Parallel Rewriting of Multisets

Gabriel Ciobanu
Institute of Computer Science
Romanian Academy, Iaşi
Email: gabriel@iit.tuiasi.ro

Eneia Nicolae Todoran
Department of Computer Science
Technical University of Cluj-Napoca, Romania
Email: eneia.todoran@cs.utcluj.ro

***Abstract*—In this paper we employ the mathematical methodology of metric semantics in defining and relating a denotational and an operational semantics for an abstract concurrent language embodying the following features: parallel composition is based on maximal parallelism, and computations are specified by means of multiset rewriting rules. We relate these semantics, and compare them in terms of this combination of concepts. The semantic models are designed by using continuations for concurrency.**

***Keywords*-metric semantics; continuations for concurrency; parallel multiset rewriting;**

I. Introduction

We study two different semantic models for a multiset concurrent language $\mathcal{L}_{MR}$ providing the following features: non-interleaving semantics given by maximal parallelism, and computations are specified by multiset rewriting rules. We present an operational semantics and a denotational semantics for $\mathcal{L}_{MR}$; both models are designed with the continuation semantics for concurrency (CSC) technique [14]. Operational semantics is based on a labelled transition system specification in the style of Plotkin's structural operational semantics [12]. Denotational semantics is based on the compositionality principle; its model was introduced initially in [8].

Different semantics of a given language can be seen as different views of the same language. The main goal of this paper is to compare the operational semantics with the denotational semantics of $\mathcal{L}_{MR}$, by providing a formal relation between them. The semantic models are designed and related formally following the methodology of metric semantics [4]. The main mathematical tool in this approach is Banach's fixed point theorem, which states that a contracting function defined on a complete metric space has a unique fixed point. We also use the general method of solving reflexive domain equations in a category of complete metric spaces presented in [2].

A. Parallel rewriting of multisets and membrane computing

The features of the language $\mathcal{L}_{MR}$ are used in natural computation, being inspired by biological systems. Membrane computing is an established and successful research field which belongs to the more general area of natural computing [10]. Membrane computing deals with parallel and nondeterministic computing models inspired by cell biology. Membrane systems are complex hierarchical structures with a flow of materials and information which underlies their functioning, involving parallel application of rules, communication between membranes and membrane dissolution. A computation is performed in the following way: starting from an initial structure, the system evolves by applying the rules in a nondeterministic and maximally parallel manner. A halting configuration is reached when no rule is applicable.

A configuration of a membrane system is given by a membrane structure and multisets of objects associated with the regions defined by this membrane structure. A rule or a multiset of rules associated to a membrane is applicable in a configuration if its left hand side is a part of the multiset contained in that membrane. Configurations evolve by having multisets of rules applied in each membrane in a nondeterministic and maximally parallel manner. The maximally parallel way of using the rules means that in each step we apply a maximal multiset of rules, namely a multiset of rules such that no further rule can be added to the multiset.

Example 1.1: To clarify how a membrane system evolves, we present the following example, together with the graphical representation of a membrane system as a Venn diagram.

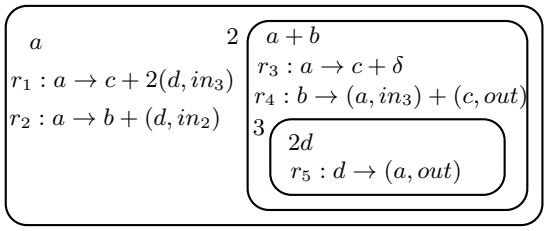

Figure 1. The initial configuration of Example 1.1

The set of objects of the system is given by $O = \{a, b, c, d\}$, and the membrane structure is clear from the Venn diagram. The initial multisets are $w_1^0 = a$, $w_2^0 = a + b$ and $w_3^0 = 2d$. The sets of rules are $R_1 = \{r_1, r_2\}$, $R_2 = \{r_3, r_4\}$ and $R_3 = \{r_5\}$, where $r_1 : a \to c + 2(d, in_3)$, $r_2 : a \to b + (d, in_2)$, $r_3 : a \to c + \delta$, $r_4 : b \to (a, in_3) + (c, out)$ and $r_5 : d \to (a, out)$. The rules contain target indications specifying the membranes where

the objects are sent. The objects either remain in the same membrane whenever they have no target attached, or they pass through membranes in two directions: they can be sent *out* of the membrane, or can be sent *in* one of the nested membranes which is precisely identified by its label. In one step, the objects can pass only through one membrane. This configuration is described graphically in Figure 1.

In membrane 1 we can only apply rule $r_2 : a \to b + (d, in_2)$ since rule $r_1 : a \to c + 2(d, in_3)$ is not eligible due to membrane 3 not being a child of membrane 1. In membrane 2 maximal parallelism requires the application of both rules r_3 and r_4. In membrane 3 rule r_5 is applied twice for the same reason. At this point we have $b+(d, in_2)$ in membrane 1, $c+(a, in_3)+(c, out)+\delta$ in membrane 2, and $2(a, out)$ in membrane 3. Now messages of form (x, out), (x, in_j) are sent to their respective destinations: d from 1 to 2, $2a$ from 3 to 2, a from 2 to 3, and c from 2 to 1. The evolution step ends with the dissolution of membrane 2, which is triggered by the presence of δ in membrane 2. Now the membrane system has only two membranes, as seen in Figure 2.

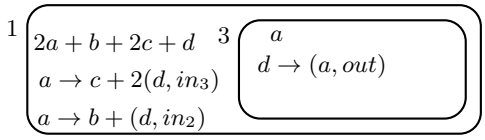

Figure 2. The P system configuration of Example 1.1 after one step

Some operational semantics for membrane systems were presented in [3] and [6]. For systems with a complex hierarchical structure, a reduction to a system with a single membrane (and additional rules) is presented in [1]. Such a reduction is achieved by encoding the semantic constraints of the hierarchical system within rule using promoters and inhibitors in the system consisting of just one membrane. This reduction is subsequently used as a technical tool to solve problems for complex systems by reducing them to simpler cases. Several applications of membrane systems are presented in [7].

B. Continuation semantics for concurrency

In the classic technique of continuations [13], a program is conceptually divided into a *current* statement and the *remainder* of the program. The continuation semantics for concurrency (CSC) is based on a similar idea. Intuitively, it is a semantic formalization of a process scheduler. In CSC, processes are grouped in what we call *continuation* in the case of denotational semantics, and respectively *resumption* in the case of operational semantics. On the other hand, there is one *active* or *current* process corresponding to the current statement. The term *process* is used here to denote a statement in the case of operational semantics, and a denotation of a statement (i.e., a computation) in the case of denotational semantics. The current statement can be evaluated either in sequence or in parallel with the remainder of the program.

In CSC, continuations are application-specific structures of computations rather than just functions to some final answer type as in the classic technique of continuations. The scheduling policy (of such a semantic scheduler) is given by the structure of continuations. In the particular case of the language $\mathcal{L}_{MR}$ studied in this paper, a continuation is a multiset of computations that are evaluated in parallel.

The evaluation by maximal parallel rewriting in $\mathcal{L}_{MR}$ allows to express repetitive computations (including non-terminating computations). Following the tradition of programming languages theory, in denotational semantics we express repetitive computation by using continuations in combination with semantic environments defined based on fixed point constructions. In the case of operational semantics we express repetitive computation by resumption manipulations and body replacement (syntactic environments).

C. Contribution

We define and then relate an operational semantics and a denotational semantics for a multiset concurrent language with computations specified by means of multiset rewriting rules and execution based on maximal parallelism. These features are encountered, e.g., in membrane computing [10]. The relation between the operational semantics and the denotational semantics is obtained within the framework of metric semantics [4], where the main mathematical tool is Banach's fixed point theorem [5]. The semantic models are defined by using continuations for concurrency [14]; the use of continuations appears to be essential for the success of our approach. To the best of our knowledge, this is the first attempt to present a study of comparative semantics for concurrent languages based on the parallel rewriting of multisets.

II. Mathematical preliminaries

A *multiset* is a generalization of a set. Intuitively, a multiset is a collection in which an element may occur more than once. We can present a multiset of elements of type X by using a functions from X to $\mathbb{N}$, or partial functions $m : X \to \mathbb{N}^+$, where $\mathbb{N}^+ = \mathbb{N} \setminus \{0\}$, namely the set of natural numbers without 0. $m(x)$ is called the *multiplicity* of x, representing its number of occurrences in m.

The notation $(x \in)X$ introduces the set X with typical element x ranging over X. Let X be a countable set. We denote by $[X]$ the set of all finite multisets of elements of type X, i.e., $[X] \stackrel{not.}{=} \bigcup_{A \in \mathcal{P}_{fin}(X)} \{m \mid m \in (A \to \mathbb{N}^+)\}$, where $\mathcal{P}_{fin}(X)$ is the powerset of all *finite* subsets of X. Since X is countable, $\mathcal{P}_{fin}(X)$ is also countable. An element $m \in [X]$ is a (finite) multiset of elements of type X, namely a function $m : A \to \mathbb{N}^+$, where $A \subseteq X$ is a finite subset of X, such that $\forall x \in A : m(x) > 0$.

We can also present a multiset $m \in [X]$ by enumerating its elements between parentheses '[' and ']'. Notice that the elements in a multiset are *not* ordered; a multiset is just an unordered list of elements. For example, [] is the empty multiset, i.e. the function with empty graph. Another example: $[x_1, x_1, x_2] = [x_1, x_2, x_1] = [x_2, x_1, x_1]$ is the multiset with two occurrences of x_1 and one occurrence of x_2, i.e. the function $m : \{x_1, x_2\} \to \mathbb{N}^+, m(x_1) = 2, m(x_2) = 1$.

We can define various operations on multisets $m_1, m_2 \in [X]$. Below, $\mathsf{dom}(\cdot)$ is the domain of function '$\cdot$'.

- *Multiset sum:* $m_1 \uplus m_2 \quad (\uplus : ([X] \times [X]) \to [X])$
  $\mathsf{dom}(m_1 \uplus m_2) = \mathsf{dom}(m_1) \cup \mathsf{dom}(m_2)$
  $(m_1 \uplus m_2)(x) =$
  $$\begin{cases} m_1(x) + m_2(x) & \text{if} \quad x \in \mathsf{dom}(m_1) \cap \mathsf{dom}(m_2) \\ m_1(x) & \text{if} \quad x \in \mathsf{dom}(m_1) \setminus \mathsf{dom}(m_2) \\ m_2(x) & \text{if} \quad x \in \mathsf{dom}(m_2) \setminus \mathsf{dom}(m_1) \end{cases}$$
- *Multiset difference:* $m_1 \setminus m_2 \quad (\setminus : ([X] \times [X]) \to [X])$
  $\mathsf{dom}(m_1 \setminus m_2) = (\, \mathsf{dom}(m_1) \setminus \mathsf{dom}(m_2) \,) \cup \{x \mid x \in \mathsf{dom}(m_1) \cap \mathsf{dom}(m_2)\,, m_1(x) > m_2(x)\}$
  $(m_1 \setminus m_2)(x) =$
  $$\begin{cases} m_1(x) & \text{if} \quad x \in \mathsf{dom}(m_1) \setminus \mathsf{dom}(m_2) \\ m_1(x) - m_2(x) & \text{if} \quad x \in \mathsf{dom}(m_1) \cap \mathsf{dom}(m_2) \end{cases}$$
- *Submultiset:* $m_1 \subseteq m_2 \quad (\subseteq : ([X] \times [X]) \to \mathit{Bool})$
  $m_1 \subseteq m_2$ iff $(\, \mathsf{dom}(m_1) \subseteq \mathsf{dom}(m_2) \,) \wedge (\forall x \in \mathsf{dom}(m_1) : m_1(x) \leq m_2(x))$.

The free commutative monoid on a set X can represent the set of finite multisets with elements from X.

Let $f \in X \to Y$ be a function. The function $(\, f \mid x \mapsto y \,) : X \to Y$ is defined, for $x, x' \in X, y \in Y$), by $(\, f \mid x \mapsto y \,)(x') =$ if $x' = x$ then y else $f(x')$. We also use the notation $(\, f \mid x_1 \mapsto y_1 \mid \cdots \mid x_n \mapsto y_n \,)$ as an abbreviation for $(\,(\, f \mid x_1 \mapsto y_1 \,) \cdots \mid x_n \mapsto y_n \,)$. If $f : X \to X$ and $f(x) = x$, we call x a *fixed point* of f. When this fixed point is unique, we write $x = fix(f)$.

The denotational and the operational semantics given in this paper are defined following the mathematical methodology of metric semantics [4]. More exactly, we work within the mathematical framework of *1-bounded complete metric spaces*. We assume the following notions are known: *metric* and *ultrametric* space, *isometry* (distance preserving bijection between metric spaces, denoted by '$\cong$'), *complete* metric space, and *compact* set. For details, the reader may consult the monograph [4], for instance.

Some metrics are frequently used in metric semantics. For example, if X is any nonempty set, we can define the *discrete metric* $d : X \times X \to [0, 1]$ as follows: $d(x, y) :=$ if $x = y$ then 0 else 1. (X, d) is a complete ultrametric space. Also, let A be a nonempty set, and $A^\infty = A^* \cup A^\omega$, where $A^*(A^\omega)$ is the set of all finite (infinite) sequences over A. A metric over A^∞ can be defined by $d(x, y) = 2^{-\sup\{\, n \mid x(n) = y(n) \,\}}$, where $x(n)$ denotes the prefix of x of length n, in case $length(x) \geq n$, and x otherwise (by convention, $2^{-\infty} = 0$). d is a Baire-like metric, and (A^∞, d) is a complete ultrametric space.

We recall that if (X, d_X), (Y, d_Y) are metric spaces, a function $f : X \to Y$ is a *contraction* if $\exists c \in \mathbb{R}$, $0 \leq c < 1$, $\forall x_1, x_2 \in X : d_Y(f(x_1), f(x_2)) \leq c \cdot d_X(x_1, x_2)$. In metric semantics, it is usual to attach a contracting factor $c = \frac{1}{2}$ to each computation step. When $c = 1$ the function f is called *nonexpansive*. In what follows, we denote by $X \stackrel{1}{\to} Y$ the set of all nonexpansive functions from X to Y.
The following theorem is at the core of metric semantics.

Theorem 2.1 (Banach): Let (X, d_X) be a complete metric space. Each contraction $f : X \to X$ has a *unique* fixed point.

Definition 2.2: Let $(X, d_X), (Y, d_Y)$ be (ultra)metric spaces. We define the following metrics over X, $X \to Y$ (function space), $X \times Y$ (Cartesian product), $X + Y$ (disjoint union defined by $X + Y = (\{1\} \times X) \cup (\{2\} \times Y)$), and $\mathcal{P}(X)$ (powerset of X), respectively.

(a) $d_{\frac{1}{2} \cdot X} : X \times X \to [0, 1] \quad d_{\frac{1}{2} \cdot X}(x_1, x_2) = \frac{1}{2} \cdot d_X(x_1, x_2)$

(b) $d_{X \to Y} : (X \to Y) \times (X \to Y) \to [0, 1]$
$d_{X \to Y}(f_1, f_2) = \sup_{x \in X} d_Y(f_1(x), f_2(x))$

(c) $d_{X \times Y} : (X \times Y) \times (X \times Y) \to [0, 1]$
$d_{X \times Y}((x_1, y_1), (x_2, y_2)) =$
$max\{d_X(x_1, x_2), d_Y(y_1, y_2)\}$;
Also, $fst : (X \times Y) \to X$ and $snd : (X \times Y) \to Y$ are defined by $fst(x, y) = x$ and $snd(x, y) = y$; both fst and snd are nonexpansive mappings.

(d) $d_{X+Y} : (X + Y) \times (X + Y) \to [0, 1]$
$d_{X+Y}(u, v) =$ if $(u, v \in X)$ then $d_X(u, v)$
else if $(u, v \in Y)$ then $d_Y(u, v)$ else 1

(e) $d_H : \mathcal{P}(X) \times \mathcal{P}(X) \to [0, 1]$:
$d_H(U, V) = max\{\sup_{u \in U} d(u, V), \sup_{v \in V} d(v, U)\}$
where $d(u, W) = \inf_{w \in W} d(u, w)$ and by convention $\sup \emptyset = 0$ and $\inf \emptyset = 1$; d_H is a *Hausdorff* metric.

We use the abbreviation $\mathcal{P}_{nco}(X)$ to denote the powerset of *non-empty and compact* subsets of X. Also, we often suppress the metrics part in domain definitions, and write only $\frac{1}{2} \cdot X$ instead of $(X, d_{\frac{1}{2} \cdot X})$.

Remark 2.3: Let $(X, d_X), (Y, d_Y), d_{\frac{1}{2} \cdot X}, d_{X \to Y}, d_{X \times Y}, d_{X+Y}$ and d_H be as in Definition 2.2. If d_X, d_Y are ultrametrics, then so are $d_{\frac{1}{2} \cdot X}, d_{X \to Y}, d_{X \times Y}, d_{X+Y}$ and d_H. Moreover, if $(X, d_X), (Y, d_Y)$ are complete then $\frac{1}{2} \cdot X$, $X \to Y$, $X \stackrel{1}{\to} Y$, $X \times Y$, $X + Y$, and $\mathcal{P}_{nco}(X)$ with their metrics defined above are also complete metric spaces [4].

We also use the abbreviation $\mathcal{P}_{fin}(X)$ to denote the powersets of *finite* subsets of X. In general, the construct $\mathcal{P}_{fin}(\cdot)$ does not give rise to a complete metric space; we use it to create a structure equipped with the discrete metric. Any set equipped with the discrete metric is a complete ultrametric space.

A. Alternative representation of multisets

Let X be a set. We use the following notation:

$$\{\!|X|\!\}^A \quad \stackrel{not.}{=} \quad \mathcal{P}_{fin}(A) \times (A \to X),$$

where A is a countable set. An element of type $\{\!|X|\!\}^A$ is a pair (π, ϖ) consisting of a finite set $\pi \in \mathcal{P}_{fin}(A)$ of identifiers, and an occurrence mapping $\varpi \in A \to X$. We use this structure to represent a finite *bag* or *multiset* of elements of type X. The set A is used to distinguish between multiple occurrences of an element in a multiset. We treat (π, ϖ) as a 'function' with finite graph $\{(\alpha, \varpi(\alpha)) \mid \alpha \in \pi\}$, thus ignoring the behavior of ϖ for any $\alpha \notin \pi$ (π is the 'domain' of the 'function'). For example, if $\pi = \{\alpha_1, \alpha_2, \alpha_3\}$ and $\varpi : A \to X$, with $\varpi(\alpha_1) = x_1, \varpi(\alpha_2) = x_2, \varpi(\alpha_3) = x_1$ then (π, ϖ) is a representation of the multiset with 2 occurrences of x_1 and 1 occurrence of x_2.

We define $id : \{\!|X|\!\}^A \to \mathcal{P}_{fin}(A)$, $id(\pi, \varpi) = \pi$. We assume that there is always a mapping $\nu : \mathcal{P}_{fin}(A) \to A$ generating a new identifier, such that $\nu(\pi) \notin \pi, \forall \pi \in \Pi$. For example, we could set $A = \mathbb{N}$, and $\nu(\pi) = 1 + max\{\alpha \mid \alpha \in \pi\}$. We introduce the functions: $(\cdot)(\cdot) : \{\!|X|\!\}^A \times A \to X$, $(\cdot \setminus \cdot) : \{\!|X|\!\}^A \times \mathcal{P}_{fin}(A) \to \{\!|X|\!\}^A$, and $(\cdot : \cdot) : X \times \{\!|X|\!\}^A \to \{\!|X|\!\}^A$, defined as follows:

$$\begin{array}{rcl} (\pi, \varpi)(\alpha) & = & \varpi(\alpha) \\ (\pi, \varpi) \setminus \pi' & = & (\pi \setminus \pi', \varpi) \\ x : (\pi, \varpi) & = & (\pi \cup \{\alpha\}, (\, \varpi \mid \alpha \mapsto x\,)) \\ & & \text{where } \ \alpha = \nu(\pi) \end{array}$$

The functions behave as follows. $id(\cdot)$ returns the collection of identifiers for the valid elements contained in the multiset, $(\cdot)(\alpha)$ returns the element with identifier α. $(\cdot) \setminus \pi$ removes the elements with identifiers in π, and $x : (\cdot)$ adds the element x to the multiset (a new, fresh identifier $\alpha(\notin \pi)$ is automatically generated for x).

Clearly, this representation of multisets is less abstract than the one given at the beginning of section II, but it can be used to model finite multisets of elements taken from an arbitrary (possibly uncountable) set X. We use this construction for both plain sets and metric domains.

Let X be a metric domain, i.e. a complete metric space. By a slight abuse, we use the (same) notation $\{\!|\mathbf{X}|\!\}^A$ for both when $\mathbf{X}$ is a metric domain and just a plain set.

$$\{\!|\mathbf{X}|\!\}^A \quad \stackrel{not.}{=} \quad \mathcal{P}_{fin}(A) \times (A \to \mathbf{X})$$

We equip both sets A and $\mathcal{P}_{fin}(A)$ with discrete metrics. By using the composite metrics given in Definition 2.2, $\{\!|\mathbf{X}|\!\}^A$ becomes also a metric domain. We treat an element of $\{\!|\mathbf{X}|\!\}^A$ as a finite multiset of computations of type $\mathbf{X}$.

We define the functions

$id : \{\!|\mathbf{X}|\!\}^A \to \mathcal{P}_{fin}(A)$ by $id(\pi, \varpi) = \pi$,

$(\cdot)(\cdot) : \{\!|\mathbf{X}|\!\}^A \times A \to \mathbf{X}$ by $(\pi, \varpi)(\alpha) = \varpi(\alpha)$,

$(\cdot \setminus \cdot) : \{\!|\mathbf{X}|\!\}^A \times \mathcal{P}_{fin}(A) \to \{\!|\mathbf{X}|\!\}^A$ by $(\pi, \varpi) \setminus \pi' = (\pi \setminus \pi', \varpi)$, and

$(\cdot : \cdot) : \mathbf{X} \times \{\!|\mathbf{X}|\!\}^A \to \{\!|\mathbf{X}|\!\}^A$ by $x : (\pi, \varpi) =$

$(\pi \cup \{\alpha\}, (\, \varpi \mid \alpha \mapsto x\,))$, where $\alpha = \nu(\pi)$, as above.

Here $\mathbf{X}$ is a metric domain (rather than just a plain set). When $\mathbf{X}$ is a metric domain (and $\{\!|\mathbf{X}|\!\}^A$, A and $\mathcal{P}_{fin}(A)$ are equipped with metrics as explained above), we can easily check that each of the functions $id(\cdot), (\cdot)(\cdot), (\cdot \setminus \cdot), (\cdot : \cdot)$ is nonexpansive.

Let X be a set, $x_1, \cdots, x_n \in X$, and $(\pi, \varpi) \in \{\!|X|\!\}^A$. It is convenient to use the following notations:

$$\{\!|x_1, \cdots, x_n|\!\}_{(\pi,\varpi)} \stackrel{not.}{=} (x_1 : \cdots (x_n : (\pi, \varpi)) \cdots)$$

$$\{\!|x_i \mid i \in \{i_1, \cdots, i_n\}|\!\}_{(\pi,\varpi)} \stackrel{not.}{=} \{\!|x_{i_1}, \cdots, x_{i_n}|\!\}_{(\pi,\varpi)}.$$

Similarly, when $\mathbf{X}$ is a metric domain, $x_1, \cdots, x_n \in \mathbf{X}$, and $(\pi, \varpi) \in \{\!|\mathbf{X}|\!\}^A$. When (π, ϖ) is understood from the context, we write $\{\!|x_1, \cdots, x_n|\!\}$ instead of $\{\!|x_1, \cdots, x_n|\!\}_{(\pi,\varpi)}$. Notice that $(\pi, \varpi) = \{\!|\,|\!\}_{(\pi,\varpi)}, \forall (\pi, \varpi) \in \{\!|X|\!\}^A$.

III. Syntax of $\mathcal{L}_{MR}$

The syntax of $\mathcal{L}_{MR}$ was introduced in [8]. Let $(o \in) O$ be an alphabet of *objects*; we assume that O is a countable set. $W = [O]$ is the set of all finite multisets of O objects. [1]

Definition 3.1: (Syntax of $\mathcal{L}_{MR}$)

(a) (Statements) $\quad x ::= o \mid x \| x$

(b) (Rules) $\quad r ::= \epsilon \mid w \Rightarrow x \,\square\, r$

(c) (Programs) $\quad (\rho \in) \mathcal{L}_{MR} = R \times X$

An $\mathcal{L}_{MR}$ program ρ is a pair (r, x) consisting of a set of rewriting rules $r \in R$ and a statement $x \in X$. The rules of a list $r = (w_1 \Rightarrow x_1 \,\square\, \cdots \,\square\, w_n \Rightarrow x_n)$ are assumed to be pairwise distinct.

An $\mathcal{L}_{MR}$ statement is either an object o or the parallel composition of two statements $(x_1 \parallel x_2)$. The semantics of an $\mathcal{L}_{MR}$ statement is a multiset of objects that are evaluated in parallel. A rule is similar to a 'procedure' declaration. Intuitively, in a construct $w \Rightarrow x$, the multiset w is the 'name', and x is the 'body' of the 'procedure'; w is composed of several objects which may be seen as 'fragments' of the 'procedure name'. This intuition is inspired by the Join Calculus [9], where procedure names are also composed of several fragments. Only when all the 'fragments' of such a 'procedure name' are prepared for interaction a rewriting rule is applied by replacing the 'name of the procedure' with its 'body'. Essentially, a construct $w \Rightarrow x$ specifies a multiset rewriting rule, and the semantics of x is a multiset of computations that are evaluated in parallel.

Notice that we incorporate the semantic notion of a multiset in the syntax of $\mathcal{L}_{MR}$. However, it would be easy to make a complete separation between syntax and semantics. For example, in Definition 3.1 we could use rules of the form $j \Rightarrow x$, where $j ::= o \mid j \,\&\, j$ is the set of procedure names (this syntax is again inspired by the Join Calculus). We decided to use multisets as procedure names because the order in which 'fragments' occur in such a 'procedure name' is irrelevant.

IV. Operational semantics

The operational semantics of $\mathcal{L}_{MR}$ is based on a transition relation embedded in a deductive system in the style of

[1] The construction $[\cdot]$ was introduced at the beginning of section II.

Plotkin's structural operational semantics [12]. We define the transition semantics of $\mathcal{L}_{MR}$ by using continuations for concurrency [14]. We use the term *resumption* as an operational counterpart of the term *continuation*.

An $\mathcal{L}_{MR}$ program consists of a finite set of rewriting rules, that are applied in a maximally parallel manner. When several combinations of rules are applicable, the selection of (the combination of) rules is nondeterministic. We model computations in $\mathcal{L}_{MR}$ as collections of sequences of multisets of objects. We employ "collections" because computation is nondeterministic. We use multisets of objects (rather than just objects) because the reduction of parallel objects proceeds simultaneously, without interleaving. We define the semantic universe $\mathbf{P}$ for $\mathcal{L}_{MR}$, both for operational and denotational semantics.

Definition 4.1: We consider $\mathbf{P} = \mathcal{P}_{nco}(\mathbf{Q})$, where $\mathbf{Q}$ is the (unique) solution of the following metric domain equation:

$$\mathbf{Q} \cong \{\epsilon\} + (W \times \frac{1}{2} \cdot \mathbf{Q})$$

The set $W = [O]$ is equipped with the discrete metric.

We use '$\cdot$' as a prefixing operator over ($\mathbf{Q}$) sequences: $w \cdot q = (w, q)$, for $q \in \mathbf{Q}$. Instead of $(w_1, (w_2, \cdots (w_n, \epsilon) \cdots))$ we write $w_1 w_2 \cdots w_n$. Also, we use the notation $w \cdot p = \{w \cdot q \mid q \in p\}$, for any $p \in \mathbf{P}$.

In Definition 4.2 we introduce an auxiliary mapping $appRules(r, w')$ which takes as arguments a set r of $\mathcal{L}_{MR}$ rewriting rules and a multiset w' of objects. The mapping $appRules(r, w')$ computes a (finite) set of pairs $\{(r''_1, w''_1), \cdots, (r''_n, w''_n)\}$. Each r''_i is a multiset of rewriting rules applicable to w and w''_i is a (sub)multiset (of w) which is irreducible with respect to r.

Definition 4.2: $appRules : (R \times W) \to \mathcal{P}_{fin}(R \times W)$

$appRules(r, w) =$
  if $aux(r, w) = \emptyset$
  then $\{(\epsilon, w)\}$
  else $\{(\overline{w} \Rightarrow \overline{x} \,\square\, r', w'')$
    $\mid ((\overline{w}, \overline{x}), w') \in aux(r, w)$
    $(r', w'') \in appRules(r, w')\}$

where

$aux : (R \times W) \to \mathcal{P}_{fin}((W \times X) \times W)$
$aux(\epsilon, w) = \emptyset$
$aux(w' \Rightarrow x' \,\square\, r, w) =$
  if $(w' \subseteq w)$
  then $\{((w', x'), w \setminus w')\} \cup aux(r, w)$
  else $aux(r, w)$.

Remarks 4.3:

(a) The definitions of $appRules(r, w)$ and $aux(r, w)$ can be justified by an easy induction (on the number of elements in the multiset w, and by induction on the length of list r, respectively).

(b) For any $r \in R$ and $w \in W$, either $appRules(r, w) = \{(\epsilon, w)\}$, or $appRules(r, w) = \{(r''_1, w''_1), \cdots, (r''_m, w''_m)\}$, where each $r''_j \neq \epsilon$, i.e. $r''_j = w_1 \Rightarrow x_1 \,\square\, \cdots \,\square\, w_n \Rightarrow x_n$ $(m \geq 1, n \geq 1)$.

We introduce the configurations of the transition system. Let Id be a (countable) set of *identifiers*. [2]

Definition 4.4: (Configurations)

(a) $Conf = (X \times C) \cup \{E\}$,
(b) $C = (K \times W)$,
(c) $K = \{\!|X|\!\}^{Id}$.

$Conf$ is the class of *configurations* of the transition system defined later by Definition 4.6. A configuration $z \in Conf$ is either a pair $(x, c) \in X \times C$ consisting of an $\mathcal{L}_{MR}$ statement $x \in X$ and a resumption $c \in C$, or the symbol E which denotes termination. A resumption $c \in C$ is a pair (k, w) consisting of a multiset $k \in \{\!|X|\!\}^{Id}$ of $\mathcal{L}_{MR}$ statements (that are evaluated in parallel), and a multiset $w \in W$ of objects that is irreducible with respect to the set of rules of the program.

Definition 4.5: Let $mset : X \to W$ defined by $mset(o) = [o]$, $mset(x_1 \parallel x_2) = mset(x_1) \uplus mset(x_2)$. Let $mset_K : \{\!|X|\!\}^{Id} \to W$ and $mset_K(k) = \biguplus_{\alpha \in id(k)} mset(k(\alpha))$; $mset$ and $mset_K$ compute the multiset of objects contained in either an X statement or an $\{\!|X|\!\}^{Id}$ multiset, respectively.

Notation: Let $\hat{o} \in O$ be a distinguished object, and $c_0 = (\{\!|\}\!\}_{k_0}, []) \in C$ the *empty resumption*, where $k_0 \in \{\!|X|\!\}^{Id}$, $k_0 = (\emptyset, \lambda\,.\,\hat{o})$.

The operational semantics of $\mathcal{L}_{MR}$ is based on a transition relation $\longrightarrow \subseteq Conf \times W \times R \times Conf$ with elements (z, w, r, z') written as $z \xrightarrow{w}_r z'$. Intuitively, z is the initial configuration, z' is the final configuration and w is the 'label' of such a transition. In general, a transition also depends on the set r of rules of the program that is evaluated. We specify the transition relation by a set of axioms $z \xrightarrow{w}_r z'$, and rules of the form $\frac{z_2 \xrightarrow{w}_r z'}{z_1 \xrightarrow{w}_r z'}$. The transitions of the system can be inferred by *backward chaining*. In order to find the transitions of z_1 we try to infer the transitions of z_2, as any transition of z_2 is also a transition of z_1. We use the following notation:

(a) $z_1 \nearrow_r z_2$ is an abbreviation for $\dfrac{z_2 \xrightarrow{w}_r z'}{z_1 \xrightarrow{w}_r z'}$.
(b) $\langle n \setminus i \rangle$ is an abbreviation for the set $\{1, \cdots, n\} \setminus \{i\}$, where $n, i \in \mathbb{N}$.

Definition 4.6: (Transition system $\mathcal{T}_{MR}$) The transition relation $\longrightarrow$ for $\mathcal{L}_{MR}$ is the smallest subset of $Conf \times R \times W \times Conf$ satisfying the axioms and rules given below. In axioms (A1) and (A2) $w' = [o] \uplus w \uplus mset_K(k)$, $\varrho = appRules(r, w')$, and k_0 is part of the empty resumption.

[2] For example, we can set $Id = \mathbb{N}$ and define $\nu : \mathcal{P}_{fin}(Id) \to Id$, $\nu(\pi) = 1 + max\{\alpha \mid \alpha \in \pi\}$, as explained in subsection II-A.

(A1) $(o,(k,w)) \xrightarrow{w'}_r E$ if $\varrho = \{(\epsilon, w')\}$

(A2) $(o,(k,w)) \xrightarrow{w'}_r (x_i,(\{\!|x_j \mid j \in \langle n \setminus i\rangle|\!\}_{k_0}, w''))$
if $(w_1 \Rightarrow x_1 \,\square\, \cdots \,\square\, w_n \Rightarrow x_n, w'') \in \varrho,\ 1 \le i \le n$

(R3) $(x_1 \parallel x_2,(k,w)) \nearrow_r (x_1,(x_2 : k,w))$

(R4) $(x_1 \parallel x_2,(k,w)) \nearrow_r (x_2,(x_1 : k,w))$

A configuration of the form $(x,(k,w))$ is a semantic representation of an $\mathcal{L}_{MR}$ program decomposed into a *current* statement x, and the remainder of the program which is encapsulated in the resumption (k,w), The current computation is evaluated in parallel with the resumption. If the current computation x is an (elementary) object o the system performs a reduction step corresponding to a multiset of rewriting rules which are applied in parallel; this is expressed in axioms (A1) and (A2).

The multiset of rewriting rules is computed by the mapping $appRules(r,w')$ which takes as arguments the set r of rules of the program which is evaluated, and a multiset w' containing the (current) object o plus all objects contained in the resumption (k,w). $appRules(r,w')$ computes a (finite) set ϱ of pairs (r'',w''), each such pair consisting of a multiset of rewriting rules $r'' = w_1 \Rightarrow x_1 \,\square\, \cdots \,\square\, w_n \Rightarrow x_n$ applicable to w' and an irreducible (sub)multiset w'' (of w'). According to axiom (A2) the multiset of rules is selected in a nondeterministic manner and all rules (in the multiset) are applied in parallel in a single step. When no rule in r is applicable to w', the mapping $appRules$ returns $\{(\epsilon,w')\}$; in this case axiom (A1) states that the $\mathcal{L}_{MR}$ program produces w' as observable, and then terminates.

According to rules (R3) and (R4), the semantics of parallel composition $x_1 \parallel x_2$ is based on a nondeterministic choice between two computations: one evaluates x_1 in parallel with x_2 added to the resumption and the other one evaluates x_2 in parallel with x_1 added to the resumption.

Definition 4.7: (Operational semantics of $\mathcal{L}_{MR}$)

(a) Let $Sem_O = Conf \to \mathbf{P}$ and let $\Psi_r : Sem_O \to Sem_O$ be defined by
$\Psi_r(S)(E) = \{\epsilon\}$,
$\Psi_r(S)(x,c) = \bigcup\{w \cdot S(z) \mid (x,c) \xrightarrow{w}_r z\}$

(b) We put $\mathcal{O}_r = fix(\Psi_r)$, and define $\mathcal{O}[\![\cdot]\!] : \mathcal{L}_{MR} \to \mathbf{P}$ by $\mathcal{O}[\![(r,x)]\!] = \mathcal{O}_r(x,c_0) = \mathcal{O}_r(x,(\{\!|\,|\!\}_{k_0},[]))$.

Remarks 4.8:

(a) It is not difficult to prove that $\mathcal{T}_{MR}$ is finitely branching (i.e., for all $z \in Conf$, $r \in R$, the set $\{(w',z') \mid z \xrightarrow{w'}_r z'\}$ is finite), and thus it induces a compact operational semantics (see [4]). E has no transitions. When $z = (x,c)$, the proof can proceed by structural induction on x (using the fact that $appRules$ yields a finite set).

(b) Ψ_r is contracting due to the "$w\cdot$"-step in its definition, and thus it has a unique fixed point

Example 4.9: Let $\rho \in \mathcal{L}_{MR}$ be $\rho = (r, o_1 \parallel o_2)$, where $r = [o_1] \Rightarrow o_2 \,\square\, [o_1] \Rightarrow o_3$. In this example we simply write $\{\!|\cdots|\!\}$ instead of $\{\!|\cdots|\!\}_{k_0}$. We compute $\mathcal{O}[\![(r, o_1 \parallel o_2)]\!] = \mathcal{O}_r(o_1 \parallel o_2, (\{\!|\,|\!\},[]))$. We have:
$(o_1 \parallel o_2,(\{\!|\,|\!\},[])) \nearrow_r (o_1,(\{\!|o_2|\!\},[]))$
$(o_1 \parallel o_2,(\{\!|\,|\!\},[])) \nearrow_r (o_2,(\{\!|o_1|\!\},[]))$
Let $z_1 = (o_1,(\{\!|o_2|\!\},[])), z_2 = (o_2,(\{\!|o_1|\!\},[]))$.
$\mathcal{O}_r(o_1 \parallel o_2,(\{\!|\,|\!\},[])) = p_1 \cup p_2$, where
$p_1 = \bigcup\{\overline{w} \cdot \mathcal{O}_r(\overline{z}) \mid z_1 \xrightarrow{\overline{w}}_r \overline{z}\}$
and $p_2 = \bigcup\{\overline{w} \cdot \mathcal{O}_r(\overline{z}) \mid z_2 \xrightarrow{\overline{w}}_r \overline{z}\}$.
We compute the transitions of $z_1 = (o_1,(\{\!|o_2|\!\},[]))$.
Let $w' = [o_1] \uplus [\,] \uplus mset_K(\{\!|o_2|\!\}) = [o_1,o_2]$.
As $appRules(r,w')$=$\{([o_1] \Rightarrow o_2,[o_2]),([o_1] \Rightarrow o_3,[o_2])\}$,
$(o_1,(\{\!|o_2|\!\},[])) \xrightarrow{[o_1,o_2]}_r (o_2,(\{\!|\,|\!\},[o_2]))$
$(o_1,(\{\!|o_2|\!\},[])) \xrightarrow{[o_1,o_2]}_r (o_3,(\{\!|\,|\!\},[o_2]))$
$[o_2] \uplus [o_2] \uplus mset_K(\{\!|\,|\!\}) = [o_2,o_2]$ and $appRules(r,[o_2,o_2]) = \{(\epsilon,[o_2,o_2])\}$, hence
$(o_2,(\{\!|\,|\!\},[o_2])) \xrightarrow{[o_2,o_2]}_r E$
Also, $[o_3] \uplus [o_2] \uplus mset_K(\{\!|\,|\!\}) = [o_2,o_3]$ and $appRules(r,[o_2,o_3]) = \{(\epsilon,[o_2,o_3])\}$, therefore
$(o_3,(\{\!|\,|\!\},[o_2])) \xrightarrow{[o_2,o_3]}_r E$
Thus,
$p_1 = [o_1,o_2] \cdot \mathcal{O}_r(o_2,(\{\!|\,|\!\},[o_2])) \cup$
$\quad [o_1,o_2] \cdot \mathcal{O}_r(o_3,(\{\!|\,|\!\},[o_2]))$
$= [o_1,o_2] \cdot ([o_2,o_2] \cdot \mathcal{O}_r(E)) \cup [o_1,o_2] \cdot ([o_2,o_3] \cdot \mathcal{O}_r(E))$
$= \{[o_1,o_2][o_2,o_2],[o_1,o_2][o_2,o_3]\}$
It turns out that $p_2 = p_1$. Therefore $\mathcal{O}[\![\rho]\!] = \mathcal{O}[\![(r,o_1 \parallel o_2)]\!] = \{[o_1,o_2][o_2,o_2],[o_1,o_2][o_2,o_3]\}$.

V. Denotational semantics

The denotational semantics $[\![\cdot]\!]$ of $\mathcal{L}_{MR}$ is similar to that presented in [8]. Here the definitions are without the formal justifications given in [8]. [3] The domain definitions are:

$(\psi \in)\mathbf{D} = W \times \mathbf{F}$
$(\phi \in)\mathbf{F} \cong \mathbf{C} \xrightarrow{1} \mathbf{P}$
$(\gamma \in)\mathbf{C} = \mathbf{K} \times W$
$(\kappa \in)\mathbf{K} = \{\!|W \times (\frac{1}{2} \cdot \mathbf{F})|\!\}^{Id}$
$(\eta \in)\mathbf{E} = O \to \mathbf{D}$

$\mathbf{D}$ is the domain of denotations. and $\mathbf{C}$ is the domain of continuations. The notation $\{\!|\cdot|\!\}^{Id}$ is explained in Subsection II-A, and the set $W = [O]$ is as in Sections III and IV. Here the sets W and Id are equipped with discrete metrics. $\mathbf{E}$ is the domain of semantic environments, which are mappings from object names O to denotations $\mathbf{D}$. The final domain $\mathbf{P}$ is given in Definition 4.1. The domain equation for $\mathbf{D}$ has

[3] There is only one 'cosmetic' difference: $\mathbf{C} = W \times \mathbf{K}$ in [8].

a unique solution up to an isomorphism '$\cong$' [2]. For further explanations see [8].

The denotational mapping $[\![\cdot]\!] : X \to \mathbf{E} \to \mathbf{D}$ is defined by

$$
\begin{array}{l}
[\![\cdot]\!] : X \to \mathbf{E} \stackrel{1}{\to} \mathbf{D} \\
[\![o]\!]\eta = \eta(o) \\
[\![x_1 \parallel x_2]\!]\eta = \\
\quad \mathsf{let}\ (w_1, \phi_1) = [\![x_1]\!]\eta \\
\qquad\ (w_2, \phi_2) = [\![x_2]\!]\eta \\
\quad \mathsf{in}\ (w_1 \uplus w_2\ ,\ \lambda(\kappa, w).(\phi_1((w_2, \phi_2) : \kappa, w) \cup \\
\qquad\qquad\qquad\qquad \phi_2((w_1, \phi_1) : \kappa, w))).
\end{array}
$$

We define a semantic environment $\eta_0 : \mathbf{E}$ as (the unique) fixed point of a higher-order mapping Φ_r, which is given (for any set $r \in R$ of $\mathcal{L}_{MR}$ rewriting rules) by:

$$
\begin{array}{l}
\Phi_r : \mathbf{E} \to \mathbf{E} \\
\Phi_r(\eta)(o) = \\
\quad ([o], \\
\quad\ \lambda(\kappa, w)\,. \\
\qquad \mathsf{let}\ w' = [o] \uplus w \uplus (\biguplus_{\alpha \in id(\kappa)} fst(\kappa(\alpha))) \\
\qquad\quad\ \varrho = appRules(r, w') \\
\qquad \mathsf{in}\ w' \cdot (\ \mathsf{if}\ \varrho = \{(\epsilon, w'')\}\ \mathsf{then}\ \{\epsilon\} \\
\qquad\qquad\qquad \mathsf{else}\ \bigcup_{(r'', w'') \in \varrho}\ exe(r'', w'', \eta))),
\end{array}
$$

where

$$
\begin{array}{l}
exe(w_1 \Rightarrow x_1 \,\square\, \cdots \,\square\, w_n \Rightarrow x_n, w'', \eta) = \\
\quad \bigcup_{i \in I}(snd([\![x_i]\!]\eta))(\{\!|[\![x_j]\!]\eta \mid j \in \langle n \setminus i \rangle|\!\}_{\kappa_0}, w''),
\end{array}
$$

and $\kappa_0 = (\emptyset, \lambda\alpha\,.\,[\![\hat{o}]\!]\eta_0)$. In [8] it is shown that Φ_r is indeed a contraction ($\Phi_r : \mathbf{E} \stackrel{\frac{1}{2}}{\to} \mathbf{E}$). We denote $\eta_0 = fix(\Phi_r)$.

Finally, we define $\mathcal{D}[\![\cdot]\!] : \mathcal{L}_{MR} \to \mathbf{P}$ by:

$$\mathcal{D}[\![(r, x)]\!] = (snd([\![x]\!]\eta_0))(\{\!|\}\!|_{\kappa_0}, [])$$

Remark 5.1: $fst([\![x]\!]\eta_0) = mset(x)$, for all $x \in X$. This can be checked easily by structural induction on x.

Examples 5.2:

(a) Let $\rho = (r, o_1 \parallel o_2)$ be the $\mathcal{L}_{MR}$ program given in example 4.8. In [8] it is shown that $\mathcal{D}[\![\rho]\!] = \{[o_1, o_2][o_2, o_2], [o_1, o_2][o_2, o_3]\}$, i.e. $\mathcal{D}[\![\rho]\!] = \mathcal{O}[\![\rho]\!]$.

(b) Let $\rho' = (r', o_1 \parallel o_2)$, $r' = [o_1] \Rightarrow o_2 \parallel o_4 \,\square\, [o_1, o_2] \Rightarrow o_2 \,\square\, [o_2] \Rightarrow o_3$; the behavior of this $\mathcal{L}_{MR}$ example program is explained in [8]. We do not give here the details of how $\mathcal{O}[\![\cdot]\!]$ and $\mathcal{D}[\![\cdot]\!]$ provide the meaning of ρ'. One can check that $\mathcal{D}[\![\rho']\!] = \mathcal{O}[\![\rho']\!] = \{[o_1, o_2][o_2, o_3, o_4][o_3, o_3, o_4], [o_1, o_2][o_2][o_3]\}$.

VI. The relation between the operational semantics and the denotational semantics

In this section we prove that $\mathcal{O}[\![\rho]\!] = \mathcal{D}[\![\rho]\!], \forall \rho \in \mathcal{L}_{MR}$. First, we introduce an auxiliary mapping $\mathcal{R}_r : Conf \to \mathbf{P}$. We show that $\mathcal{R}_r = fix(\Psi_r)$, where Ψ_r is given in Definition 4.7. The desired result is obtained in Theorem 6.4 by using Lemma 6.3; as is customary in metric semantics, the proof relies on Banach's fixed point.

Definition 6.1:

(a) Let $[\![\![\cdot]\!]\!] : K \to \mathbf{K}$ be given by

$$[\![\![k]\!]\!] = (id(k), \lambda\alpha\,.\,[\![k(\alpha)]\!]\eta_0)$$

(b) We define $\mathcal{R}_r : Conf \to \mathbf{P}$ by

$$
\begin{array}{l}
\mathcal{R}_r(E) = \{\epsilon\} \\
\mathcal{R}_r(x, (k, w)) = (snd([\![x]\!]\eta_0))([\![\![k]\!]\!], w)
\end{array}
$$

The following result is needed in the proof of Lemma 6.3.

Lemma 6.2: Let $x, x_i \in X, k \in Conf$. We have

(a) $[\![\![x : k]\!]\!] = [\![x]\!]\eta_0 : [\![\![k]\!]\!]$

(b) $[\![\![\{\!|x_i \mid i \in I|\!\}_k]\!]\!] = \{\!|[\![x_i]\!]\eta_0 \mid i \in I|\!\}_{[\![\![k]\!]\!]}$, $I \in \mathcal{P}_{fin}(\mathbb{N})$

(c) $\{\!|\}\!|_{\kappa_0} = \kappa_0 = [\![\![k_0]\!]\!] = [\![\![\{\!|\}\!|_{k_0}]\!]\!]$

where η_0, κ_0 are given in section V and k_0 is part of the empty resumption.

The following result is an important ingredient in the proof of Theorem 6.4.

Lemma 6.3: $\mathcal{R}_r = fix(\Psi_r)$.

Proof: We show that $\forall z \in Conf : \Psi_r(\mathcal{R}_r)(z) = \mathcal{R}_r(z)$. Indeed, $\Psi_r(\mathcal{R}_r)(E) = \mathcal{R}_r(E) = \{\epsilon\}$. Next, when $z = (x, (k, w))$ we proceed by structural induction on $x \in X$.

- Case $x = o$. Let $w' = [o] \uplus w \uplus mset_K(k)$ and $\varrho = appRules(r, w')$. Notice that $fst([\![\![k]\!]\!](\alpha)) = fst([\![k(\alpha)]\!]\eta_0) = mset(k(\alpha))$, by Remark 5.1, and $id([\![\![k]\!]\!]) = id(k)$. Hence $w' = [o] \uplus w \uplus (\biguplus_{\alpha \in id(k)} mset(k(\alpha)))$ $= [o] \uplus w \uplus (\biguplus_{\alpha \in id([\![\![k]\!]\!])} fst([\![\![k]\!]\!](\alpha)))$. If $\varrho = \{(\epsilon, w'')\}$ there is a single transition $(o, (k, w)) \xrightarrow{w'}_r E$. In this subcase we compute as follows:

$$
\begin{array}{l}
\Psi_r(\mathcal{R}_r)(o, (k, w)) \\
= w' \cdot \mathcal{R}_r(E) = \{w'\} \\
= (snd(\eta_0(o)))([\![\![k]\!]\!], w) \\
= (snd([\![o]\!]\eta_0))([\![\![k]\!]\!], w) \\
= \mathcal{R}_r(o, (k, w))
\end{array}
$$

  Otherwise, $\varrho = \{(r''_1, w''_1), \cdots, (r''_m, w''_m)\}$, where $r''_1 \neq \epsilon, \cdots, r''_m \neq \epsilon$, (see Remark 4.3(b)). In this subcase:

$$
\begin{array}{l}
\Psi_r(\mathcal{R}_r)(o, (k, w)) \\
= \bigcup_{(w_1 \Rightarrow x_1 \,\square\, \cdots \,\square\, w_n \Rightarrow x_n, w'') \in \varrho} \\
\qquad w' \cdot \mathcal{R}_r(x_i, (\{\!|x_j \mid j \in \langle n \setminus i \rangle|\!\}_{k_0}, w'')) \\
= \bigcup_{(w_1 \Rightarrow x_1 \,\square\, \cdots \,\square\, w_n \Rightarrow x_n, w'') \in \varrho} \\
\qquad w' \cdot (snd([\![x_i]\!]\eta_0))([\![\![\{\!|x_j \mid j \in \langle n \setminus i \rangle|\!\}_{k_0}]\!]\!], w'') \\
\qquad\qquad \text{[Lemma 6.2(b) and Lemma 6.2(c)]} \\
= \bigcup_{(w_1 \Rightarrow x_1 \,\square\, \cdots \,\square\, w_n \Rightarrow x_n, w'') \in \varrho} \\
\qquad w' \cdot (snd([\![x_i]\!]\eta_0))(\{\!|[\![x_j]\!]\eta_0 \mid j \in \langle n \setminus i \rangle|\!\}_{\kappa_0}, w'') \\
= (snd(\eta_0(o)))([\![\![k]\!]\!], w) \\
= (snd([\![o]\!]\eta_0))([\![\![k]\!]\!], w) \\
= \mathcal{R}_r(o, (k, w))
\end{array}
$$

- Case $x = x_1 \parallel x_2$. Let $z = (x_1 \parallel x_2, (k, w))$, $z_1 = (x_1, (x_2 : k, w))$ and $z_2 = (x_2, (x_1 : k, w))$.

$$
\begin{array}{l}
\Psi_r(\mathcal{R}_r)(z) = \bigcup\{w_1 \cdot \mathcal{R}_r(z'_1) \mid z_1 \xrightarrow{w_1}_r z'_1\} \cup \\
\qquad\qquad \bigcup\{w_2 \cdot \mathcal{R}_r(z'_2) \mid z_2 \xrightarrow{w_2}_r z'_2\} \\
= \Psi_r(\mathcal{R}_r)(z_1) \cup \Psi_r(\mathcal{R}_r)(z_2) \\
\qquad \text{[Induction hypothesis]}
\end{array}
$$

$= \mathcal{R}_r(z_1) \cup \mathcal{R}_r(z_2)$
$= (snd([\![x_1]\!]\eta_0))([\![\![x_2 : k]\!]\!], w) \cup$
$(snd([\![x_2]\!]\eta_0))([\![\![x_1 : k]\!]\!], w)$
[Lemma 6.2(a)]
$= (snd([\![x_1]\!]\eta_0))([\![x_2]\!]\eta_0 : [\![\![k]\!]\!], w) \cup$
$(snd([\![x_2]\!]\eta_0))([\![x_1]\!]\eta_0 : [\![\![k]\!]\!], w)$
$= (snd([\![x_1 \parallel x_2]\!]\eta_0))([\![\![k]\!]\!], w)$
$= \mathcal{R}_r(x_1 \parallel x_2, (k, w))$

■

We can present now the main result of the paper.

Theorem 6.4: $\mathcal{O}[\![\rho]\!] = \mathcal{D}[\![\rho]\!], \forall \rho \in \mathcal{L}_{MR}$.

Proof: Let $\rho = (r, x) \in \mathcal{L}_{MR}$.

$\mathcal{O}[\![(r, x)]\!] = \mathcal{O}_r(x, (\{\!\}_{k_0}, [])) =$
[Theorem 2.1, Lemma 6.3]
$= \mathcal{R}_r(x, (\{\!\}_{k_0}, [])) =$
$= (snd([\![x]\!]\eta_0))([\![\![\{\!\}_{k_0}]\!]\!], []) =$
[Lemma 6.2(c)]
$= (snd([\![x]\!]\eta_0))(\{\!\}_{\kappa_0}, []) =$
$= \mathcal{D}[\![(r, x)]\!]$.

■

VII. Concluding remarks

This paper is a continuation of the approach started in [8]. Here we use the mathematical methodology of metric semantics [4] in defining and relating an operational semantics and a denotational semantics for a multiset concurrent language $\mathcal{L}_{MR}$. The semantics of parallel composition in $\mathcal{L}_{MR}$ is based on the concept of maximal parallelism and computations are specified by means of multiset rewriting rules. These features are encountered in new formalisms of natural computing, for instance in membrane computing. The two semantic models were defined by using the continuation semantics for concurrency technique, and the formal relation between the operational semantics and the denotational semantics was established by using Banach's fixed point theorem.

References

[1] O. Agrigoroaiei, G. Ciobanu. Flattening the Transition P Systems with Dissolution. *Lecture Notes in Computer Science*, vol.6501, pp.53–64, 2011.

[2] P. America, J.J.M.M. Rutten, "Solving Reflexive Domain Equations in a Category of Complete Metric Spaces", *J. of Comput. System Sci.*, vol.39, pp.343–375, 1989.

[3] O. Andrei, G. Ciobanu and D. Lucanu, "A Rewriting Logic Framework for Operational Semantics of Membrane Systems," *Theoretical Computer Science*, vol.373, pp.163–181, 2007.

[4] J.W. de Bakker, E.P. de Vink. *Control Flow Semantics*, MIT Press, 1996.

[5] S. Banach. Sur les Operation dans les Ensembles Abstrait et leurs Application aux Equation Integrales, *Fundamenta Mathematicae*, vol.3, pp.133–181, 1922.

[6] G. Ciobanu. Semantics of P Systems, *Handbook of Membrane Computing*, Oxford University Press, pp.413–436, 2009.

[7] G. Ciobanu, Gh. Păun, M.J. Pérez-Jiménez. *Applications of Membrane Computing*, Natural Computing Series, Springer, 2006.

[8] G. Ciobanu, E.N. Todoran. Metric Denotational Semantics for Parallel Rewriting of Multisets. In *Proceedings SYNASC 2011*, pp.276–284, IEEE Computer Press, 2011.

[9] C. Fournet, G. Gonthier. The Join Calculus: a Language for Distributed Mobile Programming, *Lecture Notes in Computer Science* vol.2395, pp.268–332, 2000.

[10] Gh. Păun, *Membrane Computing. An Introduction.* Springer, 2002.

[11] G.D. Plotkin. A Powerdomain Construction, *SIAM Journal of Computing*, vol.5, pp.452–487, 1976.

[12] G.D. Plotkin. A Structural Approach to Operational Semantics, *Journal of Logic and Algebraic Programming*, vol.60-61, pp.17–139, 2004.

[13] C. Stratchey, C. Wadsworth. Continuations: a mathematical semantics for handling full jumps, *Higher Order and Symbolic Computation*, vol.13, pp.135–152, 2000.

[14] E.N. Todoran. Metric Semantics for Synchronous and Asynchronous Communication: a Continuation-Based Approach, *Electronic Notes in Theoretical Computer Science*, vol.28, pp.119–146, 2000.