Available online at www.sciencedirect.com
SCIENCE DIRECT
Electronic Notes in Theoretical Computer Science
Electronic Notes in Theoretical Computer Science 104 (2004) 199–215
www.elsevier.com/locate/entcs

A Graphical Fusion Calculus[⋆]

Ivan Lanese[1]

Dipartimento di Informatica
Università di Pisa
Pisa, Italy

Ugo Montanari[2]

Dipartimento di Informatica
Università di Pisa
Pisa, Italy

Abstract

We analyze the relationship between *Fusion Calculus* and *graph transformations* defined in the *Synchronized Hyperedge Replacement* (SHR) style. In particular we show that the underlying algebraic structure is the same when the synchronization used in SHR is Milner synchronization. The main difference we see is that Fusion Calculus has an *interleaving* behaviour while SHR is inherently *concurrent*. In the paper we introduce the interleaving semantics for SHR with Milner synchronization and show that there is a complete correspondence between the operational semantics of Fusion Calculus and of SHR systems.

Keywords: Fusion Calculus, graph transformation, Synchronized Hyperedge Replacement, mobility

1 Introduction

In this paper we compare two different kinds of models which are useful for studying concurrent, distributed and mobile systems. This kind of systems is very important in practice since it includes the Internet, which interconnects

⋆ Research supported in part by MIUR project **COMETA** and EU-FET project **AGILE** IST-2001-32747.

[1] Email: lanese@di.unipi.it

[2] Email: ugo@di.unipi.it

doi:10.1016/j.entcs.2004.08.026

hundreds of millions of users, but also other kinds of nets, such as LANs and wireless communication networks. The problem of developing large systems, as often required, in this kind of environments is complex since the traditional computational aspects interact in new ways with issues such as coordination, synchronization, security and mobility. So powerful formal models and tools able to deal with this scenario at a suitable level of abstraction are needed.

Several proposals have appeared in literature, and some of them are widely used, but no single formalism has emerged as the best model for this kind of systems. In this scenario, it is very important to analyze and understand the relationships that hold between different models, in order to find out which features are required to deal with this kind of systems, which are the possible choices and which are merits and drawbacks of each of them. In the paper, we study the relationships between *Fusion Calculus* [11] and *Synchronized Hyperedge Replacement* [1,3]. The Fusion Calculus follows the approach of *process-calculi*, where, as in CCS [9], a system is modeled by a term in a suitable algebra and the operational semantics is specified by structural induction. A successor of CCS, the π-calculus [10], allows the study of a wide range of mobility problems in a simple mathematical framework. Fusion Calculus is an evolution of π-calculus which is interesting since it is obtained by simplifying and making more symmetric the π-calculus. In fact, the input prefix of the π-calculus has been decomposed into a new input prefix which is symmetric with respect to the output prefix and a standard π-calculus restriction. Furthermore Fusion Calculus has a new kind of actions, *fusion actions*, which merge names and thus allow modeling some aspects of mobility that the π-calculus can not handle.

One of the known limitations of the process-calculi approach when applied to distributed systems is that process-calculi lack an intuitive representation because they are equipped with an interleaving semantics and they use the same constructions for representing both the agents and their configurations. A different approach that solves this kind of problems is based on *graph transformations* [12]. In this case, the configuration is explicitly represented by a graph which offers both a clean, inherently concurrent mathematical semantics and a suggestive representation. Among the various proposals for graph transformations we choose Synchronized Hyperedge Replacement (SHR) [2,1,6,3]. In our approach we represent computational entities such as processes or hosts with hyperedges (edges connected to an arbitrary number of nodes) and channels between them as shared nodes. As far as the dynamic aspect is concerned, we use productions to specify the behaviours of single hyperedges which are synchronized by exposing actions on nodes. Actions exposed by different hyperedges on the same node must be compatible. What exactly compatible means depends on the choice of the synchronization model. For instance, we can have Hoare synchronization, where all the edges must expose the same action (in the CSP style), and Milner synchronization where two corresponding actions ("input" and "output") are synchronized (in the CCS style). We use the extension of SHR with *mobility* [5,3], that allows hyperedges to send node references together with actions. Nodes whose references are matched during the synchronization are fused. This approach is very expressive as shown in [4], since it can employ multiple synchronizations to define a global rewriting step using only local rules. In literature SHR has already been used as a meta-model to study other formalisms such as π-calculus [4,5] and Ambient calculus [3,8]. Our work in particular extends the work of [5] on π-calculus to cope with the expressiveness of Fusion Calculus.

We show that Fusion Calculus (with guarded sum and recursion) is in strict correspondence with a subset of SHR. The correspondence is based on the existence in both models of two analogous operators, i. e. a parallel composition operator for building systems and a restriction operator for declaring local names. Furthermore we recognize close relationships between the respective mechanisms for mobility. The main difference is that Fusion Calculus has an *interleaving* semantics and allows just one synchronization at each step, whereas SHR is a *concurrent* model and allows multiple synchronizations. Thus in order to have a complete correspondence we have to force an interleaving semantics also for SHR, what we do using a particular set of inference rules. By using the normal inference rules, we have transitions that correspond to many Fusion Calculus steps.

Section 2 contains the required background, in particular syntax and semantics of Fusion Calculus (2.1), the algebraic representation for graphs and the rules for SHR (2.2). Section 3 contains the main contributions of this paper, that is the rules for interleaving Milner SHR (3.1) and the mapping from Fusion Calculus into interleaving Milner SHR (3.2). Finally Section 4 contains some conclusions and traces for future work.

2 Background

2.1 The Fusion Calculus

The Fusion Calculus [11] is a calculus for modeling distributed and mobile systems which is based on the concepts of fusion and scope. It is an evolution of the π-calculus [10], and the interesting point is that it is obtained by simplifying the calculus. In fact, the two action prefixes for communication are symmetric whereas in the π-calculus they are not and we have just one binding operator called scope whereas the π-calculus has two (restriction and input). As shown in [11] the π-calculus is syntactically a subcalculus of the Fusion Calculus (the key point is that the input of π-calculus is obtained using input and scope). In order to have these properties fusion actions have to be introduced.

We will present in detail the syntax and the SOS semantics of Fusion Calculus. In our work we consider only a subcalculus of the Fusion Calculus, which has no match (but we will later discuss how to reintroduce it) and no mismatch operator, and has only guarded summation and recursion. In our discussion we distinguish between sequential agents (which have a summation as topmost operator) and general agents.

We suppose to have an infinite set $\mathcal{N}$ of names ranged over by $u, v, \ldots, z$. Names represent communication channels. We use $\boldsymbol{x}$ to denote a vector of names and ϕ to denote an equivalence relation on $\mathcal{N}$ which can be represented by a finite set of equalities (we denote with 1 the identity relation).

Definition 2.1 The free actions are defined by:

$$\begin{aligned} \alpha ::= {} & u\boldsymbol{x} \text{ (Input)} \\ & \overline{u}\boldsymbol{x} \text{ (Output)} \\ & \phi \text{ (Fusion)} \end{aligned}$$

Definition 2.2 The agents are defined by:

$$S ::= \sum_i \alpha_i.P_i \text{ (Guarded sum)}$$

$$\begin{array}{rcl} P ::= & 0 & \text{(Inaction)} \\ & S & \text{(Sequential Agent)} \\ & P_1|P_2 & \text{(Composition)} \\ & (x)P & \text{(Scope)} \\ & \mathrm{rec}\, X.P & \text{(Recursion)} \\ & X & \text{(Agent variable)} \end{array}$$

The scope operator is a binder for names thus x is bound in $(x)P$.

Similarly rec is a binder for agent variables, furthermore we only consider agents which are closed with respect to agent variables and where in $\mathrm{rec}\, X.P$ each occurrence of X is within a sequential agent (guarded recursion).

We define the functions fn, bn and n that given an agent or an action compute the set of free, bound and all names respectively (the names in a fusion are the names in non singleton equivalence classes).

Processes are agents considered up-to structural axioms defined as follows.

Definition 2.3 The structural congruence $\equiv$ between agents is the least congruence satisfying the α-conversion law (both for names and for agent variables), the abelian monoid laws for summation and composition (associativity, commutativity and 0 as identity), the scope laws $(x)0 \equiv 0$, $(x)(y)P \equiv (y)(x)P$, the scope extension law $P|(z)Q \equiv (z)(P|Q)$ where $z \notin \mathrm{fn}(P)$ and the recursion law $\mathrm{rec}\, X.P \equiv P\{\mathrm{rec}\, X.P/X\}$.

Note that fn can be trivially extended to processes.

Definition 2.4 A bound action is of the form $(\boldsymbol{z})a\boldsymbol{x}$ where $|\boldsymbol{z}| > 0$ and all elements in $\boldsymbol{z}$ are also in $\boldsymbol{x}$. Names in $\boldsymbol{z}$ are bound names. The actions consist of the free actions and the bound actions.

For convenience we define $\phi \setminus z$ to mean $\phi \cap (\mathcal{N} \setminus \{z\})^2 \cup \{(z,z)\}$.
We can now present the SOS semantics.

Definition 2.5 [SOS semantics for Fusion Calculus]

$$\text{PREF} \quad \frac{-}{\alpha.P \xrightarrow{\alpha} P}$$

$$\text{SUM} \quad \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$$

$$\text{PAR} \quad \frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}$$

$$\text{COM} \quad \frac{P \xrightarrow{u\boldsymbol{x}} P', Q \xrightarrow{\overline{u}\boldsymbol{y}} Q', |\boldsymbol{x}| = |\boldsymbol{y}|}{P|Q \xrightarrow{\{\boldsymbol{x}=\boldsymbol{y}\}} P'|Q'}$$

$$\text{SCOPE} \quad \frac{P \xrightarrow{\phi} P', z\phi x, z \neq x}{(z)P \xrightarrow{\phi\setminus z} P'\{x/z\}}$$

$$\text{PASS} \quad \frac{P \xrightarrow{\alpha} P', z \notin \mathrm{n}(\alpha)}{(z)P \xrightarrow{\alpha} (z)P'}$$

$$\text{OPEN} \quad \frac{P \xrightarrow{(\boldsymbol{y})a\boldsymbol{x}} P', z \in \boldsymbol{x} \setminus \boldsymbol{y}, a \notin \{z, \overline{z}\}}{(z)P \xrightarrow{(z\boldsymbol{y})a\boldsymbol{x}} P'}$$

$$\text{STRUCT} \quad \frac{P \xrightarrow{\alpha} P' \quad P \equiv Q \quad P' \equiv Q'}{Q \xrightarrow{\alpha} Q'}$$

We show here a useful decomposition that can be defined for fusion agents.

Definition 2.6 The standard decomposition of an agent P is defined as:

$$P = \sigma_P \hat{P}$$

where σ_P is the standard substitution and $\hat{P}$ is the standard agent of P. This decomposition satisfies:

$$P = \sigma Q \text{ implies } \hat{P} = \hat{Q} \wedge \sigma_P = \sigma\sigma_Q$$

We denote with $\mathrm{fnarray}(P)$ the array of the free name occurrences in P, ordered according to some fixed order dictated by the structure of P. In particular $\sigma_P = \{\mathrm{fnarray}(P)/\,\mathrm{fnarray}(\hat{P})\}$.

The standard decomposition can be easily extended to processes.

2.2 Synchronized Hyperedge Replacement

Synchronized Hyperedge Replacement [2,1,3] is an approach to (hyper)graph transformations that allows the definition of global transformations using local productions that describe how a single hyperedge can be rewritten and the constraints that the rewriting imposes on the surrounding nodes. Thus the global transformation is obtained by combining different productions whose conditions are compatible. What exactly compatible means depends on which synchronization model we use. Possible models are for instance the Hoare [7] and the Milner [3] synchronization models. Furthermore the rewriting system also use node mobility, that is during a transformation some references to nodes can be transmitted and corresponding nodes are merged.

We give a formal description of SHR in terms of labelled transition system, but first of all we need an algebraic representation for hypergraphs. A hyperedge, or simply an edge, is an atomic item with a label (from a ranked alphabet $LE = \{LE_n\}_{n=0,1,\ldots}$) and with as many ordered tentacles as the rank of its label. A set of nodes, together with a set of such edges, forms a hypergraph (or simply a graph) if each edge is connected, by its tentacles, to its attachment nodes. A graph is equipped with a set of external nodes identified by distinct names. External nodes can be seen as the connecting points of a graph with its environment (i. e. the context). We consider graphs up-to isomorphisms that preserve connections between edges and nodes, external nodes and edge labels. Now, we present a definition of graphs as syntactic judgements, where nodes correspond to names, external nodes to free names and edges to basic terms of the form $L(x_1, \ldots, x_n)$, where x_i are arbitrary names and $L \in LE_n$.

Definition 2.7 [Graphs as syntactic judgements] Let $\mathcal{N}$ be a fixed infinite set of names and LE a ranked alphabet of labels. A syntactic judgement is of the form $\Gamma \vdash G$ where:

(i) $\Gamma \subseteq \mathcal{N}$ is a finite set of names (the external nodes of the graph).

(ii) G is a term generated by the grammar
$G ::= L(\boldsymbol{x}) \mid G|G \mid \nu y\ G \mid nil$
where $\boldsymbol{x}$ is a vector of names, L is an edge label with $\text{rank}(L) = |\boldsymbol{x}|$ and y is a name.

We define ν as a binder, so in $\nu y\ G$ all occurrences of y (also the ones in G) are bound occurrences. We define as usual the functions fn, bn and n.

We demand that $\text{fn}(G) \subseteq \Gamma$.

We use the notation Γ, x to denote the set obtained by adding x to Γ, assuming $x \notin \Gamma$. Similarly, we write Γ_1, Γ_2 to state that the resulting set of names is the disjoint union of Γ_1 and Γ_2.

Definition 2.8 [Structural congruence]

We define the structural congruence $\equiv$ on syntactic judgments which obeys the following axioms:

$$\begin{array}{ll}
(\text{AG1}) & (G_1|G_2)|G_3 \equiv G_1|(G_2|G_3) \\
(\text{AG2}) & G_1|G_2 \equiv G_2|G_1 \\
(\text{AG3}) & G|nil \equiv G \\
(\text{AG4}) & \nu x\ \nu y\ G \equiv \nu y\ \nu x\ G \\
(\text{AG5}) & \nu x\ G \equiv G \text{ if } x \notin \text{fn}(G) \\
(\text{AG6}) & \nu x\ G \equiv \nu y\ G[y/x] \text{ if } y \notin \text{fn}(G) \\
(\text{AG7}) & \nu x\ (G_1|G_2) \equiv (\nu x\ G_1)|G_2 \text{ if } x \notin \text{fn}(G_2)
\end{array}$$

Axioms (AG1), (AG2) and (AG3) define respectively the associativity, commutativity and identity over *nil* for operator $|$. Axioms (AG4) and (AG5) state that nodes of a graph can be hidden only once and in any order. Axiom (AG6) defines α-conversion of a graph with respect to its bound names. Axiom (AG7) defines the interaction between hiding and parallel composition.

Thanks to axiom (AG4), we can write νX, with $X = \bigcup_{i=1 \ldots n}\{x_i\}$, to abbreviate $\nu x_1\ \nu x_2 \ldots \nu x_n$. Note that by using the axioms, we can always put any judgement in the normal form $\Gamma \vdash \nu X\ G$, where G is a subterm containing only compositions of edges. We can also make Γ and X be disjoint sets of nodes. Note that $\text{n}(G) \subseteq \Gamma \cup X$. We can state the following correspondence theorem.

Theorem 2.9 (Correspondence of graphs and judgements)
Well-formed judgements up to structural axioms are isomorphic to graphs up

to isomorphisms.

We present here the steps of an SHR computation.

Definition 2.10 [SHR transition] Let *Act* be a ranked set of actions. The rank of $a \in \mathit{Act}$ is $\mathrm{ar}(a)$.
A SHR transition is of the form:

$$\Gamma \vdash G \xrightarrow{\Lambda,\pi} \Phi \vdash G'$$

where $\Lambda \subseteq \Gamma \times \mathit{Act} \times \mathcal{N}^*$ such that Λ is a total function in its first argument (that is $\Lambda(x_i) = (a_i, \boldsymbol{y}_i)$, and we denote $\boldsymbol{y}_i$ as $\mathrm{n}_\Lambda(x_i)$) and we demand that $\mathrm{ar}(a_i) = |\boldsymbol{y}_i|$ and $\pi : \Gamma \to \Gamma$ is an idempotent substitution. We define:

- $\mathrm{n}(\Lambda) = \{z | \exists x. z \in \mathrm{n}_\Lambda(x)\}$
- $\Phi = \pi(\Gamma) \cup (\mathrm{n}(\Lambda) \setminus \Gamma)$

We require that $\forall x \in \mathrm{n}(\Lambda). \pi(x) = x$ (what means that we communicate only the representatives of the equivalence classes defined by π).

We may drop π if it is the identity. We usually have a trivial action ϵ of arity 0, thus when not otherwise stated we suppose $\Lambda(x) = (\epsilon, \langle\rangle)$ (we write $\Lambda = \Lambda_\epsilon$ if this happens for all x).

We derive SHR transitions from basic productions which define the behaviour of a single hyperedge using some inference rules which depend on the choice of the synchronization model.

Definition 2.11 [SHR production] A production is an SHR transition of the form:

$$x_1, \ldots, x_n \vdash s(x_1, \ldots, x_n) \xrightarrow{\Lambda,\pi} \Phi \vdash G$$

where all x_i are distinct.

Productions have to be considered as schemas, that is if we have a production we also have all productions obtainable from it by applying an injective renaming.

We present here the set of inference rules for Milner synchronization model.

Definition 2.12 [Rules for Milner synchronization]

$$(\mathrm{par}) \quad \frac{\Gamma \vdash G_1 \xrightarrow{\Lambda,\pi} \Phi \vdash G_2 \qquad \Gamma' \vdash G'_1 \xrightarrow{\Lambda',\pi'} \Phi' \vdash G'_2}{\Gamma, \Gamma' \vdash G_1 | G'_1 \xrightarrow{\Lambda \cup \Lambda', \pi \cup \pi'} \Phi, \Phi' \vdash G_2 | G'_2}$$

where $(\Gamma \cup \Phi) \cap (\Gamma' \cup \Phi') = \emptyset$.

$$\text{(merge)}\quad \frac{\Gamma \vdash G \xrightarrow{\Lambda,\pi} \Phi \vdash G'}{\sigma(\Gamma) \vdash \sigma(G) \xrightarrow{\Lambda',\pi'} \Phi' \vdash \nu U\ \rho(\sigma(G'))}$$

where $\sigma : \Gamma \to \Gamma$ is an idempotent substitution and:

- $\sigma(x) = \sigma(y) \wedge \Lambda(x) \neq (\epsilon, \langle\rangle) \wedge \Lambda(y) \neq (\epsilon, \langle\rangle) \wedge x \neq y \Rightarrow$
  $(\forall z \in \mathcal{N} \setminus \{x, y\}.\sigma(z) = \sigma(x) \Rightarrow \Lambda(z) = (\epsilon, \langle\rangle))$
  $\wedge \Lambda(x) = (a, \boldsymbol{v}) \wedge \Lambda(y) = (\overline{a}, \boldsymbol{w}) \wedge a \neq \tau$
- $\rho = \text{mgu}(\{\sigma(\boldsymbol{v}) = \sigma(\boldsymbol{w}) | \sigma(x) = \sigma(y) \wedge \Lambda(x) = (a, \boldsymbol{v}) \wedge \Lambda(y) = (\overline{a}, \boldsymbol{w})\}$
  $\cup \{\sigma(x) = \sigma(y) | \pi(x) = \pi(y)\})$
- $\Lambda'(z) = \begin{cases} (\tau, \langle\rangle) & \text{if } \sigma(x) = \sigma(y) = z \wedge x \neq y \wedge \Lambda(x), \Lambda(y) \neq (\epsilon, \langle\rangle) \\ \rho(\sigma(\Lambda(x))) & \text{if } \sigma(x) = z \wedge \Lambda(x) \neq (\epsilon, \langle\rangle) \\ (\epsilon, \langle\rangle) & \textit{otherwise} \end{cases}$
- $\pi' = \rho|_{\sigma(\Gamma)}$
- $U = \rho(\sigma(\Phi)) \setminus \Phi'$

$$\text{(res)}\quad \frac{\Gamma, x \vdash G \xrightarrow{\Lambda,\pi} \Phi \vdash G'}{\Gamma \vdash \nu x\ G \xrightarrow{\Lambda|_\Gamma,\pi|_\Gamma} \Phi' \vdash \nu Z\ G'}$$

where:

- $(\pi(x) = \pi(y) \wedge x \neq y) \Rightarrow \pi(x) \neq x$
- $\Lambda(x) = (\epsilon, \langle\rangle) \vee \Lambda(x) = (\tau, \langle\rangle)$
- $Z = \{x\}$ if $x \notin \text{n}(\Lambda|_\Gamma), Z = \emptyset$ otherwise

$$\text{(idle)}\quad \Gamma \vdash G \xrightarrow{\Lambda_\epsilon,id} \Gamma \vdash G$$

Rules for Milner synchronization model suppose that actions can be normal actions ("input") or coactions ("output", denoted as $\overline{a}$). Furthermore we have two special actions ϵ (no action) and τ (internal action) of arity 0.

Rule (par) allows the composition of transitions which use disjoint sets of names. Rule (merge) deals with non injective renamings, rule (res) restricts a name where no communication occurs and rule (idle) guarantees that each edge can always make an explicit idle step.

3 Mapping Fusion Calculus into Synchronized Hyperedge Replacement

In this section, we present a mapping from Fusion Calculus to SHR that extends the work of [5] for the π-calculus and maps Fusion Calculus into a particular kind of SHR which uses Milner synchronization and has special rules in order to force an interleaving behaviour. This is necessary since the semantics of Fusion Calculus is itself interleaving. This mapping has the advantage of being natural (there is a one-to-one mapping that maps parallel composition into parallel composition and scope into restriction) and of realizing a complete correspondence between the two models.

3.1 Interleaving Milner SHR

In order to define the inference rules for Milner interleaving SHR we need to restrict ourselves to two kinds of transitions:

- communication transitions: they have $\pi = id$ and $\Lambda(x) = (\epsilon, \langle\rangle)$ for each x but one (we denote this kind of Λ with *act*);
- fusion transitions: they have $\Lambda = \Lambda_\epsilon$.

We present now the particular set of inference rules used in this case.

Definition 3.1 [Rules for interleaving Milner synchronization]

$$(\mathrm{ren})\quad \frac{\Gamma \vdash G \xrightarrow{\Lambda,\pi} \Phi \vdash G'}{\Gamma', \sigma(\Gamma) \vdash \sigma(G) \xrightarrow{\Lambda',\pi'} \Gamma', \rho(\sigma(\Phi)) \vdash \rho(\sigma(G'))}$$

where $\sigma : \Gamma \to \Gamma$ is an idempotent substitution and:

- $\rho = \mathrm{mgu}(\{\sigma(x) = \sigma(y) | \pi(x) = \pi(y)\})$
- $\Lambda'(z) = \begin{cases} \rho(\sigma(\Lambda(x))) \text{ if } \sigma(x) = z \wedge \Lambda(x) \neq (\epsilon, \langle\rangle) \\ (\epsilon, \langle\rangle) \qquad \mathit{otherwise} \end{cases}$
- $\pi' = \rho|_{\sigma(\Gamma)}$

$$(\mathrm{com/close})\quad \frac{\Gamma \vdash G_1 \xrightarrow{act_1,id} \Phi \vdash G_1' \qquad \Gamma \vdash G_2 \xrightarrow{act_2,id} \Phi' \vdash G_2'}{\Gamma \vdash G_1|G_2 \xrightarrow{\Lambda_\epsilon,\pi} \rho(\Gamma) \vdash \nu U\ \rho(G_1'|G_2')}$$

where:

- $act_1 = (x, a, \boldsymbol{y}_1)$ and $act_2 = (x, \overline{a}, \boldsymbol{y}_2)$ for some node x and pair action-coaction a and $\overline{a}$
- $\rho = \text{mgu}(\{\boldsymbol{y}_1 = \boldsymbol{y}_2\})$
- $\pi = \rho|_\Gamma$
- $U = \rho(\Phi \cup \Phi') \setminus \rho(\Gamma)$

$$\text{(par)} \quad \frac{\Gamma \vdash G_1 \xrightarrow{\Lambda,\pi} \Phi \vdash G_1' \qquad \Gamma \vdash G_2}{\Gamma \vdash G_1|G_2 \xrightarrow{\Lambda,\pi} \Phi \vdash G_1'|\pi(G_2)}$$

$$\text{(res)} \quad \frac{\Gamma, x \vdash G \xrightarrow{\Lambda,\pi} \Phi \vdash G'}{\Gamma \vdash \nu x\ G \xrightarrow{\Lambda|_\Gamma,\pi|_\Gamma} \Phi' \vdash \nu Z\ G'}$$

where:

- $(\pi(x) = \pi(y) \land x \neq y) \Rightarrow \pi(x) \neq x$
- $\Lambda(x) = (\epsilon, \langle\rangle)$
- $Z = \{x\}$ if $x \notin \text{n}(\Lambda|_\Gamma), Z = \emptyset$ otherwise

Rule (ren) deals with renamings of names in the interface and allows also the creation of new isolated nodes, rule (com/close) synchronizes two communication transitions exposing complementary actions on the same node to give a fusion transition, rule (par) adds an idle part of the system to the transition and rule (res) deals with restriction.

Note that with respect to the transition system obtained using rules in 2.12 we drop the τ action (which is substituted by the ϵ action) and that all the transitions that we obtain starting from a set of communication and fusion productions are either communication ones or fusion ones, that is they all have just one effect. This makes the semantics interleaving.

We can state the following correspondence theorem between normal SHR and interleaving SHR.

Theorem 3.2 *Given a set of communication and fusion productions $\mathcal{P}$ and a starting graph $\Gamma \vdash G$ we can derive a transition $\Gamma \vdash G \xrightarrow{\Lambda,\pi} \Phi \vdash G'$ using the inference rules in definition 3.1 iff we can derive the transition $\Gamma \vdash G \xrightarrow{\Lambda',\pi} \Phi \vdash G'$ where $\Lambda(x) = (\epsilon, \langle\rangle)$ if $\Lambda'(x) = (\tau, \langle\rangle)$ and $\Lambda(x) = \Lambda'(x)$ otherwise using the rules in definition 2.12 and either one production or two communication productions which synchronize on some node.*

3.2 Mapping Fusion into interleaving SHR

One of the differences between Fusion Calculus and SHR is that in one case fusions are represented by equivalence relations, whereas in the other one substitutions are used. The relation between the two representations is the following one.

Definition 3.3 A substitutive effect of a fusion ϕ is an idempotent substitution σ agreeing with ϕ (i. e. σ sends all members of each equivalence class of ϕ to one representative in the class).

We define now the translation between Fusion Calculus and interleaving SHR with Milner synchronization.

Definition 3.4 [Translation of actions] We define now the relation between the actions of Fusion Calculus and the synchronizations of SHR. This relation is a function that we denote with $[\![-]\!]$ in the case of communication actions (we have a free action when $\boldsymbol{y} = \emptyset$):
$[\![(\boldsymbol{y})u\boldsymbol{x}]\!] = (u, in_n, \boldsymbol{x}), id$ where $n = |\boldsymbol{x}|$
$[\![(\boldsymbol{y})\overline{u}\boldsymbol{x}]\!] = (u, out_n, \boldsymbol{x}), id$ where $n = |\boldsymbol{x}|$
We define in_n and out_n as complementary actions, i. e. $in_n = \overline{out_n}$ and $out_n = \overline{in_n}$.
A ϕ action of Fusion Calculus corresponds in the SHR setting to any substitution π which is a substitutive effect of ϕ.

Note that in the translation the distinction between free names and bound names vanishes, but for any transition $P \xrightarrow{\alpha} P'$ $\mathrm{bn}(\alpha) = \mathrm{n}(\alpha) \setminus \mathrm{fn}(P)$, thus we can retrieve the set of bound names when necessary.

Since we want a finite number of productions that describe the evolution of each process we use edges and productions only for sequential processes in standard form. In particular, each edge has a label which encapsulate the corresponding process.

Definition 3.5 [Translation of agents] We define now the translation on agents:
$[\![0]\!] = nil$
$[\![S]\!] = L_{\hat{S}}(\mathrm{fnarray}(S))$ where $\hat{S}$ is the process that corresponds to the standard agent for S
$[\![P_1|P_2]\!] = [\![P_1]\!]|[\![P_2]\!]$
$[\![(x)P]\!] = \nu x\ [\![P]\!]$
$[\![\mathrm{rec}\, X.P]\!] = [\![P\{\mathrm{rec}\, X.P/X\}]\!]$

We have no need to translate agent variables since they occur only inside sequential processes.

The translation is well-defined also on equivalence classes, i. e. on processes, since if $P_1 \equiv P_2$ then $[\![P_1]\!] \equiv [\![P_2]\!]$. Note that if a congruence rule is applied inside a sequential agent then the translation of the two agents is the same since the edge is labelled by the process itself. Furthermore abelian monoid laws for $|$ correspond to rules (AG1), (AG2) and (AG3), $(x)0 = 0$ corresponds to an instance of (AG5), $(x)(y)P = (y)(x)P$ corresponds to (AG4) and the scope extension law corresponds to (AG7). Finally α-conversion for names is (AG6). The recursion law holds since the translation of a recursively defined agent coincides with the translation of its expansion using recursion law. Note that even if this part of the definition is not by structural induction, still the termination is granted because we use only guarded recursion. Thus, after one application of the rule, we get a sequential agent, whose translation is directly defined. Finally laws for $+$ and α-conversion for agent variables can be applied only inside sequential agents.

Note also that rule (AG5) corresponds to a rule that can be obtained from $P|0 = P$, the scope extension law and $(x)0 = 0$. Thus both formalisms have the same underlying structure, that is a parallel composition operator and a binder for names, ruled by equivalent congruences.

As a last step we need to show the productions used in the SHR system.

Definition 3.6 We define productions only for standard sequential agents $\sum_i \alpha_i.P_i$. Let Γ be $\mathrm{fn}([\![\sum_i \alpha_i.P_i]\!])$.
The productions are of the following forms:

$$\Gamma \vdash [\![\sum_i \alpha_i.P_i]\!] \xrightarrow{[\![\alpha_i]\!]} \Gamma \vdash [\![P_i]\!]$$

if α_i is a communication action (note that this is a communication production) and

$$\Gamma \vdash [\![\sum_i \alpha_i.P_i]\!] \xrightarrow{\pi} \pi(\Gamma) \vdash [\![\pi(P_i)]\!]$$

if $\alpha_i = \phi$ and π is a fusion effect of ϕ (note that this is a fusion production).

Note that since, in SHR computations, nodes are never deleted, the graphs that we obtain during the computation have some more unused nodes.

The correspondence between the different methods used for computing fusions in the two systems is given by the following lemma.

Lemma 3.7 *Let S be a set of equalities representing a fusion ϕ and σ be a substitution. Then the following two propositions are equivalent:*

- *σ is a substitutive effect of ϕ;*
- *σ is an mgu of S.*

An important characteristics of graphs obtained from Fusion Calculus is that we have no generation of new names as shown by the following lemma (but we can have extrusions of bound names), we can however have infinite many names since the axiom for recursion allows a finite representation of an infinite agent (which may contain infinitely many names).

Lemma 3.8 *For each process P if $P \xrightarrow{\alpha} P'$ for some action α then all the free names that appears in α are also free names of $[\![P]\!]$.*

We can now state the two theorems that show the correctness and the completeness of the translation.

Theorem 3.9 (Correctness) *For each fusion process P and each $\Gamma \supseteq \mathrm{fn}([\![P]\!])$ if $P \xrightarrow{\alpha} P'$ and $\Gamma \cap \mathrm{bn}(\alpha) = \emptyset$ then:*

(i) $\Gamma \vdash [\![P]\!] \xrightarrow{[\![\alpha]\!]} \Gamma, \Gamma_E \vdash [\![P']\!]$ *where $\Gamma_E = \mathrm{bn}(\alpha)$ if α is a communication action;*

(ii) $\Gamma \vdash [\![P]\!] \xrightarrow{\pi} \pi(\Gamma) \vdash [\![\pi(P')]\!]$ *for each substitutive effect π of α if α is a fusion action.*

Theorem 3.10 (Completeness) *For each fusion process P if $\Gamma \vdash [\![P]\!] \xrightarrow{\Lambda,\pi} \Gamma \vdash G$ for some $\Gamma \supseteq \mathrm{fn}([\![P]\!])$ then there exists P' such that:*

(i) *either $P \xrightarrow{\alpha} P'$, $[\![\alpha]\!] = (\Lambda, id)$, $\pi = id$, $[\![P']\!] = \Gamma' \vdash G$ and $\Gamma' = \Gamma, \Gamma_E$ where $\Gamma_E = \mathrm{bn}(\alpha)$;*

(ii) *or $P \xrightarrow{\phi} P'$, π is a substitutive effect of ϕ, $\Lambda = \Lambda_\epsilon$, $[\![\pi(P')]\!] = G$ and $\Gamma' = \pi(\Gamma)$.*

We can also handle the match operator by distributing it on sequential agents and then adding to SHR a special rule for dealing with edges that correspond to sequential processes with match.

We show an example on how the translation can be used.

Example 3.11 Let us consider the fusion transition:
$(z)(P|\overline{u}xy.Q|uzw.R) \xrightarrow{\{y=w\}} (P|Q|R)\{x/z\}$.
The translation of the starting process into a graph is:
$\nu z\ L_P | L_{\overline{x_1}x_2x_3.Q}(u,x,y) | L_{x_1x_2x_3.R}(u,z,w)$ and we can choose $\Gamma = u, x, y, w$.
Furthermore we have the following productions:

$$x_1, x_2, x_3 \vdash L_{\overline{x_1}x_2x_3.Q}(x_1, x_2, x_3) \xrightarrow{(x_1, out_2, \langle x_2, x_3 \rangle)} x_1, x_2, x_3 \vdash L_Q$$

$$x_1, x_2, x_3 \vdash L_{x_1x_2x_3.R}(x_1, x_2, x_3) \xrightarrow{(x_1, in_2, \langle x_2, x_3 \rangle)} x_1, x_2, x_3 \vdash L_R$$

By applying rule (ren) we can derive:

$$u,x,y,z,w \vdash L_{\overline{x_1}x_2x_3.Q}(u,x,y) \xrightarrow{(u,out_2,\langle x,y\rangle)} u,x,y,z,w \vdash L_Q$$
$$u,x,y,z,w \vdash L_{x_1x_2x_3.R}(u,z,w) \xrightarrow{(u,in_2,\langle z,w\rangle)} u,x,y,z,w \vdash L_R$$

We can then apply rules (com/close), (par) and (res) to have:

$$\begin{aligned}
&u,x,y,z,w \vdash L_{\overline{x_1}x_2x_3.Q}(u,x,y)|L_{x_1x_2x_3.R}(u,z,w)\\
&\qquad\xrightarrow{\{x/z,y/w\}} u,x,y \vdash (L_Q|L_R)\{x/z,y/w\}\\
&u,x,y,z,w \vdash L_P|L_{\overline{x_1}x_2x_3.Q}(u,x,y)|L_{x_1x_2x_3.R}(u,z,w)\\
&\qquad\xrightarrow{\{x/z,y/w\}} u,x,y \vdash (L_P|L_Q|L_R)\{x/z,y/w\}\\
&u,x,y,w \vdash \nu z\ L_P|L_{\overline{x_1}x_2x_3.Q}(u,x,y)|L_{x_1x_2x_3.R}(u,z,w)\\
&\qquad\xrightarrow{\{y/w\}} u,x,y \vdash (L_P|L_Q|L_R)\{x/z,y/w\}
\end{aligned}$$

as desired since $\{y/w\}$ is a substitutive effect of $\{y=w\}$.

Here we show how SHR can be used to execute many Fusion Calculus transitions in one step.

Example 3.12 Let us consider the fusion process $(xy)(ux.P|yx.Q|\overline{y}z.R)$. We can have the two following computations:

$$\begin{aligned}
(xy)(ux.P|yx.Q|\overline{y}z.R) &\xrightarrow{(x)ux} (y)(P|yx.Q|\overline{y}z.R)\\
&\xrightarrow{\{x=z\}} (y)(P|Q|R)\\
(xy)(ux.P|yx.Q|\overline{y}z.R) &\xrightarrow{1} (y)((ux.P|Q|R)\{z/x\})\\
&\xrightarrow{uz} (y)((P|Q|R)\{z/x\})
\end{aligned}$$

Using the translation and the inference rules of definition 3.1 we have the two corresponding sequences of transitions:

$$\begin{aligned}
&u,z \vdash \nu x,y\ L_{x_1x_2.P}(u,x)|L_{x_1x_2.Q}(y,x)|L_{\overline{x_1}x_2.R}(y,z)\\
&\quad\xrightarrow{(u,in_1,x)} u,z,x \vdash \nu y\ L_P|L_{x_1x_2.Q}(y,x)|L_{\overline{x_1}x_2.R}(y,z)\\
&\quad\xrightarrow{\{z/x\}} u,z \vdash \nu y\ (L_P|L_Q|L_R)\{z/x\}
\end{aligned}$$

$$\begin{aligned}
&u,z \vdash \nu x,y\ (L_{x_1x_2.P}(u,x)|L_{x_1x_2.Q}(y,x)|L_{\overline{x_1}x_2.R}(y,z))\\
&\quad\longrightarrow u,z \vdash \nu y\ (L_{x_1x_2.P}(u,x)|L_Q|L_R)\{z/x\}\\
&\quad\xrightarrow{(u,in_1,z)} u,z \vdash \nu y\ (L_P|L_Q|L_R)\{z/x\}
\end{aligned}$$

Note that in this case the two results are equal, even if the observations are different because of the different ordering on the transitions. Furthermore if we choose the set of inference rules of definition 2.12 we can also have the transition:

$$u, z \vdash \nu x, y\ (L_{x_1 x_2.P}(u, x) | L_{x_1 x_2.Q}(y, x) | L_{\overline{x_1} x_2.R}(y, z))$$
$$\xrightarrow{(u, in_1, z)} u, z \vdash \nu y\ (L_P | L_Q | L_R)\{z/x\}$$

which corresponds to the concurrent execution of the two transitions.

We end this section with a simple schema on the correspondence between the two models.

Fusion	SHR	Fusion	SHR
Process	Graph	Transition	Transition
Sequential process	Edge	Parallel comp.	Parallel comp.
Name	Node	Scope	Restriction
Prefix execution	Production	0	Nil

4 Conclusions

We have analyzed the relationships between Fusion Calculus and SHR, showing that they both rely on the same underlying algebraic structure and on Milner synchronization (but SHR can also use other kinds of synchronization models) and that the main difference is the interleaving behaviour of Fusion Calculus with respect to the concurrent nature of SHR. Thus we have devised an interleaving version of SHR with Milner synchronization and proved a correspondence theorem between the two models.

In literature we have two similar works, which map respectively π-calculus [5] and Ambient calculus [3] into SHR. With respect to [3] we use the same mobility mechanism (that is we allow fusions of any node) but we do not need multiple synchronizations. Furthermore ambients have poor communication mechanisms and use only tree-structured processes, whereas we have general graphs. With respect to [5] and the π-calculus we allow general fusions of names. This is important since this is the same kind of mobility used in logic programming, as shown in [7], and thus this work is a first step from process-calculi with fusions towards logic programming.

As future work we want to go ahead in this direction, in particular analyzing the relationship between Hoare synchronization (which is used in logic programming) and Milner synchronization (which is used in the most popular process-calculi). This will bridge the gap between Fusion Calculus and logic programming.

Another direction for future developments is the transfer of techniques between Fusion Calculus and SHR. In particular it would be interesting to extend the concept of hyperequivalence to SHR (keeping the property of being a congruence) and, in the opposite direction, using SHR in order to provide a concurrent semantics to Fusion Calculus.

References

[1] Pierpaolo Degano and Ugo Montanari. A model for distributed systems based on graph rewriting. *Journal of the ACM (JACM)*, 34(2):411–449, 1987.

[2] H. Ehrig, H.-J. Kreowski, U. Montanari, and G. Rozenberg, editors. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol.3: Concurrency, Parallellism, and Distribution.* World Scientific, 1999.

[3] GianLuigi Ferrari, Ugo Montanari, and Emilio Tuosto. A lts semantics of ambients via graph synchronization with mobility. In Simona Ronchi Della Rocca Antonio Restivo and Luca Roversi, editors, *Proc. of ICTCS'01*, volume 2202 of *LNCS*, pages 1–16. Springer, October 2001.

[4] Dan Hirsch. *Graph Transformation Models for Software Architecture Styles.* PhD thesis, Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Argentina, 2003.

[5] Dan Hirsch and Ugo Montanari. Synchronized hyperedge replacement with name mobility. In Springer Verlag, editor, *Proc. of CONCUR'01*, LNCS, 2001.

[6] Barbara König and Ugo Montanari. Observational equivalence for synchronized graph rewriting. In Springer Verlag, editor, *Proc. TACS'01*, volume 2215 of *LNCS*, January 2001.

[7] Ivan Lanese. Process synchronization in distributed systems via Horn clauses. Master's thesis, University of Pisa, Computer Science Department, 2002. Downloadable from `http://www.di.unipi.it/~lanese/tesi.ps`.

[8] Ivan Lanese and Ugo Montanari. Software architectures, global computing and graph transformation via logic programming. In Leila Ribeiro, editor, *Proc SBES'2002 - 16th Brazilian Symposium on Software Engineering*, pages 11–35. Anais, 2002.

[9] Robin Milner. A calculus of communicating systems. In *Lect. Notes in Comp. Sci.*, volume 92. Springer, Berlin, 1989.

[10] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes. *Inform. and Comput.*, 100:1–77, 1992.

[11] Joachim Parrow and Björn Victor. The fusion calculus: Expressiveness and symmetry in mobile processes. In *Proceedings of LICS '98.* IEEE, Computer Society Press, June 1998.

[12] G. Rozenberg, editor. *Handbook of graph grammars and computing by graph transformations, vol. 1: Foundations.* World Scientific, 1997.