Relations on words and FIFO channel systems

Wojciech Fraczak

Université du Québec en Outaouais, Gatineau Québec J8X 3X7, Canada
fraczak@uqo.ca

Abstract. We reexamine the FIFO channel systems (also called Communicating Automata, Message-Passing Automata, or Data Flow Sequential Processes) as a virtual machine defining relations on words. Originally, the model consisted of a network of finite state machines communicating using reliable FIFO channels. The semantics of such a system was defined in terms of local state changes driven by the content of the FIFO channels. The model is simple, expressive, and apt to efficient distributed realizations.
In this paper we investigate a compositional semantics for FIFO channel systems in terms of word relations.

1 Introduction

The theory of formal languages and automata goes beyond languages seen as subsets of finitely generated free monoids, [7]. However, once the underlying monoid stops being free, the notion of a corresponding automaton and its semantics begins to be more complicated and less intuitive. In this paper we characterize a class of word relations in order to use it as a semantic domain for *FIFO channel systems*, [1]. We redefine the semantics of FIFO channel systems in terms of those relations making it fully compositional. Our semantics, however, is discriminative in the sense that it assumes that FIFO channel systems have a special property, namely, they are deadlock free. Intuitively, a FIFO channel system is deadlock free, if it never stops in the middle of producing its output. For example, Kahn's deterministic systems, as defined in [6], are a spacial case of deadlock free systems.

2 Words, tuples, languages, and relations

Let S be a finite set. By S^* we denote the set of all finite sequences of elements of S, called *words*. The empty word is denoted by ε. Let u and w be words. We write $u \cdot w$ (or uw) to denote the word concatenation and $|w|$ to denote the length of w.

By $(S^*)^n$, for a natural number n, we denote the set of all n-tuples (vectors of size n) of words over S. If $x = (w_1, \ldots, w_n) \in (S^*)^n$ and $y = (u_1, \ldots, u_m) \in (S^*)^m$, then by $x \times y$ (or $x; y$) we mean $(w_1, \ldots, w_n, u_1, \ldots, u_m) \in (S^*)^{n+m}$. If $x = (w_1, \ldots, w_n)$ and $y = (u_1, \ldots, u_n)$ are both in $(S^*)^n$, then $x \cdot y$ (or xy) denotes $(w_1u_1, \ldots, w_nu_n)$.

A set of words, i.e., a subset of S^*, is called a *language*. A set of tuples, i.e., a subset of $(S^*)^n$, is called a *relation*. Notice that a word is a special case of tuple and a language is a special case of relation. Operations of tuples extend to relations: $R_1 \cdot R_2 = \{xy \mid x \in R_1, y \in R_2\}$, $R_1 \times R_2 = \{x \times y \mid x \in R_1, y \in R_2\}$.

We say that a tuple u is a left factor of a tuple w, for $u, w \in (S^*)^n$, if there exists $x \in (S^*)^n$ such that $ux = w$. We will write $u \leq w$ to denote that u is a left factor (also called *prefix*) of w and $u \geq w$ to denote that w is a prefix of u.

Let $R \subseteq (S^*)^n$ and $i \in \{1, \ldots, n\}$. By $\downarrow_i R$ (respectively, $\uparrow_i R$) we denote the prefix (suffix[1]) closure of R on dimension i. I.e.:

$$\downarrow_i R \stackrel{\texttt{def}}{=} \big\{(u \times x \times v) \mid u \in (S^*)^{i-1}, x \in S^*, \exists y \geq x\ (u \times y \times v) \in R\big\}$$
$$\uparrow_i R \stackrel{\texttt{def}}{=} \big\{(u \times x \times v) \mid u \in (S^*)^{i-1}, x \in S^*, \exists y \leq x\ (u \times y \times v) \in R\big\}$$

Notice that $\downarrow_i\downarrow_j R = \downarrow_j\downarrow_i R$, $\uparrow_i\uparrow_j R = \uparrow_j\uparrow_i R$, for all $i, j \in \{1, \ldots, n\}$, and $\downarrow_i\uparrow_j R = \uparrow_j\downarrow_i R$, for all $i, j \in \{1, \ldots, n\}$ such that $i \neq j$. Thus, the notation $\uparrow_I\downarrow_J R$, where $I, J \subseteq \{1, \ldots, n\}$ and $I \cap J = \emptyset$ will be used to denote $(\uparrow_{i_1}\uparrow_{i_2} \ldots \downarrow_{j_1}\downarrow_{j_2} \ldots R)$ with $I = \{i_1, i_2, \ldots\}$ and $J = \{j_1, j_2, \ldots\}$.

Let $\mu : (\{1, \ldots, n\} \times S)^* \to (S^*)^n$ be the usual monoid homomorphism which maps every $(k, s) \in (\{1, \ldots, n\} \times S)$ into the tuple $(w_1, \ldots, w_n)$ of words over S with $w_k = s$ and $w_i = \varepsilon$, for $i \in \{1, \ldots, n\} \setminus \{k\}$.

Let $R \subseteq (S^*)^m$ and $i \in \{1, \ldots, m\}$. We define the hiding of dimension i in R, denoted by $\mathit{HIDE}(R, i)$, as:

$$\mathit{HIDE}(R, i) \stackrel{\texttt{def}}{=} \{u \times v \mid u \in (S^*)^{i-1}, v \in (S^*)^{m-i}, \exists x \in S^*, u \times x \times v \in L\}$$

By $\mathit{HIDE}(R, I)$ for $I \subseteq \{1, \ldots, m\}$ we denote simultaneous hiding of all dimensions from I.

3 FIFO components

Traditionally, a *FIFO channel system* is a collection of components (usually finite state machines) communicating with each other through order-preserving unbounded point-to-point channels called FIFO channels (for First-In-First-Out). In our modular presentation of FIFO channel systems, a component is characterized by: (1) an *interface*, i.e., two disjoint sets of *input* and *output ports*, which FIFO channels can connect to; and (2) a *word relation*, i.e., a set of tuples of words over *signals* read and written on the ports.

From an architectural (static) point of view, a FIFO channel consists of components whose ports are connected by channels. In Figure 1, the system C consists of three components C_1, C_2, and C_3. For example, component C_1 has one input port (numbered by 1) and two output ports (numbered by 2 and 3). In the figure, input ports are represented by black squares and output ports by

[1] The name may be misleading in the sense that the "suffix closure" has nothing to do with the suffix, i.e., right factor. By the suffix closure of a language $L \subseteq S^*$ we mean LS^* and by its prefix closure $L(S^*)^{-1}$.

empty squares. The unconnected ports of the components become the interface of the system. Such a system, in turn, can be used as a component in a definition of another system.

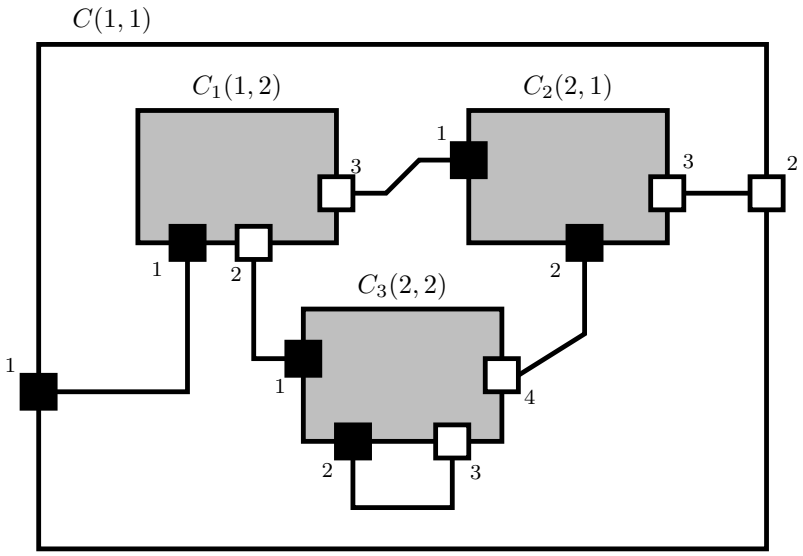

Fig. 1. A FIFO channel system architecture.

In order to simplify the notation we assume that the set of *signals*, S, which can occur on a port is the same for all ports. The semantics of a component C with m input ports and n output ports, denoted $C(m, n)$, is defined as a relation $C \subseteq (S^*)^{m+n}$. Intuitively, a tuple $(w_1, \ldots, w_m, u_1, \ldots, u_n)$ in C, means that by writing sequences of signals $w_1, \ldots, w_m$ on input ports $1, \ldots, m$, we may be able to read sequences of signals $u_1, \ldots, u_n$ on output ports $m+1, \ldots, m+n$, respectively.

We will write $(x; y) \in C$ in order to explicitly separate the tuple $x \times y \in C$ into the "writing tuple" $x \in (S^*)^m$ and the "reading tuple" $y \in (S^*)^n$, as we assume that the input ports are numbered from 1 to m and output ports from $m+1$ to $m+n$.

Definition 1 *A FIFO component C with $m \geq 0$ input ports and $n \geq 0$ output ports is a word relation $C \subseteq (S^*)^{m+n}$ such that:*

$$\uparrow_{\{1,\ldots,m\}} \downarrow_{\{m+1,\ldots,m+n\}} (C \cup \{\varepsilon^{m+n}\}) \;=\; C$$

In other words, a relation is a *FIFO component* if it is defined for the empty input, it is suffix-closed on input, and it is prefix-closed on output.

The above constraints capture the restrictions for modular implementations imposed by the real asynchronous world in the spirit of the Kahn's deterministic

processes, [6], and the Josephs's receptive and data flow processes, [4, 5]. The environment, i.e., a user of a component, can always send signals to the component (the component doesn't control what the user is pushing into its input ports) and it reads signals on the output ports of the component one-by-one, e.g., a possibility of reading a word w on a port, implies possibilities of reading any prefix of w on the port.

In a context of known number m of input ports and n of output ports, we will write $\uparrow\downarrow R$ rather than $\uparrow_{\{1,\ldots,m\}}\downarrow_{\{m+1,\ldots,m+n\}}(R \cup \{\varepsilon^{m+n}\})$. Intuitively, $\uparrow\downarrow R$ denotes the FIFO component closure of any relation $R \in (S^*)^{m+n}$.

Proposition 1 *FIFO components are closed under the union and the intersection, i.e., if C_1 and C_2 are FIFO components with m input and n output ports, then so are $C_1 \cap C_2$ and $C_1 \cup C_2$.*

Proof. Directly from the definition of the FIFO component. In fact, FIFO components seen as sets constitute a complete lattice. □

Definition 2 *Let $C \subseteq (S^*)^{m+n}$ be a FIFO component. We say that a word $t \in (\{1, \ldots, m+n\}, S)^*$ is a* trace *of C if for every prefix t' of t, $\mu(t') \in C$. By $\mathcal{T}(C)$ we will denote the set of all traces of C.*

Proposition 2 *Let C be a FIFO component. For every $x \in C$ there exists a trace $t \in \mathcal{T}(C)$, such that $\mu(t) = x$.*

Proof. Assume that C has m input and n output ports, and $x = (u; v) \in C$, where $u \in (S^*)^m$ is the m-tuple of input words and $v \in (S^*)^m$ is the n-tuple of output words in x. The sets $T_1 = \mu^{-1}(u \times \varepsilon^n)$ and $T_2 = \mu^{-1}(\varepsilon^m \times v)$ are both non-empty. Since $C = \uparrow\downarrow C$, for every $t \in T_1 \cdot T_2$ and every prefix $t' \leq t$, $\mu(t') \in C$. □

Intuitively, a trace represents a possible sequence of reads and writes of a sequential "user" of the component.

The usual *reordering* relation on traces $\ltimes$ (e.g., [5]) can be defined as the reflexive and transitive closure of:

$$\{(u(p,s)(q,t)v, u(q,t)(p,s)v) \mid p \neq q \text{ and } (p \leq m \text{ or } q > m)\}$$

where $P = \{1, \ldots, m+n\}$, $u, v \in (P \times S)^*$, and $(p,s), (q,t) \in P \times S$.

We introduce the *feasibility* relation, a pre-order relation over $\{1, \ldots, m+n\} \times S$, in order to captures the implication on "possibility" of occurring between traces. I.e., we will write $u \preceq v$, if $u \in \uparrow\downarrow t$, where $\uparrow\downarrow t \stackrel{\text{def}}{=} \mathcal{T}(\uparrow\downarrow\{\mu(t)\})$. Intuitively, $v \preceq u$ states that every system admitting a trace u will also admit trace v. For example, $v \leq u$ or $v \ltimes u$ implies $v \preceq u$.

Proposition 3 *Let $T \subseteq (\{1, \ldots, m+n\} \times S)^*$ be a set of traces of a FIFO component with m input and n output ports. T is closed with respect to the* feasibility *relation, i.e., if $t \in T$ then $t' \in T$, for every $t' \preceq t$.*

A pair $(p, s) \in \{1, \ldots, m+n\} \times S$ will be written as $p?s$ if $p \leq m$ and $p!s$ if $p > m$, in order to make it explicit that p is an input or output port, respectively.

By $\uparrow\downarrow T$ we denote the closure of set of traces T with respect to $\preceq$, i.e., $\uparrow\downarrow T \stackrel{\text{def}}{=} \{\uparrow\downarrow t \mid t \in T\}$. In [5], a non-empty set of traces T closed with respect to the *feasibility* relation, i.e., $T = \uparrow\downarrow T$, is called "*healthy*".

For every FIFO component C, $\mathcal{T}(C)$ is a healthy set of traces. However, it is important to notice that not every healthy set of traces corresponds to the set of traces of a FIFO component, since in general we have $\mathcal{T}(C_1) \cup \mathcal{T}(C_2) \subsetneq \mathcal{T}(C_1 \cup C_2)$. For example, following the Brock-Ackerman anomaly from [2], consider $C_1 = \uparrow\downarrow\{(\varepsilon, a)\}$ and $C_2 = \uparrow\downarrow\{(b, aa)\}$, i.e., two minimal FIFO components containing trace $t_1 = (2!a)$ and $t_2 = (1?b)(2!a)(2!a)$, respectively. FIFO component $C = C_1 \cup C_2$ includes the trace $t = (2!a)(1?b)(2!a)$, since $(b, aa), (b, a), (\varepsilon, a)$ are all in C. However, $t \notin \mathcal{T}(C_1) \cup \mathcal{T}(C_2)$.

3.1 Automata representation of FIFO components

An automaton, $\mathcal{A} = (Q, \Sigma, I, F, E)$, consists of a set Q of *states*, a finite alphabet Σ, a set of initial states $I \subseteq Q$, a set of final states $F \subseteq Q$, and a transition relation $E \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$. An automaton is finite if its set of states Q is finite.

Automaton $\mathcal{A} = (Q, \Sigma, I, F, E)$ defines a language $L(\mathcal{A})$ over Σ. A word $w \in \Sigma^*$ is in $L(\mathcal{A})$ if there is a path in $\mathcal{A}$ starting in a initial state $q_I \in I$ and ending in a final state $q_F \in F$ labeled by w, i.e., there exists a finite (possibly empty) sequence of transitions $(q_I, e_0, q_1), (q_1, e_1, q_2), \ldots, (q_k, e_k, q_F)$, such that $w = e_0 e_1 \ldots e_k$.

In our case, the alphabet Σ consists of port-signal pairs, i.e., $\Sigma = \{1, \ldots, m\} \times S$, where m is a natural number (of ports) and S is a finite set of signals. Therefore, an automaton $\mathcal{A} = (Q, \{1, \ldots, m\} \times S, I, F, E)$ defines also a relation $R(\mathcal{A}) \stackrel{\text{def}}{=} \mu(L(\mathcal{A})) \subseteq (S^*)^m$.

A relation $R \subseteq (S^*)^m$ is called *rational* if there is a finite automaton $\mathcal{A} = (Q, \{1, \ldots, m\} \times S, I, F, E)$ such that $R = R(\mathcal{A})$.

Lemma 4 *If $R \subseteq (S^*)^n$ is rational then $\uparrow_i R$ and $\downarrow_i R$, for $i \in \{1, \ldots, n\}$, are rational too.*

Proof. R is rational so there exists a finite automaton $A = (Q, \{1, \ldots, m\} \times S, I, F, E)$ for R. An automaton for $\downarrow_i R$ is $(Q \cup Q', \{1, \ldots, m\} \times S, I \cup I', F \cup F', E \cup E' \cup E'')$, where

- $Q' = c(Q)$ is a disjoint copy of Q by a bijection $c : Q \to Q'$;
- $I' = c(I)$ and $F' = c(F)$;
- $E' = \{(q, \varepsilon, c(q)) \mid q \in Q\} \cup \{(c(q), \varepsilon, c(q')) \mid \exists (q, (i, s), q') \in E\}$;
- $E'' = \{(c(q), (j, s), c(q')) \mid j \neq i, (q, (j, s), q') \in E\}$.

Let $I = \{\varepsilon^{i-1}\} \times S^* \times \{\varepsilon^{n-i}\}$. Since R and I are rational, so is $\uparrow_i R = R \cdot I$. □

Directly form Lemma 4 we have:

Theorem 5 *If $R \subseteq (S^*)^{m+n}$ is rational then the FIFO component closure of R, i.e., $\uparrow\downarrow R$, is rational too.*

Thanks to Theorem 5, any rational $R \subseteq (S^*)^{m+n}$ defines a rational FIFO component $\uparrow\downarrow R$.

An automaton $A = (Q, \Sigma, I, F, E)$ will be called a *FIFO state machine* if $\Sigma = \{1, \ldots, m+n\} \times S$, $F = Q$, and the automaton is *deterministic*, i.e.:

- $I = \{q_0\}$, i.e., there is one initial state;
- $E \subseteq Q \times \Sigma \times Q$, i.e., there is no *silent* transitions;
- for every $(q_1, (p_1, s_1), q_1'), (q_2, (p_2, s_2), q_2') \in E$, if $q_1 = q_2$ then $p_1 = p_2$, i.e., in every state we read or write only on one port (one dimension);
- for every $(q_1, (p_1, s_1), q_1'), (q_2, (p_2, s_2), q_2') \in E$, if $q_1 = q_2$ and $s_1 = s_2$ then $q_1' = q_2'$, i.e., in every state what we read or write defines the unique next state.

An automaton which is a FIFO state machine will be represented by a quadruple $A = (q_o, \Sigma, I, E)$.

A FIFO state machine can be seen as an abstract implementation of a component consisting of a control state machine together with FIFO queues, one for every port. The global state, called *configuration*, of the system implementing the component is described by the local state of the control state machine together with the content of the queues. The environment (user of the component) communicates with the component by writing signals into input port queues and reading signals from the output port queues.

The finite FIFO state machines are of interest for us because of two reasons. Firstly, they are effectively[2] implementable in software or in hardware. Secondly, the problem of equivalence of two finite FIFO state machines is decidable[3].

The behavior of a FIFO state machine can be described in terms of a *labeled transition system* (*lts*). An *lts* is a triple (G, T, λ), where G is the set of global states (configurations), λ is a finite set of *labels*, and $T \subseteq G \times \lambda \times G$, is the set of (global) transitions. A global transition $t = (g_1, l, g_2) \in T$ will be denoted by $g_1 \xrightarrow{l} g_2$, and its label l by $\lambda(t)$.

For the *lts* associated with a FIFO state machine $\mathcal{F}(m, n) = (\{q_0\}, \Sigma, I, E)$, the global states are $m+n+1$ tuples, $G \subseteq (S^*)^{m+n} \times Q$. The initial global state of the *lts* is $(\varepsilon, \ldots, \varepsilon, q_0)$; it represents the initial configuration where all queues are empty, and the local control state of the component is in the initial state. The global transitions of the *lts*, $T \subseteq G \times \lambda \times G$, represent configuration changes made by the environment (when a new signal is appended into an input port queue, or a signal is read from an output port queue) or by the control state machine (when a local transition $e \in E$ is executed). Thus, λ labels every global transition $t \in T$ by $\{1, \ldots, m+n\} \times S$ or by E, i.e., $\lambda = (\{1, \ldots, m+n\} \times S) \cup E$.

[2] By "effectively implementable" we mean that a deadlock-free program or a device for the FIFO component can be derived directly from its definition.

[3] The statement follows (not so directly) from the decidability result for the equivalence of deterministic multi-tape finite automata, [3].

The *lts* (G, T, λ) for a FIFO state machine $\mathcal{F}(m,n) = (Q, \Sigma, I, E)$ is defined by the following rules (with $i \in \{1, \ldots, m+n\}$, $\alpha_i \in S^*$, $s \in S$, $q, q' \in Q$):

- $(\alpha_1, \ldots, \alpha_j, \ldots, \alpha_{m+n}, q) \xrightarrow{j?s} (\alpha_1, \ldots, \alpha_j \cdot s, \ldots, \alpha_{m+n}, q)$ for every $j \in \{1, \ldots, m\}$;
- $(\alpha_1, \ldots, s \cdot \alpha_j, \ldots, \alpha_{m+n}, q) \xrightarrow{j!s} (\alpha_1, \ldots, \alpha_j, \ldots, \alpha_{m+n}, q)$ for every $j \in \{m+1, \ldots, m+n\}$;
- $(\alpha_1, \ldots, s \cdot \alpha_j, \ldots, \alpha_{m+n}, q) \xrightarrow{e} (\alpha_1, \ldots, \alpha_j, \ldots, \alpha_{m+n}, q')$, for every $j \in \{1, \ldots, m\}$ and $e \in E$ such that $e = (q, j?s, q')$;
- $(\alpha_1, \ldots, \alpha_j, \ldots, \alpha_{m+n}, q) \xrightarrow{e} (\alpha_1, \ldots, \alpha_j \cdot s, \ldots, \alpha_{m+n}, q')$, for every $j \in \{m+1, \ldots, m+n\}$ and $e \in E$ such that $e = (q, j!s, q')$.

By $\Pi(\mathcal{F})$ we will denote the set of all paths in the above *lts* starting in the initial configuration. By $\lambda(\pi) \subseteq (E \cup \{1, \ldots, m+n\} \times S)^*$, for $\pi \in \Pi$, we denote the label of π, i.e., the concatenations of all labels of the global transitions of π. Also, by $\lambda_i(\pi) \subseteq E^*$ and by $\lambda_e(\pi) \subseteq (\{1, \ldots, m+n\} \times S)^*$ we denote the "*internal*" and "*external*" label of π, i.e., where all occurrences of $\{1, \ldots, m+n\} \times S$ or of E, respectively, are erased from $\lambda(\pi)$.

Theorem 6 *Let $\mathcal{F}$ be a FIFO state machine defining FIFO component C. We have*

$$\mathcal{T}(C) = \lambda_e(\Pi(\mathcal{F}))$$

where $\lambda_e(\Pi(\mathcal{F})) \stackrel{\mathtt{def}}{=} \{\lambda_e(\pi) \mid \pi \in \Pi(\mathcal{F})\}$.

Proof. We need to prove that:

- for every $t \in \mathcal{T}(C)$ there exists $\pi \in \Pi(\mathcal{F})$ such that $t = \lambda_e(\pi)$; and
- for every $\pi \in \Pi(\mathcal{F})$ we have $\lambda_e(\pi) \in \mathcal{T}(C)$.

...

A FIFO state machine $\mathcal{F}(m,n) = (Q, \Sigma, I, E)$ is *deterministic* if:

- $I = \{q_0\}$, i.e., there is one initial state;
- $E \subseteq Q \times \Sigma \times Q$, i.e., there is no *silent* transitions;
- for every $(q_1, (p_1, s_1), q_1'), (q_2, (p_2, s_2), q_2') \in E$, if $q_1 = q_2$ then $p_1 = p_2$, i.e., in every state we read or write only on one port (one dimension);
- for every $(q_1, (p_1, s_1), q_1'), (q_2, (p_2, s_2), q_2') \in E$, if $q_1 = q_2$ and $s_1 = s_2$ then $q_1' = q_2'$, i.e., in every state what we read or write defines the unique next state.

The finite deterministic FIFO state machines are of interest for us because of two reasons. Firstly, they are effectively[4] implementable in software or in hardware. Secondly, the problem of equivalence of two finite deterministic FIFO state machines is decidable[5].

[4] By "effectively implementable" we mean that a deadlock-free program or a device for the FIFO component can be derived directly from its definition.

[5] The statement follows (not so directly) from the decidability result for the equivalence of deterministic multi-tape finite automata, [3].

4 Gathering and wiring FIFO components

A FIFO channel system is a network of FIFO components. Any such finite network can be described as the result of *grouping* and *wiring* components. The grouping operation is defined as a binary operation corresponding to the *parallel composition*. The semantics of two FIFO components, C_1 and C_2, grouped together is a new component $(C_1 \parallel C_2) \subseteq (S^*)^{m_1+m_2+n_1+n_2}$ with $m_1 + m_2$ input ports and $n_1 + n_2$ output ports, where m_i, n_i are the numbers of input and output ports of C_i, for $i \in \{1, 2\}$.

$$(C_1 \parallel C_2) \stackrel{\texttt{def}}{=} \{(x_1 \times x_2; y_1 \times y_2) \mid (x_1; y_1) \in C_1, (x_2; y_2) \in C_2\}.$$

The semantics of $(C_1 \parallel C_2)$ corresponds to $C_1 \times C_2$ followed by a reordering of the words in the tuples which moves the input words of C_2 in front of the output words of C_1. With an "appropriate" port renaming of traces of C_1 and C_2, the set of traces of $(C_1 \parallel C_2)$ corresponds to the shuffle of the renamed traces of C_1 with the renamed traces of C_2,

Proposition 7 *The grouping operation is associative.*

Proposition 8 *The grouping operation is increasing, i.e., $C_1 \subseteq C_2$ implies $(C_1 \parallel C) \subseteq (C_2 \parallel C)$ and $(C \parallel C_1) \subseteq (C \parallel C_2)$.*

Wiring operation, $C \circ [p \leftarrow q]$, takes a FIFO component C, one of its input ports p, and one of its output ports q, and connects the ports together. The connected ports disappear from the interface of $C \circ [p \leftarrow q]$. The semantics of $C \circ [p \leftarrow q]$ is built out of those traces of C in which every read of a signal on port p is preceded by a write of the signal to the port q. The set of such traces can be defined as follows:

$$\texttt{SYNCH}(C, p \leftarrow q) \stackrel{\texttt{def}}{=} \{t \in \mathcal{T}(C) \mid \forall x \leq t,\ x|_p \leq x|_q\}\ ,$$

where $t|_j$, for $t \in (\{1, \ldots, n\} \times S)^*$ and $j \in \{1, \ldots, n\}$, denotes the word over S obtained from t by its projection on port j, i.e., $\varepsilon|_j = \varepsilon$, $((i, s) \cdot t)|_j = s \cdot t|_j$ if $j = i$, and $((i, s) \cdot t)|_j = t|_j$ if $j \neq i$. Intuitively, a trace t of C is in $\texttt{SYNCH}(C, p \leftarrow q)$ if in every prefix x of t, the word on port p is a prefix of the word on port q.

Finally we can write:

$$C \circ [p \leftarrow q] \stackrel{\texttt{def}}{=} \mathit{HIDE}\,(\mu\,(\texttt{SYNCH}(C, p \leftarrow q))\,, \{p, q\})\,.$$

Proposition 9 *The wiring operation is "commutative", i.e., if $p_1 < p_2 < q_1 < q_2$ then*

$$C \circ [p_1 \leftarrow q_1] \circ [p_2 - 1 \leftarrow q_2 - 2] = C \circ [p_2 \leftarrow q_2] \circ [p_1 \leftarrow q_1' - 1],$$

and if $p_1 < p_2 < q_2 < q_1$ then

$$C \circ [p_1 \leftarrow q_1] \circ [p_2 - 1 \leftarrow q_2 - 1] = C \circ [p_2 \leftarrow q_2] \circ [p_1 \leftarrow q_1' - 2].$$

The proposition can be proved using the following simple observation.

Lemma 10 *Let $C(m,n)$ be a component with at least two input ports p_1, p_2 and two output ports q_1, q_2, i.e., let $1 \leq p_1 < p_2 \leq m < q_1 < q2 \leq m+n$.*

The component $(C \circ [p_1 \leftarrow q_1]) \circ [p_2 - 1 \leftarrow q_2 - 2]$ is equal to:

$$\mathit{HIDE}\,(\mu(\texttt{SYNCH}(C, p_1 \leftarrow q_1) \cap \texttt{SYNCH}(C, p_2 \leftarrow q_2)), \{p_1, p_2, q_1, q_2\})\,,$$

and component $(C \circ [p_1 \leftarrow q_2]) \circ [p_2 - 1 \leftarrow q_2 - 1]$ is equal to:

$$\mathit{HIDE}\,(\mu(\texttt{SYNCH}(C, p_1 \leftarrow q_2) \cap \texttt{SYNCH}(C, p_2 \leftarrow q_1)), \{p_1, p_2, q_1, q_2\})\ .$$

The system from Figure 1 could be described by the following *grouping* and *wiring* expression: $((((C_1 \parallel (C_2 \parallel C_3)) \circ [2 \leftarrow 7]) \circ [3 \leftarrow 5]) \circ [2 \leftarrow 6]) \circ [2 \leftarrow 3]$.

Notice that the port numbering changes after every operation. Thanks to Propositions 7 and 9, we can introduce the simultaneous wiring operation in order to make the expression closer to its graphical representation (see Figure 2):

$$(C_1 \parallel C_2 \parallel C_3) \circ [2 \leftarrow 7, 4 \leftarrow 6, 3 \leftarrow 10, 5 \leftarrow 9].$$

We will also allow an even more direct notation, by allowing composed port

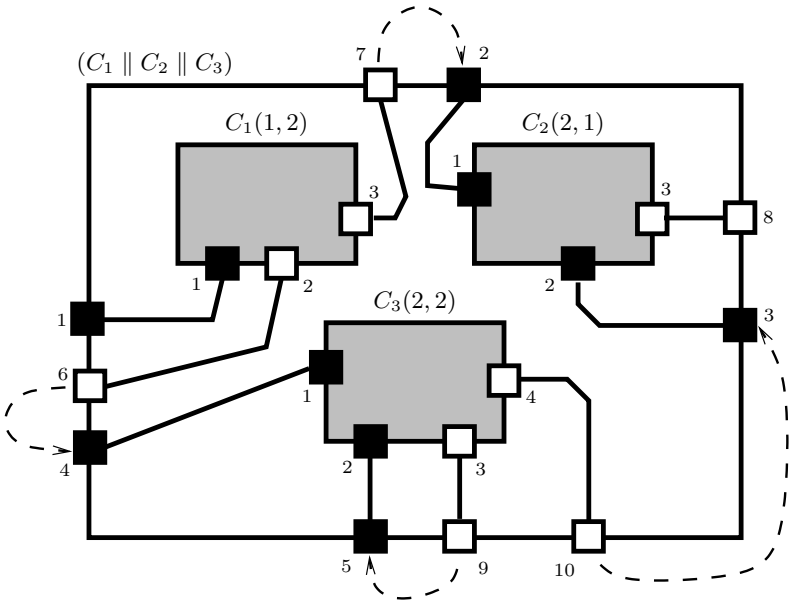

Fig. 2. Gathering $(C_1 \parallel C_2 \parallel C_3)$ and wiring $[2 \leftarrow 7, 4 \leftarrow 6, 3 \leftarrow 10, 5 \leftarrow 9]$, or equivalently $[C_2.1 \leftarrow C_1.3, C_3.1 \leftarrow C_1.2, C_2.2 \leftarrow C_3.4, C_3.2 \leftarrow C_3.3]$, yields component $C(1,1)$ from Figure 1.

identifiers of the form `component.local-port-number` when a gathering is immediately followed by a wiring. Thus, the system from Figure 1 could be written as: $(C_1 \parallel C_2 \parallel C_3) \circ [C_2.1 \leftarrow C_1.3, C_3.1 \leftarrow C_1.2, C_2.2 \leftarrow C_3.4, C_3.2 \leftarrow C_3.3]$.

Proposition 11 *The wiring operation is increasing, i.e., for any wiring H, $C_1 \subseteq C_2$ implies $C_1 \circ H \subseteq C_2 \circ H$.*

Proof. It is enough to check that `SYNC` is increasing. □

4.1 Recursive definitions of FIFO channel systems

The compositional semantics allows us to introduce recursion in system descriptions. Consider the system $X(1,1)$ from Figure 3 which, intuitively, reads one word over two letter alphabet, a, b, terminated by an end-marker, \$, and outputs the reverse of its input.

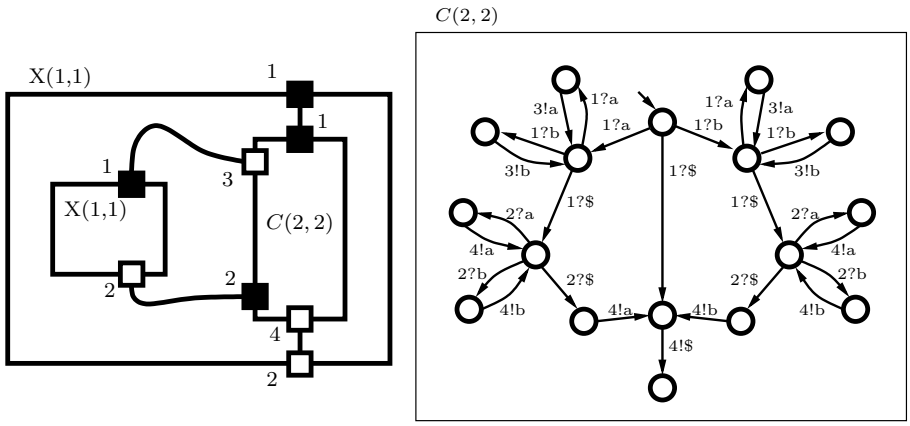

Fig. 3. Recursive description of $X = (X \parallel C) \circ [X?1 \leftarrow C!3, C?2 \leftarrow X!2]$, left, and the automaton defining component C, right.

In general, a recursive definition $X = \Psi(X)$, where $\Psi(\ldots)$ is a "gathering-wiring" expression, can be interpreted as the smallest (or the biggest) by inclusion component satisfying the equation. Since any such expression is *increasing*, Propositions 8 and 11, i.e., $C_1 \subseteq C_2$ implies $\Psi(C_1) \subseteq \Psi(C_2)$, and FIFO components constitute a complete lattice, Proposition 1, by Knaster-Tarski theorem [8], such FIFO components exist. For example, the smallest solution can be described as follows:

$$X \stackrel{\texttt{def}}{=} \bigcup_{i \geq 0} X_i \ ,$$

where $X_0 \stackrel{\texttt{def}}{=} \uparrow\downarrow(\varepsilon^{m+n})$ and $X_{k+1} \stackrel{\texttt{def}}{=} \Psi(X_k)$, for $k \geq 0$.

4.2 Execution model

In this section we assume that a FIFO channel system is built out of a finite set of finite FIFO state machines in the spirit of [1]. We recall the execution model of such a system by defining an infinite FIFO state machine implementing it.

Let $F_i(m_i, n_i) = (Q_i, \Sigma_i, I_i, E_i)$, for $i \in \{1, \ldots, k\}$, $\Sigma_i = \{1, \ldots, m_i + n_i\} \times S$, be a finite collection of components given by their finite FIFO state machines, and $H_i = \{(f_i, p_i), (f_i', p_i')\}$, for $i \in \{1, \ldots, l\}$, $f_i, f_i' \in \{1, \ldots, k\}$, $p_i \in \{m_{f_i} + 1, \ldots, m_{f_i} + n_{f_i}\}$, $p_i' \in \{1, \ldots, m_{f_i'}\}$, be a set of point-to-point FIFO channels. Intuitively, the system Ω is defined as:

$$\Omega = (F_1 \parallel \ldots \parallel F_k) \circ [H_1, \ldots, H_l].$$

Components are numbered from 1 to k and channels from 1 to l. Component number i is defined by FIFO state machine F_i with m_i input and n_i output ports. A channel number j, H_j, is defined by two pairs $\{(f_j, p_j), (f_j', p_j')\}$ of ports; H_j connects port p_j of component f_j with port p_j' of component f_j'. By "point-to-point channels" we mean that one port of a component is connected to at most one channel, i.e., if $i \neq j$ then $H_i \cap H_j = \emptyset$ for all $i, j \in \{1, \ldots, l\}$.

Let $A_\Omega = (Q, \Sigma, I, Q, E)$ be the following (infinite) automaton, where

- The set of states Q is

$$Q \stackrel{\texttt{def}}{=} Q_1 \times Q_2 \times \ldots \times Q_k \times (S^*)^l$$

- The alphabet Σ consists of local transitions, i.e.:

$$\Sigma \stackrel{\texttt{def}}{=} E_1 \cup E_2 \cup \ldots \cup E_k$$

- The set of initial states I is defined as:

$$I = I_1 \times I_2 \times \ldots \times I_k \times (\{\varepsilon\})^l$$

- The transition relation E consists of *steps*. Every step corresponds to a firing of a local transition which may change only one local state and one channel:

$$((q_1, \ldots, q_k, f_1, \ldots, f_l), e, (q_1', \ldots, q_k', f_1', \ldots, f_l')) \in E$$

  if and only if there exists a component $i \in \{1, \ldots, k\}$ and a transition $e = (q, \alpha, q') \in E_i$ such that:
  1. $q_i = q$, $q_i' = q'$, and $q_j = q_j'$ for all $j \in \{1, \ldots, k\} \setminus \{i\}$;
  2. If $\alpha = \varepsilon$, i.e., e is a silent transition, then $f_j' = f_j$, for all $j \in \{1, \ldots, l\}$;
  3. If $\alpha = p!s$, i.e., transition e writes signal $s \in S$ to output port $p \in \{m_i + 1, \ldots, m_i + n_i\}$, then, for all $j \in \{1, \ldots, l\}$,

$$f_j' = \begin{cases} f_j \cdot s & \text{if channel } j \text{ connects } p, \text{ i.e., } (i, p) \in H_j \\ f_j & \text{otherwise} \end{cases}$$

  4. If $\alpha = p?s$, i.e., transition e reads signal $s \in S$ from input port $p \in \{1, \ldots, m_i\}$, then, for all $j \in \{1, \ldots, l\}$,

$$f_j = \begin{cases} s \cdot f_j' & \text{if channel } j \text{ connects } p, \text{ i.e., } (i, p) \in H_j \\ f_j' & \text{otherwise} \end{cases}$$

Let $\phi : \Sigma \to (\{1, \ldots, n\} \times S) \cup \{\varepsilon\}$ be the relabeling of transitions from local transitions of the components into ports of Ω, with n being the number of unconnected ports in the system. Intuitively, if $e = (q, \alpha, q')$ is a local transition of component i with $\alpha = \varepsilon$ or α writing/reading on a connected port, then $\phi(e) = \varepsilon$. Otherwise, e is reading/writing signal s on an interface port p of Ω, and thus $\phi(e) = (p, s)$.

Finally, from $A_\Omega = (Q, \Sigma, I, Q, E)$ we define the FIFO state machine for Ω as $\mathcal{F}_\Omega = (Q, \{1, \ldots, n\} \times S, I, \phi(E))$, where $\phi((q, e, q')) \stackrel{\texttt{def}}{=} (q, \phi(e), q')$.

Theorem 12 *Let C_Ω be a FIFO component semantics of Ω and $\mathcal{F}_\Omega = (Q, \Sigma, I, E)$ be the FIFO state machine semantics for Ω as defined in this section. We have:*

$$\lambda_e(\Pi(\mathcal{F}_\Omega)) = \mathcal{T}(\Omega),$$

where $\Pi(\mathcal{F}_\Omega)$ is the set of paths of the lts *for $\mathcal{F}_\Omega$ as defined in Section 3.1.*

The theorem (whose proof doesn't fit into the paper) says that the FIFO channel system semantics in terms of word relation captures the visible (external) part of the traditional execution model for such systems.

5 Conclusions

The FIFO channel system model is used in many areas of the information technology: from low-level hardware design to high-level software specification and verification. Even though, in general, most decision problems for the very expressive FIFO channel systems are undecidable, there are many techniques based on model restrictions or/and by using *over-approximation* which make the systems appealing for formal verifications.

In this paper we proposed a compositional semantics for FIFO channel systems in terms of word relations. The semantics allows different realizations of a system as long as the visible (external) aspect of the system is preserved. Many practical aspects for which FIFO channel systems were used in the past, such as the performance evaluation, are not directly addressed by the semantics which says "what" not "how". We believe however that the FIFO channel system model, which is especially appealing because of its simple execution model, can make the translation from "what" to "how" relatively straightforward.

It is important to notice that the word relation semantics does not consider any notion of time. In that asynchronous model we do not distinguish between a delay and a deadlock. For example, a FIFO component $C(0, 1) = \downarrow\{(1!a)\}$, i.e., which produces once a signal a on its unique output port, does not guarantee that the output be ever produced. The observer (the user of the component) has to wait for a to be produced and, since the system is asynchronous, the waiting time is not bounded. In other words, the FIFO channel system under the word relation semantics is a *specification* of a system, rather then its *implementation*. In such a setting, the problem of deadlock, for example, becomes a problem of performance evaluation of a particular implementation.

References

1. Daniel Brand and Pitro Zafiropulo. On communicating finite-state machines. *J. ACM*, 30(2):323–342, 1983.
2. J. Dean Brock and William B. Ackerman. Scenarios: A model of non-determinate computation. In Josep Díaz and Isidro Ramos, editors, *ICFPC*, volume 107 of *Lecture Notes in Computer Science*, pages 252–259. Springer, 1981.
3. T. Harju and J. Karhumäki. The equivalence problem of multitape finite automata. *Theor. Comput. Sci.*, 78(2):347–355, 1991.
4. Mark B. Josephs. Receptive process theory. *Acta Inf.*, 29(1):17–31, 1992.
5. Mark B. Josephs. Models for data-flow sequential processes. In Ali E. Abdallah, Cliff B. Jones, and Jeff W. Sanders, editors, *25 Years Communicating Sequential Processes*, volume 3525 of *Lecture Notes in Computer Science*, pages 85–97. Springer, 2004.
6. Gilles Kahn. The semantics of a simple language for parallel programming. In J. L. Rosenfeld, editor, *Information Processing '74: Proceedings of the IFIP Congress*, pages 471–475. North-Holland, New York, NY, 1974.
7. Jaques Sakarovitch. *Eléments de théorie des automates*. Vuibert Informatique, 2003.
8. Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. Math.*, 5(2):285–309, 1955.