To appear in the *Proceedings of TACS2001*, Sendai, October 28–31, 2001.
Copyright 2001 Springer Verlag

Solo Diagrams

Cosimo Laneve* Joachim Parrow† Björn Victor‡

July 2001

Abstract

We address the problems of implementing the replication operator efficiently in the solos calculus—a calculus of mobile processes without prefix. This calculus is expressive enough to admit an encoding of the whole fusion calculus and thus the π-calculus. We show that nested occurrences of replication can be avoided, that the size of replicated terms can be limited to three particles, and that the usual unfolding semantics of replication can be replaced by three simple reduction rules. To illustrate the results and show how the calculus can be efficiently implemented we present a graphic representation of agents in the solos calculus, adapting ideas from interaction diagrams and pi-nets.

1 Introduction

The π-calculus [MPW92] has proved remarkably successful in modelling diverse computational phenomena, and it is natural to ask how much of it is really necessary in order to attain the expressive power. This has led to several interesting and expressive subcalculi. For example, in the more easily implemented asynchronous subcalculus [Bou92, HT91] the output prefix $\overline{u}\,v\,.\,P$ is replaced by the output particle $\overline{u}\,v$. In the fusion calculus [PV98] the reduction of an input and output results in a *fusion* of names rather than a substitution. In that calculus both input and output prefix can be replaced by their corresponding particles, in other words, there is no need for explicit representation of temporal precedence. These particles are called *solos* and take the general forms $u\,\widetilde{x}$ for input and $\overline{u}\,\widetilde{x}$ for output, where $\widetilde{x}$ is a sequence

*Dept. of Computer Science, University of Bologna, Italy. laneve@CS.UniBO.IT
†Royal Institute of Technology, Kista, Sweden. joachim@it.kth.se
‡Dept. of Computer Systems, Uppsala University, Sweden. victor@DoCS.UU.SE

of names. This *solos calculus* additionally includes only parallel composition $P \mid Q$, scoping $(x)P$ and replication $!\,P$, giving a very lean formalism. We refer the reader to [LV99] for further explanation of the expressive power of solos.

The replication operator $!\,P$ is often used in place of recursive definitions, since it has nice algebraic properties. For example, if the number of recursive definitions is finite, recursion can be coded in terms of replication [Mil93]. Replication can be defined in terms of unguarded recursion: $!\,P \stackrel{\mathit{def}}{=} P \mid !\,P$. This definition is, however, hard to implement—when should the unfolding stop? How many (possibly nested) replications need be expanded in order to infer a reduction? For the π-calculus Sangiorgi has shown [San98] that replication of general agents is not necessary, but it can be replaced by the guarded variant $!\,\alpha\,.\,P$. This corresponds to using guarded recursion $(!\,\alpha\,.\,P \stackrel{\mathit{def}}{=} \alpha\,.\,(P \mid !\,\alpha\,.\,P))$. In languages based on the asynchronous communication such as Pict [PT00] or Join [FG96], it is relatively easy to see that it suffices to have input-guarded replication of the form $!\,x(y)\,.\,P$.

These guarded variants of replication cannot be used in the solos calculus, where there are no prefix-guarded terms present, but only solos. However, we can replace the unguarded unfolding of $!\,P \stackrel{\mathit{def}}{=} P \mid !\,P$ with three reduction rules which pinpoint when a reduction involving a replicated agent can be inferred. We show that the new formulation of the semantics coincides with the standard one. This result rests on a flattening law, which allows us to remove nested replications.

Another problem with implementing replication is that the term being unfolded may be arbitrarily large, making interaction with a replicated term computationally expensive. We address this problem by presenting a decomposition law allowing us to limit the size of replicated terms to three solos.

The resulting formalism is thus very slender, and provides a simple canonical form for agents: $(\widetilde{x})(P \mid \prod_{i \in I} !\,(\widetilde{y_i})Q_i)$ where P and Q_i are compositions of solos, and $\widetilde{y_i}$ is a subset of the names in Q_i. We argue that this calculus can be easily and efficiently implemented, and illustrate both this fact and the general results using a new graphical formalism for the solos calculus, the *solo diagrams*.

The underlying idea is quite simple. To draw an agent, pick one node for each name, and label the node with the name. Outputs $\overline{u}\,\widetilde{x}$ become multi-edges from the nodes labelled $\widetilde{x}$ to the node labelled u, and conversely for inputs. Parallel composition is just graph union and scope restriction erases the label of nodes. Reductions between inputs and outputs are possible when two types of edges meet at the same node, and results in the corresponding object nodes being fused or merged together, preserving any additional edges connecting them. As an example, Figure 1 shows a simple agent and reduction. Note that the two kinds of edges are distinguished by the shape of the arrow which has either a head or a tail.

In the first three figures, dotted lines indicate which solos correspond to which edges. These lines are not technically part of the diagrams.

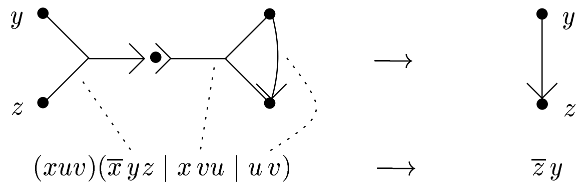


Figure 1: A simple diagram reduction and its correponding term reduction.

As a further example, Figure 2 illustrates an agent with a nondeterministic behaviour: depending on which output solo is scheduled for the reduction, the agent produces two alternative graphs.

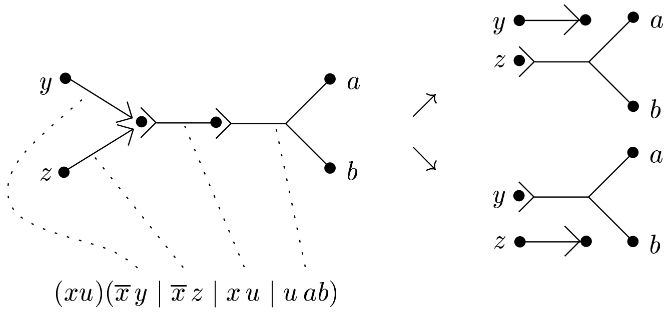
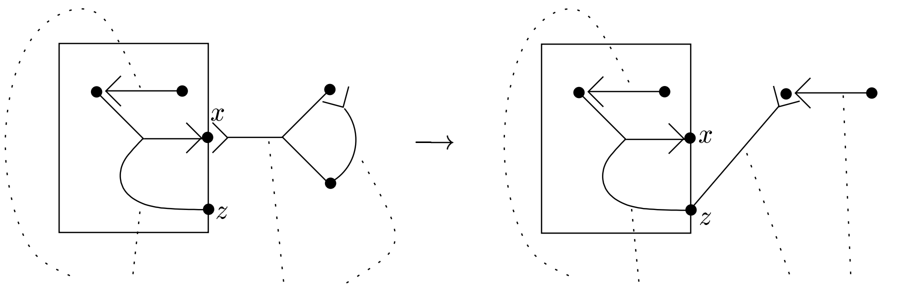


Figure 2: The reductions of the diagram corresponding to $(xu)(\overline{x}\,y \mid \overline{x}\,z \mid x\,u \mid u\,ab)$

Replication is modelled by boxes. These are drawn as rectangles surrounding the replicated graph, and are regarded as copying-machines for edges and nodes therein: when an edge of a box is involved in a reduction, all the content of the box is first copied. An example is shown in Figure 3. Here, the rightmost edge of the right-hand diagram is the remainder of the copy of the box after the reduction.

We continue by presenting the solos calculus; in Section 3 we introduce the graphical formalism. In Section 4 we formally relate this to the solos calculus; in particular, the usual structural congruence in the solos calculus is shown



$$!\,(yw)(\overline{y}\,w \mid \overline{x}\,yz) \mid (uv)(x\,uv \mid u\,v) \quad\longrightarrow\quad !\,(yw)(\overline{y}\,w \mid \overline{x}\,yz) \mid (uw')(u\,z \mid \overline{u}\,w')$$

Figure 3: The reduction of a diagram with a box, and its corresponding term reduction.)

to coincide with diagram isomorphism, and the solo diagram reductions are shown to yield the same reductions as the standard unfolding approach. In Section 5 we consider a generalisation to graphs with multiply labelled nodes, giving a finer semantics. The paper concludes with a discussion of related work.

2 The Solos Calculus

Syntax. We assume an infinite set $\mathcal{N}$ of *names* ranged over by $u, v, x, z, \cdots$. Names represent communication channels, which are also the values transmitted. We write $\widetilde{x}$ for a (possibly empty) finite sequence $x_1 \cdots x_n$ of names.

The *solos* α are *inputs* $u\,\widetilde{x}$ and *outputs* $\overline{u}\,\widetilde{x}$. The names $\widetilde{x}$ are the *objects* of the solo, and the name u is the *subject*.

The *agents* $P, Q, R, \cdots$ are defined by

$$P ::= \underset{(\mathit{Inaction})}{\mathbf{0}} \;\Big|\; \underset{(\mathit{Solo})}{\alpha} \;\Big|\; \underset{(\mathit{Composition})}{P \mid P} \;\Big|\; \underset{(\mathit{Scope})}{(x)P} \;\Big|\; \underset{(\mathit{Replication})}{!\,P}$$

The name x is said to be *bound* in $(x)P$. We write $(\widetilde{x})P$ for $(x_1)\cdots(x_n)P$, $n \geq 0$, and often $\prod_{i \in I} P_i$ for the composition of all P_i for $i \in I$, where I is a finite set. The *free names* in P, denoted *fn*(P), are the names in P with a non-bound occurrence. The (choice-free) fusion calculus [PV98] consists of the above agents and those formed using the prefix operator, namely agents of the form $\alpha\,.\,P$.

Operational semantics. We begin by defining a structural congruence which equates all agents we will never want to distinguish for any semantic reason, and then use this when giving the operational semantics.

$$(\widetilde{z})(\overline{u}\,\widetilde{x} \mid u\,\widetilde{y} \mid P) \longrightarrow P\sigma$$

$$\frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q} \qquad \frac{P \longrightarrow P'}{(x)P \longrightarrow (x)P'} \qquad \frac{P \equiv Q \quad Q \longrightarrow Q' \quad Q' \equiv P'}{P \longrightarrow P'}$$

Side conditions in the first rule:
$|\widetilde{x}| = |\widetilde{y}|, \sigma$ agrees with $\{\widetilde{x} = \widetilde{y}\}, ran(\sigma) \cap \widetilde{z} = \emptyset,$ and $dom(\sigma) = \widetilde{z}$.

Figure 4: Reduction rules for the calculus of solos.

Definition 1 *The* structural congruence, $\equiv$, *between agents is the least congruence satisfying the abelian monoid laws for Composition, alpha-renaming, and the laws for scope and replication*

$$\begin{gathered}(x)\mathbf{0} \equiv \mathbf{0}, \quad (x)(y)P \equiv (y)(x)P, \\ P \mid (z)Q \equiv (z)(P \mid Q), \quad \textit{if } z \notin \mathit{fn}(P) \\ !\,P \equiv P \mid !\,P\end{gathered}$$

The reduction relation of the calculus of solos is the least relation satisfying the rules in Figure 4. Here and in the following σ will range over name substitutions, and we use $dom(\sigma) = \{u : \sigma(u) \neq u\}$ and $ran(\sigma) = \{\sigma(u) : \sigma(u) \neq u\}$. We write $\{\widetilde{x} = \widetilde{y}\}$ for the smallest total equivalence relation on $\mathcal{N}$ relating each x_i with y_i, and say that σ *agrees with* the equivalence φ if $\forall x, y : x\,\varphi\,y \Leftrightarrow \sigma(x) = \sigma(y)$. We say that two names are *fused* by a reduction if they are made equal by the resulting substitution. The side condition of Figure 4 bans reductions that fuse two different free names. For instance, the agent $\overline{x}\,y \mid x\,z$ is irreducible. See Section 5 and [PV98] for alternative semantics allowing such reductions.

Equivalence. To define an extensional semantics, we use the standard notion of *barbed bisimulation* developed in [MS92].

Definition 2 *The* observation relation *is the least relation satisfying the rules below.*

$$\begin{array}{lll} x\,\widetilde{y} \downarrow x & (P \mid Q) \downarrow x & \textit{if } P \downarrow x \textit{ or } Q \downarrow x \\ \overline{x}\,\widetilde{y} \downarrow x & (z)P \downarrow x & \textit{if } P \downarrow x \textit{ and } x \neq z \end{array}$$

Definition 3 *A* weak barbed bisimulation *is a symmetric binary relation* $\mathcal{S}$ *between agents such that* $P\;\mathcal{S}\;Q$ *implies:*

1. *If* $P \longrightarrow P'$ *then* $Q \longrightarrow^* Q'$ *and* $P' \mathcal{S} Q'$*.*
2. *If* $P \downarrow x$ *for some* x*, then* $Q \longrightarrow^* \downarrow x$*.*

P *is* barbed bisimilar *to* Q*, written* $P \stackrel{\centerdot}{\approx} Q$*, if* $P \mathcal{S} Q$ *for some weak barbed bisimulation* $\mathcal{S}$*.* P *is* barbed congruent *to* Q*, written* $P \approx Q$*, if for all contexts* $C[\cdot]$*,* $C[P] \stackrel{\centerdot}{\approx} C[Q]$*.*

The solos calculus, although simple, is expressive enough. The next theorem recalls a result in [LV99].

Theorem 4 *There exists an encoding* $[\![\cdot]\!]$ *of the fusion calculus into the calculus of solos such that* $[\![P]\!] \approx [\![Q]\!]$ *implies* $P \approx Q$*, and* $P \stackrel{\centerdot}{\approx} Q$ *implies* $[\![P]\!] \stackrel{\centerdot}{\approx} [\![Q]\!]$*.*

This result only gives full abstraction up-to encoded contexts, i.e., $C[P] \stackrel{\centerdot}{\approx} C[Q]$ iff $[\![C[P]]\!] \stackrel{\centerdot}{\approx} [\![C[Q]]\!]$ for any fusion calculus context $C[\cdot]$.

2.1 The implementation of the replication operator

Although the standard definition of the replication operator is algebraically elegant, an unconstrained implementation of the equality $!\,P \equiv P \mid !\,P$ would quickly give an "out-of-memory error". This problem is well-known in implementations of mobile calculi. In Pict, for instance, the authors implement the so-called *replicated input* [PT00]. We cannot use the same machinery because the solos calculus has no prefix operator. Instead, we use a definition of the replication operator which is closer to a realistic implementation.

We begin by showing that nested replication may be flattened into non-nested replications [LV01]:

Theorem 5 (Flattening)

$$!\,(\widetilde{x})(P \mid !\,Q) \quad \approx \quad (y)(!\,(\widetilde{x})(P \mid \overline{y}\,\widetilde{z}) \mid !\,(\widetilde{w})(y\,\widetilde{w} \mid Q\{\widetilde{w}/\widetilde{z}\}))$$

where $\widetilde{z} = \mathrm{fn}(Q)$ *and* y *and* $\widetilde{w}$ *are fresh.*

Corollary 6 *For all* P *there exists a* Q *such that it does not contain nested replications, and* $P \approx Q$*.* Proof: By structural induction on P. The only interesting case is where P is on the form matching the left-hand side of the equality in Theorem 5.

$$(\widetilde{z})(P \mid \overline{u}\,\widetilde{y} \mid !\,(\widetilde{w})(u\,\widetilde{x} \mid Q)) \longrightarrow_r (\widetilde{w})(P \mid Q \mid !\,(\widetilde{w})(u\,\widetilde{x} \mid Q))\sigma$$

$$(\widetilde{z})(P \mid !\,(\widetilde{v})(\overline{u}\,\widetilde{y} \mid Q) \mid !\,(\widetilde{w})(u\,\widetilde{x} \mid R)) \\ \longrightarrow_r (\widetilde{v}\widetilde{w})(P \mid Q \mid R \mid !\,(\widetilde{v})(\overline{u}\,\widetilde{y} \mid Q) \mid !\,(\widetilde{w})(u\,\widetilde{x} \mid R))\sigma$$

$$(\widetilde{z})(P \mid !\,(\widetilde{w})(\overline{u}\,\widetilde{y} \mid u\,\widetilde{x} \mid Q)) \longrightarrow_r (\widetilde{w})(P \mid Q \mid !\,(\widetilde{w})(\overline{u}\,\widetilde{y} \mid u\,\widetilde{x} \mid Q))\sigma$$

Side conditions

In every rule:	$\lvert\widetilde{x}\rvert = \lvert\widetilde{y}\rvert, \sigma$ agrees with $\{\widetilde{x} = \widetilde{y}\}$, and $dom(\sigma) \cap ran(\sigma) = \emptyset$
In the first rule:	u and $\overline{u}$ may be interchanged, $u \notin \widetilde{w}, \widetilde{w} \cap fn(P) = \emptyset$, and $\widetilde{z} \subseteq dom(\sigma) \subseteq \widetilde{z} \cup \widetilde{w}$
In the second rule:	$u \notin \widetilde{v} \cup \widetilde{w}, (\widetilde{v} \cup \widetilde{w}) \cap fn(P) = \widetilde{v} \cap fn(R) = \widetilde{w} \cap fn(Q) = \emptyset$, and $\widetilde{z} \subseteq dom(\sigma) \subseteq \widetilde{z} \cup \widetilde{v} \cup \widetilde{w}$
In the third rule:	$\widetilde{z} \subseteq dom(\sigma) \subseteq \widetilde{z} \cup \widetilde{w}$

Figure 5: Reduction rules for replication.

In view of this corollary there is no substantial loss of generality to only consider non-nested replication. Therefore we from now on adopt the restriction that in $!\,Q$, the agent Q may not contain replications. This has several advantages. For example there is an attractive kind of canonical form:

Proposition 7 *Every agent P is structurally equivalent to an agent of the form $(\widetilde{x})(Q \mid (\prod_{i\in I} !\,(\widetilde{x}_i)Q_i))$, where Q, Q_i are compositions of solos, $\widetilde{x}_i \subseteq fn(Q_i)$ and $\widetilde{x} \subseteq fn(Q \mid (\prod_{i\in I} !\,(\widetilde{x}_i)Q_i))$.*

Notwithstanding these simplifications, the implementation difficulties coming from the equality $!\,P \equiv P \mid !\,P$ are almost unchanged. Therefore we give an alternative semantics of the replication operator, which has the same formal power as the standard replication but is more easily implemented.

Let $\equiv_r$ be the structural congruence introduced earlier in this section, without the rule $!\,P \equiv P \mid !\,P$. Let $\longrightarrow_r$ be the $\longrightarrow$ reduction rule, where $\equiv_r$ replaces $\equiv$ in the last rule, and with the three rules in Figure 5. These reduction rules account for interactions between two solos when (1) exactly one is replicated, (2) both are under different replications, and (3) both are under the same replication. Note that the outermost sequence of scopes in the right-hand-side of the rules may contain redundant scopes; these can be removed by structural congruence. The scopes of $\widetilde{z}$ are removed since $ran(\sigma) \cap \widetilde{z} = \emptyset$ just like in Figure 4.

The rules in Figure 5 implement a lazy usage of replications: a replicated process is duplicated only if it is used in a reduction. Their correctness with respect to the standard replication operator strongly relies on considering non-nested replications:

Proposition 8 *If P has only non-nested replications, then $P \longrightarrow Q$ if and only if $P \longrightarrow_r R$ and $Q \equiv R$.*

This proposition states that if an agent moves according to $\longrightarrow$, possibly by unfolding replications, then it moves according to $\longrightarrow_r$ without any unfolding of replication at all (and the other way around). If the agent had a nested replication, for instance $!\,!\,P$, the correspondence between $\longrightarrow$ and $\longrightarrow_r$ fails, because $\longrightarrow_r$-reduction is only defined for non-nested replication. The correspondence between $\longrightarrow$ and $\longrightarrow_r$ is proved by induction on the depth of the proofs of $\longrightarrow$ and $\longrightarrow_r$. Each time a replica is used in $\longrightarrow$ then one of the reductions in Figure 5 may be used instead; and, *vice versa*, when one reduction of Figure 5 is used in $\longrightarrow_r$, then its effects may be simulated by means of the replication law and the basic reduction of $\longrightarrow$.

3 Solo Diagrams

We now introduce a graphical formalism for solos, the *solo diagrams*. Diagrams are built out of an infinite set $\mathcal{U}$ of *nodes*, ranged over by $a, b, \ldots$.

Definition 9 *An* edge *is a tuple in $\mathcal{U}$. There are two kinds of edges: input edges $\langle a, a_1, \cdots, a_k\rangle_i$ and output edges $\langle a, a_1, \cdots, a_k\rangle_o$.*

Indexes o, i are omitted when the kind of an edge is irrelevant. Given an edge $\langle a, a_1, \cdots, a_k\rangle$, k is its *arity*, a is the subject node, and a_i are the object nodes. Let *nodes*$[\cdot]$ be the function taking a multiset of edges and returning the set of nodes therein.

Graphically, we draw output and input edges as follows:

output edge $\langle a, a_1, \cdots, a_k\rangle_o$ input edge $\langle a, a_1, \cdots, a_k\rangle_i$

We keep implicit the name of the nodes in the drawings: names are only used to identify or separate nodes. 0-arity edges are drawn as:

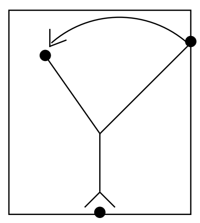
output and input edges with 0 arity

We also introduce boxes:

Definition 10 *A* box B *is a pair* $\langle \mathsf{G}, S\rangle$ *where* G *is a* graph *(a finite multiset of edges) and* $S \subseteq nodes[\mathsf{G}]$*.* S *is called the* internal nodes *of* B *and* $nodes[\mathsf{G}] \setminus S$ *the* principal nodes *of* B*.*

Boxes are ranged over by $B, B', \ldots$, and M ranges over finite multisets of boxes. We extend $nodes[\cdot]$ to boxes by stating that $nodes[\langle \mathsf{G}, S\rangle] \stackrel{def}{=} nodes[\mathsf{G}]$, and define $principals[\langle \mathsf{G}, S\rangle] = nodes[\langle \mathsf{G}, S\rangle] \setminus S$. For multisets we extend these functions pointwise.

Boxes are drawn as rectangles where principal nodes are on the perimeter and internal nodes and all edges are inside. The intuition is that everything inside the box, i.e., all edges and internal nodes, can be replicated. Principal nodes cannot be replicated and can be thought of as the interface between the box and the rest of the diagram. As an example, we illustrate a box with two edges and three nodes. Two nodes are principal, one x-labelled and one unlabelled; the third node is internal.



Definition 11 *A* solo diagram*, in brief SD, is a triple* $(\mathsf{G}, \mathsf{M}, \ell)$ *where* G *is a finite multiset of edges,* M *is a finite multiset of boxes, and* ℓ *is a partial injective function from* $nodes[\mathsf{G}] \cup principals[\mathsf{M}]$ *to* $\mathcal{N}$*, such that internal nodes of every box in* M *do not occur outside the box.*

The labelling ℓ is partially injective to enforce different nodes having different labels. Nodes in $dom(\ell)$ are *labelled nodes*, corresponding to free names; the others represent bound names. In the figures, labels will be explicitly written beside the node they mark. The condition in the definition of solo diagram says that internal nodes are not visible outside the box, i.e., they cannot occur in edges elsewhere. For instance, $(\{\langle a, a''\rangle_i\}, \{\langle \{\langle a, a'\rangle_i\}, \{a\}\rangle\}, \emptyset)$ is not an SD because the internal node a occurs outside the box. Note that internal nodes must also be unlabelled since the domain of ℓ only contains the principal nodes. Note also that an SD cannot contain isolated nodes.

In Definition 11, G represents the solos of an agent that are not replicated, while each box in M corresponds to one term under a replication operator.

The edges within the box are the solos under that operator, the internal nodes of the box are the Scope operators under it. Principal nodes may be labelled or unlabelled; unlabelled principal nodes are under a Scope that does not sit under a replication. Both G and M are multisets rather than sets, in order to model agents such as $(x \mid x)$ or $!x \mid !x$. In the following, $\uplus$ denotes multiset union.

The SDs come with four rewriting rule, whose formal definition is the following, where we use σ as a substitution on nodes and $\mathsf{G}\sigma$ (and $\mathsf{M}\sigma$, respectively) to mean the graph obtained by replacing a by $\sigma(a)$ in all edges in G (and M, respectively).

Definition 12 *The reduction relation $\longrightarrow$ of SDs consists of the following schema, where G_1 and G_2 are arbitrary graphs, M' an arbitrary finite multiset of boxes. Let $\alpha = \langle a, a'_1, \cdots, a'_k \rangle_o$, $\beta = \langle a, a_1, \cdots, a_k \rangle_i$ or vice versa with reversed i/o polarity.*

1. *edge-edge reduction:* $(\mathsf{G} \uplus \{\alpha, \beta\}, \mathsf{M}, \ell) \longrightarrow (\mathsf{G}\sigma, \mathsf{M}\sigma, \ell')$.

2. *edge-box reduction:*
   Let $\mathsf{G} = \{\alpha\} \uplus \mathsf{G}_1,$
   $\mathsf{M} = \langle \{\beta\} \uplus \mathsf{G}_2, S \rangle \uplus \mathsf{M}',$
   Then $(\mathsf{G}\,,\, \mathsf{M}\,,\, \ell) \longrightarrow ((\mathsf{G}_1 \uplus \mathsf{G}_2\rho)\sigma\,,\, \mathsf{M}\sigma\,,\, \ell')$
   where ρ is a renaming of nodes in S into fresh nodes.

3. *box-box reduction:*
   Let $B_1 = \langle \{\alpha\} \uplus \mathsf{G}_1, S_1 \rangle,$ $B_2 = \langle \{\beta\} \uplus \mathsf{G}_2, S_2 \rangle,$ *and* $\mathsf{M} = \{B_1, B_2\} \uplus \mathsf{M}'$
   Then $(\mathsf{G}\,,\, \mathsf{M}\,,\, \ell) \longrightarrow ((\mathsf{G} \uplus \mathsf{G}_1\rho \uplus \mathsf{G}_2\rho)\sigma\,,\, \mathsf{M}\sigma\,,\, \ell')$
   where ρ is a renaming from $S_1 \cup S_2$ to fresh nodes.

4. *internal box reduction:*
   Let $\mathsf{M} = \langle \{\alpha, \beta\} \uplus \mathsf{G}_1, S \rangle \uplus \mathsf{M}'$. *Then* $(\mathsf{G}\,,\, \mathsf{M}\,,\, \ell) \longrightarrow ((\mathsf{G} \uplus \mathsf{G}_1\rho)\sigma\,,\, \mathsf{M}\sigma\,,\, \ell')$
   where ρ is a renaming from S into fresh nodes.

where, for every reduction $(\mathsf{G}, \mathsf{M}, \ell) \longrightarrow (\mathsf{G}'\sigma, \mathsf{M}'\sigma, \ell')$, $ran(\sigma) \cap dom(\sigma) = \emptyset$, $dom(\sigma) \cap dom(\ell) = \emptyset$ and σ agrees with $\{a'_i = a_i : 1 \leq i \leq k\}$. Moreover, ℓ' restricts ℓ to $nodes[\mathsf{G}'\sigma] \cup nodes[\mathsf{M}'\sigma]$.

As in the solos calculus, the edge-edge reduction may fuse either two unlabelled nodes or one labelled node and an unlabelled one. This follows from the constraint that $dom(\sigma) \cap dom(\ell) = \emptyset$: a labelled node cannot be substituted. The edge-edge reduction is illustrated in Figure 1 for a simple agent.

Figures 6, 7 and 8 describe the reductions involving boxes. As said above we regard boxes as copying-machines for edges and internal nodes: when an interaction involves an edge of the box, all the contents of the box are instantiated at once. In particular, internal nodes are duplicated and renamed into fresh new nodes. Principal nodes, whether labelled or not, are not duplicated and will remain shared with the rest of the diagram. In Figure 6, we describe a box-instantiation due to the interaction on the principal node x. The overall effect is the appearance of the edge $\langle y, z\rangle_o$ on the right hand side, together with the box on the left hand side. Figure 7 shows a box-instantiation due to an interaction between edges in different boxes. This rule produces a copy of the contents of the two boxes, where the two interacting edges have been consumed. The rewriting in Figure 8 describes the box-instantiation due to a reduction internal to the box. The effect of this reduction is the fusion of two nodes, which turn out to be the arguments of an input on the principal node x. As a consequence, the instance of the box contents consists of the edge $\langle x, n, n\rangle_i$ only, where $n \in \mathcal{U}$ is fresh and unlabelled.

We conclude with few remarks about the SD reductions:

1. The definition of SD implicitly carries out a garbage collection of disconnected nodes.
2. The labelling function of an SD always decreases during the computation. In other words, no free name is ever created by internal reductions.
3. Very small parts of the graph are involved in the reduction. Apart from the removal of the interacting edges α and β, the residuals of the involved boxes are copied after renaming internal nodes, and in the next subsection we will see how this copying can be minimized. This lends to a simple and slender implementation of the graphs.

3.1 A Local Implementation of Boxes

Boxes are difficult to implement because they require code-duplication that may involve arbitrarily many edges. This imposes a synchronization among edges and hinders a completely local view of the computation. It is therefore interesting that we can restrict such code-duplications to a small constant number of edges. In this respect, we are mainly inspired by local implementations of linear logic boxes [GAL92] and Parrow's trios [Par00].

In our solos calculus, boxes may be decomposed to boxes of three edges, without affecting the expressiveness:

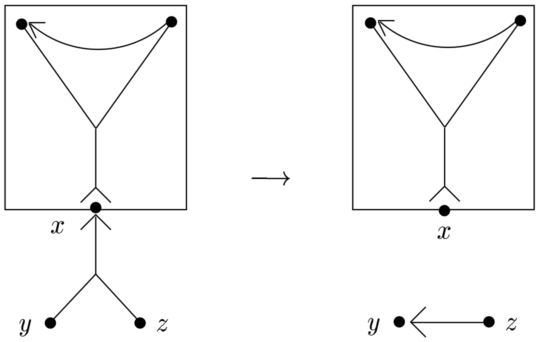


Figure 6: The edge-box reduction $\overline{x}\,yz \mid !\,(uv)(x\,uv \mid \overline{u}\,v) \longrightarrow \overline{y}\,z \mid !\,(uv)(x\,uv \mid \overline{u}\,v)$

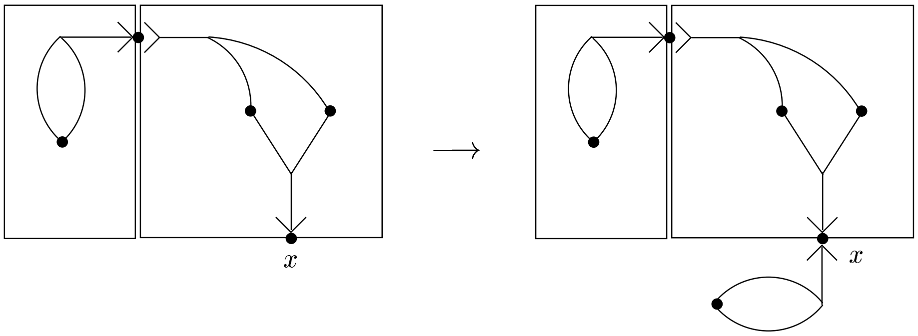


Figure 7: The box-box reduction $(z)(!\,(u)\overline{z}\,uu \mid !\,(uv)(z\,uv \mid x\,uv)) \longrightarrow (z)(!\,(u)\overline{z}\,uu \mid !\,(uv)(z\,uv \mid x\,uv)) \mid (u)\overline{x}\,uu$

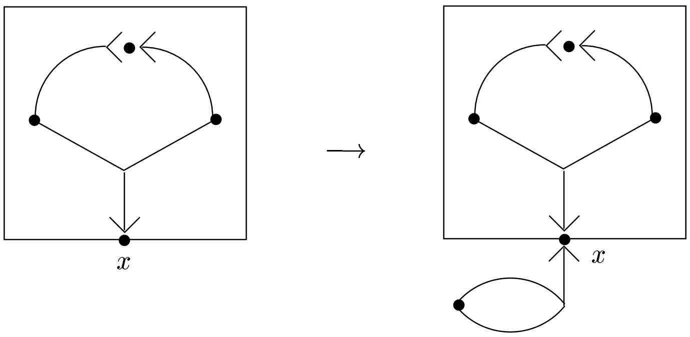


Figure 8: The internal box reduction $!\,(uvw)(x\,uv \mid \overline{w}\,v \mid w\,u) \longrightarrow (v')x\,v'v' \mid !\,(uvw)(x\,uv \mid \overline{w}\,v \mid w\,u)$

Theorem 13 (Decomposition)

$$! \, (\widetilde{u})(\prod_{i=1}^{n} \alpha_i) \quad \approx \quad (z_i{}^{1 \le i \le n}) \prod_{i=1}^{n} ! \, (\widetilde{u})(z_i \, \widetilde{u} \mid \alpha_i \mid \overline{z_{i+1}} \, \widetilde{u})$$

where α_i are solos, z_i, $1 \le i \le n$ are pairwise distinct fresh names and $z_{n+1} = z_1$.

In Figure 9 we draw a diagram showing the basic units for implementing any SD-box.

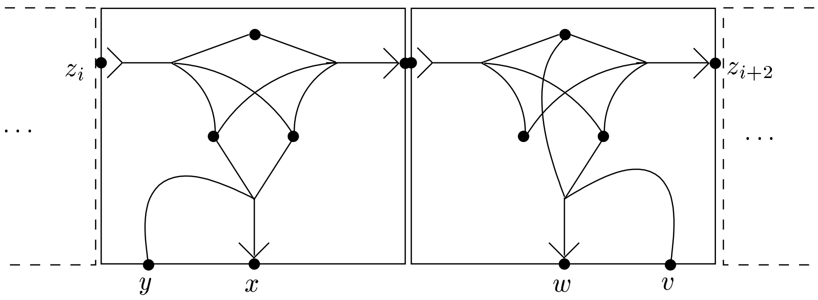


Figure 9: The decomposition of $! \, (u_1 u_2 u_3)(\cdots \mid \overline{x} \, y u_1 u_2 \mid \overline{w} \, u_3 u_2 v \mid \, \cdots)$.

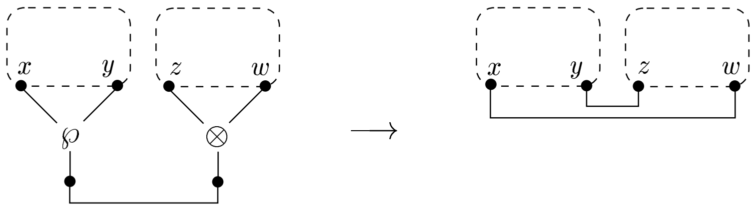
The computational cost of the restriction in Theorem 13 is to perform a number of fusions which is linear with respect to the size of the original box. These fusions are used to unify the choice of bound names $\widetilde{u}$.

We remark that if we systematically apply Theorem 13 to all boxes containing at least 2 edges, then the last reduction rule in Definition 12 (internal box reduction) becomes redundant.

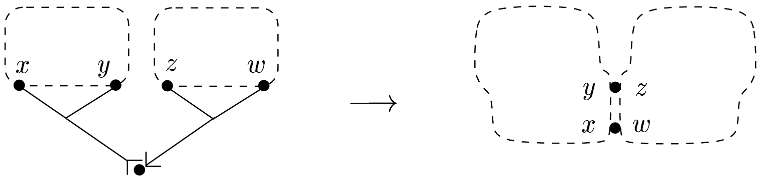
3.2 The Connection with Proof Nets

In this section we briefly discuss the connection between our SDs and linear logic proof nets [Gir87] (this topic will be detailed in the full paper). The reader without some background in linear logic may safely skip this section.

We begin with Multiplicative Proof Nets, which are graphs with four kinds of edges: *axioms*, *cuts*, *par operators*, and *tensor operators*. This subset of Proof Nets may be put in correspondence with SDs without boxes: par and tensor operators correspond to input and output edges, respectively (or, equivalently, the other way around), axiom and cut edges are interpreted by fusing the nodes they connect. This interpretation lifts to the dynamics: cut-elimination is exactly the edge-edge reduction in SDs. Graphically, Proof Net cut-elimination is illustrated by the following reduction



where, in the net on the left hand side, $\wp$ is the par operator, $\otimes$ is the tensor, and the edge in the bottom is the cut operator. We observe that this reduction is mimicked by the SD rewriting



The converse connection is complicated because solo diagrams are more expressive than Proof Nets. Nevertheless, it is possible to show that well-sorted SDs (see [LV01]) with nodes occurring at most two times in the graph correspond to a superset of Proof Nets (the *proof structures*—a collection of nets that may possibly be unsound [Gir87]).

When boxes come into the picture the above relationship is more tenuous. The reason is that boxes are used differently. In Proof Nets, boxes may be safely replaced by their content, thus disallowing the copying capability. This operation is reasonable in a functional language like Proof Nets, where resources, when plugged in some context agent, cannot be used by other agents. In a concurrent scenario, where systems are open, resources may be used by several agents who are not aware of each other. In such contexts an agent cannot erase resources just because it does not need them anymore. Resources may be garbage collected, provided they can never be accessed again, a property which usually follows by some equivalence (such as barbed congruence).

Another difference is that, in Proof Nets, a reduction inside a box does not create copies of its contents, but instead modifies the original box. This is plausible in Proof Nets, which are deterministic, and such simplifications may shorten computations. In contrast these simplifications are conceptually wrong in SDs because of the nondeterminism. Reductions inside a box can also be seen as infinitely many computation steps (affecting all future copies of the box contents), which from an operational viewpoint is unagreeable.

Notwithstanding these differences it may still be possible to relate the two systems. There are suitable evaluation strategies of proof nets that may be implemented in SDs, e.g. according to the techniques illustrated by Milner in [Mil92].

4 Agents and Solo Diagrams

This section details the formal correspondence between the calculus of solos and solo diagrams. We use the following notions and auxiliary functions:

- *graph-isomorphism*: $(\mathsf{G}, \mathsf{M}, \ell)$ and $(\mathsf{G}', \mathsf{M}', \ell')$ are *isomorphic* if there is a bijection f from $nodes[\mathsf{G}] \cup nodes[\mathsf{M}]$ to $nodes[\mathsf{G}'] \cup nodes[\mathsf{M}']$ such that

  1. $\langle a, a_1, \cdots, a_k \rangle \in \mathsf{G}$ if and only if $\langle f(a), f(a_1), \cdots, f(a_k) \rangle \in \mathsf{G}'$;
  2. for every $B \in \mathsf{M}$, the restriction of f to B is a bijection into nodes of $B' \in \mathsf{M}'$ which preserves principal nodes;
  3. $\ell(a) = \ell'(f(a))$.

  In the following we shall never be interested in distinguishing isomorphic SDs.

- *graph-composition*: Let $(\mathsf{G}, \mathsf{M}, \ell)$ and $(\mathsf{G}', \mathsf{M}', \ell')$ be two SDs such that $\ell(a) = \ell'(a')$ if and only if $a = a'$. Then $(\mathsf{G}, \mathsf{M}, \ell) \mid (\mathsf{G}', \mathsf{M}', \ell') \stackrel{def}{=} (\mathsf{G} \uplus \mathsf{G}', \mathsf{M} \uplus \mathsf{M}', \ell \cup \ell')$.

- *graph-scope*: $(x)(\mathsf{G}, \mathsf{M}, \ell) \stackrel{def}{=} (\mathsf{G}, \mathsf{M}, \ell')$, where ℓ' is the function:
  $\ell'(a) = \begin{cases} \ell(a) & \text{if } \ell(a) \neq x \\ \text{undefined} & \text{if } \ell(a) = x \end{cases}$

- $\mathsf{box}[(\mathsf{G}\,,\, \emptyset\,,\, \ell)] = (\emptyset\,,\, \langle \mathsf{G}, S \rangle\,,\, \ell)$, where $S = nodes[\mathsf{G}] \setminus dom(\ell)$.

We remark that graph-composition is a partial function. In order to compose two arbitrary SDs, we consider diagrams isomorphic to them such that nodes are equal if and only if they have the same label ("$\ell(a) = \ell'(a')$ if and only if $a = a'$"). Scope is simply removing the node labelled x from the domain of the labelling function.

We can now define the function $graph[\cdot]$ from agents to SDs such that each solo corresponds to one edge.

Definition 14 *Let $graph[\cdot]$ be the function from agents to SDs, defined inductively as follows:*

$$\begin{array}{rcll}
graph[\mathbf{0}] & = & (\emptyset, \emptyset, \emptyset) & \\
graph[x_0\, x_1 \cdots x_k] & = & (\{\langle a_0, a_1, \cdots, a_k \rangle_i\}, \emptyset, [a_i \mapsto x_i]^{i \le k}) & (x_i = x_j \textit{ iff } a_i = a_j) \\
graph[\overline{x_0}\, x_1 \cdots x_k] & = & (\{\langle a_0, a_1, \cdots, a_k \rangle_o\}, \emptyset, [a_i \mapsto x_i]^{i \le k}) & (x_i = x_j \textit{ iff } a_i = a_j) \\
graph[Q \mid R] & = & graph[Q] \mid graph[R] & \\
graph[(x)Q] & = & (x)graph[Q] & \\
graph[!\, Q] & = & \mathsf{box}[graph[Q]] &
\end{array}$$

Conversely, the function $term[\cdot]$ takes a graph and returns the corresponding canonical agent.

Definition 15 *Let $term[\cdot]$ be the function from SDs to agents defined as follows.*

$$
\begin{array}{rcl}
term[(\mathsf{G},\mathsf{M},\ell)] & = & (\widetilde{z})\Big(\prod_{e\in\mathsf{G}} term[e,\rho] \mid \prod_{B\in\mathsf{M}} term[B,\rho]\Big) \\
term[\langle\mathsf{G},S\rangle,\rho] & = & !\,\Big(\rho(S)\Big)\Big(\prod_{e\in\mathsf{G}} term[e,\rho]\Big) \\
term[\langle a,a_1,\cdots,a_k\rangle_i,\rho] & = & (a\,a_1\cdots a_k)\rho \\
term[\langle a,a_1,\cdots,a_k\rangle_o,\rho] & = & (\overline{a}\,a_1\cdots a_k)\rho
\end{array}
$$

where ρ is a bijection from $nodes[\mathsf{G}] \cup nodes[\mathsf{M}]$ into names that extends ℓ, and $\widetilde{z} = (ran(\rho) \setminus ran(\ell)) \setminus \bigcup_{\langle\mathsf{G},S\rangle\in\mathsf{M}} \rho(S)$.

It is easy to check that $graph[\cdot]$ and $term[\cdot]$ are well-defined functions. Moreover, they are in a precise sense the inverses of each other.

Proposition 16 *1. $P \equiv_r term[graph[P]]$.*

2. $(\mathsf{G},\mathsf{M},\ell)$ and $graph[term[(\mathsf{G},\mathsf{M},\ell)]]$ are isomorphic.

The graphical representation of agents identifies structurally congruent terms. This is proved with suitable isomorphisms for every structural law. The converse also holds: terms mapped onto isomorphic graphs are structurally congruent.

Proposition 17 *$P \equiv_r Q$ iff $graph[P]$ is isomorphic to $graph[Q]$.*

The correspondence between agents and SDs also lifts to the dynamics: two agents reduce provided the corresponding SDs reduce, and *vice versa*.

Theorem 18 *If $P \longrightarrow_r Q$ then $graph[P] \longrightarrow graph[Q]$; if $(\mathsf{G},\mathsf{M},\ell) \longrightarrow (\mathsf{G}',\mathsf{M}',\ell')$ then $term[(\mathsf{G},\mathsf{M},\ell)] \longrightarrow_r term[(\mathsf{G}',\mathsf{M}',\ell')]$.*

PROOF: By induction on the depth of the proof of $P \longrightarrow_r Q$ and by properties of the mappings $term[\cdot]$ and $graph[\cdot]$.

$$(\overline{u}\,\widetilde{x} \mid u\,\widetilde{y} \mid P) \xrightarrow{\{\widetilde{x}=\widetilde{y}\}} P \qquad \frac{P \xrightarrow{\varphi} P'}{P \mid Q \xrightarrow{\varphi} P' \mid Q}$$

$$\frac{P \xrightarrow{\varphi} P',\ x \notin \mathrm{n}(\varphi)}{(x)P \xrightarrow{\varphi} (x)P'} \qquad \frac{P \xrightarrow{\varphi} P',\ z\,\varphi\,x,\ z \neq x}{(x)P \xrightarrow{\varphi\backslash x} P'\{z/x\}} \qquad \frac{P \equiv Q \quad Q \xrightarrow{\varphi} Q' \quad Q' \equiv P'}{P \xrightarrow{\varphi} P'}$$

Side condition in the first rule: $|\widetilde{x}| = |\widetilde{y}|$.

Figure 10: Fusion rules for the calculus of solos.

5 Multi-Labelled Solo Diagrams

The reduction relation of SDs so far corresponds exactly to the reduction relation of the calculus of solos. In turn, this latter reduction corresponds to the internal, unobservable transition of the fusion calculus, which in the labelled transition semantics is the $\xrightarrow{\mathbf{1}}$ relation, where $\mathbf{1}$ is the identity fusion [PV98].

It may be interesting to extend the reduction relation to include the general fusion actions of the labelled transition semantics, noted $\xrightarrow{\varphi}$, where φ is an equivalence relation on names such as $\{\widetilde{x} = \widetilde{y}\}$. Ignoring replication, the rules for fusion actions in the calculus of solos are defined in Figure 10, which relaxes the side conditions of the previous rules (Figure 4). We define $\varphi\backslash z$ to mean $\varphi \cap (\mathcal{N} - \{z\})^2 \cup \{(z,z)\}$, i.e., the equivalence relation φ with all references to z removed (except for the identity). For example, $\{x = z, z = y\}\backslash z = \{x = y\}$, and $\{x = y\}\backslash y = \mathbf{1}$. We write $n(\varphi)$ for the names in non-singular equivalence classes of φ.

The corresponding SDs are defined by changing the labelling function to be a total function $\mathcal{U} \to \mathcal{P}_f(\mathcal{N})$, where $\mathcal{P}_f(\mathcal{N})$ is the set of finite subsets of $\mathcal{N}$. Where ℓ previously was undefined, it now returns $\emptyset$; where it returned a single name, it now returns a singleton.

Definition 19 *A (basic)* Multi-labelled solo diagram*, in brief MSD, is a pair* (G, ℓ)*, where* G *is a finite multiset of edges, and* ℓ*, the* labelling function*, is a function from* $\mathrm{nodes}[\mathsf{G}]$ *to* $\mathcal{P}_f(\mathcal{N})$ *such that, for every* $a, a' \in \mathrm{dom}(\ell)$*,* $a \neq a'$ *implies* $\ell(a) \cap \ell(a') = \emptyset$*.*

In MSDs, the labelling function has the constraint that sets in the range have empty intersection. Intuitively, these sets represent names that have been fused, and the condition guarantees that the same name cannot occur in different sets. This is a generalization of Definition 11 where the same name cannot mark two different nodes. It may thus be easier to think of a mapping from names to nodes: for each name the node (if any) is uniquely defined. This is true in SDs as well as MSDs; the extension to MSD means this mapping is not necessarily injective since different names can label the same node.

Definition 20 *The reduction relation $\longrightarrow$ of MSDs is defined by the following rule schema:*

- *edge-edge reduction:* $(\mathsf{G} \uplus \{\langle a, a_1, \cdots, a_k\rangle_o, \langle a, a'_1, \cdots, a'_k\rangle_i\}, \ell) \longrightarrow (\mathsf{G}\sigma, \ell')$ *where σ agrees with $\{a_i = a'_i : 1 \leq i \leq k\}$, $ran(\sigma) \cap dom(\sigma) = \emptyset$ and $\forall a : \ell'(a) = \bigcup_{\sigma(a')=a} \ell(a')$*

As an example of MSD reduction, in Figure 11 we illustrate the dynamics of an agent that is irreducible with the semantics of section 3.

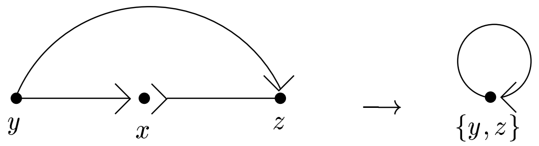


Figure 11: The transition $(\overline{x}\, y \mid x\, z \mid \overline{z}\, y) \xrightarrow{\{y=z\}} \overline{z}\, y$.

As the reader may suspect, MSDs are strongly related to agents. Given an equivalence relation on names (representing the fusions that have occurred so far), we can define a mapping from agents to MSDs, and given the equivalence relation induced by ℓ, we can define a mapping from MSDs to agents. This relationship lifts to the dynamics: every reduction in MSD corresponds to a fusion action in the calculus of solos with the semantics of Figure 10, and *vice versa*. Finally, basic MSDs can be extended with boxes, in the style of section 3.

6 Related Work

The direct ancestor of our solo diagrams are Parrow's Interaction Diagrams [Par95], which were presented as a graphical form of the π-calculus. Interaction Diagrams are more complex, due to the asymmetries of π-calculus. Three types of nodes are used, corresponding to parameters, input bound names, and other names, respectively. To encode input prefixing, Parrow uses a method similar to that of [LV99], increasing the polyadicity to express causal dependency. Parrow's replication box is considered as a regeneration mechanism that can generate an unbounded number of copies of itself. In this respect, it is a straight implementation of the law $!\,P \equiv P \mid !\,P$.

π-nets [Mil94] were introduced by Milner as a graphical form of the action calculus PIC, where the π-calculus can be encoded. π-nets are also very similar to our solo diagrams, except for the treatment of boxes/replication, where an additional *link* arrow is introduced, and for prefixing, which uses a box inhibiting reductions inside the box until its box arc (representing the guard) has been consumed.

In retrospect it is interesting to see that the purely graphical parts of both Interaction Diagrams and π-nets have all the power of our solo diagrams, but Parrow and Milner, respectively, constrain how they may be constructed and thereby avoid constructing the solo diagrams and possibly the calculus of solos.

In [Fu98], Fu investigated the communication mechanism of mobile calculi in terms of cut-elimination in multiplicative proof nets, and inspired by Interaction Diagrams introduced *reaction graphs* to this end. (This investigation led to the χ-calculus [Fu97].) Reaction graphs have some elements in common with solo diagrams, e.g. symmetry of input and output edges. Only a limited form of guarded replication box is handled: no labelled nodes are allowed inside, neither reaction between boxes, nor nested boxes are treated. Fu only uses monadic edges, and thus cannot encode prefixing without these guarded boxes.

Bellin and Scott also give a detailed interpretation of proofs in terms of π-calculus processes [BS94]. Overall, their interpretations use prefixes. As a consequence, reductions that do not occur at the bottom of the proofs are not mirrored.

Yoshida in [Yos95] discusses the relationship between processes and a general framework for proof nets, the so-called Interaction Nets [Laf90]. The analogy mostly concerns the modes of connecting edges in the two systems (by using polarities and principal/auxiliary ports) and of defining interactions (as connections between principal ports). A behavioural analysis is missing.

References

[Bou92] G. Boudol. Asynchrony and the π-calculus (note). Rapport de Recherche 1702, INRIA Sophia-Antipolis, May 1992.

[BS94] G. Bellin and P. Scott. On the π-calculus and linear logic. *Theoretical Computer Science*, 135:11–65, 1994.

[FG96] C. Fournet and G. Gonthier. The reflexive chemical abstract machine and the join-calculus. In *Proc. of POPL '96*, pages 372–385. ACM, Jan. 1996.

[Fu97] Y. Fu. A proof-theoretical approach to communication. In P. Degano, R. Gorrieri and A. Marchetti-Spaccamela, eds, *Proc. of ICALP '97*, volume 1256 of *LNCS*, pages 325–335. Springer, 1997.

[Fu98] Y. Fu. Reaction graph. *Journal of Computer Science and Technology, Science Press, China*, 13(6):510–530, 1998.

[GAL92] G. Gonthier, M. Abadi and J.-J. Lévy. The geometry of optimal lambda reduction. In *Proc. of POPL '92*, pages 15–26. ACM Press, 1992.

[Gir87] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50, 1987.

[HT91] K. Honda and M. Tokoro. An object calculus for asynchronous communication. In P. America, ed, *Proc. of ECOOP '91*, volume 512 of *LNCS*, pages 133–147. Springer, July 1991.

[Laf90] Y. Lafont. Interaction nets. In *Proc. of POPL '90*, pages 95–108. ACM Press, 1990.

[LV99] C. Laneve and B. Victor. Solos in concert. In J. Wiederman, P. van Emde Boas and M. Nielsen, eds, *Proc. of ICALP '99*, volume 1644 of *LNCS*, pages 513–523. Springer, July 1999.

[LV01] C. Laneve and B. Victor. Solos in concert. Full version of [LV99], submitted for journal publication, February 2001.

[Mil92] R. Milner. Functions as processes. *Journal of Mathematical Structures in Computer Science*, 2(2):119–141, 1992.

[Mil93] R. Milner. The polyadic π-calculus: A tutorial. In F. L. Bauer, W. Brauer and H. Schwichtenberg, eds, *Logic and Algebra of Specification*, volume 94 of *Series F*. NATO ASI, Springer, 1993.

[Mil94] R. Milner. Pi-nets: A graphical form of π-calculus. In D. Sannella, ed, *Proc. of ESOP '94*, volume 788 of *LNCS*, pages 26–42. Springer, 1994.

[MPW92] R. Milner, J. Parrow and D. Walker. A calculus of mobile processes, part I/II. *Information and Computation*, 100:1–77, Sept. 1992.

[MS92] R. Milner and D. Sangiorgi. Barbed bisimulation. In W. Kuich, ed, *Proc. of ICALP '92*, volume 623 of *LNCS*, pages 685–695. Springer, 1992.

[Par95] J. Parrow. Interaction diagrams. *Nordic Journal of Computing*, 2:407–443, 1995.

[Par00] J. Parrow. Trios in concert. In G. Plotkin, C. Stirling and M. Tofte, eds, *Proof, Language and Interaction: Essays in Honour of Robin Milner*, Foundations of Computing. MIT Press, May 2000.

[PT00] B. C. Pierce and D. N. Turner. Pict: A programming language based on the pi-calculus. In G. Plotkin, C. Stirling and M. Tofte, eds, *Proof, Language and Interaction: Essays in Honour of Robin Milner*, Foundations of Computing. MIT Press, May 2000.

[PV98] J. Parrow and B. Victor. The fusion calculus: Expressiveness and symmetry in mobile processes. In *Proc. of LICS '98*, pages 176–185. IEEE, Computer Society Press, July 1998.

[San98] D. Sangiorgi. On the bisimulation proof method. *Mathematical Structures in Computer Science*, 8(5):447–479, 1998.

[Yos95] N. Yoshida. Graph notation for concurrent combinators. In T. Ito and A. Yonezawa, eds, *Proc. of TPPP '94*, volume 907 of *LNCS*, pages 393–412. Springer, 1995.