Solo Diagrams

Cosimo Laneve[1], Joachim Parrow[2], and Björn Victor[3]

[1] Dept. of Computer Science, University of Bologna, Italy. `laneve@CS.UniBO.IT`
[2] Royal Institute of Technology, Kista, Sweden. `joachim@it.kth.se`
[3] Dept. of Computer Systems, Uppsala University, Sweden. `victor@DoCS.UU.SE`

Abstract. The calculus of solos is the prefix-less subcalculus of the fusion calculus: it contains only atomic communication actions and the operators for parallel and scoping. It is expressive enough to admit an encoding of the whole fusion calculus. We here present a graphic representation of agents in the solo calculus, adapting ideas from interaction diagrams and pi-nets. Our so-called solo diagrams are less complicated than these and clarify the important aspects, for example structural congruence in the calculus is precisely isomorphism in the diagrams. Perhaps most importantly the diagrams contribute to an implementation, replacing the usual unfolding with a few tractable rules for reducing replication boxes. Finally we demonstrate the strong similarity with proof nets in linear logic; it turns out that par and tensor edges have same behaviour as inputs and outputs.

1 Introduction

When the π-calculus was introduced almost fifteen years ago one of the main motivations was to find fundamental mechanisms of computation with high expressive power. These mechanisms include transfer of an atomic name between parallel processes and generation of new names at need. The π-calculus embodies them in a small set of operators to build *agents*, the computationally active entities. Among the operators are input prefix $u\,(x)\,.\,P$ (receive an input for x along u and continue as P), output prefix $\overline{u}\,v\,.\,P$ (ditto for output), parallel composition, and restriction of channel scope. When an input $u\,(x)\,.\,P$ and output $\overline{u}\,v\,.\,Q$ are combined in parallel the system reduces, such that the output object v substitutes x in the receiver P.

The π-calculus 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 the output prefix $\overline{u}\,v\,.\,P$ is replaced by the output particle $\overline{u}\,v$, in other words an output cannot enable further actions. In the fusion calculus 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, their 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 of names. The solo calculus additionally includes only binary parallel composition $P \mid Q$ and scoping $(x)P$, giving a very lean formalism. We refer the reader to [9] for further explanation of the expressive power of solos.

The main contribution in this paper is the presentation and use of a graphical formalism for solos. The underlying idea is quite simple. In order to draw an agent pick one node for each name and label the node with the name. Outputs and inputs become two types of multiedges, the solo $u\,\widetilde{x}$ an input edge from the nodes labelled $\widetilde{x}$ to the node labelled u, and conversely for output. Parallel composition is just graph union and scope restriction entails erasing the label of nodes. Reductions between inputs and outputs means that the corresponding nodes are fused, i.e., the nodes are merged, preserving any additional edges connecting them. As an example, Figure 1 shows a simple agent and reduction. Note that the two kinds of edges, input and output, are distinguished by the shape of the arrow which has either a head or a tail.

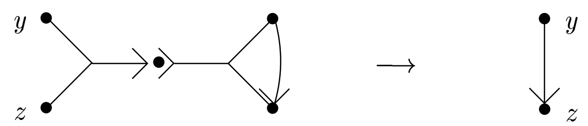


Fig. 1. The solo diagram corresponding to $(xuv)(\overline{x}\,yz \mid x\,vu \mid u\,v) \longrightarrow \overline{z}\,y$

In order to model potentially infinite behaviour some mechanism for recursion or iteration must be added. We adopt the standard replication operator in the π-calculus, written $!\,P$, standing for arbitrarily many copies of P. In the diagrams replications are drawn as boxes surrounding the replicated graph. The standard semantics is to use unfolding: $!\,P$ can unfold to $P \mid\, !\,P$. This definition is of debatable value for an implementation since it does not prescribe exactly when unfoldings should happen. In the diagrams we therefore give the reductions through a small set of rules that do not explicitly mention unfolding. An example is shown in Figure 2. As can be seen in this example the box remains

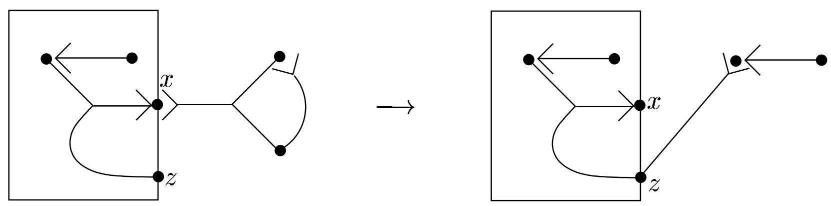


Fig. 2. The solo diagram corresponding to $(uv)(!(yw)(\overline{x}\,yz \mid \overline{y}\,w) \mid x\,uv \mid u\,v) \longrightarrow (uw')(!(yw)(\overline{x}\,yz \mid \overline{y}\,w) \mid \overline{u}\,w' \mid u\,z)$

after the reduction, which generates a copy of the contents.

Within this framework we derive several results. The usual structural congruence in the calculus is shown to coincide with diagram isomorphism, and we get a simple kind of normal form for agents. We prove that our semantics yields the same reductions as the standard unfolding approach. A flattening law can be used to remove replications within replications, in other words, we need not deal with boxes within boxes. A decomposition law can be used to limit the size of boxes to at most three solos. An interesting correspondence is established with the proof nets of linear logic: if inputs and outputs are replaced with par and tensor edges, the resulting net will have the same reduction behaviour, at least for an interesting subset of the diagrams. Finally we consider a generalisation to graphs with multiply labelled nodes; in some sense this gives a better correspondence to the standard semantics of the fusion calculus.

In the following we begin with the basic calculus and diagrams, i.e., without replication and boxes. These features are introduced and studied in section 6. The paper concludes with a discussion of related work.

2 The Calculus of Solos

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

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

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 $\mathit{fn}(P)$, are the names in P with a non-bound occurrence.

The (finite, choice-free) fusion calculus [16] consists of the above agents and the prefix operator, namely agents of the form $\alpha\,.\,P$.

Operational semantics. In the style of the Chemical Abstract Machine [2], 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.

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

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

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

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

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

Fig. 3. Reduction rules for the calculus of solos.

The following fact is easily shown.

Proposition 2. *Every agent P is structurally congruent to an agent of the form $(\widetilde{x})Q$, where Q is a parallel compositions of solos and $\widetilde{x} \subseteq fn(Q)$. These forms are called* canonical.

The reduction relation of the calculus of solos is the least relation satisfying the rules in Figure 3. 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 3 bans reductions that fuse two different free names. For instance, the agent $\overline{x}\,y \mid x\,z$ is irreducible. See section 7 and [16] for alternative semantics allowing such reductions.

Equivalence. To define an extensional semantics, we use the standard notion of *barbed bisimulation* developed in [13, 17].

Definition 3. *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 4. *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 $P \stackrel{\cdot}{\approx} Q$, 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 $C[\cdot]$, $C[P] \stackrel{\cdot}{\approx} C[Q]$.*

The calculus of solos, although simple, is expressive enough. The next theorem recalls a result in [9].

Theorem 5. *There exists an encoding* $[\![\cdot]\!]$ *of the full fusion calculus into the calculus of solos such that* $[\![P]\!] \approx [\![Q]\!]$ *implies* $P \approx Q$*, and* $P \dot{\approx} Q$ *implies* $[\![P]\!] \dot{\approx} [\![Q]\!]$.

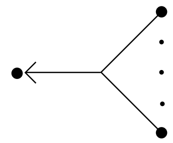
This implies that $C[P] \dot{\approx} C[Q]$ iff $[\![C[P]]\!] \dot{\approx} [\![C[Q]]\!]$ for any fusion calculus context $C[\cdot]$.

3 Basic Solo Diagrams

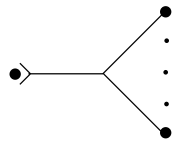
Our graphs will be built out of an infinite set $\mathcal{U}$ of *nodes*, ranged over by $a, b, \ldots$.

Definition 6. *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. 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

Definition 7. *A (Basic)* solo diagram*, in brief SD, is a pair* (G, ℓ)*, where* G*, the* graph*, is a finite multiset of edges, and* ℓ*, the* labelling*, is a partial injective function from the set of nodes occurring in* G*, noted* $nodes[\mathsf{G}]$*, to* $\mathcal{N}$.

Graphs are multisets rather than sets, in order to model agents such as $(x \mid x)$. In the following, $\uplus$ denotes multiset union. Nodes in $dom(\ell)$ are *labelled nodes*, corresponding to free names; the others represent bound names. Two different labelled nodes have different labels, in other words, ℓ is required to be injective. In the figures, labels will be explicitly written beside the node they mark. Note that a SD cannot contain isolated nodes.

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

Definition 8. *The reduction relation* $\longrightarrow$ *of SDs 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 $dom(\sigma) \cap dom(\ell) = \emptyset$;*
  - *ℓ' is the restriction of ℓ to $nodes[\mathsf{G}\sigma]$.*

As for the calculus of solos, 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 definition of SD implicitly carries out a garbage collection of disconnected nodes. The edge-edge reduction is illustrated in Figure 1 for a simple agent.

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*: (G, ℓ) and (G', ℓ') are *isomorphic* if there is a bijection f on nodes such that (1) $\langle a, a_1, \cdots, a_k\rangle \in \mathsf{G}$ if and only if $\langle f(a), f(a_1), \cdots, f(a_k)\rangle \in \mathsf{G}'$; and (2) $\ell(a) = \ell'(f(a))$.
  In the following we shall never be interested in distinguishing isomorphic SDs.

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

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

We remark that graph-composition is a partial function. In order to compose two arbitrary SDs, we consider two isomorphic ones such that the condition "$\ell(a) = \ell'(a')$ if and only if $a = a'$" holds. 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 9. *Let $graph[\cdot]$ be the function from agents to SDs, defined inductively as follows:*

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

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

Definition 10. *Let term*$[\cdot]$ *be the function from SDs to agents defined as follows.*

$$
\begin{aligned}
term[(\mathsf{G},\ell)] &= (\widetilde{z})(\prod_{e\in\mathsf{G}} term[e,\rho]) \\
term[\langle a, a_1, \cdots, a_k\rangle_i, \rho] &= (a\, a_1 \cdots a_k)\rho \\
term[\langle a, a_1, \cdots, a_k\rangle_o, \rho] &= (\overline{a}\, a_1 \cdots a_k)\rho
\end{aligned}
$$

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

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 11. *1.* $P \equiv term[graph[P]]$.
2. (G,ℓ) *and* $graph[term[(\mathsf{G},\ell)]]$ *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 12. $P \equiv Q$ *if and only if* $graph[P]$ *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*.

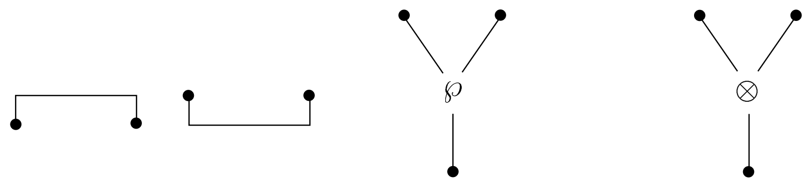
Theorem 13. *If* $P \longrightarrow Q$ *then* $graph[P] \longrightarrow graph[Q]$*; if* $(\mathsf{G},\ell) \longrightarrow (\mathsf{G}',\ell')$ *then* $term[(\mathsf{G},\ell)] \longrightarrow term[(\mathsf{G}',\ell')]$.

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

5 The Connection with Multiplicative Proof Nets

In this section we discuss the relationship between our (basic) SDs and linear logic [5] multiplicative proof nets. This analysis will shed some new light on the contrast between the “functional dynamics” of proof nets and the “concurrent dynamics” of SDs.

In multiplicative proof nets there are four kinds of edges; *axioms*, *cuts*, *par operators* and *tensor operators*, which are drawn as follows:



axiom cut par edge tensor edge

In addition, proof nets are well-sorted: two edges ending into the same node have suitable opposite sorts. Therefore, to establish a direct correspondence with SDs, we should adopt a sorting discipline for the latter. This is not difficult (see e.g. [10]), but does not convey anything interesting for our analysis, so we shall omit the sorts in the present paper.

Before detailing the formal correspondence, let us provide some intuition. Clearly, par and tensor operators correspond to input and output edges, respectively (or, equivalently, the other way around). Axiom and cut edges have no corresponding edges in SDs. We interpret these edges as "node fusions" still to be carried out. Therefore we implement axioms and cuts by fusing the corresponding nodes. (Girard, in [6], also removes cut edges by directly gluing the connected nodes.) For instance, Figure 4 illustrates a proof net and its related SD.

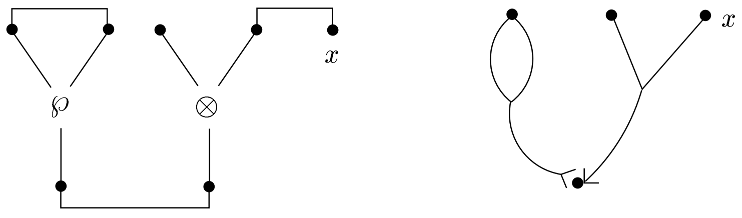
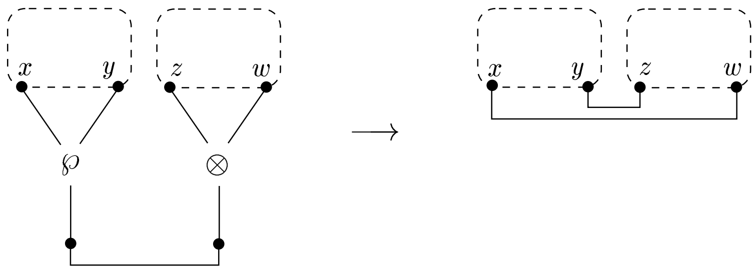
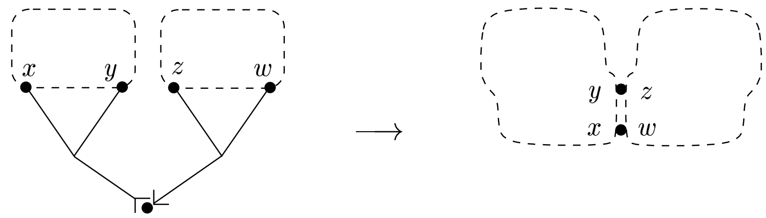


Fig. 4. Proof net style and solo diagram style of drawing.

More interestingly, the connection lifts to the dynamics. Proof nets come with two reductions: (1) a cut elimination between a par edge and a tensor edge and (2) a cut between an axiom and another edge. The interesting reduction is (1), which is illustrated below



We observe that this reduction is mimicked by the SD rewriting



(In the right hand side graph, with an abuse of notation, we have nodes with two labels—see Section 7.)

Formally, the connection is as follows.

Definition 14. *A (multiplicative) proof net is a pair* (N, ℓ) *representing a proof in multiplicative linear logic [5], where* N *is a finite collection of edges* $\langle a, a' \rangle_{axiom}$, $\langle a, a' \rangle_{cut}$, $\langle a, a', a'' \rangle_{\wp}$, *and* $\langle a, a', a'' \rangle_{\otimes}$; ℓ *is a partial injective function from nodes*$[\mathsf{N}]$ *to* $\mathcal{N}$ *such that* $a \in dom(\ell)$ *if and only if* $\langle a', a \rangle_{axiom} \in \mathsf{N}$ *and* a *occurs exactly once in* N.

Definition 14 does not characterize proof nets in an intrinsic way, because a linear logic proof is required. For the purpose of the following discussion, the reader may safely assume that proof nets are labelled graphs. An inductive, intrinsic definition of proof nets may be found in [5].

Proof nets have the following two reductions:

1. *par-tensor cut-elimination:*
   $\Big(\mathsf{N} \cup \{\langle a, a_1, a_2 \rangle_{\otimes}, \langle a', a'_1, a'_2 \rangle_{\wp}, \langle a, a' \rangle_{cut}\},\ \ell\Big) \longrightarrow \Big(\mathsf{N} \cup \{\langle a_1, a'_1 \rangle_{cut}, \langle a_2, a'_2 \rangle_{cut}\},\ \ell\Big)$
2. *axiom cut-elimination:*
   $\Big(\mathsf{N} \cup \{\langle a, a' \rangle_{axiom}, \langle a', a'' \rangle_{cut}\},\ \ell\Big) \longrightarrow \Big(\mathsf{N}\{a/a''\},\ \ell\Big)$
   $\Big(\mathsf{N} \cup \{\langle a'', a' \rangle_{cut}, \langle a', a \rangle_{axiom}\},\ \ell\Big) \longrightarrow \Big(\mathsf{N}\{a/a''\}, \ell\Big)$

Notice that the labelling function ℓ (also called *the interface of the net*) is invariant in proof net reductions.

Let $\mathcal{T}$ be the translation from proof nets to SDs defined as follows:

$\mathcal{T}[(\mathsf{N}, \ell)] = \Big(\{\langle a, a_1, a_2 \rangle_i, \langle a', a'_1, a'_2 \rangle_o : \langle a, a_1, a_2 \rangle_{\wp}, \langle a', a'_1, a'_2 \rangle_{\otimes} \in \mathsf{N}\}\sigma,\ \ell\Big)$

where

1. σ agrees with $\{a = a' : \langle a, a' \rangle_{cut} \in \mathsf{N}$ or $\langle a, a' \rangle_{axiom} \in \mathsf{N}\}$;
2. $ran(\sigma) \subseteq \{a, a' : \langle a, a' \rangle_{cut} \in \mathsf{N}$ or $\langle a, a' \rangle_{axiom} \in \mathsf{N}\}$ and
3. $ran(\sigma) \supseteq dom(\ell) \cap \{a, a' : \langle a, a' \rangle_{cut} \in \mathsf{N}$ or $\langle a, a' \rangle_{axiom} \in \mathsf{N}\}$.

Thus, $\mathcal{T}[(\mathsf{N}, \ell)]$ is obtained by replacing each par edge in N by an input edge and each tensor edge by an output edge and then fusing nodes which are connected by cut and axiom edges. The next theorem brings together the two different operational perspectives discussed at present: cut-eliminations and fusions.

Theorem 15. $(\mathsf{N}, \ell) \longrightarrow (\mathsf{N}', \ell)$, *due to a par-tensor cut, iff* $\mathcal{T}[(\mathsf{N}, \ell)] \longrightarrow \mathcal{T}[(\mathsf{N}', \ell)]$. $(\mathsf{N}, \ell) \longrightarrow (\mathsf{N}', \ell)$, *due to an axiom cut, iff* $\mathcal{T}[(\mathsf{N}, \ell)] = \mathcal{T}[(\mathsf{N}', \ell)]$.

It is important to observe that axiom cuts have no counterpart in SDs because their effects are computed during the translation.

The converse connection is more complicated since solo diagrams are more expressive than proof nets. For instance, the former are nondeterministic, while the latter are deterministic. Furthermore, a fusion during a reduction in an SD may affect several edges, while it affects at most two edges in proof nets. Notably, if we restrict basic SDs such that every node occurs at most two times in the graph, then these differences disappear. It turns out that such restriction allows us to relate SDs (with sorts) with a superset of proof nets (the *proof structures*—a collection of nets that may possibly be unsound [5]).

6 Replication

The agents of Section 2 have finite behaviour. This is a severe restriction of expressiveness which, in mobile calculi, is usually overcome by introducing a *replication operator* $!\,P$ and a structural law $!\,P \equiv P \mid !\,P$ [11]. Although this addition is algebraically elegant, its SD implementation could hardly have the property of Proposition 12. In other words, such an unconstrained implementation of replication would quickly give an "out-of-memory error". In this section we first define the replication operator as above, and then show that we can use a more implementation-friendly definition which has a direct correspondence to solo diagrams extended with boxes.

The agents of the calculus of solos (with replication) are defined by:

$$\begin{array}{rll} P ::= & \cdots & \text{as in Section 2} \\ & !\,Q & \text{(Replication)} \end{array}$$

Structural congruence is defined in the same way as in Section 2. The dynamics is defined by the rules of Figure 3 and the new rule

$$\frac{P \equiv_r Q \quad Q \longrightarrow Q' \quad Q' \equiv_r P'}{P \longrightarrow P'}$$

where $\equiv_r$ is $\equiv$ with the additional rule $!\,P \equiv_r P \mid !\,P$.

The definitions of barbed bisimulation and congruence are unchanged. The following theorem shows that nested replication may be flattened into non-nested replications.

Theorem 16 (Flattening).

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

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

In view of this theorem 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 17. *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, $\widetilde{x}_i \subseteq \mathrm{fn}(Q_i)$ and $\widetilde{x} \subseteq \mathrm{fn}(Q \mid (\prod_{i\in I} !\,(\widetilde{x}_i)Q_i))$.*

The agents of the above form are called canonical.

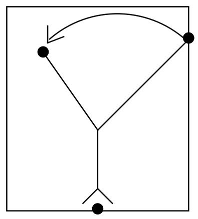
6.1 Solo Diagrams with Boxes

In order to extend SDs with replication we introduce the notion of a box.

Definition 18. *A* box *B is a pair $\langle \mathsf{G}, S\rangle$ where G is a graph and $S \subseteq \mathit{nodes}[\mathsf{G}]$. S is called the* internal nodes *of B and $\mathit{nodes}[\mathsf{G}] \setminus S$ the* principal nodes *of B.*

Boxes are ranged over by $B, B', \ldots$, and M ranges over finite multisets of boxes. We extend $\mathit{nodes}[\cdot]$ to boxes by stating that $\mathit{nodes}[\langle \mathsf{G}, S\rangle] \stackrel{def}{=} \mathit{nodes}[\mathsf{G}]$, and define $\mathit{principals}[\langle \mathsf{G}, S\rangle] = \mathit{nodes}[\langle \mathsf{G}, S\rangle] \setminus S$. 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 a diagram. We illustrate, as an example, a box with two edges and three nodes. Two nodes are principal, a labelled and unlabelled one; the other node is internal.



Definition 19. *A* solo diagram *(with boxes) is a triple $(\mathsf{G}, \mathsf{M}, \ell)$ where G is a graph, M is a finite multiset of boxes, and ℓ is a partial injective function from $\mathit{nodes}[\mathsf{G}] \cup \mathit{principals}[\mathsf{M}]$ to $\mathcal{N}$, such that internal nodes of every box in M do not occur outside the box.*

Here G represent the edges that are not replicated. Intuitively, each box corresponds to one replication operator; the edges within the box are the solos under that operator and the internal nodes 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. With a slight abuse of notation, in the following we shorten solo diagram (with boxes) into SD.

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.

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 is 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. This behaviour is detailed in Definition 20, which is illustrated in Figures 5, 6 and 7 for sample agents.

Definition 20. *The reduction relation $\longrightarrow$ of SDs consists of the edge-edge reduction in Definition 8 (extended to triples $(\mathsf{G},\mathsf{M},\ell)$) and the following, where G_1 and G_2 are arbitrary graphs.*

- *edge-box reduction:*
  Let $\mathsf{G} = \{\langle a, a'_1, \cdots, a'_k\rangle_o\} \uplus \mathsf{G}_1$,
  $B = \langle\{\langle a, a_1, \cdots, a_k\rangle_i\} \uplus \mathsf{G}_2, S\rangle \in \mathsf{M}$,
  and ρ be a renaming of nodes in S into fresh nodes.
  Then $(\mathsf{G}\,,\,\mathsf{M}\,,\,\ell) \longrightarrow ((\mathsf{G}_1 \uplus \mathsf{G}_2\rho)\sigma\,,\,\mathsf{M}\sigma\,,\,\ell')$
  and similarly by reverting input and output edges;
- *box-box reduction:*
  Let $B_1 = \langle\{\langle a, a_1, \cdots, a_k\rangle_i\} \uplus \mathsf{G}_1, S_1\rangle$, $B_2 = \langle\{\langle a, a'_1, \cdots, a'_k\rangle_o\} \uplus \mathsf{G}_2, S_2\rangle$, *where* $\{B_1, B_2\} \subseteq \mathsf{M}$
  and ρ a renaming from $S_1 \cup S_2$ to fresh nodes.
  Then $(\mathsf{G}\,,\,\mathsf{M}\,,\,\ell) \longrightarrow ((\mathsf{G} \uplus \mathsf{G}_1\rho \uplus \mathsf{G}_2\rho)\sigma\,,\,\mathsf{M}\sigma\,,\,\ell')$
- *internal box reduction:*
  Let $B = \langle\{\langle a, a_1, \cdots, a_k\rangle_i, \langle a, a'_1, \cdots, a'_k\rangle_o\} \uplus \mathsf{G}_1, S\rangle \in \mathsf{M}$ *and ρ be a renaming from S into fresh nodes. Then* $(\mathsf{G}\,,\,\mathsf{M}\,,\,\ell) \longrightarrow ((\mathsf{G} \uplus \mathsf{G}_1\rho)\sigma\,,\,\mathsf{M}\sigma\,,\,\ell')$.

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 $nodes[\mathsf{G}'\sigma] \cup nodes[\mathsf{M}'\sigma]$.

The labelling function of an SD always decreases during the computation. In other words, no free name is ever created by internal reductions.

The graph rewriting in Figure 5 describes 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 6 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 7 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.

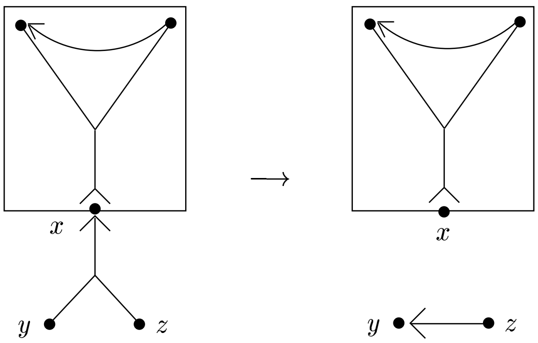


Fig. 5. The edge-box reduction $\overline{x}\, y z \mid !\,(uv)(x\, uv \mid \overline{u}\, v) \longrightarrow \overline{y}\, z \mid !\,(uv)(x\, uv \mid \overline{u}\, v)$

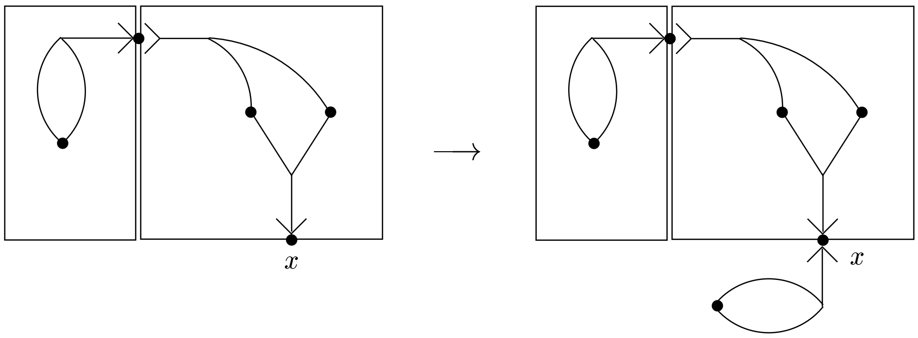


Fig. 6. The box-box reduction $(z)\Big(!\,(u)\overline{z}\, uu \mid !\,(uv)(z\, uv \mid x\, uv)\Big) \longrightarrow (z)\Big(!\,(u)\overline{z}\, uu \mid !\,(uv)(z\, uv \mid x\, uv)\Big) \mid (u)\overline{x}\, uu$

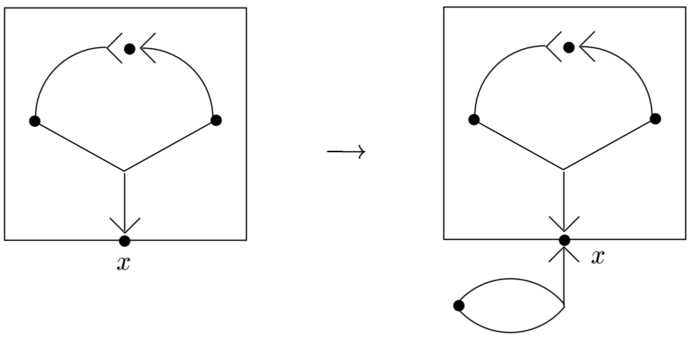


Fig. 7. The internal box reduction $!\,(uvw)(x\, uv \mid \overline{w}\, v \mid w\, u) \longrightarrow (v')x\, v'v' \mid !\,(uvw)(x\, uv \mid \overline{w}\, v \mid w\, u)$

6.2 Replication and Boxes

The correspondence of Section 4 may be easily extended to cope with replication and boxes.

Let $\mathsf{box}[(\mathsf{G}\,,\,\emptyset\,,\,\ell)]$ be the diagram $(\emptyset\,,\,\langle\mathsf{G},S\rangle\,,\,\ell)$, where $S = nodes[\mathsf{G}]\backslash dom(\ell)$, and extend the definitions of *graph-isomorphism*, *graph-composition* and *graph-scope* in the obvious ways, to handle triples $(\mathsf{G},\mathsf{M},\ell)$ instead of pairs (G,ℓ).

Definition 21. *The functions graph*$[\cdot]$ *and term*$[\cdot]$ *of Definitions 9 and 10 are extended to triples, and as follows:*

$$graph[!\,Q] = \mathsf{box}[graph[Q]]$$

$$term[(\mathsf{G},\mathsf{M},\ell)] = (\widetilde{z})\Big(\prod_{e\in\mathsf{G}} term[e,\rho] \mid \prod_{B\in\mathsf{M}} term[B,\rho]\Big)$$
$$term[\langle\mathsf{G},S\rangle,\rho] = !\,\Big(\rho(S)\Big)\Big(\prod_{e\in\mathsf{G}} term[e,\rho]\Big)$$

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

Propositions 11, 12 and Theorem 13 still hold for the present extension and can be demonstrated in similar ways by first replacing the $\equiv_r$ reduction rule with three new reduction rules, accounting 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.

6.3 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 [7] and Parrow's trios [15].

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

Theorem 22 (Decomposition).

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

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

In Figure 8 we draw a diagram showing the basic units for implementing any SD-box.

The computational cost of the restriction in Theorem 22 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 $\widetilde{u}$.

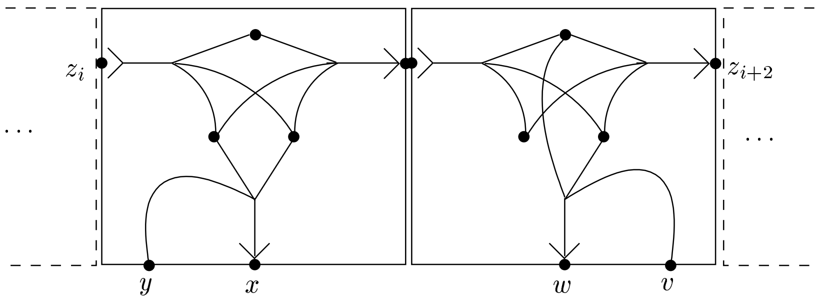


Fig. 8. The decomposition of $!\,(u_1u_2u_3)(\cdots \mid \overline{x}\,yu_1u_2 \mid \overline{w}\,u_3u_2v \mid \cdots)$.

We remark that if we systematically apply Theorem 22 to all boxes containing at least 2 edges, then the third reduction rule in Definition 20 (internal box reduction) becomes redundant.

6.4 Proof Nets with Exponentials

Most of the beauty of the connection in Section 5 between multiplicative proof nets and SDs is lost when boxes come into the picture. The reason is that boxes are used differently. Let us discuss this question is some detail.

In linear logic, exponentials consist of two operators: the *of-course*, noted !, which models resources that can be instantiated at wish, and the *why-not*, noted ?, which models resource instantiations. The operator of-course is close (but not equivalent) to our replication operator, while why-not has no counterpart. With the why-not we can represent nets that transform a resource $!\mathsf{N}$ to a single instance of the resource N. Such operations are reasonable in a functional language like proof nets, where resources, when plugged in some context agent, cannot be used by other agents. In this respect, a user instantiates the resource as many times as it needs and then erases the resource.

In a concurrent scenario, where systems are open, resources may be used by several agents who are not aware of each other. This situation is particularly complicated for mobile agents where new agents may dynamically come into play. It is evident that 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 concerns the of-course operator. A reduction inside the of-course does not imply creating copies of its contents, but instead modifies the original term. 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.

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

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

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

Fig. 9. Fusion rules for the calculus of solos.

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 [11].

7 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 [16].

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 9, which relaxes the side conditions of the previous rules (Figure 3). 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 23. *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* nodes$[\mathsf{G}]$ *to* $\mathcal{P}_f(\mathcal{N})$ *such that, for every* $a, a' \in dom(\ell)$*,* $a \neq a'$ *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 7 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 24. *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* $\{a_i = a'_i : 1 \leq i \leq k\}$, $ran(\sigma) \cap dom(\sigma) = \emptyset$*;*
  - $\forall a : \ell'(a) = \bigcup_{\sigma(a')=a} \ell(a')$

As an example of MSD reduction, in Figure 10 we illustrate the dynamics of an agent that is irreducible with the semantics of section 3.

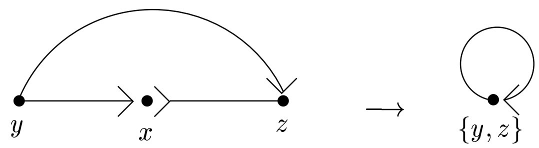


Fig. 10. The transition $(\overline{x}\, y \mid x\, z \mid \overline{z}\, y) \xrightarrow{\{y=z\}} \overline{z}\, y$.

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 9, and *vice versa*. Finally, basic MSDs can be extended with boxes, in the style of section 6.

8 Related Work

The direct ancestor of our solo diagrams are Parrow's Interaction Diagrams [14], 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 [9], 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 [12] 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 [4], 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 [3].) 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. His proof net encoding in terms of reaction graphs is informal.

Bellin and Scott also give a detailed interpretation of proofs in terms of π-calculus processes [1]. Overall, their interpretations use prefixes. As a consequence, reductions that do not occur at the bottom of the proofs are not mirrored. Also, no correspondence between proof nets and π-calculus graphs is provided.

Yoshida in [18] discusses the relationship between processes and a general framework for proof nets, the so-called Interaction Nets [8]. 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

[1] G. Bellin and P. Scott. On the π-calculus and linear logic. *Theoretical Computer Science*, 135:11–65, 1994. Also published as LFCS report ECS-LFCS-92-232, LFCS, University of Edinburgh.

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

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

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

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

[6] J.-Y. Girard. Notes on linear logic and ludics. The Linear Summer School, Azores, August-September 2000.

[7] G. Gonthier, M. Abadi, and J.-J. Lévy. The geometry of optimal lambda reduction. In *POPL '92. Proceedings of the nineteenth annual ACM Symposium on Principles of Programming Languages*, pages 15–26. ACM Press, 1992.

[8] Y. Lafont. Interaction nets. In *POPL '90. Proceedings of the seventeenth annual ACM Symposium on Principles of Programming Languages*, pages 95–108. ACM Press, 1990.

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

[10] R. Milner. Sorts in the π-calculus (extended abstract). In E. Best and G. Rozenberg, editors, *Proceedings of the 3rd Workshop on Concurrency and Compositionality*, volume 191 of *GMD-Studien*. GMD Bonn, St. Augustin, 1991. Also available as Report 6/91 from University of Hildesheim.

[11] R. Milner. Functions as processes. *Journal of Mathematical Structures in Computer Science*, 2(2):119–141, 1992. Previous version as Rapport de Recherche 1154, INRIA Sophia-Antipolis, 1990, and in *Proceedings of ICALP '91*, LNCS 443.

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

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

[14] J. Parrow. Interaction diagrams. *Nordic Journal of Computing*, 2:407–443, 1995. A previous version appeared in *Proceedings of A Decade in Concurrency*, LNCS 803: 477–508, 1993.

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

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

[17] D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms*. PhD thesis, LFCS, University of Edinburgh, 1993. CST-99-93 (also published as ECS-LFCS-93-266).

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