# Main Concepts of Networks of Transformation Units with Interlinking Semantics*

Dirk Janssens[1], Hans-Jörg Kreowski[2], and Grzegorz Rozenberg[3]

[1] University of Antwerp,
Department of Mathematics and Computer Science,
Antwerp, Belgium
Dirk.Janssens@ua.ac.be
[2] University of Bremen,
Department of Mathematics and Computer Science,
Bremen, Germany
kreo@tzi.de
[3] Leiden University,
Leiden Institute of Advanced Computer Science,
Leiden, The Netherlands
rozenber@liacs.nl

**Abstract.** The aim of this paper is to introduce a modelling concept and structuring principle for rule-based systems the semantics of which is not restricted to a sequential behavior, but can be applied to various types of parallelism and concurrency. The central syntactic notion is that of a transformation unit that encapsulates a set of rules, imports other transformation units, and regulates the use and interaction of both by means of a control condition. The semantics is given by interlinking the applications of rules with the semantics of the imported units using a given collection of semantic operations. As the main result, the interlinking semantics turns out to be the least fixed point of the interlinking operator. The interlinking semantics generalizes the earlier introduced interleaving semantics of rule-based transformation units, which is obtained by the sequential composition of binary relations as only semantic operation.

## 1 Introduction

In this paper, we introduce networks of transformation units with interlinking semantics as a modelling concept and structuring principle for rule-based systems the semantics of which may be non-sequential. The key concept is a transformation unit encapsulating a set of local rules and importing other transformation units. Moreover, each transformation unit has a control condition that regulates

* Research partially supported by the EC Research Training Network SegraVis (Syntactic and Semantic Integration of Visual Modeling Techniques) and by the German Research Foundation (DFG) as part of the Collaborative Research Centre 637 *Autonomous Cooperating Logistic Processes – A Paradigm Shift and its Limitations.*

H.-J. Kreowski et al. (Eds.): Formal Methods (Ehrig Festschrift), LNCS 3393, pp. 325–342, 2005.

the application of the local rules and the interaction with the imported components. If a set of transformation units is closed under import, its import structure forms a network. In this way, large sets of rules can be organized and structured in such a way that each local unit may contain only a small set of rules while the effects of other units can be used by importing them.

In [4–7] transformation units have been introduced for graphs as well as for more general configurations as underlying data structures and provided with a purely sequential semantics. It is obtained by interleaving rule applications and the semantics of the imported components in such a way that the control condition is obeyed. In this paper, we generalize the framework of transformation units such that also non-sequential systems can be specified. For this purpose, we replace the underlying domain of binary relations on configurations by a domain of more general semantic entities and the sequential composition of binary relations by a set of arbitrary operations on semantic entities. But to be able to use set-theoretic operations and their properties, we assume that the domain of semantic entities is the power set of a set of semantics items.

The operations on semantic entities may be chosen as sequential, parallel or concurrent compositions or as any other operations one wants to use to model the type of semantics one is interested in. We show that the new framework covers nicely elementary net systems with their non-sequential processes as well as rule-based systems with sequential and parallel derivations, covering Chomsky grammars and various types of graph grammars in particular. This means that not only these approaches can be seen in a unified framework, but are also provided with a common structuring principle as a novel feature.

The paper is organized in the following way. In the next section, the basic notions and notations of transformation units with interlinking semantics are introduced. Networks of transformation units and their iterated interlinking semantics are studied in Section 3. Finally, the main result of this paper is formulated in Section 4. It states that the iterated interlinking semantics is the least fixed point of the interlinking operator if the used semantic operations and the control conditions are continuous. As running examples, we discuss elementary net systems and binary relations as semantic entities of grammatical systems of various kinds. Because of lack of space, the proofs are omitted.

## 2 Transformation Units with Interlinking Semantics

In this section, we introduce the notion of transformation units with interlinking semantics, which generalizes the formerly defined interleaving semantics.

The basis is the notion of a semantic domain (2.1) consisting of a set of semantic items together with operations on semantic entities being sets of semantic items. Typical semantic items are derivations, computations, and processes; typical operations are sequential and parallel compositions of derivations, computations, and processes or their embedding into larger context. To be able to deal with semantic entities, a semantic domain is first equipped with rules yielding a rule base (2.2) where a rule is some abstract syntactic feature that specifies

a semantic entity. In many examples, rules rewrite some kind of configurations defining direct derivations and computation steps or rules are actions and events that describe elementary processes. Therefore, a set of rules provides a set of semantic entities the union of which may be closed under the operations of the semantic domain. For example, if one applies the sequential composition to direct derivations and computation steps, one gets all derivations and computations resp. Or if one applies certain kinds of parallel composition to elementary processes, one obtains all parallel processes of a set of actions or events. Often this is not enough to describe the behavior of a system. In addition, one may like to choose certain initial and terminal configurations or to regulate the rule applications by imposing a certain order or in some other way. For this purpose, a rule base is additionally equipped with control conditions (2.3) that allow one to restrict semantic effects. Formally, a control condition specifies a semantic entity depending on some environment which associates semantic entities to a given set of identifiers. The idea of this is the following. A control condition as a syntactic feature may use the identifiers to demand or forbid the applications of certain operations to the semantic entities associated to the identifiers and may restrict the free operational closure of these entities in this way.

### 2.1 Semantic Domains

While interleaving semantics is based on the sequential composition of binary relations, the generalization employs an arbitrary set of operations on arbitrary semantic entities. But to keep the technicalities simple, we assume that the semantic entities are the subsets of a set of semantic items such that we have union, intersection, inclusion and all other set-theoretic operations and all their properties for free.

A *semantic domain* $\mathcal{D} = (X, OP)$ consists of a set $X$ of *semantic items* and a set $OP$ of (partial) *operations* on the power set $2^X$ of $X$.

The arities of the operations can vary. The set of operations with arity $k \in \mathbb{N}$ is denoted by $OP_k$. The elements of $2^X$ are called *semantic entities*.

Such a semantic domain provides the operational closure for every set of semantic entities, which can be defined in the usual recursive way.

Let $M \subseteq 2^X$. Then the *operational closure* of $M$, $OP^*(M) \subseteq 2^X$, is recursively defined by

(i) $M \cup OP_0 \subseteq OP^*(M)$, and
(ii) $op(t_1, \ldots, t_k) \in OP^*(M)$ for $op \in OP_k$ and $t_1, \ldots, t_k \in OP^*(M)$.

Starting from the nullary operations and the given semantic entities, the operations are applied repeatedly to all semantic entities that are obtained in this way ad infinitum. The operational closure yields a set of subsets. If one wants to consider the union of them, this may be denoted by $\bigcup OP^*(M)$.

#### Examples

As running examples, we discuss elementary net systems with processes as semantic items (see, e.g., [3, 8, 13]) and binary relations on configurations like words and graphs as the semantic entities of grammatical rules (see, e.g., [12, 9]).

**Elementary Net Systems.** Let $\overline{N} = (\overline{B}, \overline{E}, \overline{F})$ be some contact-free elementary net where $\overline{B}$ is a set of *conditions*, $\overline{E}$ is a set of *events*, and $\overline{F} \subseteq (\overline{B} \times \overline{E}) \cup (\overline{E} \times \overline{B})$ is a *flow relation*. Then one may consider all processes on $\overline{N}$ as semantic items. More formally, $PROC(\overline{N})$ is the set of all pairs $proc = (N, p)$ where $N = (B, E, F)$ is an occurrence net, i.e. an acyclic and conflict-free net, and $p : N \to \overline{N}$ is a net morphism which is injective on cuts.

In particular, each acyclic and conflict-free subnet $N$ of $\overline{N}$ together with the inclusion $incl_N : N \to \overline{N}$ yields a process. As a case $C \subseteq \overline{B}$ can be seen as a subnet $sub(C) = (C, \emptyset, \emptyset)$ with the empty set of events and the empty flow relation, $C$ induces a particular process $proc(C) = (sub(C), incl_{sub(C)})$. In this way, the set of cases can be seen as a subset of the set of processes. Moreover, an event $e \in \overline{E}$ (together with its pre- and post conditions) induces a subnet $sub(e) = ({}^\bullet e \cup e^\bullet, \{e\}, ({}^\bullet e \times \{e\}) \cup (\{e\} \times e^\bullet))$ which is acyclic due to the contact-freeness of $\overline{N}$ and conflict-free by definition. Hence each event provides an elementary process $proc(e) = (sub(e), incl_{sub(e)})$.

There are two natural binary operations on processes: parallel and sequential compositions. Given two processes $proc = (N, p)$ and $proc' = (N', p')$ with $p(N) \cap p(N') = \emptyset$, then the parallel process is given by $proc + proc' = (N + N', <p, p'>)$ where $N + N'$ is the disjoint union of $N$ and $N'$ and $<p, p'>$ is the induced net morphism defined as $p$ on $N$ and as $p'$ on $N'$.

To define the sequential composition, we need the notion of input and output conditions of an occurrence net $N$. The set of conditions with indegree 0 is denoted by $in(N)$ and the set of conditions of outdegree 0 by $out(N)$. Then the sequential composition of two processes $proc = (N, p)$ and $proc' = (N', p')$ requires that $p(out(N)) = p'(in(N'))$ and that there is no further overlap between $p(N)$ and $p'(N')$. The result is given $proc \circ proc' = (N + N' / out(N) = (in(N'), <p, p'>)$ where the occurrence net is the disjoint union of $N$ and $N'$ which is merged in each condition $c$ of $N$ and $c'$ of $N'$ with $p(c) = p'(c')$. The net morphism $<p, p'>$ is defined as in the parallel case by $p$ on elements of $N$ and by $p'$ on elements on $N'$. It is a mapping as $p$ and $p'$ coincide on the merged conditions. Note that the sequential composition of processes $proc$ and $proc'$ is only partially defined by $proc \circ proc'$ if this is a process again.

Based on these preliminaries, we can consider sets of processes on $\overline{N}$ as semantic entities and extend the binary operations elementwise to such sets. Moreover, each condition $c \in \overline{B}$ provides a nullary operation $\hat{c} = \{(proc(\{c\})\} = \{(sub(\{c\}), incl_{sub(\{c\})})\}$ containing as only semantic item the process induced by $\{c\}$. Altogether we get the semantic domain $\mathcal{D}(\overline{N}) = (PROC(\overline{N}), OP(\overline{N}))$ with $OP(\overline{N}) = \{+, \circ\} \cup \{\hat{c} \mid c \in \overline{B}\}$, which is associated to the given elementary net $\overline{N}$.

It should be noted that the set of processes corresponding to cases $C \subseteq \overline{B}$ is just the closure of the nullary operations under parallel compositions. Moreover, this set is trivially closed under sequential composition because we have obviously $in(sub(C)) = C = out(sub(C))$ such that the only defined sequential composition is $proc(C) \circ proc(C)$ and yields $proc(C)$. Another significant operational closure is considered in the next subsection.

**Binary Relations.** Let $K$ be a set of configurations like strings, trees, or graphs. Then the subsets of $K \times K$ can be considered as semantic entities describing, for example, input/output relations.

There is always the sequential composition $R \circ R'$ of relations $R, R' \subseteq K \times K$ given by $R \circ R' = \{(x,z) \mid (x,y) \in R, (y,z) \in R' \text{ for some } y \in K\}$.

If $K$ has got some binary operation $\cdot : K \times K \to K$, this gives rise to a parallel composition $R \parallel R'$ given by $R \parallel R' = \{(x \cdot x', z \cdot z') \mid (x,z) \in R, (x',z') \in R'\}$.

A typical example is the concatenation of strings if $K$ is the set $A^*$ of all strings over an alphabet $A$. In this case, we also get an interesting unary operation *context* that embeds a given relation $R$ into all possible contexts, i.e. $context(R) = \{(xuy, xvy) \mid (u,v) \in R, x, y \in A^*\}$.

## 2.2 Rule Bases

A rule base equips a semantic domain with rules as a first syntactic feature. A rule provides a semantic entity describing basic computations.

A *rule base* $\mathcal{DR} = (X, OP; \mathcal{R}, \Longrightarrow)$ consists of a semantic domain $(X, OP)$, a class of *rules* $\mathcal{R}$, and a *rule application operator* $\Longrightarrow$ being a mapping $\Longrightarrow: \mathcal{R} \to 2^X$ which assigns a semantic entity $\underset{r}{\Longrightarrow} \in 2^X$ to each $r \in \mathcal{R}$.

As a rule specifies a semantic entity, a set of rules, $P \subseteq \mathcal{R}$, provides a set of semantic entities, $\{\underset{r}{\Longrightarrow} \mid r \in P\}$, which can be closed under the operations of the semantic domain. Accordingly, we denote $OP^*(\{\underset{r}{\Longrightarrow} \mid r \in P\})$ by $OP^*(P)$ for short. In this way, a set of rules $P$ specifies a semantic entity $\bigcup OP^*(P)$, which contains all semantic items that are obtained by the operational closure of all applications of rules in $P$.

### Examples

**Elementary Net Systems.** The events of $\overline{N}$ may be considered as rules. Each event $e \in \overline{E}$ induces a basic process $proc(e)$ such that the singleton set $\{proc(e)\}$ is a suitable semantic interpretation of an event as a rule. In other words, there is a rule base $D\mathcal{R}(\overline{N}) = (PROC(\overline{N}), OP(\overline{N}); \overline{E}, Proc : \overline{E} \to 2^{PROC(\overline{N})})$ with $Proc(e) = \{proc(e)\}$ for all $e \in \overline{E}$.

As $proc(e) \in PROC(\overline{N})$ for all $e \in \overline{E}$ and as the processes on $\overline{N}$ are closed under the operations in $OP(\overline{N})$, we get

$$\bigcup OP(\overline{N})^*(Proc(\overline{E})) \subseteq PROC(\overline{N})$$

for $Proc(\overline{E}) = \{Proc(e) \mid e \in \overline{E}\}$.

Conversely, let $proc = (N, p)$ with $N = (B, E, F)$ be a process on $\overline{N}$. If $E = \emptyset$, then $proc$ equals $proc(B)$ which is the parallel composition of all $\hat{c}$ for $c \in p(B)$.

For $E \neq \emptyset$, we show by induction on the number of elements in $E$ that $proc \in \bigcup OP(\overline{N})^*(Proc(\overline{E}))$.

If $E = \{e\}$, then $proc$ is the parallel composition of $proc(p(e))$ with all $\hat{c}$ for $c \in p(B) - in(sub(p(e))$. This case can be used as induction base.

If $E$ has more than one element, then it is well-known that *proc* is the sequential composition of two subnet processes $proc_i = (N_i, p_i)$ with $N_i = (B_i, E_i, F_i)$ for $i = 1, 2$ and $E_1 \neq \emptyset \neq E_2$. In particular, $E_1$ and $E_2$ are smaller sets than $E$ such that we may assume by induction that $proc_1$ and $proc_2$ are in the operational closure of $Proc(\overline{E})$. Because of $proc = proc_1 \circ proc_2$, *proc* is also in the closure.

Altogether, we have proved

$$\bigcup OP(\overline{N})^*(Proc(\overline{E})) = PROC(\overline{N}).$$

**Binary Relations.** Grammatical rules and all rules like these can be applied to some kind of configurations and derive configurations from them. Such a rule provides one with a binary relation of configurations the elements of which are often called direct derivations or computation steps.

A well-known explicit example of this type is the rule of a semi-Thue system or Chomsky grammar $p = (u, v)$ for $u, v \in A^*$ and some alphabet $A$. This rule specifies a binary relation $\underset{p}{\longrightarrow} \subseteq A^* \times A^*$ which is defined in infix notation by

$$xuy \underset{p}{\longrightarrow} xvy \text{ for all } \mathrm{x, y} \in \mathrm{A}^*.$$

Similarly, all kinds of graph transformation rules define a binary relation on the proper kinds of graphs by means of direct derivations.

If a rule $r$ is composed of a pair $(L, R)$ of configurations as in the case of the rules $(u, v)$ with $u, v \in A^*$, then there is a simple alternative to the relation of direct derivations. This is the singleton set $simple(r) = \{(L, R)\}$.

Let us first consider the rule bases $\mathcal{DR}_1(A) = (A^* \times A^*, OP_1; A^* \times A^*, \longrightarrow)$ with $OP_1 = \{\circ\}$ and $\mathcal{DR}_i(A) = (A^* \times A^*, OP_i; A^* \times A^*, simple)$ for $i = 2, 3$ with $OP_2 = OP_1 \cup \{context\}$ and $OP_3 = OP_2 \cup \{\|\}$.

Then the following holds for a set of rules, $P \subseteq A^* \times A^*$ :

$$\bigcup OP_1^*(P) = \bigcup OP_2^*(P) = \bigcup OP_3^*(P).$$

Note that the first operational closure is done for the direct derivations of $P$ while the other two start from the simple relations $simple(p)$ for $p \in P$. The first equality follows from the obvious fact that $context(simple(p)) = \underset{p}{\longrightarrow}$ for all $p = (u, v) \in P$. The second follows from the well-known fact that the parallel composition can be expressed by context embeddings and sequential composition, i.e.

$$(xx', zz') = (xx', zx') \circ (zx', zz'.)$$

Given a set $P$ of rules, the derivability relation $\overset{*}{\underset{p}{\longrightarrow}}$ is the sequential closure of all applications of rules in $P$, i.e. $\{\circ\}^*(\underset{P}{\longrightarrow})$ with $\underset{P}{\longrightarrow} = \bigcup\limits_{p \in P} \underset{p}{\longrightarrow}$ which equals $\bigcup OP_1^*(P)$. In other words, all three rules bases $\mathcal{DR}_i(A)$ for $i = 1, 2, 3$ describe sequential derivability through sets of rules.

### 2.3 Rule Bases with Control Conditions

A rule base may be additionally equipped with control conditions that allow to regulate computations. For this purpose, its semantics depends on the environment given by semantic entities for a set of identifiers.

A *rule base with control conditions* $\mathcal{DRC} = (X, OP; \mathcal{R}, \Longrightarrow; ID, \mathcal{C}, SEM)$ consists of a rule base $(X, OP; \mathcal{R}, \Longrightarrow)$, a set $ID$ of *identifiers*, a class of *control conditions* $\mathcal{C}$, and a semantic interpretation $SEM$ which associates each condition $C \in \mathcal{C}$ and each semantic mapping $Env : ID \to 2^X$, called *environment*, with a semantic entity $SEM(C, Env) \in 2^X$.

Depending on the environment $Env$, a control condition $C$ can be used to restrict the operational closure of a set $M$ of semantic entities by means of the intersection $\bigcup OP^*(M) \cap SEM(C, Env)$.

#### Examples

**Elementary Net Systems.** An elementary net becomes an elementary net system if an initial case is added. The idea of an initial case is that semantically only processes starting in this case are considered. Initial cases are typical examples of control conditions. Let $C_{in} \subseteq \overline{B}$ be some initial case. Then its semantics $SEM(C_{in})$ consists of all processes $(N, p)$ with $p(in(N)) = C_{in}$. Therefore, the process semantics $PROC((\overline{B}, \overline{E}, \overline{F}, C_{in}))$ of the elementary net system $(\overline{B}, \overline{E}, \overline{F}, C_{in})$ coincides with the intersection of the operational closure $\bigcup OP(\overline{N})^*(Proc(\overline{E}))$ and $SEM(C_{in})$, i.e.

$$PROC((\overline{B}, \overline{E}, \overline{F}, C_{in})) = (\bigcup OP(\overline{N})^*(Proc(\overline{E}))) \cap SEM(C_{in}).$$

Let $ID$ be a set of identifiers and $Env : ID \to 2^X$ some semantic mapping. Then the semantics of an initial case can be extended to the environment $Env$ in a trivial way as a case does not refer to any identifier:

$$SEM(C_{in}, Env) = SEM(C_{in}).$$

Each elementary net $\overline{N} = (\overline{B}, \overline{E}, \overline{F})$ induces a rule base with cases as control conditions $\mathcal{DRC}(\overline{N}) = (PROC(\overline{N}), OP(\overline{N}); \overline{E}, Proc; ID, 2^{\overline{B}}, SEM)$ where $SEM$ is defined as above. We have shown that this rule base describes elementary net systems with their processes starting in the initial case as semantics by using the events as rules and the initial cases as control conditions. In 2.4 the notion of a basic transformation unit is introduced as a syntactic modelling concept that allows one the use of rules and control conditions explicitly.

**Binary Relations.** In grammatical systems, the most frequently used kind of control condition is the choice of start symbols or some other configurations as axioms to begin derivations and the choice of terminal alphabets to describe the configurations at which derivations may end. For example, given an alphabet $A$, each pair $(S, T)$ with $T \subseteq A$ and $S \in A \setminus T$ specifies a binary

relation $SEM((S,T)) = \{S\} \times T^* \subseteq A^* \times A^*$. The intersection of this relation with the derivability relation of a set of rules, $P \subseteq A^* \times A^*$, contains pairs $(S,w)$ where $S$ derives $w$ and $w$ is terminal. In other words, the intersection $(\bigcup OP_1^*(P)) \cap SEM((S,T)) = \xrightarrow[P]{*} \cap (\{S\} \times T^*)$ represents the generated language of the Chomsky grammars $G = (N,T,P,S)$ with $N = A \setminus T$ in a unique way. This type of control condition is independent of any environment in the same way as initial cases above:

$$SEM((S,T),Env) = SEM((S,T))$$

for all $Env : ID \to 2^{A^* \times A^*}$ where $ID$ is some set of identifiers. The same remains true if $S$ is replaced by an arbitrary start word or axiom $z \in A^*$.

In other words, we may extend the rule bases $\mathcal{DR}_i(A)$ for $i = 1,2,3$ into rule bases with control conditions:

$$\mathcal{DRC}_i(A) = (\mathcal{DR}_i(A); ID, A^* \times 2^A, SEM).$$

As shown above, they allow one to describe Chomsky grammars and their generated languages within our framework.

Using a more sophisticated type of control conditions, one can also specify Lindenmayer systems (see, e.g., [10, 11]) as grammatical systems with a massively parallel mode of rewriting. We introduce this mode in a general way to demonstrate the role of identifiers and environments.

Let $M \subseteq 2^{A^* \times A^*}$ be a set of binary relations on $A^*$, which may be some kind of basic computations. Then the sequential closure of the parallel closure of $M, \{\circ\}^*(\{\|\}^*(M))$, describes massive parallelism on the semantic level as each step of a sequence consists just of parallel computations. To express this on the syntactic level of control conditions, one needs access to the members of $M$ for which we offer two ways. The first possibility is given by a set $P \subseteq \mathcal{R}$ of rules and the second one by a set $U \subseteq ID$ of identifiers together with an environment. Formally, we introduce the control condition $mp(P,U)$ with

$$SEM(mp(P,U),Env) = \{\circ\}^*(\{\|\}^*(\{simple(p) \mid p \in P\} \cup \{Env(id) \mid id \in U\})),$$

where $mp$ refers to the term *massive parallelism*. If $P$ is a set of context-free rules, i.e. $P \subseteq A \times A^*$, and $U$ is empty, $mp(P,\emptyset)$ describes the derivation mode of 0L systems.

In order to combine massive parallelism explicitly with the rule based features for binary relations, one may consider the rule bases with control conditions $\mathcal{DRC}_i(A, mp)$ for $i = 1,2,3$ which are obtained from $\mathcal{DRC}_i(A)$ by adding the control conditions $\{mp(P,U) \mid P \subseteq \mathcal{R}, U \subseteq ID\}$ with $SEM(mp(P,U),Env)$ as defined above and the combined control conditions $A^* \times 2^A \times \{mp(P,U) \mid P \subseteq \mathcal{R}, U \subseteq ID\}$ with $SEM((z,T,mp(P,U)),Env) = SEM(z,T) \cap SEM(mp(P,U), Env)$.

Instead of massive parallelism, there are other derivation modes that allow one to control the application of grammatical rules. Further typical examples are $\leq k$ $(= k, \geq k)$ for some $k \in \mathbb{N}$ requiring that the number of rule applications

for a given set of rules is not greater than $k$ (equals $k$, is not less than $k$) and $t$ (for terminating) requiring that the given rules are applied as long as possible (see, e.g., [1] for more details).

### 2.4 Transformation Units

A rule base provides the computational framework in which rule-based specifications can be defined. The most elementary kind of such a specification in our framework is a transformation unit that comprises a local set of rules, a set of identifiers, and a control condition. The identifiers refer to used or imported components. The control condition regulates the interaction of rules and imported components.

Let $\mathcal{DRC} = (X, OP; \mathcal{R}, \Longrightarrow; ID, \mathcal{C}, SEM)$ be an arbitrary, but fixed rule base with control conditions. Then a *transformation unit* (over $\mathcal{DRC}$) is a system $tu = (P, U, C)$ where $P \subseteq \mathcal{R}$ is a finite set of rules, $U \subseteq ID$ is a finite set of identifiers, which is called the *use* or *import interface*, and $C \in \mathcal{C}$ is a control condition.

The unit is called *basic* if $U$ is empty.

Examples are discussed together with the interlinking semantics at the end of the next subsection.

### 2.5 Interlinking Semantics of Transformation Units

Given a semantic entity for each import identifier, i.e. a mapping $Imp : U \to 2^X$, the transformation unit $tu$ specifies a semantic entity which is constructed as the operational closure of the semantic entities given by the local rules and the import as far as it meets the control condition. Because the rules and the import are interlinked with each other through the semantic operations, the resulting semantic entity is called *interlinking semantics* which is formally defined as follows:

$$INTER_{Imp}(tu) = (\bigcup OP^*(P, Imp)) \cap SEM(C, Imp_+)$$

where $OP^*(P, Imp)$ is the operational closure of the semantic entities given by the rules and the import mapping, i.e.

$$OP^*(P, Imp) = OP^*(\{\underset{r}{\Longrightarrow} \mid r \in P\} \cup \{Imp(id) \mid id \in U\}),$$

and where the environment $Imp_+ : ID \to 2^X$ is the trivial extension of $Imp$ to $ID$, i.e. $Imp_+(id) = Imp(id)$ for $id \in U$ and $Imp_+(id) = \emptyset$ otherwise.

Altogether, the interlinking semantics interlinks the semantic effects of the local rules of the transformation unit with the imported semantic entities according to the control condition. It should be noted that the notion of interlinking semantics of transformation units generalizes the interleaving semantics introduced in [5, 6]. The interleaving semantics concerns binary relations on graphs or configurations resp. as semantic entities and the sequential composition of binary relations as only semantic operator.

**Examples**

In this subsection, only examples of basic transformation units are presented. Examples of units with import can be found in 3.2 and 3.3.

**Elementary Net Systems.** Consider the rule base with control conditions $\mathcal{DRC}(\overline{N}) = (PROC(\overline{N}), OP(\overline{N}); \overline{E}, Proc; ID, 2^{\overline{B}}, SEM)$ for a given elementary net $\overline{N} = (\overline{B}, \overline{E}, \overline{F})$. Then an elementary net system $(N, C_{in})$ with a subnet $N = (B, E, F)$ of $\overline{N}$ and an initial case $C_{in}$ can be interpreted as a basic transformation unit $tu(N, C_{in}) = (E, \emptyset, C_{in})$ such that the process semantics of $(N, C_{in})$ coincides with the interlinking semantics of $tu(N, C_{in})$ using the empty mapping $Empty : \emptyset \to 2^X$ as the only import mapping.

$$PROC(N, C_{in}) = INTER_{Empty}(tu(N, C_{in})).$$

This follows directly from the definition of the interlinking semantics and the considerations in 2.3.

Conversely, a basic transformation unit $tu = (E, \emptyset, C_{in})$ induces an elementary net system $N(tu) = (\overline{B}, E, F(tu), C_{in})$ with $F(tu) = \overline{F} \cap ((\overline{B} \times E) \cup (E \times \overline{B}))$ where the underlying net is the subnet of $\overline{N}$ induced by $E$.

**Binary Relations.** Consider the rule base with control condition $\mathcal{DRC}_1(A)$ as given in 2.3. It is shown there that the language $L(G) = \{w \in T^* \mid S \xrightarrow[P]{*} w\}$ generated by the Chomsky grammar $G = (N, T, P, S)$ corresponds one-to-one to the binary relation $(\bigcup OP_1^*(P)) \cap SEM((S, T)) = \xrightarrow[P]{*} \cap (\{S\} \times T^*)$. In other words, the grammar $G$ gives rise to the basic transformation unit $tu(G) = (P, \emptyset, (S, T))$ over $\mathcal{DRC}_1(A)$ such that its interlinking semantics coincides with the generated language $L(G)$ up to representation. According to 2.2, this remains true if $\mathcal{DRC}_1(A)$ is replaced by $\mathcal{DRC}_2(A)$ or $\mathcal{DRC}_3(A)$.

Similarly, many other kinds of grammars, like for example tree and graph grammars, can be seen as basic transformation units such that the generated languages correspond to the interlinking semantics if one chooses the set of configurations, the set of rules, the rule application operator, single configurations as axioms and terminal configurations properly.

As a Chomsky grammar, an 0L system $G'=(A, P, z)$ with $P \subseteq A \times A^*$ and $z \in A^*$ can be modelled as the basic transformation unit $tu(G') = (P, \emptyset, (z, A, mp(P, \emptyset)))$ over one of the rule bases with control conditions $\mathcal{DRC}_i(A, mp)$ for $i = 1, 2, 3$ such that the generated language $L(G')$ corresponds again to the interlinking semantics of $tu(G')$.

### 2.6 Monotony of the Interlinking Semantics

The interlinking semantics depends on the imported semantic entities. If they are replaced by larger sets, the environment of a transformation unit increases automatically. The interlinking semantics and the operational closure are also increasing if the control condition and the operations are monotone. This helps to show in the following sections that the interlinking semantics is a fixed-point semantics.

An operation $op \in OP_k$ for some $k$ is *monotone* if $op(t_1, \ldots, t_k) \subseteq op(t'_1, \ldots, t'_k)$ for all $t_i, t'_i \in 2^X$ with $t_i \subseteq t'_i$ and $i = 1, \ldots, k$. Accordingly, a set of operations is *monotone* if each of its elements is monotone. A control condition $C \in \mathcal{C}$ is *monotone* if $SEM(C, Env) \subseteq SEM(C, Env')$ for all environments $Env, Env' : ID \to 2^X$ with $Env(id) \subseteq Env'(id)$ for all $id \in ID$.

**Observation 1** *Let $tu = (P, U, C)$ be a transformation unit over a rule base with control conditions $\mathcal{DRC} = (X, OP; \mathcal{R}, \Longrightarrow; ID, \mathcal{C}, SEM)$, and let $Imp, Imp' : U \to 2^X$ be import mappings with $Imp \subseteq Imp'$. Then the following hold:*

*(1) $Imp_+ \subseteq Imp'_+$.*
*(2) $\bigcup OP^*(P, Imp) \subseteq \bigcup OP^*(P, Imp')$ provided that $OP$ is monotone.*
*(3) $INTER_{Imp}(tu) \subseteq INTER_{Imp'}(tu)$ provided that $C$ is monotone in addition.*

**Examples**

If an operation on the powerset of a set $X$ is the natural elementwise extension of an operation on the underlying set $X$, then the extension is obviously monotone. All operations considered for elementary net systems and binary relations are of this kind. Moreover, all control conditions considered in the examples are monotone because they control the composition of semantic entities independent of their size such that the results get larger if the arguments are replaced by larger sets.

## 3 Networks of Transformation Units with Iterated Interlinking Semantics

A transformation unit is a rule-based system with a generic import. An import identifier represents a semantic entity, but it is not fixed how it is specified. A simple way to specify the import is to assume that the identifiers name again transformation units. In this case the import structure forms a directed graph leading to the notion of a network of transformation units. If the network is finite and acyclic or if the network has no infinite path, the interlinking semantics can be defined for all transformation units in the network. If the network has a cycle or an infinite path, one may start with the empty semantic entity for each transformation unit and then iterate the interlinking semantics ad infinitum.

### 3.1 Networks of Transformation Units

A *network of transformation units* over a rule base with control conditions $\mathcal{DRC} = (X, OP; \mathcal{R}, \Longrightarrow; ID, \mathcal{C}, SEM)$, is a system $N = (V, \tau)$ where $V$ is a set of nodes and $\tau$ is a mapping assigning a transformation unit $\tau(v) = (P(v), U(v), C(v))$ to each node $v \in V$ with $U(v) \subseteq V$.

A network of transformation units can be seen as a directed graph where the elements of $V$ are the nodes and the ordered pairs of nodes $(v, v')$ with

$v' \in U(v)$ the edges. A network of transformation units is *well-founded* if it does not contain an infinite path.

Note that a finite network is well-founded if and only if it is acyclic. In case of a well-founded network, the set of nodes can be divided into pairwise disjoint *levels* $V_k$ for $k \in \mathbb{N}$ which are inductively defined by

(i) $V_0$ containing all nodes with basic units and

(ii) $V_{k+1}$ containing all nodes $u \in V \setminus \bigcup_{i=0}^{k} V_i$ with $U(v) \subseteq \bigcup_{i=0}^{k} V_i$.

### 3.2 Interlinking Semantics of Well-Founded Networks

The interlinking semantics of transformation units is easily extended to well-founded networks because it can be defined inductively level by level.

Let $N = (V, \tau)$ be a well-founded network of transformation units. Then the interlinking semantics $INTER : V \to 2^X$ is inductively defined in the following way:

(1) for $v \in V_0$, we have $U(v) = \emptyset$ such that the empty mapping *Empty* is the only choice for the semantics of the import part. Therefore, the interlinking semantics of $tu(v)$ with *Empty* is defined yielding

$$INTER(v) = INTER_{Empty}(tu(v)).$$

(2) Let us assume that $INTER(v)$ is defined for all $v' \in \bigcup_{i=0}^{k} V_k$ for some $k \in \mathbb{N}$.

(3) Consider $v \in V_{k+1}$. Then we have $U(v) \subseteq \bigcup_{i=0}^{k} V_k$ such that $Imp_k(v') = INTER(v')$ is defined for all $v' \in U(v)$. Therefore, the interlinking semantics of $tu(v)$ with $Imp_k$ is defined yielding

$$INTER(v) = INTER_{Imp_k}(tu(v)).$$

**Examples**

The first examples of networks of transformation units are well-founded and have all the same simple structure with $n$ nodes $v_1, \ldots, v_n$ at level 0 and one node $v_0$ at level 1, i.e. there is a main unit importing the other units. The examples differ only in the choices of transformation units for the nodes.

**Elementary Net Systems.** In 2.5, an elementary net system $(N, C_{in})$ with $N = (B, E, F)$ is transformed into the basic transformation unit $tu(N, C_{in})$. But as each event gives rise to a basic transformation unit separately with the event as the only rule, the elementary net system can be seen as a transformation unit that imports its event units, i.e. $\tau(N, C_{in})(v_0) = (\emptyset, \{v_1, \ldots, v_n\}, C_{in})$ and $\tau(N, C_{in})(v_i) = tu(e_i) = (\{e_i\}, \emptyset, all)$ for $i = 1, \ldots, n$ and $E = \{e_1, \ldots, e_n\}$. The control condition *all* is a void condition the semantics of which is always the set of all processes so that the intersection with any other semantic entity has no

effect. Hence, the interlinking semantics of $tu(e_i)$ is the closure of the process $proc(e_i)$ and the processes $proc(c)$ for each $c \in \overline{B}$ under parallel and sequential composition. Because the sequential composition with $proc(C)$ for $C \subseteq \overline{B}$ has no effect and the event $e_i$ is not enabled directly after its occurrence, the sequential compositions can be ignored, and one gets as interlinking semantics of $tu(e_i)$ all cases and all single occurrences of $e_i$, i.e.

$$INTER(v_i) = INTER_{Empty}(tu(e_i)) = 2^{\overline{B}} \cup \{proc(e_i) + C \mid C \subseteq \overline{B} \setminus ({}^\bullet e_i \cup e_i^\bullet)\}.$$

This provides also the import mapping $Imp_0$ for the interlinking semantics of the level-1 node $v_0$ consisting of all sequential and parallel compositions of the imported processes that start with $C_{in}$ yielding the process semantics of the elementary net system $(N, C_{in})$, i.e.

$$INTER(v_0) = INTER_{Imp_0}(\tau(N, C_{in})(v_0)) = PROC(N, C_{in}).$$

**Binary Relations.** Grammar systems (see, e.g., [1]) are typical examples of the same form. The system becomes the main unit, and its components are imported. More formally, a *cooperating distributed grammar system* $\Gamma{=}(N, T, S, P_1, \ldots, P_n)$ consists of a set of *nonterminals* $N$, a set of *terminals* $T$, a *start symbol* $S \in N$, and a collection of finite sets of rules $P_1, \ldots, P_n$ with $P_i \subseteq (N \cup T)^* \times (N \cup T)^*$ for $i = 1, \ldots, n$. Choosing a derivation mode $f$ (according to examples in 2.3), $\Gamma$ generates the language $L_f(\Gamma)$ which contains all terminal words $w$ that are derived from the start symbol $S$ by a sequence of derivations in the mode $f$ of the form

$$S \xrightarrow[P_{i_1}]{f} w_1 \xrightarrow[P_{i_2}]{f} \ldots \xrightarrow[P_{i_m}]{f} w_m = w$$

with $m \geq 1$ and $1 \leq i_j \leq n$ for $j = 1, \ldots, m$. The corresponding transformation units are defined by $\tau(\Gamma)(v_0) = (\emptyset, \{v_1, \ldots, v_n\}, (S, T))$ and $\tau(\Gamma)(v_i) = (P_i, \emptyset, f)$ for $i = 1, \ldots, n$. The interlinking semantics of the latter units coincides obviously with the derivation relations with respect to the derivation mode $f$. And the interlinking semantics of $\tau(\Gamma)(v_0)$ imports these, constructs the closure under the operations including sequential composition, and intersects the result with $\{S\} \times T^*$ due to the control condition. Consequently, a word $w$ is in interlinking relation to $S$ if and only if $w \in L_f(\Gamma)$.

Similarly, a T0L system $G'' = (A, P_1 \ldots, P_n, z)$ with an *alphabet* $A$, a *start word* $z \in A^*$ and a collection of finite sets of context-free rules $P_i \subseteq A \times A^*$ gives rise to a network of transformation unit. The network structure is the same as in the last two examples, and the transformation units of the nodes are defined by $\tau(G'')(v_0) = (\emptyset, \{v_1, \ldots, v_n\}, (z, A))$ and $\tau(G'')(v_i) = (P_i, \emptyset, mp(P_i, \emptyset))$ for $i = 1, \ldots, n$. Accordingly, the language generated by $G''$ corresponds to the interlinking semantics of the root node $v_0$.

### 3.3 Iterated Interlinking Semantics of Arbitrary Networks

The problem of networks which are not well-founded is that the semantics of the import parts of some units at the network may not be known at the moment

when one wants to apply the interlinking semantics. But if one assumes to have at least a preliminary semantics for all nodes, i.e. some semantic mapping $Sem : V \to 2^X$, then the interleaving semantics is defined for the unit of each node wrt to $Sem$ restricted to the import part, i.e. $Sem'(v) = INTER_{Sem(U(v))}(tu(v))$ for $v \in V$ with $Sem(U(v))(v') = Sem(v')$ for all $v' \in U(v)$.

The resulting semantic mapping $Sem'$ is denoted by $INTER(Sem)$, and the operator $INTER$, that yields semantic mappings from semantic mappings by interlinking them with the semantic entities of the respective rules, is called *interlinking operator*.

In this way, one gets another semantic mapping, which may be used as a next preliminary semantics such that this process can be iterated ad infinitum whenever one starts from some semantic mapping. An obvious candidate to start is the mapping that assigns the empty set to each node of the network. Therefore, the *iterated interlinking semantics* $ITERATE : V \to 2^X$ of a network of transformation units $N = (V, \tau)$ may be defined for all $v \in V$ as follows:

$$ITERATE(v) = \bigcup_{i \in \mathbb{N}} ITERATE_i(v)$$

with $ITERATE_0(v) = \emptyset$ and $ITERATE_{i+1}(v) = INTER_{ITERATE_i(U(v))}(tu(v))$.

**Examples**

The concept of networks of transformation units beyond the examples in 3.2 provide new possibilities of cooperation and distribution in the framework of elementary net systems and of grammar systems. The examples of 3.2 may be reconsidered. Instead of a main unit which imports all others, the main unit imports only one of the other units which import each other. While the main unit takes care of the global control condition only, the other units do the computational work interactively.

**Elementary Net Systems.** Let $(N, C_{in})$ be an elementary net system, let $E = \{e_1, \ldots, e_n\}$ be its set of events, and let $V = \{v_1, \ldots, v_n\}$. Then the second network $\hat{\tau}(N, C_{in})$ associated to $(N, C_{in})$ is given by $\hat{\tau}(N, C_{in})(v_0) = (\emptyset, \{v_1\}, C_{in})$ and $\hat{\tau}(N, C_{in})(v_i) = (\{e_i\}, V, all)$ for $i = 1, \ldots, n$. Starting with the empty set of processes at each node, the first application of the interlinking operator yields the singleton set $\{C_{in}\}$ as semantics of $v_0$ and all cases and all single occurrences of the event $e_i$ as semantics of $v_i$ for $i = 1, \ldots, n$. As the import is empty at the first step, the unit at $v_i$ behaves as $tu(e_i)$. The second application of the interlinking operator yields the set of all processes as semantics at $v_i$ for $i = 1, \ldots, n$ as all single occurrences of all events are imported and closed under sequential and parallel composition. The semantics at $v_0$ contains, besides $C_{in}$, the sequential compositions of $C_{in}$ with single occurrences of $e_1$, i.e. the single occurrence of $e_1$ under $C_{in}$ if ${}^\bullet e_1 \subseteq C_{in}$. The third application of the interlinking operator yields the processes of $N$ that start in $C_{in}$ at the node $v_0$. The other semantic entities are kept. Further iteration is not changing the semantics.

**Binary Relations.** Analogously, grammar systems and T0L systems can be reconstructed as networks of transformation units of the given form.

The same remains true if the subnetwork induced by $v_1, \ldots, v_n$ is not complete, but there is a path from each node to $v_1$. In this case, the interlinking operator must be iterated $m + 2$ times if $m$ is the length of the longest shortest path of a node $v_i$ to $v_1$ for $i = 1, \ldots, n$.

### 3.4 Iterated Interlinking Semantics of Well-Founded Networks

If one applies the iterated interlinking semantics to well-founded networks, the result coincides with the ordinary interlinking semantics. This is a first indication that the interlinking semantics is meaningful.

**Observation 2** *Let $N = (V, \tau)$ be a well-founded network of transformation units. Then we have*

$$INTER = ITERATE.$$

### 3.5 Monotony and Continuity of the Interlinking Operator

Given a semantic entity for each node of a network of transformation units, the interlinking semantics is defined for each node yielding another semantic entity. This is the basic operator which is iterated in the iterated interlinking semantics. This operator turns out to be monotone and even continuous if the used operations and control conditions are monotone resp. continuous.

An operation $op \in OP_k$ for some $k$ is *continuous* if

$$\bigcup_{i \in \mathbb{N}} op(t_1, \ldots, t_{j-1}, t_{j_i}, t_{j+1}, \ldots, t_k) = op(t_1, \ldots, t_{j-1}, \bigcup_{i \in \mathbb{N}} t_{j_i}, t_{j+1}, \ldots, t_k).$$

for each $j$ with $1 \leq j \leq k$ and each increasing chains of semantic entities $t_{j_0} \subseteq t_{j_1} \subseteq \ldots \subseteq t_{j_i} \subseteq \ldots$. Accordingly, a set of operations is *continuous* if each of its elements is continuous.

A control condition $C$ is *continuous* if

$$\bigcup_{i \in \mathbb{N}} SEM(C, Env_i) = SEM(C, \bigcup_{i \in \mathbb{N}} Env_i)$$

for each increasing chain of environments $Env_0 \subseteq Env_1 \subseteq \ldots \subseteq Env_i \subseteq \ldots$ with $Env_i : ID \to 2^X$ for $i \in \mathbb{N}$.

#### Examples

All operations in the examples of this paper are operations on semantic items, which are extended elementwise to semantic entities. Such operations are obviously monotone and continuous. The same applies to the control conditions as they restrict the application of semantic operators independent of the content of the semantic entities.

**Observation 3** *Let $\mathcal{DRC} = (X, OP; \mathcal{R}, \Longrightarrow; ID, \mathcal{C}, SEM)$ be a rule base with control conditions and $N = (V, \tau)$ be a network of transformation units over $\mathcal{DRC}$ such that $OP$ and $C(v)$ for all $v \in V$ are continuous. Let $INTER$ be the corresponding interlinking operator. Then the following hold:*

*(1) The interlinking operator is monotone, i.e.*

$$INTER(Sem) \subseteq INTER(Sem')$$

*for all semantic mappings $Sem, Sem' : V \to 2^X$ with $Sem \subseteq Sem'$.*

*(2) The interlinking operator is continuous, i.e.*

$$\bigcup_{i \in \mathbb{N}} INTER(Sem_i) = INTER(\bigcup_{i \in \mathbb{N}} Sem_i)$$

*for all increasing chains of semantic mappings $Sem_0 \subseteq Sem_1 \subseteq \ldots \subseteq Sem_i \subseteq \ldots$.*

## 4 Fixed-Point Theorem

Let $N = (V, \tau)$, be a network of transformation units $N = (V, \tau)$, and $INTER$ the corresponding interlinking operator on the semantics mappings $Sem : V \to 2^X$ defined by

$$INTER(Sem)(v) = INTER_{Sem(U(v))}(tu(v))$$

for all $v \in V$. Because their domain is a power set, it is a well-known fact that the set of semantic mappings is a complete partial order with respect to the argumentwise inclusion where the union of every increasing chain is the least upper bound. In Observation 3, the interlinking operator is shown to be monotone and continuous with respect to this complete partial order such that Kleene's fixed-point theorem applies. This means that the iterated interlinking semantics is the least fixed point of the interlinking operator.

**Theorem 1.** *Let $\mathcal{DRC} = (X, OP; \mathcal{R}, \Longrightarrow; ID, \mathcal{C}, SEM)$ be a rule base with control conditions, and let $N = (V, \tau)$ be a network of transformation units over $\mathcal{DRC}$ such that each operation is continuous and $C(v)$ as well for each $v \in V$. Then the iterated interlinking semantics $ITERATE : V \to 2^X$ is the least fixed point of the interlinking operator $INTER$, i.e.*

$$INTER(ITERATE) = ITERATE.$$

It should be noted that this result generalizes the fixed-point theorem in [7], which deals with binary relations on some set of graphs as semantic entities and with the sequential composition of binary relations as the only semantic operator.

## 5 Conclusion

In this paper, we have introduced the notion of interlinking semantics of networks of transition units generalizing the purely sequential interleaving semantics of earlier work. We have demonstrated that the new concept covers parallelism and concurrency of elementary net systems and various types of grammars. The main result is a fixed-point theorem stating that the iterated interlinking semantics is the smallest fixed-point of the interlinking operator.

Future work should shed some more light on the significance of this approach in two respects at least. On one hand, it should be investigated how the fixed-point theorem can be used to analyze rule-based systems and to prove their properties. On the other hand, further case studies would be helpful to fit in further approaches to parallelism and concurrency into our new framework. In particular, we would like to relate parallelism and concurrency in the area of graph transformation, which have been intensively investigated by Hartmut Ehrig and others in the last three decades (see, e.g., the Chapters 3 and 4 in [9] and [2] for an overview), with interlinking semantics.

## Acknowledgement

We would like to thank Peter Knirsch and Gabriele Taentzer for the helpful comments on an earlier version of this paper.

## References

1. Jürgen Dassow, Gheorghe Păun and Grzegorz Rozenberg. Grammar systems. In G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages, Vol. 2.* pages 155–213, Springer, 1997.
2. Hartmut Ehrig, Hans-Jörg Kreowski, Ugo Montanari and Grzegorz Rozenberg, editors. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 3.* World Scientific, 1999.
3. Cesar Fernandez. Non-sequential processes. In W. Brauer, W. Reisig and G. Rozenberg, editors. *Petri nets: Central models and their properties, Advances in Petri nets, Part I. Lecture Notes in Computer Science*, vol. 254, pages 95–115, Springer, 1986.
4. Hans-Jörg Kreowski and Sabine Kuske. On the interleaving semantics of transformation units – A step into GRACE. In Janice E. Cuny, Hartmut Ehrig, Gregor Engels and Grzegorz Rozenberg, editors. *Proc. Graph Grammars and Their Application to Computer Science. Lecture Notes in Computer Science*, vol. 1073, pages 89–108, Springer, 1996.
5. Hans-Jörg Kreowski and Sabine Kuske. Graph transformation units with interleaving semantics. *Formal Aspects of Computing*, vol. 11, no. 6, pages 690–723, 1999.
6. Hans-Jörg Kreowski and Sabine Kuske. Approach-independent structuring concepts for rule-based systems. In Martin Wirsing, Dirk Pattison, Rolf Hennicker, editors. *Proc. 16th Int. Workshop on Algebraic Development Techniques (WADT 2002). Lecture Notes in Computer Science*, vol. 2755, pages 299–311, Springer, 2003.

7. Hans-Jörg Kreowski, Sabine Kuske and Andy Schürr. Nested graph transformation units, *International Journal on Software Engineering and Knowledge Engineering*, vol. 7, no. 4, pages 479–502, 1997.
8. Grzegorz Rozenberg. Behaviour of elementary net systems. In W. Brauer, W. Reisig and G. Rozenberg, editors. *Petri nets: Central models and their properties, Advances in Petri nets, Part I. Lecture Notes in Computer Science*, vol. 254, pages 60–94, Springer, 1986.
9. Grzegorz Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 1.* World Scientific, 1997.
10. Grzegorz Rozenberg and Arto Salomaa, editors. *The Book of L.* Springer, 1986.
11. Grzegorz Rozenberg and Arto Salomaa, editors. *Lindenmayer Systems.* Springer, 1992.
12. Grzegorz Rozenberg and Arto Salomaa, editors. *Handbook of Formal Languages, Vol. 1–3.* Springer, 1997.
13. R.S. Thiagarajan. Elementary net systems. In W. Brauer, W. Reisig and G. Rozenberg, editors. *Petri nets: Central models and their properties, Advances in Petri nets, Part I. Lecture Notes in Computer Science*, vol. 254, pages 26–59, Springer, 1986.