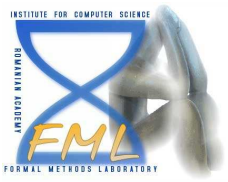

Formal Methods Laboratory

Romanian Academy

FML Technical Report

TR no. FML-14-02

Printed: September 2014
Revised: April 2015

Correct Metric Semantics for a Biologically-Inspired Formalism

G.Ciobanu and E.N.Todoran

Prepared by
Formal Methods Laboratory.

Series of
Technical Reports (Institute of Computer Science Iaşi)

ISSN 1842 - 1490

FML is the Formal Methods Laboratory from the Institute of Computer Science of the Romanian Academy and is located in Iaşi. It is a research institution aimed at developing new formalisms for challenging open problems in computer science, systems biology and other emerging research fields. FML is headed by Dr. Gabriel Ciobanu.

IIT is the Institute of Computer Science of the Romanian Academy and is concerned with basic research in computer science.

Romanian Academy of Science is the supreme forum of science from Romania.
Calea Victoriei 125, Sector 1, 010071, Bucharest, ROMANIA.
Telephone: +40 21 2128640
Telefax: +40 21 2116608
Web: http://www.academiaromana.ro/

Printed in Romania and reproduced directly from the authors original copy.

Blvd. Carol I, nr.8, 700505, Iaşi, ROMANIA (RO)
Institute of Computer Science, Romanian Academy, branch Iaşi.
Telephone/Telefax: +40 232 241708
Web: http://iit.iit.tuiasi.ro/fml
Contact: gabriel@iit.tuiasi.ro

Series of Technical Reports (Institute of Computer Science Iaşi)

ISSN 1842 - 1490

Correct Metric Semantics for a Biologically-Inspired Formalism

by

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

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

ABSTRACT

We investigate a language similar to a process algebra for DNA computing introduced by Cardelli. For such a language we relate two formal semantics. We define a new denotational semantics by using complete metric spaces, in which various semantic functions are defined as fixed points of appropriate higher-order mappings. We compare this denotational semantics with an operational semantics, and establish a formal relationship between them by using an abstraction operator and a fixed point argument. In this way we prove the correctness of the denotational semantics with respect to the operational one.

Contents

1 Introduction

We design and relate formally operational and denotational semantic models for a process algebra language - that we name $\mathcal{L}_{DNA}$ - which incorporates some basic concepts of DNA computing. The language $\mathcal{L}_{DNA}$ is based on the combinatorial "strand algebra" introduced in section 2 of [7].[1] The relevance of $\mathcal{L}_{DNA}$ for DNA computing is explained in [7]. We define an operational semantics $\mathcal{O}$ for $\mathcal{L}_{DNA}$ based on a DNA *reaction relation* introduced in [7]. We also define a new denotational semantics $\mathcal{D}$ for $\mathcal{L}_{DNA}$. We show that the operational semantics is equal to the denotational meaning to which an abstraction operator *abs* is applied: $\mathcal{O} = abs \circ \mathcal{D}$. In this way we establish the *correctness* of the denotational semantics with respect to the operational semantics ($\mathcal{D}$ makes *at least* the same distinction between meanings as $\mathcal{O}$ does).

In this paper $\mathcal{L}_{DNA}$ is a formal language with typical elements P, Q, R, that we call *components*. We describe the syntax of $\mathcal{L}_{DNA}$ in BNF as follows:

$$P ::= \mathbf{0} \mid x \mid g \mid P \parallel P \mid P^*$$

An *inert component* is indicated by $\mathbf{0}$. x is a *signal*. g is a *gate*. $P_1 \parallel P_2$ is the *parallel composition* of P_1 and P_2. Intuitively, an $\mathcal{L}_{DNA}$ component describes a concurrent combination of several signals and gates that can interact as explained below. The construction P^* describes an unbounded (inexhaustible) *population*. Intuitively, a population P^* is an infinite resource, storing an infinite number of copies of the component P. The construction for populations is based on the replication operator from the π calculus [15]. P^* and $P \parallel P^*$ behave the same; formally, we express this as follows: $P^* \equiv P \parallel P^*$, where $\equiv (\subseteq \mathcal{L}_{DNA} \times \mathcal{L}_{DNA})$ is a congruence relation called *mixing* ($\equiv$ was introduced [7]).

We assume given a countable set X of *signals* with typical elements x, y. A gate is an operator from signals to signals $([x_1, \ldots, x_n].[y_1, \ldots, y_m])$, able to join the signals $x_1, \ldots, x_n$ and to fork the signals $y_1, \ldots, y_m$. We consider that the order of signals is irrelevant, hence a gate is a pair of multisets.[2] In $\mathcal{L}_{DNA}$ signals and gates combine in a multiset of elements (a 'chemical soup') that proceed concurrently.[3]

In such a multiset of concurrent $\mathcal{L}_{DNA}$ components, when n signals $x_1, \ldots, x_n$ combine in parallel with a gate $([x_1, \ldots, x_n].[y_1, \ldots, y_m])$ they can react (or interact). The signals $x_1, \ldots, x_n$ and the gate are consumed in the interaction. The signals $y_1, \ldots, y_m$ are released and become available for further interactions. Cell structures cannot grow arbitrarily large. It is reasonable to assume that there exists $k \in N$ such that for any gate $([x_1, \ldots, x_n].[y_1, \ldots, y_m])$ we have $n \leq k$, i.e., the number of signals in the input part of the gate is always lesser than or equal to k. The behavior of such a system is expressed formally in [7] by a *reaction* relation $\longrightarrow (\subseteq \mathcal{L}_{DNA} \times \mathcal{L}_{DNA})$ and the following rule:

$$x_1 \parallel \cdots \parallel x_n \parallel ([x_1, \ldots, x_n].[y_1, \ldots, y_m]) \longrightarrow y_1 \parallel \cdots \parallel y_m$$

[1] In [7] the language is named $\mathcal{P}$.

[2] Intuitively, a *multiset* is a collection where an element may occur more than once, an unordered list; a formal definition is provided in section 2.

[3] The concurrent combination of several processes is a *multiset* because, in general, multiple copies of a process may be executed in parallel.

In this rule the multiset $[x_1, \ldots, x_n]$ behaves as a join pattern. Join patterns and join synchronization were investigated in the context of join calculus [10]. The ability to join and fork signals and to combine signals and gates into populations are specific of the strand process algebras given in [7].

Process algebras describe formally the behavior of multiple processes running concurrently. In general, a process algebra only provides compositionality at the level of the syntax, in terms of operators which can combine simple components into more complex ones.

Denotational semantics (initially known as *mathematical* or Scott-Strachey semantics) is an approach of formalizing the meanings of programming or specification languages. The main principle in the denotational approach is the *semantic compositionality*: the denotation of a composite construction is defined solely based on the denotation of its syntactic components. An elegant theory of domains has been developed, which may use order-theoretic structures [11] or metric spaces [3, 5].

In this paper we use the metric approach to semantics described in the monograph [5]. The main mathematical tool is Banach's theorem, which states that contracting functions defined on complete metric spaces have *unique* fixed points. We define various semantic operators as fixed points of appropriate higher-order mappings. Also, we use semantic domains that are complete metric spaces. The semantic domains are defined as solutions of reflexive domain equations solved by using the method introduced in [3].

We extend the binary reaction relation introduced in [7] to a ternary relation $\longrightarrow \subseteq \mathcal{L}_{DNA} \times G \times \mathcal{L}_{DNA}$, where $(g \in) G$ is the set of gates. Each element of $\longrightarrow$ is a triple (P, g, P') written as $P \stackrel{g}{\longrightarrow} P'$. We label each reaction with the gate that is involved in the reaction. If $g = ([x_1, \ldots, x_n].[y_1, \ldots, y_m])$ then

$$x_1 \parallel \cdots \parallel x_n \parallel g \stackrel{g}{\longrightarrow} y_1 \parallel \cdots \parallel y_m$$

The gates become *observable* items that capture the information expressed by interactions. We obtain a labeled reaction relation that we use in the definition of an operational semantics $\mathcal{O} : \mathcal{L}_{DNA} \to \mathbf{P}$. The semantic universe $\mathbf{P}$ is a linear-time domain. The elements of $\mathbf{P}$ are (non-empty and compact) collections of sequences of observables (gates).

The semantic universe of $\mathcal{D}$ is a branching time domain $\mathbf{P}_D$. An element of $\mathbf{P}_D$ is a tree-like structure whose nodes represent nondeterministic choice points. We need a branching domain to model in a denotational (compositional) manner then synchronization mechanism on which $\mathcal{L}_{DNA}$ is based. The elements of the domain $\mathbf{P}$ (that we use as semantic universe for the operational semantics) are collections of sequences, i.e., linear (or non-branching) structures. The terminology 'linear time' versus 'branching time' is often used in denotational semantics [4],[5].

We investigate the correctness of the denotational model $\mathcal{D}$. A denotational semantics is said to be *correct* with respect to a corresponding operational semantics if whenever the denotational meanings of two language constructs are equal the operational semantics of the two language constructs are also equal in any syntactic context. A formal definition is provided in section 2.4. As it is known, in order to prove the correctness of $\mathcal{D}$ it is sufficient to find an (abstraction) operator $abs : \mathbf{P}_D \to \mathbf{P}$ (which, in general, is not injective) such that $\mathcal{O} = abs \circ \mathcal{D}$, where $\circ$ is the operator for function composition [5]. We define such a function abs that takes $\mathbf{P}_D$ processes, which are tree-like structures, as arguments and

yields collections of sequences as results. In order to prove that $\mathcal{O} = abs \circ \mathcal{D}$, we introduce an alternative operational semantics $\mathcal{O}_A : \mathcal{L}_{DNA} \to \mathbf{P}$ and a (branching time) intermediate operational semantics: $\mathcal{O}_D : \mathcal{L}_{DNA} \to \mathbf{P}_D$. In the definitions of $\mathcal{O}_A$ and $\mathcal{O}_D$ we use a new ternary relation $\Longrightarrow \subseteq \mathcal{L}_{DNA} \times A \times \mathcal{L}_{DNA}$, that we call a *transition* relation. $(\alpha \in)A$ is a strict superset of $G(G \subset A)$. An element of $\Longrightarrow$ is a triple (P, α, P'), written as $P \stackrel{\alpha}{\Longrightarrow} P'$.

The transition relation $\Longrightarrow$ extends the reaction relation $\longrightarrow$ in order to express the interaction capabilities of $\mathcal{L}_{DNA}$ components. In the case of $\Longrightarrow$, a label may be either a DNA gate $(g \in G)$ representing a successful reaction, or an attempt to participate in a DNA interaction. Such a DNA interaction attempt is a multiset, comprising zero or more signals and (eventually) a gate, that could participate in a successful DNA reaction.

We prove that $\mathcal{O} = \mathcal{O}_A$, $\mathcal{O}_A = abs \circ \mathcal{O}_D$ and $\mathcal{O}_D = \mathcal{D}$. It follows that $\mathcal{O} = abs \circ \mathcal{D}$. We conclude that the denotational semantics is *correct* with respect to the operational semantics.

2 Mathematical preliminaries

The notation $(x, y, \ldots \in)X$ introduces the set X with typical elements $x, y, \ldots$ ranging over X. By $\mathcal{P}(X)$ we denote the power set of X, i.e. the set of all subsets of X. The notation $\mathcal{P}_\pi(X)$ denotes the power set of X that have property π. For example, $\mathcal{P}_{fin}(X)$ is the set of all finite subsets of X, and $\mathcal{P}_{nfin}(X)$ is the set of all finite and nonempty subsets of X.

2.1 Multisets

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 function from X to $\mathbf{N}$, or a partial function $m : X \to \mathbf{N}^+$, where $\mathbf{N}^+ = \mathbf{N} \setminus \{0\}$, namely the set of natural numbers without 0. $m(x)$ is called the *multiplicity* of x (the number of occurrences of x in m). More about the mathematics of multisets can be found in [1].

Let X be a countable set. We denote by $[X]$ the set of all finite multisets of elements of type X, i.e., $[X] \stackrel{not.}{=} \bigcup_{A \in \mathcal{P}_{fin}(X)} \{m \mid m \in (A \to \mathbf{N}^+)\}$. Since X is countable, $\mathcal{P}_{fin}(X)$ is also countable. An element $m \in [X]$ is a multiset of elements of type X, namely a function $m : A \to \mathbf{N}^+$, where $A \in \mathcal{P}_{fin}(X)$ is such that $\forall x \in A : m(x) > 0$. 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 union:* $m_1 \cup m_2 \quad (\cup : ([X] \times [X]) \to [X])$

$$\mathsf{dom}(m_1 \cup m_2) = \mathsf{dom}(m_1) \cup \mathsf{dom}(m_2)$$

$$(m_1 \cup m_2)(x) = \begin{cases} max\{m_1(x), m_2(x)\} & \text{if} \quad x \in \mathsf{dom}\,(m_1) \cap \mathsf{dom}\,(m_2) \\ m_1(x) & \text{if} \quad x \in \mathsf{dom}\,(m_1) \setminus \mathsf{dom}\,(m_2) \\ m_2(x) & \text{if} \quad x \in \mathsf{dom}\,(m_2) \setminus \mathsf{dom}\,(m_1) \end{cases}$$

- *Multiset difference:* $m_1 \setminus m_2$ $(\setminus : ([X] \times [X]) \to [X])$

  $\mathsf{dom}\,(m_1 \setminus m_2) =$

  $(\,\mathsf{dom}\,(m_1) \setminus \mathsf{dom}\,(m_2)\,) \cup \{x \mid x \in \mathsf{dom}\,(m_1) \cap \mathsf{dom}\,(m_2)\,, m_1(x) > m_2(x)\}$

  $$(m_1 \setminus m_2)(x) = \begin{cases} m_1(x) & \text{if} \quad x \in \mathsf{dom}\,(m_1) \setminus \mathsf{dom}\,(m_2) \\ m_1(x) - m_2(x) & \text{if} \quad x \in \mathsf{dom}\,(m_1) \cap \mathsf{dom}\,(m_2)\,, m_1(x) > m_2(x) \end{cases}$$

- *Submultiset:* $m_1 \subseteq m_2$ $(\subseteq: ([X] \times [X]) \to \mathit{Bool})$

  $m_1 \subseteq m_2$ iff $(\,\mathsf{dom}\,(m_1) \subseteq \mathsf{dom}\,(m_2)\,) \;\wedge\; (\forall x \in \mathsf{dom}\,(m_1) : m_1(x) \leq m_2(x))$.

- *Cardinal number:* $|m|$ $(\in \mathbf{N})$

  $$|m| = \sum_{x \in \mathsf{dom}\,(m)} m(x)$$

- *Domain restriction:* $m_1 \setminus\setminus m_2$ $(\setminus\setminus : [X] \times [X] \to [X])$

  $\mathsf{dom}\,(m_1 \setminus\setminus m_2) = \mathsf{dom}\,(m_1) \setminus \mathsf{dom}\,(m_2)$

  $(m_1 \setminus\setminus m_2)(x) = m_1(x)$, $\forall x \in \mathsf{dom}\,(m_1) \setminus \mathsf{dom}\,(m_2)$

- *Duplicates removal:* $\{\!|m|\!\}$ $(\{\!|\cdot|\!\} : [X] \to [X])$

  $\mathsf{dom}\,(\{\!|m|\!\}) = \mathsf{dom}\,(m)$

  $\{\!|m|\!\}(x) = 1$, $\forall x \in \mathsf{dom}\,(m)$

We write $m_1 = m_2$ to express that the multisets m_1 and m_2 are equal. $m_1 = m_2$ iff $\mathsf{dom}\,(m_1) = \mathsf{dom}\,(m_2)$ and $\forall x \in \mathsf{dom}\,(m_1) : m_1(x) = m_2(x)$. The notation $m_1 \neq m_2$ means that the multisets m_1 and m_2 are not equal, i.e., $\neg(m_1 = m_2)$. Also, we write $m_1 \subset m_2$ whenever $m_1 \subseteq m_2$ and $m_1 \neq m_2$, i.e. m_1 is a *strict* submultiset of m_2.

We can also represent a multiset $m \in [X]$ by enumerating its elements between parentheses '[' and ']'. Notice that the elements in a multiset are *not* ordered; intuitively, a multiset is 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 $m : \{x_1, x_2\} \to \mathbf{N}^+, m(x_1) = 2, m(x_2) = 1$. Intuitively, the cardinality (or size) of a multiset is the total number of elements that it contains (taking into account the multiplicities). For example, $|[x_1, x_1, x_2]| = 3$, and $|[]| = 0$. Notice that $m = [] \Leftrightarrow |m| = 0$.

The definitions of multiset sum and multiset union should be clear. We consider the following examples: $[x_1, x_1] \cup [x_1, x_2] = [x_1, x_1, x_2]$, and $[x_1, x_1] \uplus [x_1, x_2] = [x_1, x_1, x_1, x_2]$.

$m_1 \setminus m_2$ is a multiset difference operation. Notice that an element $x \in \mathsf{dom}(m_1) \cap \mathsf{dom}(m_2)$ occurs in $(m_1(x) - m_2(x))$ times $m_1 \setminus m_2$ whenever $m_1(x) > m_2(x)$. $m_1 \setminus\!\setminus m_2$ is a more strict operation which simply restricts the domain of $m_1 \setminus\!\setminus m_2$ to $\mathsf{dom}(m_1) \setminus \mathsf{dom}(m_2)$. For example, $[x_1, x_1, x_2, x_2] \setminus [x_1] = [x_1, x_2, x_2]$, and $[x_1, x_1, x_2, x_2] \setminus\!\setminus [x_1] = [x_2, x_2]$. We also give an example for the duplicates removal operation: $\{\!|[x_1, x_1, x_2, x_2]|\!\} = [x_1, x_2]$.

When $(x \in)X$ is a countable set we use the following notation convention. We denote by $\overline{x}$ typical elements of $[X]$, i.e., $\overline{x} \in [X]$ is a multiset of elements of the type X. Also, we denote by $[X]_{\leq k}$ the set of all all finite multisets over X that have cardinality less than or equal to k, for any $k \in \mathbf{N}^+$:

$$[X]_{\leq k} = \{\overline{x} \mid \overline{x} \in [X], |\overline{x}| \leq k\}$$

2.2 Metric spaces

The semantic models given in this paper are designed following the mathematical methodology of metric semantics [5]. 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 [5], for instance.

Example 2.1 *Let $(a, b \in)A$ be a set. We employ the following metric structures, which are frequently used in semantics.*

(a) The discrete metric *$d : A \times A \to [0, 1]$ is defined as $d(x, y) =$ $\mathsf{if}\ x{=}y\ \mathsf{then}\ 0\ \mathsf{else}\ 1$. (A, d) is a complete ultrametric space.*

(b) Let $(x, y \in)A^\infty = A^* \cup A^\omega$, where A^* (A^ω) 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. (A^∞, d) is a complete ultrametric space.

We write a sequence in A^∞ by listing its elements. For example, $a_1 \ldots a_n$ is a finite sequence of length n, and $a^\omega = aaa \ldots$ is an infinite sequence. We use the symbol '$\cdot$' as a concatenation operator over sequences and also as a prefixing operator. In particular, if $a \in A$, $x \in A^\infty$, $a \cdot x$ is the sequence obtained by prefixing a to x. Note that, if $x, y \in A^\infty$ then $d(a \cdot x, a \cdot y) = \frac{1}{2} \cdot d(x, y)$.

We recall that if (X, d_X), (Y, d_Y) are metric spaces, a function $f{:}X \to Y$ is a *contraction* if $\exists c \in \mathbf{R}$, $0 \leq c < 1$, $\forall x_1, x_2 \in X : d_Y(f(x_1), f(x_2)) \leq c \cdot d_X(x_1, x_2)$. 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*. We denote by $X \stackrel{1}{\to} Y$ the set of all nonexpansive functions from X to Y.

Let $f : X \to X$ be a function. When $x \in X$ is such that $f(x) = x$, we call x a *fixed point* of f. When this fixed point is unique, we write $x = fix(f)$. The following theorem is at the core of metric semantics.

Theorem 2.2 (Banach) *Let (X, d_X) be a complete metric space. Each contraction $f : X \to X$ has a* unique *fixed point.*

Definition 2.3 *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) $d_{\frac{1}{2}\cdot X} : X \times X \to [0,1]$, $d_{\frac{1}{2}\cdot X}(x_1, x_2) = \frac{1}{2} \cdot d_X(x_1, x_2)$

(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) $d_{X \times Y} : (X \times Y) \times (X \times Y) \to [0,1]$, $d_{X \times Y}((x_1, y_1), (x_2, y_2)) = max\{d_X(x_1, x_2), d_Y(y_1, y_2)\}$;

(d) $d_{X+Y} : (X + Y) \times (X + Y) \to [0,1]$,

$$d_{X+Y}(u, v) = \text{ if } (u, v \in X) \text{ then } d_X(u, v) \text{ else if } (u, v \in Y) \text{ then } d_Y(u, v) \text{ else } 1$$

(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* $d(u, W) = \inf_{w \in W} d_X(u, w)$ *(by convention* $\sup \emptyset = 0$ *and* $\inf \emptyset = 1$*);* d_H *is the* Hausdorff *metric.*

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

Remark 2.4 *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 2.3. 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 $\frac{1}{2} \cdot X$, $X \to Y$, $X \overset{1}{\to} Y$, $X \times Y$, $X + Y$, $\mathcal{P}_{co}(X)$ and $\mathcal{P}_{nco}(X)$ with their metrics defined above are also complete metric spaces [5].*

2.3 Bisimulation semantics

We assume the reader is familiar with the way an operational semantics can be defined based on a (labelled) *transition system*, embedded in a deductive system in the style of structured operational semantics [16]. To reason about the behavior specified by a transition system we use the notion of a *strong bisimulation* relation [15].

Definition 2.5 *A transition system T is a triple $(X, A, \to)$, where $(x \in) X$ is a set of* configurations*, $(\alpha \in) A$ is a set of* observations*, and $\to$ is a subset of $X \times A \times X$ ($\to \subseteq X \times A \times X$). One usually writes $x \xrightarrow{\alpha} x'$ to express that $(x, \alpha, x') \in \to$.*

Definition 2.6 *Let $T = (X, A, \to)$ be a transition system. A* strong bisimulation *on T is a relation $B \subseteq X \times X$ which satisfies the following property. For each $x_1, x_2 \in X$, if $x_1 B x_2$ then (i) and (ii) hold*

(i) If $x_1 \xrightarrow{\alpha} x_1'$ then there exists x_2' such that $x_2 \xrightarrow{\alpha} x_2'$ and $x_1' B x_2'$.

(ii) If $x_2 \xrightarrow{\alpha} x_2'$ then there exists x_1' such that $x_1 \xrightarrow{\alpha} x_1'$ and $x_1' B x_2'$.

where we write xBx' to express that $(x, x') \in B$.

We say that x_1 and x_2 are strongly bisimilar*, written $x_1 \sim x_2$, if there exists a strong bisimulation B such that $x_1 B x_2$.*

2.4 Semantic correctness

A semantics is a function $\mathcal{M} : \mathcal{L} \to \mathbf{M}$, where $\mathcal{L}$ is a (formal) language and $\mathbf{M}$ is a mathematical domain (of meanings). A language is a collection of syntactic forms. We assume the reader is familiar with the formal description techniques that we employ in this paper, namely denotational semantics and operational semantics.

Let $(P, Q \in)\mathcal{L}$ be a language, C a typical element of a class of syntactic contexts for $\mathcal{L}$, $\mathcal{D} : \mathcal{L} \to \mathbf{D}$ a denotational (compositional) semantics, and $\mathcal{O} : \mathcal{L} \to \mathbf{O}$ an operational semantics. Intuitively, a *syntactic context* is a language construct with 'holes'. For a given context C we denote by $C(P)$ the result of replacing all occurrences of the 'hole' symbol $(\cdot)$ with P in C. The notion of a syntactic context is exemplified in Definition 3.4 for the language $\mathcal{L}_{DNA}$.

The denotational semantics $\mathcal{D}$ is said to be *correct* with respect to the operational semantics $\mathcal{O}$ if whenever the denotations of two language elements are equal then the operational meanings of the two language elements in any syntactic context are also equal. Formally, $\mathcal{D}$ is correct with respect to $\mathcal{O}$ if the following condition holds:

$$\forall P, Q \in \mathcal{L}\,[\,\mathcal{D}[\![P]\!] = \mathcal{D}[\![Q]\!] \Rightarrow \forall C\,[\,\mathcal{O}[\![C(P)]\!] = \mathcal{O}[\![C(Q)]\!]\,]]$$

Remark 2.7 *As it is well-known (see chapter 17 in [5]), in order to prove the correctness of $\mathcal{D}$ it is sufficient to find an operator $abs : \mathbf{D} \to \mathbf{O}$ (which, in general, is not injective) such that: $\mathcal{O} = abs \circ \mathcal{D}$.*

If the denotational semantics $\mathcal{D}$ is correct and it is also *complete* with respect to $\mathcal{O}$ then $\mathcal{D}$ is said to be *fully abstract*. In this paper we do not need to define formally the notion of *semantic completeness*. The full-abstraction problem was first raised by Robin Milner [13, 14].

3 Syntax of $\mathcal{L}_{DNA}$

An informal description of $\mathcal{L}_{DNA}$ was already presented in Section 1. In this section we give the formal definition of the syntax of $\mathcal{L}_{DNA}$. We assume given a countable set $(x, y \in)X$ of signals. We model a gate as a pair of multisets, written as $([x_1, \ldots, x_n].[y_1, \ldots, y_m])$. Cell structures cannot grow arbitrarily large. It is reasonable to assume that there exists $k \in N$ such that for any gate $([x_1, \ldots, x_n].[y_1, \ldots, y_m])$ we have $n \leq k$, i.e., the number of signals in the input part of the gate is always lesser than or equal to k.

Remark 3.1 *In the rest of the paper we assume given a positive natural number $k \in \mathbf{N}^+$ that imposes an upper limit on the size of the input part of any gate. However, note that the k can be chosen arbitrarily large. k is a parameter of the specification of $\mathcal{L}_{DNA}$. The same k will be used in all subsequent sections, both in syntactic and in semantic definitions.*

We define the set of *gates* by $(g \in)G = ([X]_{\leq k} \setminus \{[]\}) \times [X]$.[4] A gate $g(\in G)$ is a pair of multisets of signals $(\overline{x}, \overline{y})$, such that $\overline{x}$ is not empty $(\overline{x} \neq [])$ and the cardinality of $\overline{x}$ is lesser

[4]The notations $[X]$ and $[X]_{\leq k}$ were introduced in Section 2.1.

than or equal to k ($0 < |\overline{x}| \leq k$). In [7] a gate $(\overline{x}, \overline{y})$ is written as $\overline{x}.\overline{y}$. We use a similar (but slightly different) notation. We write a gate $(\overline{x}, \overline{y})$ in the form $(\overline{x}.\overline{y})$. For easier readability we will use the same notation for gates both in the syntactic representation and in semantic computations.

Definition 3.2 *The language* $(P, Q, R \in)\mathcal{L}_{DNA}$ *is defined by:*

$$P ::= \mathbf{0} \mid x \mid g \mid P \parallel P \mid P^*$$

The syntax of $\mathcal{L}_{DNA}$ is given in Definition 3.2. We use the term *component* to refer to any (syntactic) element of $\mathcal{L}_{DNA}$. $\mathcal{L}_{DNA}$ provides elementary components called signals and gates, a construction for *parallel (concurrent) composition*, and a construction that can be used to express unbounded (inexhaustible) *populations* of components [7].

We denote typical elements of $\mathcal{L}_{DNA}$ by P, Q, R. $\mathbf{0}$ is the inert component. $x \in X$ is a *signal*. $g \in G$ is a *gate* of the form $([x_1, \ldots, x_n].[y_1, \ldots, y_m])$, with $0 < n \leq k, 0 \leq m$.[5] $P_1 \parallel P_2$ is the parallel composition of P_1 and P_2. P^* is an unbounded population. The construction P^* can be used to generate an arbitrary number of concurrent copies of P.

A gate $([x_1, \ldots, x_n].[y_1, \ldots, y_m])$ can interact with n concurrent signals $x_1, \ldots, x_n$ as explained informally in Section 1. The multiset $[x_1, \ldots, x_n]$ behaves as a join pattern [10]. In the subsequent sections we will describe such interactions formally.

By using the operators for parallel composition and populations, signals and gates can be combined into a multiset (a 'soup') of concurrent components that can interact. When a gate The gate joins the signals $x_1, \ldots, x_n$, forks the signals $y_1, \ldots, y_m$ and is consumed in the interaction [7].

Remark 3.3 *The language* $\mathcal{L}_{DNA}$ *is similar to the process algebra* $\mathcal{P}$ *introduced in Section 2 of [7].*[6] *The two languages provide the same constructions, but the size of (the input part of) any gate is limited by* k *in* $\mathcal{L}_{DNA}$.

Let $g = ([x_1, \ldots, x_n].\overline{y})$ *be a gate whose input part is a multiset of size (cardinality)* n. *Such a gate can participate in interactions that involve* $n + 1$ *concurrent components:* g *and the signals* $x_1, \ldots, x_n$. *Hence, in* $\mathcal{L}_{DNA}$ *an interaction involves at most* $k + 1$ *concurrent components: a gate of size at most* k *and at most* k *concurrent signals. By choosing* k *sufficiently large any interaction that can be modeled in* $\mathcal{P}$ *can also be modeled in* $\mathcal{L}_{DNA}$. *In this sense,* $\mathcal{L}_{DNA}$ *is not less expressive than* $\mathcal{P}$.

We will present a denotational model for $\mathcal{L}_{DNA}$ which is *correct* with respect to a corresponding operational model. The concept of *semantic correctness* (see Section 2.4) is defined by using the notion of a *syntactic context*, which is specific of the language under investigation.

[5] The reader may wonder why we use the semantic notion of a multiset in the syntax definition of $\mathcal{L}_{DNA}$. It would be easy to make a complete separation between syntax and semantics. For example, we could define the class of gates by $g ::= (x^+.x^*)$, where x^+ is a finite and nonempty sequence of signals of length less than or equal to k, and x^* is a finite, possibly empty, sequence of signals. Instead, we use multisets because the order of signals in a gate is irrelevant. In this way we also avoid some obvious conversions between sequences and multisets.

[6] $\mathcal{P}$ is called *combinatorial strand algebra* in [7].

Definition 3.4 *The class of* syntactic contexts *for $\mathcal{L}_{DNA}$ is given by: $C ::= (\cdot) \mid \mathbf{0} \mid x \mid g \mid C^* \mid C \parallel C$. We use the notation $C(P)$ to represent the $\mathcal{L}_{DNA}$ component that is obtained by substituting P for all occurrences of the 'hole' symbol $(\cdot)$ in the context C. $C(P)$ can be defined inductively: $(\cdot)(P) = P, \mathbf{0}(P) = \mathbf{0}, x(P) = x, g(P) = g, C^*(P) = (C(P))^*$ and $(C_1 \parallel C_2)(P) = C_1(P) \parallel C_2(P)$.*

4 Reactions and operational semantics ($\mathcal{O}$)

The operational semantics of the combinatorial strand algebra $\mathcal{P}$ presented in section 2 of [7] is based on two relations, called *mixing* and *reaction*, respectively. In [7] the both relations are defined as binary relations. In this paper we extend the reaction relation to a ternary relation $\longrightarrow \subseteq \mathcal{L}_{DNA} \times G \times \mathcal{L}_{DNA}$. The elements of the set G (of gates) are used as observables in the definition of the operational semantics. The operational semantics yields (nonempty and compact) collections of sequences of gates. Intuitively, in this semantic model a gate $g = ([x_1, \ldots, x_n].[y_1, \ldots, y_m]) \in G$ is an observation that describes an interaction between g and n concurrent signals $x_1, \ldots, x_n$.

We introduce the mixing relation $\equiv$ for $\mathcal{L}_{DNA}$ in Definition 4.1. Mixing is a congruence relation axiomatizing a well-mixed solution [7, 6]. Next, in Definition 4.2 we introduce the reaction relation $\longrightarrow$ for $\mathcal{L}_{DNA}$. The axioms and rules that describe $\equiv$ and $\longrightarrow$ are taken from [7], and adapted to $\mathcal{L}_{DNA}$. The configurations that we use in the definitions of $\equiv$ and $\longrightarrow$ are $\mathcal{L}_{DNA}$ components. Also, $\longrightarrow$ is extended to a ternary relation.

Definition 4.1 *The relation $\equiv \subseteq \mathcal{L}_{DNA} \times \mathcal{L}_{DNA}$, called* mixing*, is the smallest relation satisfying the following properties:*

(E1) $P \equiv P$ *(equivalence)*

(E2) $P \equiv Q \Rightarrow Q \equiv P$

(E3) $P \equiv Q, Q \equiv R \Rightarrow P \equiv R$

(C1) $P \equiv Q \Rightarrow P \parallel R \equiv Q \parallel R$ *(congruence)*

(C2) $P \equiv Q \Rightarrow P^* \equiv Q^*$

(D1) $P \parallel \mathbf{0} \equiv P$ *(diffusion)*

(D2) $P \parallel Q \equiv Q \parallel P$

(D3) $P \parallel (Q \parallel R) \equiv (P \parallel Q) \parallel R$

(P1) $P^* \equiv P \parallel P^*$ *(population)*

(P2) $\mathbf{0}^* \equiv \mathbf{0}$

(P3) $(P \parallel Q)^* \equiv P^* \parallel Q^*$

(P4) $P^{**} \equiv P^*$

Definition 4.2 *The relation* $\longrightarrow \subseteq \mathcal{L}_{DNA} \times G \times \mathcal{L}_{DNA}$*, called* reaction*, is the smallest relation satisfying the rules given below. The elements of* $\longrightarrow$ *are triples* $(P, g, P') \in \mathcal{L}_{DNA} \times G \times \mathcal{L}_{DNA}$*, that we call* reactions*. We write* $P \xrightarrow{g} P'$ *to express that* $(P, g, P') \in \longrightarrow$*.*

(R1) $x_1 \parallel \cdots \parallel x_n \parallel g \xrightarrow{g} y_1 \parallel \cdots \parallel y_m$ *(gate)*

$$\text{where } g = ([x_1, \ldots, x_n].[y_1, \ldots, y_m]) \in G$$

(R2) $$\frac{P \xrightarrow{g} P'}{P \parallel Q \xrightarrow{g} P' \parallel Q}$$ *(dilution)*

(R3) $$\frac{P \equiv Q, P' \equiv Q', P \xrightarrow{g} P'}{Q \xrightarrow{g} Q'}$$ *(well mixing)*

Rule (R1) was explained informally in Section 1. It describes the interaction between a gate and a corresponding multiset of signals. If $m = 0$ in (R1) then $y_1 \parallel \cdots \parallel y_m$ is replaced with $\mathbf{0}$ [7]. Notice that, according to properties (D2) and (D3), the parallel composition operator $\parallel$ is associative and commutative. Hence, in rule (R1) the components can be written in any order and any order of association can be used. We use a standard notation and terminology for inference rules. Rule (R1) is an axiom. According to rule (R2), whenever $P \xrightarrow{g} P'$ we also have $P \parallel Q \xrightarrow{g} P' \parallel Q$. Rule (R3) allows mixing $\equiv$ to be used at any point in inferring reactions.

In Definition 4.3 we present an operational semantics $\mathcal{O}$ for $\mathcal{L}_{DNA}$. The definition of $\mathcal{O}$ is based on observable interactions, each interaction representing a (successful) DNA reaction.

Definition 4.3 *(Operational semantics* $\mathcal{O}$*) We write* $P \not\longrightarrow$ *to express that* P *has no reactions, i.e., there is no* g *and* P' *such that* $P \xrightarrow{g} P'$*. Let* $(q \in) G^\infty$ *be the complete space of all finite and infinite sequences over the set* G *(of gates). We use the symbol* ϵ *to represent the empty sequence over* G*.*[7] *We define* $\mathcal{O} : \mathcal{L}_{DNA} \to \mathcal{P}(G^\infty)$ *by:*

$$\mathcal{O}[\![P]\!] = \{\epsilon\} \qquad \text{if } P \not\longrightarrow$$

$$\mathcal{O}[\![P]\!] = \{\, g_1 g_2 \cdots g_n \in G^* \mid P_0 = P, P_{i-1} \xrightarrow{g_i} P_i, \forall 1 \le i \le n, P_n \not\longrightarrow \} \cup$$

$$\{\, g_1 g_2 \cdots \in G^\omega \mid P_0 = P, P_{i-1} \xrightarrow{g_i} P_i, \forall i \ge 1\}$$

Remark 4.4 *For any* $P \in \mathcal{L}_{DNA}$*,* $\mathcal{O}[\![P]\!] \neq \emptyset$*, i.e.,* $\mathcal{O}[\![P]\!]$ *is always non-empty. We will prove that, actually,* $\mathcal{O}[\![P]\!]$ *yields a (non-empty and)* compact *collection of sequences of interactions (see Remark 7.14).*

Examples 4.5

(a) $\mathcal{O}[\![x]\!] = \mathcal{O}[\![([x].[])]\!] = \{\epsilon\}$*, and* $\mathcal{O}[\![x \parallel ([x].[])]\!] = \{([x].[])\}, \forall x \in X$*.*

[7] The construction $G^\infty = G^* \cup G^\omega$ was introduced in Section 2.2. The empty sequence ϵ is an element of G^* ($\epsilon \in G^*$).

(b) $x \parallel ([x].[y_1]) \parallel ([x].[y_2]) \xrightarrow{([x].[y_1])} y_1 \parallel ([x].[y_2])$ *and* $y_1 \parallel ([x].[y_2]) \not\longrightarrow$. *Also,* $x \parallel ([x].[y_1]) \parallel ([x].[y_2]) \xrightarrow{([x].[y_2])} y_2 \parallel ([x].[y_1])$ *and* $y_2 \parallel ([x].[y_1]) \not\longrightarrow$. *Therefore we obtain:*
$\mathcal{O}[\![x \parallel ([x].[y_1]) \parallel ([x].[y_2])]\!] = \{([x].[y_1]), ([x].[y_2])\}$.

Lemma 4.6 *Mixing ($\equiv$) is a strong bisimulation with respect to reaction relation ($\longrightarrow$).*

Proof: It suffices to notice that, assuming $P \equiv Q$ and $P \xrightarrow{g} P'$, we can take $Q' = P'$ (which implies $Q' \equiv P'$) and, by using rule (R3), we infer $Q \xrightarrow{g} Q'$.

Remark 4.7 *For any $P \in \mathcal{L}_{DNA}$, the operational semantics $\mathcal{O}[\![P]\!]$ is a collection of sequences of gates (DNA reactions) expressing the behavior of component P. It may not be obvious that $\mathcal{O}[\![P]\!]$ contains sufficient information to express the behavior of P.*

We can prove the following property: (*) for any $\mathcal{L}_{DNA}$ component P and gate g (either there is no P' such that $P \xrightarrow{g} P'$ or) there is a unique component P' (up to mixing bisimulation) such that $P \xrightarrow{g} P'$. Therefore, any sequence of DNA reactions (gates) $g_1 \dots g_i \dots \in \mathcal{O}[\![P]\!]$ can be used to recover a corresponding sequence of $\mathcal{L}_{DNA}$ components $PP_1 \dots P_i \dots$. The sequence $PP_1 \dots P_i \dots$ is uniquely determined up to mixing bisimulation. More precisely, for a given sequence $g_1 \dots g_i \dots \in \mathcal{O}[\![P]\!]$, if we put $P_0 = P'_0 = P$, and if $P_{i-1} \xrightarrow{g_i} P_i, \forall i \geq 1$, and $P'_{i-1} \xrightarrow{g_i} P'_i, \forall i \geq 1$, then $P_i \equiv P'_i, \forall i \geq 0$.

The property (*) justifies our design decision to use $\mathcal{L}_{DNA}$ gates as observable elements in the definition of the operational semantics $\mathcal{O}[\![\cdot]\!]$. However, the property is not needed in the proof of the main semantic correctness result given in Section 8. Hence, we postpone its presentation to Section 9. The property (*) is stated formally in Section 9 as Proposition 9.1.

According to Example 4.5(a), the operational semantics $\mathcal{O}$ is *not* a compositional semantics. In the next section we present a denotational (compositional) semantics $\mathcal{D}$ for $\mathcal{L}_{DNA}$.

5 Denotational semantics ($\mathcal{D}$)

We offer a denotational (compositional) semantics for $\mathcal{L}_{DNA}$. In order to express the multiparty (join) synchronization mechanism on which $\mathcal{L}_{DNA}$ is based in a compositional manner we employ a branching domain $\mathbf{P}_D$. The domain $\mathbf{P}_D$ is specified as the solution of a domain equation. The presentation is organized as follows. In Section 5.1 we describe the semantics of interaction at the level of elementary $\mathcal{L}_{DNA}$ components. Next, in Section 5.2 we introduce the domain $\mathbf{P}_D$ and we define the semantics of parallel composition. In Section 5.3 we give the equations that define $\mathcal{D}$ in as a compositional mapping.

5.1 Semantics of interaction

In order to achieve a compositional semantics for $\mathcal{L}_{DNA}$ we introduce a set I of *interaction attempts*.

Definition 5.1 *The set I of* interaction attempts *is defined by:*

$$I = ([X]_{\leq k} \setminus \{[]\}) \cup \{(g, \overline{x}') \mid g \in G, g = (\overline{x}.\overline{y}), \overline{x}' \in [X], \overline{x}' \subset \overline{x}\}$$

Let $(\alpha \in)A = G \cup I$. *Obviously,* $G \cap I = \emptyset$.

In our semantic models a gate $g = ([x_1, \ldots, x_n].[y_1, \ldots, y_m])$ describes a successful interaction between g and n concurrent signals $x_1, \ldots, x_n$. An interaction attempt contains only partial information related to such an interaction. An interaction attempt could be a nonempty multiset of signals; recall that at most k signals can participate in an interaction in $\mathcal{L}_{DNA}$ (see Remark 3.1), hence such a multiset representing an interaction attempt has cardinality lesser than or equal to k. An interaction attempt could also consist of a gate $g = (\overline{x}.\overline{y})$ and a (possibly empty) multiset $\overline{x}'$, such that $\overline{x}' \subset \overline{x}$ ($\overline{x}'$ is a *strict* submultiset of $\overline{x}$).

Definition 5.2 *Let* $(\alpha \in)A' = A \cup \{\uparrow\}$, *with* $\uparrow$ *a distinct element,* $\uparrow \notin A$. *We define an* interaction function $\gamma : A' \times A' \to A'$ *as follows:*

$$\gamma(\uparrow, \alpha) = \gamma(\alpha, \uparrow) = \gamma(\uparrow, \uparrow) = \uparrow$$

$$\gamma((g_1, \overline{x}'_1), (g_2, \overline{x}'_2)) = \uparrow$$

$$\gamma(g, \alpha) = \gamma(\alpha, g) = \uparrow$$

$$\gamma(\overline{x}_1, \overline{x}_2) = \begin{cases} \overline{x}_1 \uplus \overline{x}_2 & \text{if } |\overline{x}_1 \uplus \overline{x}_2| \leq k \\ \uparrow & \text{otherwise} \end{cases}$$

$$\gamma(\overline{x}_1, (g_2, \overline{x}'_2)) = \begin{cases} g_2 & \text{if } g_2 = (\overline{x}.\overline{y}), \overline{x}_1 \uplus \overline{x}'_2 = \overline{x} \\ (g_2, \overline{x}_1 \uplus \overline{x}'_2) & \text{if } g_2 = (\overline{x}.\overline{y}), \overline{x}_1 \uplus \overline{x}'_2 \subset \overline{x} \\ \uparrow & \text{otherwise} \end{cases}$$

$$\gamma((g_1, \overline{x}'_1), \overline{x}_2) = \begin{cases} g_1 & \text{if } g_1 = (\overline{x}.\overline{y}), \overline{x}_2 \uplus \overline{x}'_1 = \overline{x} \\ (g_1, \overline{x}_2 \uplus \overline{x}'_1) & \text{if } g_1 = (\overline{x}.\overline{y}), \overline{x}_2 \uplus \overline{x}'_1 \subset \overline{x} \\ \uparrow & \text{otherwise} \end{cases}$$

for any $\alpha \in A$, $g, g_1, g_2 \in G$, $\overline{x}_1, \overline{x}_2, (g_1, \overline{x}'_1), (g_2, \overline{x}'_2) \in I$.

One can check that the interaction function γ is associative and commutative. The proof of the following Lemma is (laborious but) straightforward.

Lemma 5.3

(a) $\gamma(\alpha_1, \alpha_2) \in A \Rightarrow \alpha_1 \in I$, $\alpha_2 \in I$.

(b) γ *is commutative and associative.*

5.2 Semantic domain and semantic operators

The denotational semantics of $\mathcal{L}_{DNA}$ is defined based on the branching domain $\mathbf{P}_D$ given in Definition 5.4. $\mathbf{P}_D$ is specified as the solution of a domain equation. The technique for solving such domain equations was already introduced in [3].

Definition 5.4 *We define* $(p \in)\mathbf{P}_D$ *ss the (unique) metric domain satisfying:*

$$\mathbf{P}_D \cong \mathcal{P}_{co}(A \times \frac{1}{2} \cdot \mathbf{P}_D)$$

The set A (introduced in Definition 5.1) is endowed with the discrete metric. The composed metric spaces are built up using the composite metrics of Definition 2.3.

Definition 5.5 *We define* $p_\cdot : [X] \to \mathbf{P}_D$ *inductively:*

$p_{[]} = \emptyset$, *and*

$p_{\overline{x}} = \{(\overline{x} \setminus \overline{y}, p_{\overline{y}}) \mid \overline{y} \subset \overline{x}, \overline{x} \setminus \overline{y} \in I\}$, *if* $\overline{x} \neq []$.[8]

In particular, $p_{[y]} = \{([y], \emptyset)\}$.

We can now define the semantic operator $\|$ for parallel composition on $\mathbf{P}_D$ processes.

Definition 5.6 *Let* $(\phi \in)Op = \mathbf{P}_D \times \mathbf{P}_D \xrightarrow{1} \mathbf{P}_D$. *Let* $\Omega_\|, \Omega_{\lfloor\!\lfloor}, \Omega_| : Op \to Op$ *be given by:*

(a) $\Omega_\|(\phi)(\emptyset, p) = \Omega_\|(\phi)(p, \emptyset) = p$

$\Omega_\|(\phi)(p_1, p_2) = \Omega_{\lfloor\!\lfloor}(\phi)(p_1, p_2) \cup \Omega_{\lfloor\!\lfloor}(\phi)(p_2, p_1) \cup \Omega_|(\phi)(p_1, p_2)$, *if* $p_1 \neq \emptyset, p_2 \neq \emptyset$

(b) $\Omega_{\lfloor\!\lfloor}(\phi)(p_1, p_2) = \{(\alpha, \phi(p_1', p_2)) \mid (\alpha, p_1') \in p_1\}$

(c) $\Omega_|(\phi)(p_1, p_2) = \{(\gamma(\alpha_1', \alpha_2'), \phi(p_1', p_2')) \mid (\alpha_1', p_1') \in p_1, (\alpha_2', p_2') \in p_2, \gamma(\alpha_1', \alpha_2') \in I\} \cup$

$\{((\overline{x}.\overline{y}), \phi(p_{\overline{y}}, \phi(p_1', p_2'))) \mid (\alpha_1', p_1') \in p_1, (\alpha_2', p_2') \in p_2, (\overline{x}.\overline{y}) = \gamma(\alpha_1', \alpha_2') \in G\}$

We define: $\| = fix(\Omega_\|)$, $\lfloor\!\lfloor = \Omega_{\lfloor\!\lfloor}(\|)$ *and* $| = \Omega_|(\|)$.

$\|$ is the operator for *parallel composition* in $\mathcal{L}_{DNA}$, also called *merge*. $|$ and $\lfloor\!\lfloor$ are called *synchronization merge* and *left merge*, respectively. $|$ is a general (multiparty) join synchronization operator, specific of $\mathcal{L}_{DNA}$. The left merge operator is similar to the parallel composition (or merge) operator, but $\lfloor\!\lfloor$ imposes the restriction that the first step must come from p_1.

Remark 5.7

(a) The higher order mapping $\Omega_\|$ *given in Definition 5.6 is a contraction.* $\Omega_\|$ *is a contraction, essentially, because all occurrences of the argument* ϕ *in the right-hand sides of the equations are used to yield values that are stored in the space* $\frac{1}{2} \cdot \mathbf{P}_D$.

[8] Recall that we use the notation $\overline{y} \subset \overline{x}$ to express that $\overline{y}$ is a *strict* submultiset of $\overline{x}$.

(b) It is easy to check that the operators $\cup, \|, \lfloor\!\lfloor, |$ *are well-defined and non-expansive in both their arguments. The reader may consult [5], where many similar operators defined in this way.*

Lemma 5.8 states that the semantic operators $\cup, \|, \lfloor\!\lfloor, |$ satisfy some properties that can be encountered in various concurrency theories. In particular, $\|$ is commutative and associative. As a consequence, notice that in the definition of $|$ any order of association of $p_{\overline{y}}, p_1', p_2'$ can be used.

Lemma 5.8 *For any* $p_1, p_2, p_3 \in \mathbf{P}_D$

(a) $(p_1 \cup p_2) \lfloor\!\lfloor p_3 = (p_1 \lfloor\!\lfloor p_3) \cup (p_2 \lfloor\!\lfloor p_3)$

(b) $(p_1 \cup p_2) \mid p_3 = (p_1 \mid p_3) \cup (p_2 \mid p_3)$

(c) $p_1 \mid p_2 = p_2 \mid p_1$

(d) $p_1 \parallel p_2 = p_2 \parallel p_1$

(e) $p_1 \parallel (p_2 \parallel p_3) = (p_1 \parallel p_2) \parallel p_3$

(f) $p_1 \mid (p_2 \mid p_3) = (p_1 \mid p_2) \mid p_3$

(g) $(p_1 \lfloor\!\lfloor p_2) \lfloor\!\lfloor p_3 = p_1 \lfloor\!\lfloor (p_2 \parallel p_3)$

(h) $p_1 \mid (p_2 \lfloor\!\lfloor p_3) = (p_1 \mid p_2) \lfloor\!\lfloor p_3$

Proof: The properties stated by Lemma 5.8(a)-(d) follow easily by definition.

For the other properties we use an argument of the kind '$\varepsilon \leq \frac{1}{2} \cdot \varepsilon \Rightarrow \varepsilon = 0$', which is standard in metric semantics. Let

$$\sigma_1 = sup_{p_1,p_2,p_3 \in \mathbf{P}_D} d(p_1 \parallel (p_2 \parallel p_3), (p_1 \parallel p_2) \parallel p_3)$$

$$\sigma_2 = sup_{p_1,p_2,p_3 \in \mathbf{P}_D} d(p_1 \mid (p_2 \mid p_3), (p_1 \mid p_2) \mid p_3)$$

$$\sigma_3 = sup_{p_1,p_2,p_3 \in \mathbf{P}_D} d((p_1 \lfloor\!\lfloor p_2) \lfloor\!\lfloor p_3, p_1 \lfloor\!\lfloor (p_2 \parallel p_3)$$

$$\sigma_4 = sup_{p_1,p_2,p_3 \in \mathbf{P}_D} d(p_1 \mid (p_2 \lfloor\!\lfloor p_3), (p_1 \mid p_2) \lfloor\!\lfloor p_3)$$

One can show that $\sigma_1 \leq max\{\sigma_2, \sigma_3, \sigma_4\}$, and $\sigma_i \leq \frac{1}{2} \cdot \sigma_1, i = 2, 3, 4$. Therefore $\sigma_1 = \sigma_2 = \sigma_3 = \sigma_4 = 0$. $\sigma_1 = 0$ implies Lemma 5.8(e): $p_1 \parallel (p_2 \parallel p_3) = (p_1 \parallel p_2) \parallel p_3$. From $\sigma_2 = \sigma_3 = \sigma_4 = 0$ we infer the properties stated by Lemma 5.8(f)-(h).

For σ_1 we compute as follows:

$$d(p_1 \parallel (p_2 \parallel p_3), (p_1 \parallel p_2) \parallel p_3)$$

$$= d(p_1 \lfloor\!\lfloor (p_2 \parallel p_3) \cup (p_2 \lfloor\!\lfloor p_3 \cup p_3 \lfloor\!\lfloor p_2 \cup p_2 \mid p_3) \lfloor\!\lfloor p_1 \cup p_1 \mid (p_2 \lfloor\!\lfloor p_3 \cup p_3 \lfloor\!\lfloor p_2 \cup p_2 \mid p_3),$$

$$(p_1 \lfloor\!\lfloor p_2 \cup p_2 \lfloor\!\lfloor p_1 \cup p_1 \mid p_2) \lfloor\!\lfloor p_3 \cup p_3 \lfloor\!\lfloor (p_1 \parallel p_2) \cup (p_1 \lfloor\!\lfloor p_2 \cup p_2 \lfloor\!\lfloor p_1 \cup p_1 \mid p_2) \mid p_3)$$

$$
\begin{aligned}
&= d(p_1 \between (p_2 \parallel p_3) \cup (p_2 \between p_3) \between p_1 \cup (p_3 \between p_2) \between p_1 \cup (p_2 \mid p_3) \between p_1 \cup \\
&\quad p_1 \mid (p_2 \between p_3) \cup p_1 \mid (p_3 \between p_2) \cup p_1 \mid (p_2 \mid p_3), \\
&\quad (p_1 \between p_2) \between p_3 \cup (p_2 \between p_1) \between p_3 \cup (p_1 \mid p_2) \between p_3 \cup p_3 \between (p_1 \parallel p_2) \cup \\
&\quad p_3 \mid (p_1 \between p_2) \cup p_3 \mid (p_2 \between p_1) \cup (p_1 \mid p_2) \mid p_3) \\
&\leq max\{d(p_1 \between (p_2 \parallel p_3), (p_1 \between p_2) \between p_3),\ d((p_2 \between p_3) \between p_1, (p_2 \between p_1) \between p_3), \\
&\qquad d((p_3 \between p_2) \between p_1, p_3 \between (p_1 \parallel p_2)),\ d((p_2 \mid p_3) \between p_1, p_3 \mid (p_2 \between p_1)), \\
&\qquad d(p_1 \mid (p_2 \between p_3), (p_1 \mid p_2) \between p_3),\ d(p_1 \mid (p_3 \between p_2), p_3 \mid (p_1 \between p_2)), \\
&\qquad d(p_1 \mid (p_2 \mid p_3), (p_1 \mid p_2) \mid p_3)\} \\
&\quad \text{[Triangle inequality, commutativity of } \parallel \text{ and } \mid\text{]} \\
&\leq max\{d(p_1 \between (p_2 \parallel p_3), (p_1 \between p_2) \between p_3),\ d((p_2 \between p_3) \between p_1, p_2 \between (p_3 \parallel p_1)), \\
&\qquad d(p_2 \between (p_1 \parallel p_3), (p_2 \between p_1) \between p_3),\ d((p_3 \between p_2) \between p_1, p_3 \between (p_2 \parallel p_1)), \\
&\qquad d((p_3 \mid p_2) \between p_1, p_3 \mid (p_2 \between p_1)),\ d(p_1 \mid (p_2 \between p_3), (p_1 \mid p_2) \between p_3), \\
&\qquad d(p_1 \mid (p_3 \between p_2), (p_1 \mid p_3) \between p_2),\ d((p_3 \mid p_1) \between p_2, p_3 \mid (p_1 \between p_2)), \\
&\qquad d(p_1 \mid (p_2 \mid p_3), (p_1 \mid p_2) \mid p_3)\}
\end{aligned}
$$

In the sequel it is sufficient to prove that $\sigma_2 \leq \frac{1}{2} \cdot \sigma_1$, $\sigma_3 \leq \frac{1}{2} \cdot \sigma_1$ and $\sigma_4 \leq \frac{1}{2} \cdot \sigma_1$.

For σ_2 we prove that $d(p_1 \mid (p_2 \mid p_3), (p_1 \mid p_2) \mid p_3) \leq \frac{1}{2} \cdot \sigma_1$, for arbitrary $p_1, p_2, p_3 \in \mathbf{P}_D$, which implies $\sigma_2 \leq \frac{1}{2} \cdot \sigma_1$. We compute as follows:

$$
\begin{aligned}
&d(p_1 \mid (p_2 \mid p_3), (p_1 \mid p_2) \mid p_3) \\
&= d(p_1 \mid (\{(\gamma(\alpha_2', \alpha_3'), p_2' \parallel p_3') \mid (\alpha_2', p_2') \in p_2, (\alpha_3', p_3') \in p_3, \gamma(\alpha_2', \alpha_3') \in I\} \cup \\
&\qquad \{((\overline{x}.\overline{y}), p_{\overline{y}} \parallel (p_2' \parallel p_3')) \mid (\alpha_2', p_2') \in p_2, (\alpha_3', p_3') \in p_3, (\overline{x}.\overline{y}) = \gamma(\alpha_2', \alpha_3') \in G\}), \\
&\quad (\{(\gamma(\alpha_1', \alpha_2'), p_1' \parallel p_2') \mid (\alpha_1', p_1') \in p_1, (\alpha_2', p_2') \in p_2, \gamma(\alpha_1', \alpha_2') \in I\} \cup \\
&\qquad \{((\overline{x}.\overline{y}), p_{\overline{y}} \parallel (p_1' \parallel p_2')) \mid (\alpha_1', p_1') \in p_1, (\alpha_2', p_2') \in p_2, (\overline{x}.\overline{y}) = \gamma(\alpha_1', \alpha_2') \in G\}) \mid p_3) \\
&\quad [(\overline{x}.\overline{y}) \in G \Rightarrow \gamma((\overline{x}.\overline{y}), \alpha) = \gamma(\alpha, (\overline{x}.\overline{y})) = \uparrow, \forall \alpha \in A, \text{ Lemma 5.3(a)}] \\
&= d(\{(\gamma(\alpha_1', \gamma(\alpha_2', \alpha_3')), p_1' \parallel (p_2' \parallel p_3')) \mid (\alpha_1', p_1') \in p_1, (\alpha_2', p_2') \in p_2, (\alpha_3', p_3') \in p_3, \\
&\qquad\qquad \gamma(\alpha_1', \gamma(\alpha_2', \alpha_3')) \in I\} \cup \\
&\qquad \{((\overline{x}.\overline{y}), p_{\overline{y}} \parallel (p_1' \parallel (p_2' \parallel p_3'))) \mid (\alpha_1', p_1') \in p_1, (\alpha_2', p_2') \in p_2, (\alpha_3', p_3') \in p_3, \\
&\qquad\qquad (\overline{x}.\overline{y}) = \gamma(\alpha_1', \gamma(\alpha_2', \alpha_3')) \in G\}), \\
&\qquad \{(\gamma(\gamma(\alpha_1', \alpha_2'), \alpha_3'), (p_1' \parallel p_2') \parallel p_3') \mid (\alpha_1', p_1') \in p_1, (\alpha_2', p_2') \in p_2, (\alpha_3', p_3') \in p_3,
\end{aligned}
$$

$$\gamma(\gamma(\alpha'_1,\alpha'_2),\alpha'_3) \in I\} \cup$$
$$\{((\overline{x}.\overline{y}), p_{\overline{y}} \parallel ((p'_1 \parallel p'_2) \parallel p'_3)) \mid (\alpha'_1,p'_1) \in p_1, (\alpha'_2,p'_2) \in p_2, (\alpha'_3,p'_3) \in p_3,$$
$$(\overline{x}.\overline{y}) = \gamma(\gamma(\alpha'_1,\alpha'_2),\alpha'_3) \in G\})$$

[γ is associative (Lemma 5.3(b)), $\cup$ and $\parallel$ are nonexpansive]

$\leq \frac{1}{2} \cdot \sigma_1$

$\parallel$ is nonexpansive, hence in the last step of the proof one can use the fact that

$$d(p_{\overline{y}} \parallel ((p'_1 \parallel p'_2) \parallel p'_3), p_{\overline{y}} \parallel (p'_1 \parallel (p'_2 \parallel p'_3))) \leq d((p'_1 \parallel p'_2) \parallel p'_3, p'_1 \parallel (p'_2 \parallel p'_3))$$

One can also show that $\sigma_3 \leq \frac{1}{2} \cdot \sigma_1$ and $\sigma_4 \leq \frac{1}{2} \cdot \sigma_1$.

5.3 Denotational semantics

The denotational semantics $\mathcal{D}$ of $\mathcal{L}_{DNA}$ maps $\mathcal{L}_{DNA}$ components to $\mathbf{P}_D$ processes: $\mathcal{D} : \mathcal{L}_{DNA} \to \mathbf{P}_D$.

Definition 5.9 *(Denotational semantics $\mathcal{D}$) We define $p^n \in \mathbf{P}_D$ inductively: $p^1 = p$, $p^{n+1} = p \parallel p^n$, for any $p \in \mathbf{P}_D$, $n \in \mathbf{N}^+$. We define $\mathcal{D} : \mathcal{L}_{DNA} \to \mathbf{P}_D$ by:*

$$\begin{aligned}
\mathcal{D}[\![\mathbf{0}]\!] &= \emptyset \\
\mathcal{D}[\![x]\!] &= \{([x],\emptyset)\} \\
\mathcal{D}[\![g]\!] &= \{((g,[]),\emptyset)\} \\
\mathcal{D}[\![P^*]\!] &= fix\,(\lambda p\,.\,(\mathcal{D}[\![P]\!]^k \between p)) \\
\mathcal{D}[\![P_1 \parallel P_2]\!] &= \mathcal{D}[\![P_1]\!] \parallel \mathcal{D}[\![P_2]\!]
\end{aligned}$$

One can check that $d(p \between p_1, p \between p_2) \leq \frac{1}{2} \cdot d(p_1,p_2)$, for any $p, p_1, p_2 \in \mathbf{P}_D$. Hence, $\lambda p\,.\,(\mathcal{D}[\![P]\!]^k \between p)$ is a contraction and has a *unique* fixed point.

It may not be clear why in Definition 5.9 in the equation that describes the behavior of P^* we use the operator $\between$ rather than $\parallel$.[9] Intuitively, in $\mathcal{L}_{DNA}$ a population P^* can participate in an interaction step with at most one copy of a gate and at most k copies of a signal (see Remark 3.1). Hence, in each interaction step it is enough if a population P^* generates k copies of P; the rest of the population may only be involved in subsequent interaction steps. Therefore, we use $\between$ instead of $\parallel$. $\between$ imposes the restriction that the first step must be taken by $\mathcal{D}[\![P]\!]^k$. Moreover, notice that $\lambda p\,.\,(\mathcal{D}[\![P]\!]^k \between p)$ is a contraction. $\lambda p\,.\,(\mathcal{D}[\![P]\!] \parallel p)$ is *not* a contraction, hence we cannot simply put $fix\,(\lambda p\,.\,(\mathcal{D}[\![P]\!] \parallel p))$.

Example 5.10 $\mathcal{D}[\![x \parallel ([x].[])]\!] = \{([x],\{((([x].[]),[]),\emptyset)\}),((([x].[]),[]),\{([x],\emptyset)\}),(([x].[]),\emptyset)\}$; *see also Example 4.5(a).*

Remarks 5.11

(a) Let $\overline{y} = [y_1,\ldots,y_m]$. One can check (by induction on $|\overline{y}|$) that $p_{\overline{y}} = p_{[y_1]} \parallel \cdots \parallel p_{[y_m]}$. Also, $\mathcal{D}[\![x]\!] = p_{[x]}$. Hence $p_{[y_1,\ldots,y_m]} = \mathcal{D}[\![y_1]\!] \parallel \cdots \parallel \mathcal{D}[\![y_m]\!]$.

[9] Recall that, according to rule (P1), $P^* \equiv P \parallel P^*$.

(b) $\mathcal{D}[\![P^*]\!] = \mathcal{D}[\![P]\!]^k \between \mathcal{D}[\![P^*]\!]$. *Indeed:*

$$\begin{aligned}
&\mathcal{D}[\![P^*]\!] = fix(\lambda p\,.\,(\mathcal{D}[\![P]\!]^k \between p)) \\
&= (\lambda p\,.\,(\mathcal{D}[\![P]\!]^k \between p))(fix(\lambda p\,.\,(\mathcal{D}[\![P]\!]^k \between p))) \\
&= \mathcal{D}[\![P]\!]^k \between (fix(\lambda p\,.\,(\mathcal{D}[\![P]\!]^k \between p))) \\
&= \mathcal{D}[\![P]\!]^k \between \mathcal{D}[\![P^*]\!]
\end{aligned}$$

6 Transitions and alternative operational semantics ($\mathcal{O}_A$)

In order to establish the relation between the denotational semantics $\mathcal{D}$ defined in Section 5 and the operational semantics $\mathcal{O}$ presented in Section 4 we introduce an alternative operational semantics $\mathcal{O}_A$ for $\mathcal{L}_{DNA}$. We define $\mathcal{O}_A$ based on a *transition relation* $\Longrightarrow \subseteq \mathcal{L}_{DNA} \times A \times \mathcal{L}_{DNA}$, introduced in Definition 6.1. The label α of a transition $P \stackrel{\alpha}{\Longrightarrow} P'$ is an element of the set $(\alpha \in) A = G \cup I$ introduced in Definition 5.1, but note that only (histories of) gates are recorded by $\mathcal{O}_A$. In section 7 we will show that $\mathcal{O}$ and $\mathcal{O}_A$ behave the same.

Definition 6.1 *(Transition relation) The relation* $\Longrightarrow \subseteq \mathcal{L}_{DNA} \times A \times \mathcal{L}_{DNA}$ *is the smallest relation satisfying the rules given below. The elements of* $\Longrightarrow$ *are triples* (P, α, P') *that we call* transitions. *We write* $P \stackrel{\alpha}{\Longrightarrow} P'$ *to express that* $(P, \alpha, P') \in \Longrightarrow$. *For any* $P \in \mathcal{L}_{DNA}$, $n \in \mathbf{N}^+$, *we define* $P^n \in \mathcal{L}_{DNA}$ *by induction on* n: $P^1 = P$, $P^{n+1} = P \parallel P^n$.

(T1) $x \stackrel{[x]}{\Longrightarrow} \mathbf{0}$

(T2) $g \stackrel{(g,[])}{\Longrightarrow} \mathbf{0}$

(T3) $$\frac{P^k \stackrel{\alpha}{\Longrightarrow} P'}{P^* \stackrel{\alpha}{\Longrightarrow} P' \parallel P^*}$$

(T4) $$\frac{P_1 \stackrel{\alpha_1}{\Longrightarrow} P_1',\ P_2 \stackrel{\alpha_2}{\Longrightarrow} P_2',\ \alpha = \gamma(\alpha_1, \alpha_2) \in I}{P_1 \parallel P_2 \stackrel{\alpha}{\Longrightarrow} P_1' \parallel P_2'}$$

(T5) $$\frac{P_1 \stackrel{\alpha_1}{\Longrightarrow} P_1',\ P_2 \stackrel{\alpha_2}{\Longrightarrow} P_2',\ g = \gamma(\alpha_1, \alpha_2) \in G}{P_1 \parallel P_2 \stackrel{g}{\Longrightarrow} y_1 \parallel \cdots \parallel y_m \parallel P_1' \parallel P_2'} \qquad \textit{where } g = (\overline{x}.[y_1, \ldots, y_m])$$

(T6) $$\frac{P_1 \stackrel{\alpha_1}{\Longrightarrow} P_1'}{P_1 \parallel P_2 \stackrel{\alpha_1}{\Longrightarrow} P_1' \parallel P_2}$$

(T7) $$\frac{P_2 \stackrel{\alpha_2}{\Longrightarrow} P_2'}{P_1 \parallel P_2 \stackrel{\alpha_2}{\Longrightarrow} P_1 \parallel P_2'}$$

Rules (T1) and (T2) describe elementary interaction attempts. Rule (T3) states that whenever P^k can make a transition with label α to some P', then P^* can make a transition with label α to $P' \parallel P^*$. P^k is an abbreviation for k parallel copies of P. A transition represents either a successful interaction or an interaction attempt. According to rules (T4), (T5), (T6) and (T7) when two components are combined in parallel they can either interact or proceed independently. Interaction attempts can be combined into more complex ones by using rule (T4). Rule (T4) is needed when more than two concurrent components interact. Rule (T5) expresses a successful interaction. To illustrate how the rules (T4) and (T5) can be combined to model DNA interactions we consider an example. Assuming that $P_i \stackrel{[x_i]}{\Longrightarrow} P_i'$, for $i = 1, 2, 3$, and $P \stackrel{(g,[])}{\Longrightarrow} P'$, where $g = ([x_1, x_2, x_3].[])$, one can infer the transition $(P_1 \parallel P_2) \parallel (P \parallel P_3) \stackrel{g}{\Longrightarrow} (P_1' \parallel P_2') \parallel (P' \parallel P_3')$ as follows:

$$(T5)\frac{(T4)\dfrac{P_1 \stackrel{[x_1]}{\Longrightarrow} P_1' \quad P_2 \stackrel{[x_2]}{\Longrightarrow} P_2'}{P_1 \parallel P_2 \stackrel{[x_1,x_2]}{\Longrightarrow} P_1' \parallel P_2'} \qquad (T4)\dfrac{P \stackrel{(g,[])}{\Longrightarrow} P' \quad P_3 \stackrel{[x_3]}{\Longrightarrow} P_3'}{P \parallel P_3 \stackrel{(g,[x_3])}{\Longrightarrow} P' \parallel P_3'}}{(P_1 \parallel P_2) \parallel (P \parallel P_3) \stackrel{g}{\Longrightarrow} (P_1' \parallel P_2') \parallel (P' \parallel P_3')}$$

Remark 6.2 *In rule (T5) any order of association, e.g.* $y_1 \parallel (y_2 \parallel \cdots \parallel (y_m \parallel (P_1' \parallel P_2'))\cdots)$*, can be used, since they all behave the same, according to Definition 4.1 and Lemma 7.1. According to Lemma 7.1, two* $\mathcal{L}_{DNA}$ *components* P *and* Q *such that* $P \equiv Q$ *are strongly bisimilar, thus* P *and* Q *behave the same. Moreover, in the metric framework that we employ in this paper, strong bisimilarity coincides with the equality relation; see [12], or [5], chapter 10. A particular consequence of Lemma 7.1 is that parallel composition is commutative and associative. In the sequel, we will ignore the order of association when several* $\mathcal{L}_{DNA}$ *components are combined in parallel, both in rules that describe reactions (*$\longrightarrow$*) and in rules that describe transitions (*$\Longrightarrow$*).*

Definition 6.3 *(Operational semantics* $\mathcal{O}_A$*) We write* $P \not\Longrightarrow$ *to express that* P *has no* g*-transitions, i.e., there is no* $g \in G$ *and* P' *such that* $P \stackrel{g}{\Longrightarrow} P'$*. We define* $\mathcal{O}_A : \mathcal{L}_{DNA} \to \mathcal{P}(G^\infty)$ *by:*

$$\mathcal{O}_A[\![P]\!] = \{\epsilon\} \qquad \text{if } P \not\Longrightarrow$$

$$\mathcal{O}_A[\![P]\!] = \{\, g_1 g_2 \cdots g_n \in G^* \mid P_0 = P, P_{i-1} \stackrel{g_i}{\Longrightarrow} P_i, \forall 1 \le i \le n, P_n \not\Longrightarrow \} \cup$$

$$\{\, g_1 g_2 \cdots \in G^\omega \mid P_0 = P, P_{i-1} \stackrel{g_i}{\Longrightarrow} P_i, \forall i \ge 1\}$$

Notice that in Definition 6.3 we only consider transitions $P \stackrel{g}{\Longrightarrow} P'$ labelled with gates $g \in G$.

In some inductive proofs it is convenient to use the syntactic complexity measure introduced in Definition 6.4. $c(P)$ is clearly well-defined: by structural induction on P one can check that $c(P) \in \mathbf{N}$, for any $P \in \mathcal{L}_{DNA}$.

Definition 6.4 *We define* $c : \mathcal{L}_{DNA} \to \mathbf{N}$ *by:*

$$\begin{aligned} c(\mathbf{0}) &= c(x) = c(g) = 1 \\ c(P^*) &= k + c(P) \\ c(P_1 \parallel P_2) &= 1 + max\{c(P_1), c(P_2)\} \end{aligned}$$

Although P^k is *not* syntactically simpler than P^* (when $k > 1$), we can prove the following:

Lemma 6.5 *$c(P^*) > c(P^k)$, for any $k \in \mathbf{N}^+, P \in \mathcal{L}_{DNA}$.*

Proof: An easy induction on k.

Lemma 6.6 *The set $Succ(P) = \{(\alpha, P') \mid P \stackrel{\alpha}{\Longrightarrow} P'\}$ is finite, for any $P \in \mathcal{L}_{DNA}$.*[10]

Proof: In this proof we denote by $|\cdot|$ the cardinality (number of elements) of the set '$\cdot$'. This notation should not be confused with the notation for the cardinality of a multiset introduced in Section 2.1 (in the case of a multiset the cardinality also takes into account the multiplicity of each element).

We proceed by induction on $c(P)$. We shall prove that $|Succ(P)| \in \mathbf{N}$, for any $P \in \mathcal{L}_{DNA}$, where $\mathbf{N}$ is the set of natural numbers. We consider two cases. If $P = \mathbf{0}$ then $c(P) = 1$. $\mathbf{0}$ has no transitions. Hence $|Succ(\mathbf{0})| = |\emptyset| = 0 \in \mathbf{N}$. We also consider the case when $P = Q^*$, for some $Q \in \mathcal{L}_{DNA}$. By Lemma 6.5, $c(P) = c(Q^*) > c(Q^k)$. By the induction hypothesis we can assume that $|Succ(Q^k)| \in \mathbf{N}$. Hence $|Succ(Q^*)| = |\{(\alpha, R) \mid Q^* \stackrel{\alpha}{\Longrightarrow} R\}| = |\{(\alpha, Q' \parallel Q^*) \mid Q^k \stackrel{\alpha}{\Longrightarrow} Q'\}| = |\{(\alpha, Q') \mid Q^k \stackrel{\alpha}{\Longrightarrow} Q'\}| = |Succ(Q^k)| \in \mathbf{N}$.

Definition 6.7 *Let $(S \in) Sem_A = \mathcal{L}_{DNA} \to \mathbf{P}$, where $\mathbf{P} = \mathcal{P}_{nco}(G^\infty)$. Let $\Psi_A : Sem_A \to Sem_A$ be given by:*

$$\Psi_A(S)(P) = \{\epsilon\} \qquad \text{if } P \not\Longrightarrow$$

$$\Psi_A(S)(P) = \cup\{g \cdot S(P') \mid P \stackrel{g}{\Longrightarrow} P'\} \quad \text{otherwise}$$

Remark 6.8 *According to Lemma 6.6, the transition system induced by the transition relation $\Longrightarrow$ is* finitely branching. *It is a standard result in metric semantics that a finitely branching transition system gives rise to a* compact *operational semantics [5]. Ψ_A is a contraction (hence it has a unique fixed point) in particular as a consequence of the "$g \cdot \ldots$"-step in its definition. The following properties can be proved essentially as shown in [5] (see chapter 2 and Appendix B):*

(a) $\mathcal{O}_A[\![P]\!] \in \mathbf{P}, \forall P \in \mathcal{L}_{DNA}$ (i.e., $\mathcal{O}_A[\![P]\!]$ is non-empty and compact for any $P \in \mathcal{L}_{DNA}$), and

(b) $\mathcal{O}_A = fix(\Psi_A)$.

7 Equivalence of $\mathcal{O}$ and $\mathcal{O}_A$

Both $\mathcal{O}$ and $\mathcal{O}_A$ yield collections of sequences of gates, but their definitions are different. In this section we prove that $\mathcal{O}$ and $\mathcal{O}_A$ behave the same.

Lemma 7.1

[10] In [5] $Succ(P)$ is called the *succesor set* of P.

(a) Mixing ($\equiv$) is a strong bisimulation with respect to transition relation ($\Longrightarrow$).

(b) If $P \equiv Q$ then $P \sim Q$.

Proof: Let $P \equiv Q$. We prove that whenever $P \stackrel{\alpha}{\Longrightarrow} P'$ there exists Q' such that $Q \stackrel{\alpha}{\Longrightarrow} Q'$ and $P' \equiv Q'$. Also, we prove that whenever $Q \stackrel{\alpha}{\Longrightarrow} Q'$ there exists P' such that $P \stackrel{\alpha}{\Longrightarrow} P'$ and $P' \equiv Q'$. We proceed by induction on depth of inference of $P \equiv Q$. We consider several cases for the last step of inference. To emphasize that we proceed by induction on the depth of inference we represent each rule given in Definition 4.1 in the form $premise_1, \cdots, premise_n \Rightarrow conclusion$ by using the notation

$$\frac{premise_1, \cdots, premise_n}{conclusion}$$

- Assume that in the last step of inference of $P \equiv Q$ is used rule (E1)

$$\frac{P = P}{P \equiv P}$$

  In this case $P = Q$ and the result is obvious.

- Assume that in the last step of inference of $P \equiv Q$ is used rule (E2)

$$\frac{Q \equiv P}{P \equiv Q}$$

  By the induction hypothesis the property holds for $Q \equiv P$, and the desired result follows easily.

- Assume that in the last step of inference of $P \equiv Q$ is used rule (E3)

$$\frac{P \equiv R, R \equiv Q}{P \equiv Q}$$

  Assume that $P \stackrel{\alpha}{\Longrightarrow} P'$. By the induction hypothesis there exists R' such that $R \stackrel{\alpha}{\Longrightarrow} R'$ and $P' \equiv R'$. Also, from $R \stackrel{\alpha}{\Longrightarrow} R'$, by the induction hypothesis we infer that there exists Q' such that $Q \stackrel{\alpha}{\Longrightarrow} Q'$ and $R' \equiv Q'$. By using rule (E3), from $P' \equiv R'$ and $R' \equiv Q'$ we obtain $P' \equiv Q'$, as required. By a similar argument, from $Q \stackrel{\alpha}{\Longrightarrow} Q'$ we also infer that there exists P' such that $P \stackrel{\alpha}{\Longrightarrow} P'$ and $P' \equiv Q'$.

- Assume that in the last step of inference of $P \equiv Q$ is used rule (C1). In this case $P = P_1 \parallel R$ and $Q = Q_1 \parallel R$, for some $P_1, Q_1, R \in \mathcal{L}_{DNA}$ and

$$\frac{P_1 \equiv Q_1}{P_1 \parallel R \equiv Q_1 \parallel R}$$

  In this case $P_1 \equiv Q_1$ and there are four subcases, depending on the rule used to infer $P \stackrel{\alpha}{\Longrightarrow} P'$ or $Q \stackrel{\alpha}{\Longrightarrow} Q'$: (T4), (T5),(T6) or (T7). We only consider the subcases when one of the rules (T6),(T7) or (T5) is used.

– Assume that $P \stackrel{\alpha}{\Longrightarrow} P'$ is inferred by using rule (T6). In this case there exists P_1' such that $P_1 \stackrel{\alpha}{\Longrightarrow} P_1'$:

$$\frac{P_1 \stackrel{\alpha}{\Longrightarrow} P_1'}{P_1 \parallel R \stackrel{\alpha}{\Longrightarrow} P_1' \parallel R}$$

and $P' = P_1' \parallel R$. By the induction hypothesis there exists Q_1' such that $Q_1 \stackrel{\alpha}{\Longrightarrow} Q_1'$ and $P_1' \equiv Q_1'$. By using rule (T6) again:

$$\frac{Q_1 \stackrel{\alpha}{\Longrightarrow} Q_1'}{Q_1 \parallel R \stackrel{\alpha}{\Longrightarrow} Q_1' \parallel R}$$

If we put $Q' = Q_1' \parallel R$ we see that $Q \stackrel{\alpha}{\Longrightarrow} Q'$. Also, as $P_1' \equiv Q_1'$, by using rule (C1) $P_1' \parallel R \equiv Q_1' \parallel R$, i.e., $P' \equiv Q'$, as required.

– Assume that $Q \stackrel{\alpha}{\Longrightarrow} Q'$ is inferred by using rule (T6). By a symmetric argument we can show that there exists P' such that $P \stackrel{\alpha}{\Longrightarrow} P'$ and $P' \equiv Q'$.

– Assume that $Q \stackrel{\alpha}{\Longrightarrow} Q'$ is inferred by using rule (T7), which means that there exists R' such that $R \stackrel{\alpha}{\Longrightarrow} R'$ and

$$\frac{R \stackrel{\alpha}{\Longrightarrow} R'}{Q_1 \parallel R \stackrel{\alpha}{\Longrightarrow} Q_1 \parallel R'}$$

In this case $Q' = Q_1 \parallel R'$ (and $Q \stackrel{\alpha}{\Longrightarrow} Q'$). By using rule (T7) again we obtain

$$\frac{R \stackrel{\alpha}{\Longrightarrow} R'}{P_1 \parallel R \stackrel{\alpha}{\Longrightarrow} P_1 \parallel R'}$$

If we put $P' = P_1 \parallel R'$ we obtain $P \stackrel{\alpha}{\Longrightarrow} P'$ and (as $P_1 \equiv Q_1$) $P' = P_1 \parallel R' \equiv Q_1 \parallel R' = Q'$, as required.

– Assume that $P \stackrel{\alpha}{\Longrightarrow} P'$ is inferred by using rule (T7). By a symmetric argument we can show that there exists Q' such that $Q \stackrel{\alpha}{\Longrightarrow} Q'$ and $P' \equiv Q'$.

– Assume that $P \stackrel{\alpha}{\Longrightarrow} P'$ is inferred by using rule (T5). In this case there exists P_1', R' such that

$$\frac{P_1 \stackrel{\alpha_1}{\Longrightarrow} P_1', R \stackrel{\alpha_2}{\Longrightarrow} R', g = \gamma(\alpha_1, \alpha_2) \in G}{P_1 \parallel R \stackrel{g}{\Longrightarrow} y_1 \parallel \cdots \parallel y_m \parallel P_1' \parallel R'}$$

where $g = (\overline{x}.[y_1, \ldots, y_m])$. In this case $P = P_1 \parallel R$ and $P' = y_1 \parallel \cdots \parallel y_m \parallel P_1' \parallel R'$. As $P_1 \equiv Q_1$ and $P_1 \stackrel{\alpha_1}{\Longrightarrow} P_1'$, by the induction hypothesis, there exists Q_1' such that $Q_1 \stackrel{\alpha_1}{\Longrightarrow} Q_1'$, and $P_1' \equiv Q_1'$. We can apply rule (T5) again and we obtain:

$$\frac{Q_1 \stackrel{\alpha_1}{\Longrightarrow} Q_1', R \stackrel{\alpha_2}{\Longrightarrow} R', g = \gamma(\alpha_1, \alpha_2) \in G}{Q_1 \parallel R \stackrel{g}{\Longrightarrow} y_1 \parallel \cdots \parallel y_m \parallel Q_1' \parallel R'}$$

with $g = (\overline{x}.[y_1, \ldots, y_m])$. Let $Q' = y_1 \parallel \cdots \parallel y_m \parallel Q_1' \parallel R'$. As $P_1' \equiv Q_1'$, we infer that ($Q \stackrel{g}{\Longrightarrow} Q'$ and) $P' \equiv Q'$, as required.

– Assume that $Q \stackrel{g}{\Longrightarrow} Q'$ is inferred by using rule (T5). By a symmetric argument we can show that there exists P' such that $P \stackrel{g}{\Longrightarrow} P'$ and $P' \equiv Q'$.

- Assume that in the last step of inference of $P \equiv Q$ is used rule (D2). In this case $P = R_1 \parallel R_2$, $Q = R_2 \parallel R_1$, for some $R_1, R_2 \in \mathcal{L}_{DNA}$. We assume that $P \stackrel{\alpha}{\Longrightarrow} P'$. $P \stackrel{\alpha}{\Longrightarrow} P'$ can be inferred by using either of the rules (T4), (T5), (T6) or (T7). We only consider the case when $P \stackrel{\alpha}{\Longrightarrow} P'$ is inferred by using rule (T4). In this subcase there exist $R_1', R_2' \in \mathcal{L}_{DNA}$, $\alpha_1, \alpha_2 \in I$ (see Lemma 5.3(a)) such that $R_1 \stackrel{\alpha_1}{\Longrightarrow} R_1'$, $R_2 \stackrel{\alpha_2}{\Longrightarrow} R_2'$ and $\alpha = \gamma(\alpha_1, \alpha_2) \in I$. $P \stackrel{\alpha}{\Longrightarrow} P'$ is inferred by using rule (T4) as follows:

$$\frac{R_1 \stackrel{\alpha_1}{\Longrightarrow} R_1', R_2 \stackrel{\alpha_2}{\Longrightarrow} R_2', \alpha = \gamma(\alpha_1, \alpha_2) \in I}{R_1 \parallel R_2 \stackrel{\alpha}{\Longrightarrow} R_1' \parallel R_2'}$$

In this subcase $P' = R_1' \parallel R_2'$. By using rule (T4) again, taking into account that γ is commutative (Lemma 5.3(b)), we also have:

$$\frac{R_2 \stackrel{\alpha_2}{\Longrightarrow} R_2', R_1 \stackrel{\alpha_1}{\Longrightarrow} R_1', \alpha = \gamma(\alpha_2, \alpha_1) = \gamma(\alpha_1, \alpha_2) \in I}{R_2 \parallel R_1 \stackrel{\alpha}{\Longrightarrow} R_2' \parallel R_1'}$$

Let $Q' = R_2' \parallel R_1'$. Then we have $Q \stackrel{\alpha}{\Longrightarrow} Q'$, and $P' = R_1' \parallel R_2' \equiv R_2' \parallel R_1' = Q'$, as required.

The proof that $Q \stackrel{\alpha}{\Longrightarrow} Q'$ implies that there exists P' such that $P \stackrel{\alpha}{\Longrightarrow} P'$ and $P' \equiv Q'$ is symmetric.

- Assume that in the last step of inference of $P \equiv Q$ is used rule (P1). In this case $P = R^*$ and $Q = R \parallel R^*$, for some $R \in \mathcal{L}_{DNA}$. First we consider the case when we know that $P \stackrel{\alpha}{\Longrightarrow} P'$. $P \stackrel{\alpha}{\Longrightarrow} P'$ can only be inferred by using rule (T3) as follows. There exists $R' \in \mathcal{L}_{DNA}$ such that

$$\frac{R^k \stackrel{\alpha}{\Longrightarrow} R'}{R^* \stackrel{\alpha}{\Longrightarrow} R' \parallel R^*}$$

Hence ($P = R^*$ and) $P' = R' \parallel R^*$. By using rules (T7) and (T3) we also obtain:

$$\frac{\dfrac{R^k \stackrel{\alpha}{\Longrightarrow} R'}{R^* \stackrel{\alpha}{\Longrightarrow} R' \parallel R^*}}{R \parallel R^* \stackrel{\alpha}{\Longrightarrow} R \parallel (R' \parallel R^*)}$$

Let $Q' = R \parallel (R' \parallel R^*)$. Hence $Q \stackrel{\alpha}{\Longrightarrow} Q'$, and $P' = R' \parallel R^* \equiv R \parallel (R' \parallel R^*) = Q'$, as required.

Next, assume that $Q \stackrel{\alpha}{\Longrightarrow} Q'$, i.e., $R \parallel R^* \stackrel{\alpha}{\Longrightarrow} Q'$ ($Q = R \parallel R^*$). By Lemma 7.4(a), there exists $R' \in \mathcal{L}_{DNA}$ such that $R^{k+1} \stackrel{\alpha}{\Longrightarrow} R'$ and $Q' \equiv R' \parallel R^*$. By Lemma 7.4(e), there exists $R'' \in \mathcal{L}_{DNA}$ such that $R^k \stackrel{\alpha}{\Longrightarrow} R''$ and $R' \equiv R'' \parallel R$. Hence, by using (T3)

$$\frac{R^k \stackrel{\alpha}{\Longrightarrow} R''}{R^* \stackrel{\alpha}{\Longrightarrow} R'' \parallel R^*}$$

Let $P' = R'' \parallel R^*$ (and recall that $P = R^*$). We obtained $P \stackrel{\alpha}{\Longrightarrow} P'$, and $P' = R'' \parallel R^* \equiv (R'' \parallel R) \parallel R^* = R' \parallel R^* = Q'$, as required.

- Assume that in the last step of inference of $P \equiv Q$ is used rule (P3). In this case $P = (P_1 \parallel P_2)^*$ and $Q = P_1^* \parallel P_2^*$, for some $P_1, P_2 \in \mathcal{L}_{DNA}$. In this case the desired property follows by using Lemma 7.5 and Lemma 7.6. We only handle the case when we know that $P \stackrel{\alpha}{\Longrightarrow} P'$ (i.e., when $(P_1 \parallel P_2)^* \stackrel{\alpha}{\Longrightarrow} P'$, for some P') and we show that there exists Q' such that $Q \stackrel{\alpha}{\Longrightarrow} Q'$ (i.e., $P_1^* \parallel P_2^* \stackrel{\alpha}{\Longrightarrow} Q'$) and $P' \equiv Q'$. The transition $P \stackrel{\alpha}{\Longrightarrow} P'$ can only be inferred by using (T3) as follows. For some $R \in \mathcal{L}_{DNA}$ we have:

$$(T3)\frac{(P_1 \parallel P_2)^k \stackrel{\alpha}{\Longrightarrow} R}{(P_1 \parallel P_2)^* \stackrel{\alpha}{\Longrightarrow} R \parallel (P_1 \parallel P_2)^*}$$

  and in this case $P' = R \parallel (P_1 \parallel P_2)^*$. By Lemma 7.5(b), as $(P_1 \parallel P_2)^k \stackrel{\alpha}{\Longrightarrow} R$, there exists R' such that $P_1^k \parallel P_2^k \stackrel{\alpha}{\Longrightarrow} R'$ and $R' \equiv R$. By Lemma 7.6(a), there exists Q' such that $P_1^* \parallel P_2^* \stackrel{\alpha}{\Longrightarrow} Q'$ and $Q' \equiv R' \parallel P_1^* \parallel P_2^*$. Hence, $P' = R \parallel (P_1 \parallel P_2)^* \equiv R' \parallel (P_1 \parallel P_2)^* \equiv R' \parallel P_1^* \parallel P_2^* = Q'$, as required.

- Assume that in the last step of inference of $P \equiv Q$ is used rule (P4). In this case $P = R^{**}$, $Q = R^*$, $P = R^{**} \equiv R^* = Q$.

  First, we prove that whenever $Q \stackrel{\alpha}{\Longrightarrow} Q'$ (i.e., whenever $R^* \stackrel{\alpha}{\Longrightarrow} Q'$) there exists P' such that $P \stackrel{\alpha}{\Longrightarrow} P'$ and $P' \equiv Q'$. $R^* \stackrel{\alpha}{\Longrightarrow} Q'$ implies that there exists R' such that $Q' = R' \parallel R^*$ and:

$$\frac{R^k \stackrel{\alpha}{\Longrightarrow} R'}{R^* \stackrel{\alpha}{\Longrightarrow} R' \parallel R^*}$$

  By using rules (T3) and (T6) we obtain:

$$\frac{\dfrac{\dfrac{R^k \stackrel{\alpha}{\Longrightarrow} R'}{R^* \stackrel{\alpha}{\Longrightarrow} R' \parallel R^*}}{(R^*)^k \stackrel{\alpha}{\Longrightarrow} (R' \parallel R^*) \parallel (R^*)^{k-1}}}{R^{**} \stackrel{\alpha}{\Longrightarrow} ((R' \parallel R^*) \parallel (R^*)^{k-1}) \parallel R^{**}}$$

  Hence $P \stackrel{\alpha}{\Longrightarrow} P'$, where $P' = ((R' \parallel R^*) \parallel (R^*)^{k-1}) \parallel R^{**} \equiv R' \parallel (R^*)^k \parallel R^{**} \equiv R' \parallel R^* = Q'$. We see that whenever $Q \stackrel{\alpha}{\Longrightarrow} Q'$, there exists P' such that $P \stackrel{\alpha}{\Longrightarrow} P'$, and $P' \equiv Q'$, as required.

  Next, we prove that, whenever $P \stackrel{\alpha}{\Longrightarrow} P'$ there exists Q' such that $Q \stackrel{\alpha}{\Longrightarrow} Q'$ and $P' \equiv Q'$. We have to prove that if $R^{**} \stackrel{\alpha}{\Longrightarrow} P'$ then there exists Q' such that $R^* \stackrel{\alpha}{\Longrightarrow} Q'$ and $P' \equiv Q'$. The transition $R^{**} \stackrel{\alpha}{\Longrightarrow} P'$ can only be inferred by an instance of rule (T3). More precisely, there exists $P'' \in \mathcal{L}_{DNA}$ such that

$$\frac{(R^*)^k \stackrel{\alpha}{\Longrightarrow} P''}{R^{**} \stackrel{\alpha}{\Longrightarrow} P'' \parallel R^{**}}$$

  Let $P' = P'' \parallel R^{**}$. By Lemma 7.7, there exists $R_0 \in \mathcal{L}_{DNA}$ such that $P'' \equiv R_0 \parallel R^*$ and $R^k \stackrel{\alpha}{\Longrightarrow} R_0$. Hence, by using rule (T3) again we obtain:

$$\frac{R^k \stackrel{\alpha}{\Longrightarrow} R_0}{R^* \stackrel{\alpha}{\Longrightarrow} R_0 \parallel R^*}$$

Let $Q' = R_0 \parallel R^*$. Notice that $P' = P'' \parallel R^{**} \equiv R_0 \parallel R^* \parallel R^{**} \equiv R_0 \parallel R^* = Q'$. From $P = R^{**} \equiv R^* = Q$ and $P \stackrel{\alpha}{\Longrightarrow} P'$ we inferred that there exists Q' such that $Q \stackrel{\alpha}{\Longrightarrow} Q'$ and $P' \equiv Q'$, as desired.

Definition 7.2 *We define* $card : A \to \mathbf{N}^+$ *by:* $card(\overline{x}) = |\overline{x}|$, $card(g, \overline{x}') = 1 + |\overline{x}|$ *and* $card(\overline{x}.\overline{y}) = 1 + |\overline{x}|$.

Remark 7.3 *Let* $\alpha_1, \alpha_2 \in A$.

(a) If $\gamma(\alpha_1, \alpha_2) \in A$ *then* $card(\gamma(\alpha_1, \alpha_2)) = card(\alpha_1) + card(\alpha_2)$.

(b) $card(\alpha) \leq k + 1$, *for any* $\alpha \in A$.

(c) If $card(\alpha) = k + 1$ *then* $\alpha \in G$.

Part (a) follows immediately by the definition of γ. *For parts (b) and (c) see Remark 3.1 and Remark 3.3.*

Lemma 7.4

(a) If $P \parallel P^* \stackrel{\alpha}{\Longrightarrow} Q$ *then there exists* $R \in \mathcal{L}_{DNA}$ *such that* $P^{k+1} \stackrel{\alpha}{\Longrightarrow} R$ *and* $Q \equiv R \parallel P^*$.[11]

(b) For any $1 \leq r \leq k$, $\forall P \in \mathcal{L}_{DNA}, \alpha \in A$, *if* $P^{r+1} \stackrel{\alpha}{\Longrightarrow} R$ *and* $card(\alpha) \leq r$ *then there exists* $R' \in \mathcal{L}_{DNA}$ *such that* $P^r \stackrel{\alpha}{\Longrightarrow} R'$ *and* $R \equiv R' \parallel P$.

(c) If $P \stackrel{[x]}{\Longrightarrow} R_1$, $P \stackrel{(g,[])}{\Longrightarrow} R_2$, *for some* $g \in G$, *and* $\alpha = \gamma([x], (g, []))$, *then there exists* $R' \in \mathcal{L}_{DNA}$ *such that* $P \stackrel{\alpha}{\Longrightarrow} R'$ *and*

- *if* $\gamma([x], (g, [])) \in I$ *then* $R_1 \parallel R_2 \equiv R' \parallel P$,
- *if* $\gamma([x], (g, [])) = (\overline{x}.[y_1, \ldots, y_m]) \in G$ *then* $y_1 \parallel \cdots \parallel y_m \parallel R_1 \parallel R_2 \equiv R' \parallel P$.

(d) For any $1 \leq r \leq k$, *if* $P^{r+1} \stackrel{\alpha}{\Longrightarrow} R$, $\alpha = g \in G$, *or* $\alpha = (g, \overline{x}') \in I$ *(for some* $g = (\overline{x}.\overline{y}) \in G$, $\overline{x}' \in [X]$, $\overline{x}' \subset \overline{x}$*) and* $card(\alpha) \leq r + 1$, *then there exists* $R' \in \mathcal{L}_{DNA}$ *such that* $P^r \stackrel{\alpha}{\Longrightarrow} R'$ *and* $R \equiv R' \parallel P$.

(e) If $P^{k+1} \stackrel{\alpha}{\Longrightarrow} R$ *then there exists* $R' \in \mathcal{L}_{DNA}$ *such that* $P^k \stackrel{\alpha}{\Longrightarrow} R'$, *and* $R \equiv R' \parallel P$.

Proof: First, recall that k is the maximum size of the input part of any gate in $\mathcal{L}_{DNA}$; see Remark 3.1. Lemma 7.4(a) follows by considering the four possible ways in which $P \parallel P^* \stackrel{\alpha}{\Longrightarrow} Q$ (by using the rule (T6), (T7), (T4) or (T5)). The proof of Lemma 7.4(b) can proceed by induction on r. The proof of Lemma 7.4(c) can proceed by induction on $c(P)$ (see Definition 6.4). The proof of Lemma 7.4(d) can proceed by induction on r. When $card(\alpha) \leq k$, Lemma 7.4(e) is an easy consequence of Lemma 7.4(b). When $card(\alpha) = k+1$, Lemma 7.4(e) follows by using Lemma 7.4(d) and Remark 7.3(c). Note that, according to Remark 7.3(b), $card(\alpha) \leq k + 1$, for any $\alpha \in A$.

Lemma 7.5 *For any* $n \in \mathbf{N}^+$

[11] We recall that $P^1 = P$, $P^{n+1} = P \parallel P^n$, for any $n > 0$.

(a) $(P \parallel Q)^n \equiv P^n \parallel Q^n$

(b) If $(P \parallel Q)^n \stackrel{\alpha}{\Longrightarrow} R$ *then there exists* R' *such that* $P^n \parallel Q^n \stackrel{\alpha}{\Longrightarrow} R'$ *and* $R \equiv R'$.

(c) If $P^n \parallel Q^n \stackrel{\alpha}{\Longrightarrow} R'$ *then there exists* R *such that* $(P \parallel Q)^n \stackrel{\alpha}{\Longrightarrow} R$ *and* $R \equiv R'$.

Proof: In each case the proof can proceed by induction on n. The case when $n = 1$ is obvious. For $n > 1$ one must consider the various ways in which the transitions involved are inferred. We only consider one sub-case for Lemma 7.5(b). Assume that $(P \parallel Q)^n \stackrel{\alpha}{\Longrightarrow} R$ is derived by using (T4) and (T6) as follows:

$$(T6)\frac{(T4)\dfrac{P \stackrel{\alpha_1}{\Longrightarrow} P_1 \qquad Q \stackrel{\alpha_2}{\Longrightarrow} Q_2 \qquad \gamma(\alpha_1,\alpha_2) \in I}{P \parallel Q \stackrel{\gamma(\alpha_1,\alpha_2)}{\Longrightarrow} P_1 \parallel Q_2}}{(P \parallel Q)^n \stackrel{\gamma(\alpha_1,\alpha_2)}{\Longrightarrow} P_1 \parallel Q_2 \parallel (P \parallel Q)^{n-1}}$$

This transition is matched by (recall that $P^n = P \parallel P^{n-1}$, when $n > 1$)

$$(T4)\frac{(T6)\dfrac{P \stackrel{\alpha_1}{\Longrightarrow} P_1}{P^n \stackrel{\alpha_1}{\Longrightarrow} P_1 \parallel P^{n-1}} \qquad (T6)\dfrac{Q \stackrel{\alpha_2}{\Longrightarrow} Q_2}{Q^n \stackrel{\alpha_2}{\Longrightarrow} Q_2 \parallel Q^{n-1}} \qquad \gamma(\alpha_1,\alpha_2) \in I}{P^n \parallel Q^n \stackrel{\gamma(\alpha_1,\alpha_2)}{\Longrightarrow} P_1 \parallel P^{n-1} \parallel Q_2 \parallel Q^{n-1}}$$

By using Lemma 7.6(a), we see that $R = P_1 \parallel Q_2 \parallel (P \parallel Q)^{n-1} \equiv P_1 \parallel P^{n-1} \parallel Q_2 \parallel Q^{n-1} = R'$, as required.

Lemma 7.6 *For any* $P, Q \in \mathcal{L}_{DNA}$

(a) If $P^k \parallel Q^k \stackrel{\alpha}{\Longrightarrow} R$ *then there exists* R' *such that* $P^* \parallel Q^* \stackrel{\alpha}{\Longrightarrow} R'$ *and* $R' \equiv R \parallel P^* \parallel Q^*$.

(b) If $P^* \parallel Q^* \stackrel{\alpha}{\Longrightarrow} R'$ *then there exists* R *such that* $R' \equiv R \parallel P^* \parallel Q^*$ *and* $P^k \parallel Q^k \stackrel{\alpha}{\Longrightarrow} R'$.

where k *is the maximum size of the input part of any gate in* $\mathcal{L}_{DNA}$*; see Remark 3.1.*

Proof: One must consider the possible ways in which the transition in the assumption is derived. We only consider one sub-case for Lemma 7.6(b). Assume that $P^* \parallel Q^* \stackrel{\alpha}{\Longrightarrow} R'$ is derived as follows:

$$(T6)\frac{(T3)\dfrac{P^k \stackrel{\alpha}{\Longrightarrow} P'}{P^* \stackrel{\alpha}{\Longrightarrow} P' \parallel P^*}}{P^* \parallel Q^* \stackrel{\alpha}{\Longrightarrow} P' \parallel P^* \parallel Q^*}$$

We also have:

$$(T6)\frac{P^k \stackrel{\alpha}{\Longrightarrow} P'}{P^k \parallel Q^k \stackrel{\alpha}{\Longrightarrow} P' \parallel Q^k}$$

and we see that $R' = P' \parallel P^* \parallel Q^* \equiv P' \parallel Q^k \parallel P^* \parallel Q^* = R \parallel P^* \parallel Q^*$, as required.

Lemma 7.7 *For any $1 \leq r \leq k$, if $(R^*)^r \stackrel{\alpha}{\Longrightarrow} P''$ then there exists R_0 such that $P'' \equiv R_0 \parallel R^*$ and $R^k \stackrel{\alpha}{\Longrightarrow} R_0$.*

Proof: By induction on r.

Notation 7.8 *Let $P, P', P'' \in \mathcal{L}_{DNA}$, $g \in G$. We write $P \stackrel{g}{\Longrightarrow}\equiv P'$ to express that, for some P'', $P \stackrel{g}{\Longrightarrow} P''$ and $P'' \equiv P'$. $P \stackrel{g}{\Longrightarrow}\equiv P'$ is an instance of relational composition.*

In Lemma 7.11 we prove that $\longrightarrow$ and $\Longrightarrow$ coincide, up to mixing $\equiv$. Lemma 7.9 and Lemma 7.10 are needed in the proof of Lemma 7.11. We omit the proofs of Lemma 7.9 and Lemma 7.10, which can be approached by induction on depth of inference of $P \stackrel{\overline{x}}{\Longrightarrow} P'$ and $P \stackrel{(g,\overline{x}')}{\Longrightarrow} P'$, respectively.

Lemma 7.9 *Let $\overline{x} = [x_1, \ldots, x_n] \in I$ $(0 < n \leq k)$. If $P \stackrel{\overline{x}}{\Longrightarrow} P'$ then $P \equiv x_1 \parallel \cdots \parallel x_n \parallel P'$.*

Lemma 7.10 *Let $(g, \overline{x}') \in I, \overline{x}' = [x_1, \ldots, x_n]$. If $P \stackrel{(g,\overline{x}')}{\Longrightarrow} P'$ then $P \equiv x_1 \parallel \cdots \parallel x_n \parallel g \parallel P'$. If $n = 0$ then $x_1 \parallel \cdots \parallel x_n$ is replaced with $\mathbf{0}$.*

Lemma 7.11 *(Reaction agrees with g-transition) $P \stackrel{g}{\longrightarrow} P'$ if and only if $P \stackrel{g}{\Longrightarrow}\equiv P'$.*

Proof: ($\Rightarrow$) By induction on depth of inference of $P \stackrel{g}{\longrightarrow} P'$. We consider the possible cases for the last step of inference.

- If in the last step of inference is used rule (R1) then $P = x_1 \parallel \cdots \parallel x_n \parallel g$ and

$$P \stackrel{g}{\longrightarrow} y_1 \parallel \cdots \parallel y_m$$

where $g = ([x_1, \ldots, x_n].[y_1, \ldots, y_m])$. We have to prove that $P \stackrel{g}{\Longrightarrow}\equiv y_1 \parallel \cdots \parallel y_m$, i.e., we have to show that there exists P'' such that $P \stackrel{g}{\Longrightarrow} P''$ and $P'' \equiv y_1 \parallel \cdots \parallel y_m$.

Rule (R1) is actually ambiguous, because the order of association is not specified for the term $P \equiv x_1 \parallel \cdots \parallel x_n \parallel g$. In fact, the order of association is not important, because rule (R1) can be combined with rule (R3) which allows mixing $\equiv$ to be used at any point in inferring reactions. In particular, this implies that the operator for parallel composition $\parallel$ is associative and commutative.

Anyway, there are two possibilities:

  - either $P = (x_1 \parallel \cdots \parallel x_n) \parallel g$, for some order of association of $x_1 \parallel \cdots \parallel x_n$,
  - or $P = x_1 \parallel (x_2 \parallel \cdots \parallel x_n \parallel g)$, for some order of association of $x_2 \parallel \cdots \parallel x_n \parallel g$.

It is easy to prove the following facts:

(1) $x_1 \parallel \cdots \parallel x_n \stackrel{\overline{x}}{\Longrightarrow} R$, for some $R \in \mathcal{L}_{DNA}$, satisfying $R \equiv \mathbf{0}$, where $\overline{x} = [x_1, \ldots, x_n]$, $0 < n \leq k$,

(2) $x_1 \parallel \cdots \parallel x_p \parallel g \stackrel{(g,\overline{x}')}{\Longrightarrow} R$, for some $R \in \mathcal{L}_{DNA}$, satisfying $R \equiv \mathbf{0}$, where $\overline{x}' = [x_1, \ldots, x_p]$, $g = ([x_1, \ldots, x_n].[y_1, \ldots, y_m])$, $0 \leq p < n$.

For any order of association, (1) can be proved by induction on n, and (2) can be proved by induction on p.[12]

If $P = (x_1 \parallel \cdots \parallel x_n) \parallel g$, for some order of association of $x_1 \parallel \cdots \parallel x_n$, and $g = ([x_1, \ldots, x_n].[y_1, \ldots, y_m])$, then, according to (1), $x_1 \parallel \cdots \parallel x_n \stackrel{\overline{x}}{\Longrightarrow} R$, for some R, with $R \equiv \mathbf{0}$, where $\overline{x} = [x_1, \ldots, x_n]$, . Hence

$$(T5)\ \frac{x_1 \parallel \cdots \parallel x_n \stackrel{\overline{x}}{\Longrightarrow} R \qquad g \stackrel{(g,[])}{\Longrightarrow} \mathbf{0}}{(x_1 \parallel \cdots \parallel x_n) \parallel g \stackrel{g}{\Longrightarrow} y_1 \parallel \cdots \parallel y_m \parallel R \parallel \mathbf{0}}$$

and $y_1 \parallel \cdots \parallel y_m \parallel R \parallel \mathbf{0} \equiv y_1 \parallel \cdots \parallel y_m$, as required.

If $P = x_1 \parallel (x_2 \parallel \cdots \parallel x_n \parallel g)$, for some order of association of $x_2 \parallel \cdots \parallel x_n \parallel g$, and $g = ([x_1, \ldots, x_n].[y_1, \ldots, y_m])$, then, according to (2), $x_2 \parallel \cdots \parallel x_n \parallel g \stackrel{(g,\overline{x}')}{\Longrightarrow} R$, for some R, with $R \equiv \mathbf{0}$, where $\overline{x}' = [x_2, \ldots, x_n]$. Hence

$$(T5)\ \frac{x_1 \stackrel{[x_1]}{\Longrightarrow} \mathbf{0} \qquad x_2 \parallel \cdots \parallel x_n \parallel g \stackrel{(g,\overline{x}')}{\Longrightarrow} R}{x_1 \parallel (x_2 \parallel \cdots \parallel x_n \parallel g) \stackrel{g}{\Longrightarrow} y_1 \parallel \cdots \parallel y_m \parallel \mathbf{0} \parallel R}$$

and $y_1 \parallel \cdots \parallel y_m \parallel \mathbf{0} \parallel R \equiv y_1 \parallel \cdots \parallel y_m$, as required.

- If in the last step of inference of $P \stackrel{g}{\longrightarrow} P'$ is used rule (R2) then $P = Q \parallel R$, for some $Q, R \in \mathcal{L}_{DNA}$, and we have

$$\frac{Q \stackrel{g}{\longrightarrow} Q'}{Q \parallel R \stackrel{g}{\longrightarrow} Q' \parallel R}$$

Let $P' = Q' \parallel R$. As $Q \stackrel{g}{\longrightarrow} Q'$ is obtained by a shorter inference, by the induction hypothesis we have $Q \stackrel{g}{\Longrightarrow} \equiv Q'$, i.e., there exists Q'' such that $Q \stackrel{g}{\Longrightarrow} Q''$ and $Q'' \equiv Q'$. By using rule (T6)

$$\frac{Q \stackrel{g}{\Longrightarrow} Q''}{Q \parallel R \stackrel{g}{\longrightarrow} Q'' \parallel R}$$

Let $P'' = Q'' \parallel R$. Hence, $P \stackrel{g}{\Longrightarrow} P''$, $P'' = Q'' \parallel R \equiv Q' \parallel R = P'$, i.e. $P \stackrel{g}{\Longrightarrow} \equiv P'$, as required.

[12] If $x_1 \parallel \cdots \parallel x_p \parallel g$ has the form $(x_1 \parallel \cdots \parallel x_p) \parallel g$ then in the proof of (2) one also uses (1).

- If in the last step of inference of $P \xrightarrow{g} P'$ is used rule (R3) then there exist $Q, Q' \in \mathcal{L}_{DNA}$ such that:

$$\frac{Q \equiv P, Q \xrightarrow{g} Q', Q' \equiv P'}{P \xrightarrow{g} P'}$$

where $Q \xrightarrow{g} Q'$ is inferred by a shorter inference. Hence by the induction hypothesis we have $Q \xrightarrow{g}\!\!\equiv Q'$, i.e., there exists $Q'' \in \mathcal{L}_{DNA}$ such that $Q \xRightarrow{g} Q''$ and $Q'' \equiv Q'$. By Lemma 7.1 mixing is a strong bisimulation with respect to transition relation ($\Longrightarrow$). Hence, as $P \equiv Q$, there exists P'' such that $P \xRightarrow{g} P''$ and $P'' \equiv Q''$. As $P'' \equiv Q'' \equiv Q' \equiv P'$, it follows that $P \xrightarrow{g}\!\!\equiv P'$, as required.

For ($\Leftarrow$) it is enough to prove that $P \xRightarrow{g} P'$ implies $P \xrightarrow{g}\!\!\equiv P'$. We use the notation $P \xrightarrow{g}\!\!\equiv P'$ to express that, for some P'', $P \xrightarrow{g} P''$ and $P'' \equiv P'$. We proceed by induction on the depth of inference of $P \xRightarrow{g} P'$. We consider the possible cases for the last step of inference.

- Notice that $P \xRightarrow{g} P'$ cannot be inferred by using rule (T1), which states that $x \xRightarrow{[x]} \mathbf{0}$, because $[x] \notin G$. Similarly, $P \xRightarrow{g} P$ cannot be inferred by using (T2).

- We handle the case when $P \xRightarrow{g} P'$ is inferred by using rule (T5). In this case $P = P_1 \parallel P_2$ and

$$\frac{P_1 \xRightarrow{\alpha_1} P_1',\ P_2 \xRightarrow{\alpha_2} P_2',\ g = \gamma(\alpha_1, \alpha_2) \in G}{P_1 \parallel P_2 \xRightarrow{g} y_1 \parallel \cdots \parallel y_m \parallel P_1' \parallel P_2'}$$

with $g = (\overline{x}.[y_1, \ldots, y_m])$. Let $P' = y_1 \parallel \cdots \parallel y_m \parallel P_1' \parallel P_2'$. Without loss of generality, we consider that $\alpha_1 = \overline{x}_1 \in I$, $\alpha_2 = ((\overline{x}.\overline{y}), \overline{x}_2') \in I$, $\overline{x} = \overline{x}_1 \uplus \overline{x}_2'$ and $\overline{y} = [y_1, \ldots, y_m]$. Let $\overline{x}_1 = [x_{11}, \ldots, x_{1p}]$, and $\overline{x}_2' = [x_{21}, \ldots, x_{2r}]$, $\overline{x} = [x_{11}, \ldots, x_{1p}, x_{21}, \ldots, x_{2r}]$.

By Lemma 7.9 $P_1 \equiv x_{11} \parallel \cdots \parallel x_{1p} \parallel P_1'$. By Lemma 7.10 $P_2 \equiv x_{21} \parallel \cdots \parallel x_{2r} \parallel (\overline{x}.\overline{y}) \parallel P_2'$. Let $Q = x_{11} \parallel \cdots \parallel x_{1p} \parallel x_{21} \parallel \cdots \parallel x_{2r} \parallel (\overline{x}.\overline{y}) \parallel P_1' \parallel P_2'$. As $P = P_1 \parallel P_2$, $P \equiv Q$. By using rules (R1) and (R2), $Q \xrightarrow{g} P'$, where $g = (\overline{x}.\overline{y})$, $P' = y_1 \parallel \cdots y_m \parallel P_1' \parallel P_2'$. As $P \equiv Q$, by using rule (R3) (called "well mixing" in [7]), we infer $P \xrightarrow{g} P'$, hence, as $P' \equiv P'$, we obtain $P \xrightarrow{g}\!\!\equiv P'$.

- We also consider the case when $P \xRightarrow{g} P'$ is inferred by using rule (T3), i.e. $P = Q^*$, for some $Q \in \mathcal{L}_{DNA}$. In this case we have:

$$\frac{Q^k \xRightarrow{g} Q'}{Q^* \xRightarrow{g} Q' \parallel Q^*}$$

and $P' = Q' \parallel Q^*$. $Q^k \xRightarrow{g} Q'$ is inferred by a shorter inference, therefore, by induction hypothesis, $Q^k \xrightarrow{g}\!\!\equiv Q'$; this means that there exists Q'' such that $Q^k \xrightarrow{g} Q''$ and $Q' \equiv Q''$. By rule (R2)

$$\frac{Q^k \xrightarrow{g} Q''}{Q^k \parallel Q^* \xrightarrow{g} Q'' \parallel Q^*}$$

We have: $P = Q^* \equiv Q^k \parallel Q^*$, $Q^k \parallel Q^* \xrightarrow{g} Q'' \parallel Q^*$, and $Q'' \parallel Q^* \equiv Q' \parallel Q^* = P'$. Hence, by using rules (R3) (called "well mixing" in [7]) we obtain the desired result $P \xrightarrow{g} \equiv P'$.

Lemma 7.12

(a) If $P \xrightarrow{g} P'$ *and* $Q \equiv P$, *then there exists* Q' *such that* $Q \stackrel{g}{\Longrightarrow} Q'$ *and* $Q' \equiv P'$.

(b) $P \not\rightarrow \Leftrightarrow P \not\Rightarrow$.

(c) If $P \not\rightarrow$ *and* $Q \equiv P$ *then* $Q \not\Rightarrow$.

Proof: We only prove Lemma 7.12(a). By Lemma 7.11 $P \stackrel{g}{\Longrightarrow} \equiv P'$, i.e., for some P'', $P \stackrel{g}{\Longrightarrow} P''$ and $P'' \equiv P'$. By Lemma 7.1, there exists Q' such that $Q \stackrel{g}{\Longrightarrow} Q'$, $Q' \equiv P''$. By rule (E3) $Q' \equiv P'$.

Lemma 7.13 $\mathcal{O}[\![P]\!] = \mathcal{O}_A[\![P]\!], \forall P \in \mathcal{L}_{DNA}$.

Proof: Let $P \in \mathcal{L}_{DNA}$, $q \in G^\infty$. We prove that $q \in \mathcal{O}[\![P]\!]$ implies $q \in \mathcal{O}_A[\![P]\!]$, for any $q \in \mathcal{O}[\![P]\!]$. Also, we prove that $q \in \mathcal{O}_A[\![P]\!]$ implies $q \in \mathcal{O}[\![P]\!]$, for any $q \in \mathcal{O}_A[\![P]\!]$. We conclude that $\mathcal{O}[\![P]\!] = \mathcal{O}_A[\![P]\!], \forall P \in \mathcal{L}_{DNA}$. By Lemma 7.12(b), P has no reactions iff P has no g-transitions. Hence, $\mathcal{O}[\![P]\!] = \{\epsilon\} \Leftrightarrow \mathcal{O}_A[\![P]\!] = \{\epsilon\}$.

If P has reactions and transitions we proceed as follows. First, let $q = g_1 g_2 \cdots \in \mathcal{O}[\![P]\!]$. We want to show that $q \in \mathcal{O}_A[\![P]\!]$. As $q \in \mathcal{O}[\![P]\!]$, there exists a (finite or infinite) sequence of reactions $P = P_0 \xrightarrow{g_1} P_1, P_1 \xrightarrow{g_2} P_2 \cdots$ (in case the sequence is finite then $P_n \not\rightarrow$, for some $n \in \mathbf{N}$).

If we put $P'_0 = P_0 = P$ (which implies $P'_0 \equiv P_0$), by using Lemma 7.12(a), we obtain a corresponding (finite or infinite) sequence of transitions $P = P'_0 \stackrel{g_1}{\Longrightarrow} P'_1, P'_1 \stackrel{g_2}{\Longrightarrow} P'_2 \cdots$ (with $P'_i \equiv P_i$), which yields the same sequence of observables (gates) in $\mathcal{O}_A[\![P]\!]$. By Lemma 7.12(c), if $P_n \not\rightarrow$, for some $n \in \mathbf{N}$, then we also have $P'_n \not\Rightarrow$ (the sequence of reactions is finite iff the sequence of transitions is finite and they have the same length). Whence, $q \in \mathcal{O}_A[\![P]\!]$, as required.

Next, let $q = g_1 g_2 \cdots \in \mathcal{O}_A[\![P]\!]$. We want to show that $q \in \mathcal{O}[\![P]\!]$. $P \stackrel{g}{\Longrightarrow} P'$ implies $P \stackrel{g}{\Longrightarrow} \equiv P'$, which implies $P \xrightarrow{g} P'$, by Lemma 7.11. Therefore, whenever we have a sequence of transitions $P = P_0 \stackrel{g_1}{\Longrightarrow} P_1, P_1 \stackrel{g_2}{\Longrightarrow} P_2 \cdots$ with $q = g_1 g_2 \cdots \in \mathcal{O}_A[\![P]\!]$, we can construct a corresponding sequence of reactions $P = P_0 \xrightarrow{g_1} P_1, P_1 \xrightarrow{g_2} P_2 \cdots$, hence $q \in \mathcal{O}[\![P]\!]$.

Remark 7.14 *According to Remark 6.8(a) and Lemma 7.13,* $\mathcal{O}[\![P]\!]$ *is* non-empty and compact, *for any* $P \in \mathcal{L}_{DNA}$.

8 Semantic correctness

For any $P \in \mathcal{L}_{DNA}$, $\mathcal{D}[\![P]\!]$ yields an element of $\mathbf{P}_D$, which is a tree-like structure. $\mathcal{D}[\![P]\!]$ contains more information than the linear outcome of $\mathcal{O}[\![P]\!]$, which is an element of $\mathbf{P}$. Hence we cannot expect that $\mathcal{D}[\![P]\!] = \mathcal{O}[\![P]\!]$ on $\mathcal{L}_{DNA}$.

Our aim in this section is to establish the formal relation between $\mathcal{D}$ and $\mathcal{O}$. We introduce yet another intermediate operational semantics $\mathcal{O}_D$, which delivers $\mathbf{P}_D$ processes as a result, i.e., elements with a branching structure. Next, we introduce an abstraction operator $abs : \mathbf{P}_D \to \mathbf{P}$. We will prove that $\mathcal{O} = abs \circ \mathcal{D}$. We conclude that $\mathcal{D}$ is *correct* with respect to $\mathcal{O}$.

Definition 8.1 *Let* $(S \in)Sem_D = \mathcal{L}_{DNA} \to \mathbf{P}_D$. *Let* $\Psi_D : Sem_D \to Sem_D$ *be given by:*

$$\Psi_D(S)(P) = \{(\alpha, S(P')) \mid P \stackrel{\alpha}{\Longrightarrow} P'\}$$

We put $\mathcal{O}_D = fix(\Psi_D)$.

Well-definedness of Ψ_D and $\mathcal{O}_D$ rely on the property that the transition system induced by $\Longrightarrow$ is finitely branching (by Lemma 6.6). As usual, Ψ_D is contracting, essentially, because $S(P')$ is stored in the space $\frac{1}{2} \cdot \mathbf{P}_D$.

Definition 8.2 *(Abstraction operator) Let* $(\phi \in)Op = \mathbf{P}_D \stackrel{1}{\to} \mathbf{P}$. *We define* $\Omega_{abs} : Op \to Op$ *by:*

$$\Omega_{abs}(\phi)(p) = \{\epsilon\} \qquad \text{if } p \subseteq I \times \tfrac{1}{2} \cdot \mathbf{P}_D$$

$$\Omega_{abs}(\phi)(p) = \{g \cdot \phi(p') \mid (g, p') \in p\} \qquad \text{otherwise}$$

We put $abs = fix(\Omega_{abs})$.

Lemma 8.3 $\mathcal{O}_A = abs \circ \mathcal{O}_D$, *on* $\mathcal{L}_{DNA}$.

We omit the proof of this Lemma. A very similar Lemma is proved in [5], chapter 11.

Lemma 8.4 $\mathcal{D}[\![P]\!] = \mathcal{O}_D[\![P]\!]$, $\forall P \in \mathcal{L}_{DNA}$.

Proof: According to Theorem 2.2 (Banach) it is enough to prove that $\mathcal{D} = fix(\Psi_D)$. We show that $\Psi_D(\mathcal{D})(P) = \mathcal{D}[\![P]\!]$, for any $P \in \mathcal{L}_{DNA}$. We proceed by induction on $c(P)$. We handle two cases.

- Case $P = \mathbf{0}$. In this case $c(P) = c(\mathbf{0}) = 1$. $\Psi_D(\mathcal{D})(\mathbf{0}) = \emptyset = \mathcal{D}[\![\mathbf{0}]\!]$.
- Case $P = x$. In this case $c(P) = c(x) = 1$. x has just one transition ($x \stackrel{[x]}{\Longrightarrow} \mathbf{0}$), and $\mathcal{D}(\mathbf{0}) = \emptyset$.

  $\Psi_D(\mathcal{D})(x) = \{([x], \emptyset)\} = \mathcal{D}(x)$
- Case $P = g$. In this case $c(P) = c(g) = 1$. g has just one transition ($g \stackrel{(g,[])}{\Longrightarrow} \mathbf{0}$), and $\mathcal{D}(\mathbf{0}) = \emptyset$.

  $\Psi_D(\mathcal{D})(g) = \{((g, []), \emptyset)\} = \mathcal{D}(x)$
- Case $P = P_1 \parallel P_2$. In this case $c(P_1 \parallel P_2) = 1 + max\{c(P_1), c(P_2)\}$, hence, $c(P_1) < c(P)$, $c(P_2) < c(P)$. We compute as follows:

$$
\begin{aligned}
&\Psi(\mathcal{D})(P_1 \parallel P_2) = \\
&\quad \{(\alpha_1, \mathcal{D}[\![P_1' \parallel P_2]\!]) \mid P_1 \stackrel{\alpha_1}{\Longrightarrow} P_1'\} \cup \\
&\quad \{(\alpha_2, \mathcal{D}[\![P_1 \parallel P_2']\!]) \mid P_2 \stackrel{\alpha_2}{\Longrightarrow} P_2'\} \cup \\
&\quad \{(\gamma(\alpha_1,\alpha_2), \mathcal{D}[\![P_1' \parallel P_2']\!]) \mid P_1 \stackrel{\alpha_1}{\Longrightarrow} P_1', P_2 \stackrel{\alpha_2}{\Longrightarrow} P_2', \gamma(\alpha_1,\alpha_2) \in I\} \cup \\
&\quad \{(g, \mathcal{D}[\![y_1 \parallel \cdots \parallel y_m \parallel P_1' \parallel P_2']\!]) \mid \\
&\qquad P_1 \stackrel{\alpha_1}{\Longrightarrow} P_1', P_2 \stackrel{\alpha_2}{\Longrightarrow} P_2', \gamma(\alpha_1,\alpha_2) = g = (\overline{x}.[y_1,\ldots,y_m]) \in G\} \\
&\quad [\parallel \text{is commutative and associative, Remark 5.11(a)}] \\
&= \{(\alpha_1, \mathcal{D}[\![P_1']\!] \parallel \mathcal{D}[\![P_2]\!]) \mid P_1 \stackrel{\alpha_1}{\Longrightarrow} P_1'\} \cup \\
&\quad \{(\alpha_2, \mathcal{D}[\![P_2']\!] \parallel \mathcal{D}[\![P_1]\!]) \mid P_2 \stackrel{\alpha_2}{\Longrightarrow} P_2'\} \cup \\
&\quad \{(\gamma(\alpha_1,\alpha_2), \mathcal{D}[\![P_1']\!] \parallel \mathcal{D}[\![P_2']\!]) \mid P_1 \stackrel{\alpha_1}{\Longrightarrow} P_1', P_2 \stackrel{\alpha_2}{\Longrightarrow} P_2', \gamma(\alpha_1,\alpha_2) \in I\} \cup \\
&\quad \{(g, p_{[y_1,\ldots,y_m]} \parallel \mathcal{D}[\![P_1']\!] \parallel \mathcal{D}[\![P_2']\!]) \mid \\
&\qquad P_1 \stackrel{\alpha_1}{\Longrightarrow} P_1', P_2 \stackrel{\alpha_2}{\Longrightarrow} P_2', \gamma(\alpha_1,\alpha_2) = g = (\overline{x}.[y_1,\ldots,y_m]) \in G\} \\
&= (\Psi_D(\mathcal{D})(P_1) \between \mathcal{D}[\![P_2]\!]) \cup \\
&\quad (\Psi_D(\mathcal{D})(P_2) \between \mathcal{D}[\![P_1]\!]) \cup \\
&\quad (\Psi_D(\mathcal{D})(P_1) \mid \Psi_D(\mathcal{D})(P_2)) \\
&\quad [\text{Induction hypothesis}] \\
&= (\mathcal{D}[\![P_1]\!] \between \mathcal{D}[\![P_2]\!]) \cup (\mathcal{D}[\![P_2]\!] \between \mathcal{D}[\![P_1]\!]) \cup (\mathcal{D}[\![P_1]\!] \mid \mathcal{D}[\![P_2]\!]) \\
&= \mathcal{D}[\![P_1]\!] \parallel \mathcal{D}[\![P_2]\!] \\
&= \mathcal{D}[\![P_1 \parallel P_2]\!]
\end{aligned}
$$

- Case $P = Q^*$. By Lemma 6.5 we know that $c(Q^k) < c(Q^*)$.

$$
\begin{aligned}
&\Psi_D(\mathcal{D})(Q^*) = \{(\alpha, \mathcal{D}[\![R]\!]) \mid Q^* \stackrel{\alpha}{\Longrightarrow} R\} \\
&= \{(\alpha, \mathcal{D}[\![Q' \parallel Q^*]\!]) \mid Q^k \stackrel{\alpha}{\Longrightarrow} Q'\} \\
&= \{(\alpha, \mathcal{D}[\![Q']\!] \parallel \mathcal{D}[\![Q^*]\!]) \mid Q^k \stackrel{\alpha}{\Longrightarrow} Q'\} \\
&= \{(\alpha, \mathcal{D}[\![Q']\!]) \mid Q^k \stackrel{\alpha}{\Longrightarrow} Q'\} \between \mathcal{D}[\![Q^*]\!] \\
&= (\Psi_D(\mathcal{D})(Q^k)) \between \mathcal{D}[\![Q^*]\!] \qquad [\text{induction hypothesis } (c(Q^k) < c(Q^*))] \\
&= \mathcal{D}[\![Q^k]\!] \between \mathcal{D}[\![Q^*]\!] \\
&= \mathcal{D}[\![Q]\!]^k \between \mathcal{D}[\![Q^*]\!] \qquad [\text{Remark 5.11(b)}]
\end{aligned}
$$

$= \mathcal{D}[\![Q^*]\!]$

Theorem 8.5 *$\mathcal{O} = abs \circ \mathcal{D}$, on $\mathcal{L}_{DNA}$.*

Proof: For any $P \in \mathcal{L}_{DNA}$ we have:

$\mathcal{O}[\![P]\!]$ [Lemma 7.13]

$= \mathcal{O}_A[\![P]\!]$ [Lemma 8.3]

$= abs(\mathcal{O}_D[\![P]\!])$ [Lemma 8.4]

$= abs(\mathcal{D}[\![P]\!])$

As an immediate consequence of Theorem 8.5 and Remark 2.7 we obtain the following

Corollary 8.6 *$\mathcal{D}$ is* correct *for $\mathcal{L}_{DNA}$ with respect to $\mathcal{O}$.*

9 Reactions and observables

In this section (in Proposition 9.1) we show that if P is an $\mathcal{L}_{DNA}$ component and g is a G gate then (either P has no reactions or) there is a unique component P' (up to mixing bisimulation $\equiv$) such that $P \xrightarrow{g} P'$. The property is not needed in the proof of the correctness result presented in Section 8 (which is the main result of the paper). However, as explained in Remark 4.7, the property justifies our decision to use G gates as observable elements in the design of the operational semantics $\mathcal{O}[\![\cdot]\!]$. We recall that $\mathcal{O}[\![\cdot]\!]$ was defined in Section 4 based on the reaction relation $\longrightarrow \subseteq \mathcal{L}_{DNA} \times G \times \mathcal{L}_{DNA}$. Also, we recall that the mixing relation $\equiv$ is a strong bisimulation with respect to reaction relation $\longrightarrow$ (Lemma 4.6). In the proof of Proposition 9.1 we use Lemma 9.3. In the proof of Lemma 9.3(b) we employ a concept of *standard form*, introduced in Definition 9.2. The proof of Lemma 9.3 also relies on some auxiliary results presented separately in Section 9.1 and Section 9.2.

Propozition 9.1 *If $P \xrightarrow{g} P'$ and $P \xrightarrow{g} P''$ then $P' \equiv P''$.*

Proof: We assume that $P \xrightarrow{g} P'$ and $P \xrightarrow{g} P''$, with $g = ([x_1, \ldots, x_n].[y_1, \ldots, y_m])$, $g \in G$. By Lemma 9.3(a) there exist Q', Q'' such that $P \equiv x_1 \parallel \cdots \parallel x_n \parallel g \parallel Q'$, $P' \equiv y_1 \parallel \cdots \parallel y_m \parallel Q'$, $P \equiv x_1 \parallel \cdots \parallel x_n \parallel g \parallel Q''$, and $P'' \equiv y_1 \parallel \cdots \parallel y_m \parallel Q''$. By Lemma 9.3(b), $Q' \equiv Q''$. Hence (by using rule (C1) in Definition 4.1) we obtain $P' \equiv P''$.

Definition 9.2

(a) Let $(a, b \in)\mathcal{A}_{DNA}$ be the set of elementary $\mathcal{L}_{DNA}$ components: $a ::= x \mid g$. Let $(M \in)\mathcal{L}_M$, $M ::= \mathbf{0} \mid a \parallel M$. We write a nonempty structure $a_1 \parallel (\cdots \parallel (a_n \parallel \mathbf{0}) \cdots)$ as a list $a_1 \parallel \cdots \parallel a_n$ (the order of association can be ignored).[13] *Let $(S \in)\mathcal{L}_S$, be the set of $\mathcal{L}_M$ lists without duplicates. $S \in \mathcal{L}_S$ if $S = \mathbf{0}$ or $S = a_1 \parallel \cdots \parallel a_n \in \mathcal{L}_M$, and $a_i \neq a_j, \forall 1 \leq i \neq j \leq n$. Obviously, $\mathcal{L}_S \subseteq \mathcal{L}_M \subseteq \mathcal{L}_{DNA}$.*

[13] We could also put $M ::= \mathbf{0} \mid N$, $N ::= a \mid N \parallel N$, but in this case some details of the proof are more complicated.

(b) We say that a component $M \parallel S^*$, with $M \in \mathcal{L}_M$, $S \in \mathcal{L}_S$, is in standard form *if M and S are disjoint, i.e., either $M = \mathbf{0}$, or $S = \mathbf{0}$, or $M = a_1 \parallel \cdots \parallel a_n$, $S = a'_1 \parallel \cdots \parallel a'_m$ and $a_i \neq a'_j, \forall 1 \leq i \leq n, 1 \leq j \leq m$.*

The proof of Lemma 9.3(b) is based on the following main ideas. Every $\mathcal{L}_{DNA}$ component P is equivalent to a standard form $M \parallel S^*$: $P \equiv M \parallel S^*$. Let $a \in \mathcal{A}_{DNA}, M_1, M_2 \in \mathcal{L}_M$, and $S_1, S_2 \in \mathcal{L}_S$.If $M_1 \parallel S_1^*$ and $M_2 \parallel S_2^*$ are in standard form then $(M_1 \parallel S_1^* \equiv M_2 \parallel S_2^*)$ $\Leftrightarrow$ ($M_1 \equiv M_2$ and $S_1 \equiv S_2$). Also, $a \parallel M_1 \equiv a \parallel M_2 \Leftrightarrow M_1 \equiv M_2$. By using these properties and some further technical lemmas, we prove that $a \parallel P \equiv a \parallel Q \Leftrightarrow P \equiv Q$, for any $a \in \mathcal{A}_{DNA}, P, Q \in \mathcal{L}_{DNA}$.

Lemma 9.3

(a) Let $g = ([x_1, \ldots, x_n].[y_1, \ldots, y_m]) \in G$. If $P \xrightarrow{g} P'$ then there exists a component $Q \in \mathcal{L}_{DNA}$ such that $P \equiv x_1 \parallel \cdots \parallel x_n \parallel g \parallel Q$ and $P' \equiv y_1 \parallel \cdots \parallel y_m \parallel Q$.

(b) Let $(a \in)\mathcal{A}_{DNA}$ be the set of elementary $\mathcal{L}_{DNA}$ components introduced in Definition 9.2(a). For any $a \in \mathcal{A}_{DNA}$, and $P, Q \in \mathcal{L}_{DNA}$: $a \parallel P \equiv a \parallel Q \Leftrightarrow P \equiv Q$. More generally, for any $a_1, \ldots, a_n \in \mathcal{A}_{DNA}$, and $P, Q \in \mathcal{L}_{DNA}$: $a_1 \parallel \cdots \parallel a_n \parallel P \equiv a_1 \parallel \cdots \parallel a_n \parallel Q \Leftrightarrow P \equiv Q$.

Proof: Lemma 9.3(a) follows by an easy induction on the depth of the inference of $P \xrightarrow{g} P'$.

For Lemma 9.3(b) it is enough to prove that: $a \parallel P \equiv a \parallel Q \Leftrightarrow P \equiv Q$, for any $a \in \mathcal{A}_{DNA}$, and $P, Q \in \mathcal{L}_{DNA}$. Next, the more general property $a_1 \parallel \cdots \parallel a_n \parallel P \equiv a_1 \parallel \cdots \parallel a_n \parallel Q$ $\Leftrightarrow P \equiv Q$, for any $a_1, \ldots, a_n \in \mathcal{A}_{DNA}$, and $P, Q \in \mathcal{L}_{DNA}$, follows immediately.

The implication $P \equiv Q \Rightarrow a \parallel P \equiv a \parallel Q$ follows easily by using property (C1) (Definition 4.1). In the sequel we prove that: $a \parallel P \equiv a \parallel Q \Rightarrow P \equiv Q$, for any $P, Q \in \mathcal{L}_{DNA}, a \in \mathcal{A}_{DNA}$.

Let $(M_P, S_P) \in SF(P)$, $(M_Q, S_Q) \in SF(Q)$. By Lemma 9.26, $M_P \parallel S_P^*$ and $M_Q \parallel S_Q^*$ are in standard form and $P \equiv M_P \parallel S_P^*$, $Q \equiv M_Q \parallel S_Q^*$. By Lemma 9.8, if $a \in S_P$ then $S_P \equiv a \parallel (S_P \setminus a)$. Hence, by Lemma 9.13(c), if $a \in S_P$ then $a \parallel S_P^* \equiv S_P^*$. Hence:

$$a \parallel P \equiv a \parallel (M_P \parallel S_P^*) \equiv \begin{cases} M_P \parallel S_P^* & \text{if } a \in S_P \\ (a \parallel M_P) \parallel S_P^* & \text{if } a \notin S_P \end{cases}$$

$$a \parallel Q \equiv a \parallel (M_Q \parallel S_Q^*) \equiv \begin{cases} M_Q \parallel S_Q^* & \text{if } a \in S_Q \\ (a \parallel M_Q) \parallel S_Q^* & \text{if } a \notin S_Q \end{cases}$$

By Remark 9.7, $a \in S_P \Leftrightarrow a \in ms(S_P)$ and $a \in S_Q \Leftrightarrow a \in ms(S_Q)$. By using Lemma 9.28, from $a \parallel P \equiv a \parallel Q$ we infer that $S_P \equiv S_Q$. By Lemma 9.10, $ms(S_P) = ms(S_Q)$. By using Remark 9.7: $a \in S_P \Leftrightarrow a \in ms(S_P) \Leftrightarrow a \in ms(S_Q) \Leftrightarrow a \in S_Q$. Therefore we have two subcases.

- If $a \in S_P (\Leftrightarrow a \in S_Q)$ then $P \equiv M_P \parallel S_P^* \equiv M_Q \parallel S_Q^* \equiv Q$,
- If $a \notin S_P (\Leftrightarrow a \notin S_Q)$ then
  - $a \parallel P \equiv (a \parallel M_P) \parallel S_P^*$, and $(a \parallel M_P) \parallel S_P^*$ is in standard form, and
  - $a \parallel Q \equiv (a \parallel M_Q) \parallel S_Q^*$, and $(a \parallel M_Q) \parallel S_Q^*$ is in standard form,

In this case $(a \parallel M_P) \parallel S_P^* \equiv (a \parallel M_Q) \parallel S_Q^*$. By Lemma 9.28, $S_P \equiv S_Q$ and $a \parallel M_P \equiv a \parallel M_Q$. By Lemma 9.12, $M_P \equiv M_Q$. Hence again, $P \equiv M_P \parallel S_P^* \equiv M_P \parallel S_P^* \equiv Q$. □

In subsections 9.1 and 9.2 we present some auxiliary properties required in the proof of Lemma 9.3(b).

9.1 Properties related to $\mathcal{L}_M$ components

In this subsection we study the properties of $\mathcal{L}_M$ components. The sets $\mathcal{A}_{DNA}$, $\mathcal{L}_M$ and $\mathcal{L}_S$ are introduced in Definition 9.2.

Definition 9.4 *Let $[\mathcal{A}_{DNA}]$ be the set of all finite multisets of elementary $\mathcal{L}_{DNA}$ components. Let $\overline{\infty}$ be a distinct element, $\overline{\infty} \notin [\mathcal{A}_{DNA}]$. Formally, $\overline{\infty}$ is just a symbol. Intuitively, we use $\overline{\infty}$ to denote any infinite multiset. We define a mapping $ms : \mathcal{L}_{DNA} \to ([\mathcal{A}_{DNA}] \cup \{\overline{\infty}\})$ by:*

$$ms(\mathbf{0}) = [],\ ms(a) = [a]$$

$$ms(P^*) = \begin{cases} [] & \text{if } ms(P) = [] \\ \overline{\infty} & \text{otherwise} \end{cases}$$

$$ms(P_1 \parallel P_2) = ms(P_1) \uplus ms(P_1)$$

where, when $\overline{a}, \overline{a}_1, \overline{a}_2 \in [\mathcal{A}_{DNA}]$, $\overline{a}_1 \uplus \overline{a}_2$ is the standard multiset sum (introduced in Section 2.1) and with $\overline{\infty}$ we compute as follows: $\infty \uplus \overline{a} = \overline{a} \uplus \overline{\infty} = \overline{\infty} \uplus \overline{\infty} = \overline{\infty}$.

Remarks 9.5

(a) $ms(P)$ is clearly well defined, by structural on $P \in \mathcal{L}_{DNA}$.

(b) On $\mathcal{L}_M$, ms behaves as follows:

$$ms(\mathbf{0}) = []$$
$$ms(a \parallel M) = [a] \uplus ms(M)$$

$ms(M) \in [\mathcal{A}_{DNA}]$ (i.e., $ms(M) \neq \overline{\infty}$), for any $M \in \mathcal{L}_M$. Also, $|ms(M)| \in \mathbf{N}$, for any $M \in \mathcal{L}_M$.

Definition 9.6 *We define $a \in M$ ($\in : \mathcal{A}_{DNA} \times \mathcal{L}_M \to Bool$) by:*

$$a \in \mathbf{0} = false$$

$$a \in (a' \parallel M') = (a = a') \vee (a \in M')$$

We also define $M \setminus a$ ($\setminus : \mathcal{L}_M \times \mathcal{A}_{DNA} \to \mathcal{L}_M$) by:

$$\mathbf{0} \setminus a = \mathbf{0}$$

$$(a' \parallel M') \setminus a = \begin{cases} M' & \text{if } a' = a \\ a' \parallel (M' \setminus a) & \text{if } a' \neq a \end{cases}$$

$a \in M$ and $M \setminus a$ are mappings defined by structural induction on M. We write $a \notin M$ to express that $\neg(a \in M)$. By a slight abuse for the function names we use symbols that are traditionally used to represent (multi)set membership and (multi)set difference. However, it will always be clear from the context which is the type of $\in$ and $\setminus$.

Remark 9.7 *$a \in M \Leftrightarrow a \in ms(M), \forall M \in \mathcal{L}_M$.*

Proof: Easy structural induction on M.

Lemma 9.8 *For any $a \in \mathcal{A}_{DNA}, M \in \mathcal{L}_M$:*

(a) $a \in M \Leftrightarrow a \in ms(M)$.

(b) If $a \in M$ then $a \parallel (M \setminus a) \equiv M$

Proof: For Lemma 9.8(a), the proof of the implication ($\Rightarrow$) is an easy structural induction on M, and the proof of the implication ($\Leftarrow$) can proceed by induction on $|ms(M)|$.

The proof of Lemma 9.8(b) can proceed by structural induction on M.

- Case $M = 0$. In this case the property holds trivially, because $a \notin M$ (the assumption of the Lemma fails in this case).
- Case $M = a' \parallel M'$. Two sub-cases:
  - If $a' = a$ then $M \setminus a = M'$, hence $a \parallel (M \setminus a) = a' \parallel M' = M \equiv M$.
  - If $a' \neq a$ then $M \setminus a = (a' \parallel M') \setminus a = a' \parallel (M' \setminus a)$. By the induction hypothesis $a \parallel (M' \setminus a) \equiv M'$. Therefore, $M = a' \parallel M' \equiv a' \parallel (a \parallel (M' \setminus a)) \equiv a \parallel (a' \parallel (M' \setminus a)) \equiv a \parallel (M \setminus a)$.

Lemma 9.9 *For any $a \in \mathcal{A}_{DNA}, M \in \mathcal{L}_M$: $ms(M) \setminus [a] = ms(M \setminus a)$.*

Proof: By structural induction on M.

Lemma 9.10

(a) For any $P \in \mathcal{L}_{DNA}$: $ms(P) = []$ implies $P \equiv \mathbf{0}$.

(b) For any $P \in \mathcal{L}_{DNA}$: $ms(P) = [a]$ implies $P \equiv a$.

(c) For any $P_1, P_2 \in \mathcal{L}_{DNA}$: $P_1 \equiv P_2$ implies $ms(P_1) = ms(P_2)$.

Proof: The proofs of Lemma 9.10(a) and Lemma 9.10(b) can proceed by structural induction on P. The proof of Lemma 9.10(c) can proceed by induction on the depth of inference of $P_1 \equiv P_2$, considering the various cases for the last step of inference of $P_1 \equiv P_2$.

Lemma 9.11 *For any $M_1, M_2 \in \mathcal{L}_M$: $M_1 \equiv M_2 \Leftrightarrow ms(M_1) = ms(M_2)$.*

Proof: The proof of ($\Leftarrow$) can proceed by induction on $|ms(M_1)| = |ms(M_2)|$. By Remark 9.5, $ms(M) \in [\mathcal{A}_{DNA}]$, and $|ms(M)| \in \mathbf{N}$, for any $M \in \mathcal{L}_M$.

($\Rightarrow$) is an immediate consequence of Lemma 9.10(b).

Lemma 9.12 *For any* $M_1, M_2 \in \mathcal{L}_M, a \in \mathcal{A}_{DNA}$*:* $a \parallel M_1 \equiv a \parallel M_2 \Leftrightarrow M_1 \equiv M_2$.

Proof: ($\Leftarrow$) is obvious.

For ($\Rightarrow$) we notice that $a \parallel M_1 \equiv a \parallel M_2$ implies $ms(a \parallel M_1) = ms(a \parallel M_2)$, by Lemma 9.11. Therefore, $[a] \uplus ms(M_1) = [a] \uplus ms(M_2)$, which implies $ms(M_1) = ms(M_2)$, hence $M_1 \equiv M_2$, by Lemma 9.11.

9.2 Properties related to standard forms

We recall that the notion of a standard form was introduced in Definition 9.2. In this subsection we present a recursive algorithm for computing standard forms. It is easy to establish the following Lemma.

Lemma 9.13 *For any* $P, Q \in \mathcal{L}_{DNA}$*:*

(a) $(P^* \parallel Q)^* \equiv (P \parallel Q)^*$,

(b) $(P^n \parallel Q)^* \equiv (P \parallel Q)^*$*, where (as in Definition 6.1), for any* $P \in \mathcal{L}_{DNA}, n \in \mathbf{N}^+$ *we define* P^n *by induction on* n*:* $P^1 = P, P^{n+1} = P \parallel P^n$,

(c) $P \parallel (P \parallel Q)^* \equiv (P \parallel Q)^*$*, and*

(d) $(P^n)^* \equiv P^*$*, for any* $n \in \mathbf{N}^+$.

We also recall that $(a, b \in)\mathcal{A}_{DNA}$ is the set of elementary $\mathcal{L}_{DNA}$ components: $a ::= x \mid g$. According to our convention presented in Section 2.1, $\overline{a}, \overline{b} \in [\mathcal{A}_{DNA}]$ denote multisets of elementary components.

Definition 9.14 *We define a mapping* $sf : \mathcal{L}_{DNA} \to ([\mathcal{A}_{DNA}] \times [\mathcal{A}_{DNA}])$ *by:*

$$sf(\mathbf{0}) = ([], [])$$

$$sf(a) = ([a], [])$$

$$sf(P_1 \parallel P_2) = \ \mathsf{let}\ (\overline{a}_1, \overline{b}_1) = sf(P_1), (\overline{a}_2, \overline{b}_2) = sf(P_2)$$
$$\mathsf{in}\ ((\overline{a}_1 \uplus \overline{a}_2) \setminus\!\setminus (\overline{b}_1 \cup \overline{b}_2), \overline{b}_1 \cup \overline{b}_2)$$

$$sf(P^*) = \ \mathsf{let}\ (\overline{a}, \overline{b}) = sf(P)\ \mathsf{in}\ ([], \{\!|\overline{a}|\!\} \cup \overline{b})$$

Remark 9.15 *If* $P \in \mathcal{L}_{DNA}$ *and* $(\overline{a}, \overline{b}) = sf(P)$ *then* $\overline{a} \setminus\!\setminus \overline{b} = \overline{a}$*, and* $\{\!|\overline{b}|\!\} = \overline{b}$.

Proof: By structural induction on P.

Lemma 9.16 *For any* $M \in \mathcal{L}_M$*:* $sf(M) = (ms(M), [])$.

Proof: By structural induction on M.

Lemma 9.17

(a) For any $S \in \mathcal{L}_S$*:* $\{\!|ms(S)|\!\} = ms(S)$.

(b) $a \parallel S \in \mathcal{L}_S$ *implies* $S \in \mathcal{L}_S$.

(c) For any $S \in \mathcal{L}_S$*:* $sf(S^*) = ([], ms(S))$.

Proof: Parts (a) and (b) follow by the definition of $\mathcal{L}_S$. The proof of part (c) can proceed by structural induction on S.

Lemma 9.18 *If* $M \in \mathcal{L}_M, S \in \mathcal{L}_S$ *are such that:* $ms(M) \setminus\!\setminus ms(S) = ms(M)$ *then* $sf(M \parallel S^*) = (ms(M), ms(S))$.

Proof: By Lemma 9.16 and Lemma 9.17, $sf(M) = (ms(M), [])$, and $sf(S) = ([], ms(S))$. By the definition of sf, $sf(P_1 \parallel P_2) = ((ms(M) \uplus []) \setminus\!\setminus ms(S), ms(S)) = (ms(M), ms(S))$

Lemma 9.19 *For any* $P_1, P_2 \in \mathcal{L}_{DNA}$*:* $P_1 \equiv P_2$ *implies* $sf(P_1) = sf(P_2)$.

Proof: By induction on the depth of inference of $P_1 \equiv P_2$, considering the various cases for the last step of inference. We consider two cases.

- Assume that rule (D2) is used in the last step, i.e., for some $P, Q \in \mathcal{L}_{DNA}$: $P_1 = P \parallel Q \equiv Q \parallel P = P_2$. In this case

  $$\begin{aligned}
  & sf(P_1) = sf(P \parallel Q) \\
  & = \ \mathsf{let}\ (\overline{a}_P, \overline{b}_P) = sf(P), (\overline{a}_Q, \overline{b}_Q) = sf(Q)\ \ \mathsf{in}\ \ ((\overline{a}_P \uplus \overline{a}_Q) \setminus\!\setminus (\overline{b}_P \cup \overline{b}_Q), \overline{b}_P \cup \overline{b}_Q) \\
  & [\uplus \text{ and } \cup \text{ are commutative}] \\
  & = \ \mathsf{let}\ (\overline{a}_Q, \overline{b}_Q) = sf(Q), (\overline{a}_P, \overline{b}_P) = sf(P)\ \ \mathsf{in}\ \ ((\overline{a}_Q \uplus \overline{a}_P) \setminus\!\setminus (\overline{b}_Q \cup \overline{b}_P), \overline{b}_Q \cup \overline{b}_P) \\
  & = sf(Q \parallel P) = sf(P_2)
  \end{aligned}$$

- Assume that rule (P1) is used in the last step, i.e., for some $P \in \mathcal{L}_{DNA}$: $P_1 = P^* \equiv P \parallel P^* = P_2$. In this case

  $$sf(P_1) = sf(P^*) = \ \mathsf{let}\ (\overline{a}, \overline{b}) = sf(P)\ \ \mathsf{in}\ \ ([], \{\!|\overline{a}|\!\} \cup \overline{b})$$

  and

  $$\begin{aligned}
  & sf(P_2) = sf(P \parallel P^*) \\
  & = \ \mathsf{let}\ (\overline{a}, \overline{b}) = sf(P), ([], \{\!|\overline{a}|\!\} \cup \overline{b}) = sf(P^*)\ \ \mathsf{in}\ \ ((\overline{a} \uplus []) \setminus\!\setminus (\overline{b} \cup (\{\!|\overline{a}|\!\} \cup \overline{b})), \overline{b} \cup (\{\!|\overline{a}|\!\} \cup \overline{b})) \\
  & [\overline{a} \setminus\!\setminus \{\!|\overline{a}|\!\} = []] \\
  & = ([], \{\!|\overline{a}|\!\} \cup \overline{b}) = sf(P^*) = sf(P_1)
  \end{aligned}$$

Lemma 9.20 *If* $S, S_1, S_2 \in \mathcal{L}_S$ *are such that* $ms(S) = ms(S_1) \cup ms(S_2)$ *then* $S^* \equiv (S_1 \parallel S_2)^*$.

Proof: By structural induction on S.

Lemma 9.21 *If $M, M' \in \mathcal{L}_M, S \in \mathcal{L}_S$ are such that $ms(M) = ms(M') \setminus\setminus ms(S)$ then $M' \parallel S^* \equiv M \parallel S^*$.*

Proof: By structural induction on M'.

Lemma 9.22 *If $M \in \mathcal{L}_M, S, S' \in \mathcal{L}_S$ are such that $ms(S) = \{\!|ms(M)|\!\} \cup ms(S')$ then $S^* \equiv M^* \parallel S'^*$. obs.2.9*

Proof: By structural induction on M.

Definition 9.23 *We define a mapping $SF : \mathcal{L}_{DNA} \to \mathcal{P}_{nfin}(\mathcal{L}_M \times \mathcal{L}_S)$ by:*[14]

$$SF(P) = \{(M, S) \mid (\overline{a}, \overline{b}) = sf(P), ms(M) = \overline{a}, ms(S) = \overline{b}\}$$

Lemma 9.24 *For any $\overline{a} \in [\mathcal{A}_{DNA}]$ there exists $M \in \mathcal{L}_M$ such that $\overline{a} = ms(M)$.*

Proof: By induction on $|\overline{a}|$.

Remarks 9.25 *For any $P \in \mathcal{L}_{DNA}$*

(b) The set $SF(P)$ is finite and nonempty (according to Lemma 9.24), for any $P \in \mathcal{L}_{DNA}$, hence SF is well defined.

(c) Let $(M, S) \in SF(P)$. According to Remark 9.11, $ms(M) \setminus\setminus ms(S) = ms(M)$.

(d) If $(M, S) \in SF(P)$ then $M \parallel S^*$ is in standard form.

Lemma 9.26 *For any $P \in \mathcal{L}_{DNA}$ and for any $(M, S) \in SF(P)$: $P \equiv M \parallel S^*$, and $M \parallel S^*$ is in standard form.*

Proof: Let $(\overline{a}, \overline{b}) = sf(P)$. Let $(M, S) \in SF(P)$. By Definition 9.23, $ms(M) = \overline{a}$, $ms(S) = \overline{b}$. We must prove that $P \equiv M \parallel S^*$ and $M \parallel S^*$ is in standard form. We proceed by structural induction on P. We consider two subcases.

- Case $P = a$. In this case $sf(P) = sf(a) = ([a], [])$, and $ms(M) = [a]$ $ms(S) = []$. By Lemma 9.10(a) and Lemma 9.10(b), $M = a$, $S = 0$. Hence $P = a \equiv a \parallel \mathbf{0}^* = M \parallel S^*$. Obviously, $a \parallel \mathbf{0}^*$ is in standard form.

- Case $P = Q^*$. Let $sf(Q) = (\overline{a}, \overline{b})$. By the induction hypothesis, for any $(M_Q, S_Q) \in SF(Q)$ (i.e., for any $M_Q \in \mathcal{L}_M$, $S_Q \in \mathcal{L}_S$, such that $ms(M_Q) = \overline{a}$, $ms(S_Q) = \overline{b}$) we know that $Q \equiv M_Q \parallel S_Q^*$, and $M_Q \parallel S_Q^*$ is in standard form.

  $sf(P) = sf(Q^*) = ([], \{\!|\overline{a}|\!\} \cup \overline{b})$. We must prove that if $M \in \mathcal{L}_M$, $S \in \mathcal{L}_S$ are such that $ms(M) = []$ and $ms(S) = \{\!|\overline{a}|\!\} \cup \overline{b} = \{\!|ms(M_Q)|\!\} \cup ms(S_Q)$ then $P \equiv M \parallel S^*$. This follows by using Lemma 9.22:

  $$P = Q^* \equiv (M_Q \parallel S_Q^*)^* \qquad \text{[Lemma 9.13(a)]}$$

[14] We recall that $\mathcal{P}_{nfin}(\cdot)$ is the set of all nonempty and finite subsets of '$\cdot$'; see Section 2.

$\equiv M_Q^* \parallel S_Q^*$ [Lemma 9.22]

$\equiv S^*$ [$ms(M) = []$, hence, by Lemma 9.10(a), $M \equiv \mathbf{0}$]

$\equiv M \parallel S^*$

Obviously, $ms(M) \setminus\setminus ms(S) = [] \setminus\setminus ms(S) = [] = ms(M)$, hence $M \parallel S^*$ is in standard form.

Lemma 9.27 *For any $P, Q \in \mathcal{L}_{DNA}$ if $sf(P) = sf(Q)$ then $P \equiv Q$.*

Proof: $sf(P) = sf(Q)$ implies $SF(P) = SF(Q)$. Let $(M, S) \in SF(P) = SF(Q)$. By Lemma 9.26, $P \equiv M \parallel S^* \equiv Q$.

Lemma 9.28 *If $M_1, M_2 \in \mathcal{L}_M, S_1, S_2 \in \mathcal{L}_S$ are such that $ms(M_1) \setminus\setminus ms(S_1) = ms(M_1)$, and*
$ms(M_2) \setminus\setminus ms(S_2) = ms(M_2)$ then $M_1 \parallel S_1^* \equiv M_2 \parallel S_2^* \Leftrightarrow M_1 \equiv M_2, S_1 \equiv S_2$.

Proof: ($\Leftarrow$) is obvious.

For ($\Rightarrow$) we compute as follows:

$(ms(M_1), ms(S_1))$ [Lemma 9.18]

$= sf(M_1 \parallel S_1^*)$ [Lemma 9.19]

$= sf(M_2 \parallel S_2^*)$ [Lemma 9.18]

$= (ms(M_2), ms(S_2))$

By Lemma 9.11, $M_1 \equiv M_2$, $S_1 \equiv S_2$.

10 Concluding remarks

By using metric semantics, we related formally different semantic models for a language based on the combinatorial strand algebra introduced in [7]. The formalism consists of a process algebra language which incorporates some basic concepts of DNA computing: signals, gates, join synchronization [10] and unbound populations. We presented a denotational semantics and we proved the correctness of the denotational semantics with respect to an operational semantics introduced in [7]. To the best of our knowledge this is the first work (apart from our conference paper [9]) providing a study of comparative semantics for a concurrent language inspired by DNA computing. In [8] and [17] we use metric spaces in designing denotational models for parallel rewriting of multisets and join synchronization, respectively. However, the semantic models presented in [8] and [17] are designed with continuations and are not close to the DNA strand algebra investigated in this paper.

We intend to continue the research concerning the behavior of DNA systems by using methods in the tradition of programming languages semantics. Our next aim is to design a fully abstract denotational model for the formalism that we investigated in this paper.

References

[1] A. Alexandru, G. Ciobanu. Mathematics of multisets in the Fraenkel-Mostowski framework. *Bull. Math. Soc. Sci. Math. Roumanie*, tome 58(106), no.1, 3–18, 2015.

[2] B. Aman, G. Ciobanu. *Mobility in Process Calculi and Natural Computing.* Natural Computing Series, Springer, 2011.

[3] J.W. de Bakker, J.I. Zucker. Processes and the denotational semantics of concurrency. *Information and Control* 54:70–120, 1982.

[4] J.W. de Bakker, J.A. Bergstra, J.W. Klop, J.J.-Ch. Meyer. Linear time and branching time semantics for recursion with merge. *Lecture Notes in Computer Science* vol.154, 39–51, 1983.

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

[6] G. Berry, G. Baudol. The chemical abstract machine. *Theoretical Computer Science*, 96:217–248, 1992.

[7] L. Cardelli. Strand algebras for DNA computing. *Natural Computing*, 10:407-428, 2011.

[8] G. Ciobanu, E.N. Todoran. Metric denotational semantics for parallel rewriting of multisets. In *Proc. SYNASC 2011*, 276–284, IEEE Computer Press, 2011.

[9] G. Ciobanu, E.N. Todoran. Correct metric semantics for a biologically-inspired formalism. In *Proc. SYNASC 2014*, 317–324, IEEE Computer Press, 2014.

[10] C. Fournet, G. Gonthier. The Join calculus: a language for distributed mobile programming. *Lecture Notes in Computer Science* vol.25, 268–332, 2002.

[11] G. Giertz, D.S. Scott, *Continuous Lattices and Domains.* Cambridge University Press, 2003.

[12] R.J. van Glabbeek, J.J.M.M. Rutten. The processes of De Bakker and Zucker represent bisimulation equivalence classes. In *J.W. de Bakker 25 Jaar Semantick, Liber Amicorum*, 243–246, CWI Amsterdam, 1989.

[13] R. Milner. Processes: a mathematical model of computing agents. In *Proc. Logic Colloquium*, 157–173, North-Holland, 1973.

[14] R. Milner. Fully abstract models of typed λ-calculi. *Theoretical Computer Science*, 4: 1–22, 1977.

[15] R. Milner. *Communicating and Mobile Systems: the π-calculus.* Cambridge University Press, 1999.

[16] G. Plotkin. A structural approach to operational semantics. *Journal of Logic and Algebraic Programming*, (60-61):17–139, 2004.

[17] E.N. Todoran. Comparative semantics for modern communication abstractions, *Proc. ICCP*, 153–160, IEEE Computer Press, 2008.