# The Fusion Calculus: Expressiveness and Symmetry in Mobile Processes (extended abstract) *

Joachim Parrow
Royal Institute of Technology
Sweden

Björn Victor
Uppsala University
Sweden

## Abstract

*We present the* fusion calculus *as a significant step towards a canonical calculus of concurrency. It simplifies and extends the π-calculus.*

*The fusion calculus contains the polyadic π-calculus as a proper subcalculus and thus inherits all its expressive power. The gain is that fusion contains actions akin to updating a shared state, and a scoping construct for bounding their effects. Therefore it is easier to represent computational models such as concurrent constraints formalisms. It is also easy to represent the so called strong reduction strategies in the λ-calculus, involving reduction under abstraction. In the π-calculus these tasks require elaborate encodings.*

*Our results on the fusion calculus in this paper are the following. We give a structured operational semantics in the traditional style. The novelty lies in a new kind of action,* fusion *actions for emulating updates of a shared state. We prove that the calculus contains the π-calculus as a subcalculus. We define and motivate the bisimulation equivalence and prove a simple characterization of its induced congruence, which is given two versions of a complete axiomatization for finite terms. The expressive power of the calculus is demonstrated by giving a straight-forward encoding of the strong lazy λ-calculus, which admits reduction under λ abstraction.*



## 1 Introduction

We present the *fusion calculus*, which we claim is a significant step towards a canonical calculus of concurrency. It simplifies and extends the π-calculus. We mention three indications of its greater expressiveness:

1. Computational models involving a shared state, such as concurrent constraints formalisms, can only be represented indirectly in the π-calculus by introducing extra processes for the state. In the fusion calculus the basic mechanisms for updating a shared state are already present and the models have simpler representations.

2. Strong reduction strategies in the λ-calculus, involving reduction under abstraction, are awkward to encode in the π-calculus. These are important e.g. for optimization techniques such as partial evaluation. In the fusion calculus reduction under abstraction can be represented at no extra cost.

3. The polyadic π-calculus is a subcalculus of the fusion calculus. Therefore everything that can be done in the polyadic π-calculus can be done in the fusion calculus without added complications.

The dramatic and unexpected result is that we achieve this improvement by *simplifying* the π-calculus rather than adding features to it. Proofs about the calculus actually get shorter and clearer. We mention three reasons why the fusion calculus is simpler:

1. It has only one binding operator. All other process calculi for value-passing including the π-calculus has at least two: binding of input variables and restriction of local ports.

2. It has a complete symmetry between input and output actions; they are dual versions of a single kind of communication action. In the π-calculus they are not duals because input entails a binding where output does not.

3. It has only one sensible bisimulation equivalence, where the other value-passing calculi have a proliferation into "early", "late" and "open" varieties due to different strategies to instantiate variables.

The price to be paid is the introduction of a new kind of actions, "fusion" actions. These actions are easily and cleanly incorporated into the formal framework. In the following we shall elaborate on these points, comparing with the polyadic $\pi$-calculus.

**Background on the $\pi$-calculus:** The $\pi$-calculus postulates a set of *names* $u, v, \ldots, z$ used to represent communication ports, values, variables etc. (more refined versions of $\pi$ have type systems where these can be distinguished but that is not necessary for the development in this paper). We use the *polyadic* calculus where an output or input along a port may carry more than one object. An interaction occurs between two processes, a sender performing an output and a receiver performing an input. The effect of the interaction is local to the receiver: its input variables become instantiated with the objects emitted by the sender. An example is:

$$\overline{u}\, vw \,.\, P \mid u(xy) \,.\, Q \;\xrightarrow{\tau}\; P \mid Q\{v/x, w/y\}$$

The output prefix $\overline{u}\, vw$ sends two objects $v$ and $w$ along $u$, and the input prefix $u(xy)$, which binds $x$ and $y$, receives two objects. Their interaction gives rise to a substitution which is local to $Q$. Additionally these prefixes imply temporal precedence: $P$ and $Q$ may not perform any transitions until the prefixes have been performed.

Another kind of binding occurs in the *restriction* operator: in $(\boldsymbol{\nu} x)P$ the name $x$ in $P$ is local to $P$. This means that it is distinct from any name occurring outside $P$, and moreover that as $P$ and the environment evolve during execution it will remain distinct.

Other kinds of bindings have been suggested though not developed, notably a *delayed input* construct such as $u(x){:}\, P$ which is input without the temporal precedence. Here $P$ may execute and can also at any time receive something for $x$ along $u$. The binding of $x$ extends into $P$ which is the reason this is different from $u(x) \,.\, \mathbf{0} \mid P$.

**Fusions:** In the fusion calculus the input primitive does *not* bind names. The effect of an interaction is that the output objects and input objects become identified in what we call a *fusion.* There are two differences between fusions and $\pi$-calculus interactions: the effect of a fusion is not necessarily local but regulated by the use of a *scope* operator, and fusions are symmetric with respect to input-output polarities. We now explain these points.

Formally a fusion is just an equivalence relation on names signifying that names related by it should be considered equal. This effect is applicable not only to the receiver but also to the sender and to any process running in parallel with them. Therefore it can be thought of as an update of a (not explicitly represented) shared state. For example, an interaction between $\overline{u}\, vw \,.\, P$ and $u\, xy \,.\, Q$ will result in a fusion of $v$ and $x$, and of $w$ and $y$. We write this as $\{v = x, w = y\}$. An interaction is written

$$\overline{u}\, vw \,.\, P \mid u\, xy \,.\, Q \;\xrightarrow{\{v=x, w=y\}}\; P \mid Q$$

The fusion is placed above the arrow to signify its unbounded scope. When further transitions from the derivative $P \mid Q$ are considered, they will be treated in an environment which identifies $v$ with $x$ and $w$ with $y$. Our definition of equivalence between processes will ensure this effect. The fact that the fusion is "global" can be seen by including further parallel processes, these will also be similarly affected by the environment, for example in

$$\overline{u}\, vw \,.\, P \mid u\, xy \,.\, Q \mid R \mid S \;\xrightarrow{\{v=x, w=y\}}\; P \mid Q \mid R \mid S$$

This ability to express global effects is the reason why the fusion calculus gains in expressiveness over the $\pi$-calculus.

The fact that the interaction is symmetric can be seen by inverting the input-output polarities of the actions:

$$u\, vw \,.\, P \mid \overline{u}\, xy \,.\, Q \;\xrightarrow{\{v=x, w=y\}}\; P \mid Q$$

The fusion here is the same as before. Therefore the designations "input" and "output" are arbitrary and a more appropriate terminology might be "action" and "co-action", though in this paper we shall retain the more familiar terms.

**Scopes:** The single binding operator is called *scope* and is written $(x)P$, meaning that the $x$ in $P$ is local. In a sense it is the common denominator of input binding and restriction in the $\pi$-calculus: an instantiation of the bound name is neither mandatory (like in input) nor impossible (like in restriction). The main idea is that scopes can be used to delimit the extent of fusions. As a first example we show how the $\pi$-calculus interaction can be emulated by scoping the input names:

$$\overline{u}\, vw \,.\, P \mid \Big((x)(y) u\, xy \,.\, Q\Big) \mid R \mid S$$

$$\xrightarrow{\mathbf{1}} \quad P \mid \Big(Q\{v/x, w/y\}\Big) \mid R \mid S$$

Here $\mathbf{1}$ is an inert action (formally it is just the identity fusion) and the effect of the interaction does not extend beyond the scopes. Since $x$ and $y$ are bound they are substituted by some non-bound names fused to them.

The increased expressiveness over the $\pi$-calculus is that scopes may be placed in arbitrary places and not only directly over the input actions. For example

$$\begin{array}{l}(x)\Big(\overline{u}\,vw\,.\,P \mid (y)\Big(u\,xy\,.\,Q \mid R\Big) \mid S\Big) \\ \quad \xrightarrow{\mathbf{1}} \quad \Big(P \mid \Big(Q \mid R\Big)\{w/y\} \mid S\Big)\{v/x\}\end{array}$$

where the scopes determine the extent and direction of the resulting substitutions.

As an example of the expressiveness the delayed input is now easy to define: in

$$(x)(ux \mid P)$$

it is possible for $P$ to execute before $ux$ interacts with an action of kind $\overline{u}v$, fusing $x$ in $P$ with $v$. Another example is the polyadic input of the same name twice:

$$(z)u\,zz \mid P$$

Here any two names emitted by $P$ along $u$ will be fused, even if they are bound in $P$, for example

$$(z)u\,zz \mid \overline{u}\,vw\,.\,P \xrightarrow{\{v=w\}} P$$

A similar effect is obtained by two monadic inputs in $(z)uz\,.\,uz \mid P$.

**Related work:** Our earlier work [11] attacks some of these problems in a monadic calculus where only one object is transmitted in each interaction. The main result there, the *update calculus*, unfortunately turned out not to generalize to polyadic interactions. The reason is that updates (like substitutions) are directed, and it is unclear what a polyadic update like $[a/x, b/x]$ should mean. In contrast, the polyadic fusion $\{a = x, b = x\}$ is unproblematic: it is just the equivalence equating all of $a, b$ and $x$. The fusion calculus has a higher degree of symmetry and this has profound implications for our technical proofs.

In [21] we explore the weak version of the equivalence defined in the present paper, give a reduction semantics of the fusion calculus, and show how to encode three variants of the $\rho$-calculus [9], a foundational calculus for concurrent constraints.

Another way to obtain an input-output symmetry is in the $\pi$I-calculus developed by Sangiorgi [16]. $\pi$I is a subcalculus of $\pi$: output objects must be bound by restriction, so both input and output bind their objects. Through clever encodings it has been established that $\pi$I exhibits a considerable expressive power despite this limitation. In contrast, the fusion calculus achieves the same symmetry with an increase in expressiveness, by allowing both input and output to carry non-bound objects.

The problem of modelling strong lazy reductions in the $\lambda$-calculus has been considered by Fu [4] and by Sangiorgi [16]. Both have postulated solutions though not published any analysis. Fu extends the $\pi$-calculus to something more complicated, where we instead simplify it. Sangiorgi's solution is a comparatively complicated encoding in the $\pi$I-calculus, where we instead can formulate a straightforward modification of Milner's original encoding and proofs for the (non-strong) lazy $\lambda$-calculus.

There is a host of work on the $\pi$-calculus following the original presentation [8]. We here only mention that inspiration for the present paper comes from Milner's encoding of the $\lambda$-calculus [6], the axiomatizations by Parrow and Sangiorgi [10], and Sangiorgi's "open" bisimulation equivalence [17]. Milner's action calculi [7] also aim to find a conceptually more fundamental formalism, though in a different direction.

**Future work:** To model full $\lambda$-calculus reduction, we will look at extensions to multiway communication. Such extensions are probably easier in the fusion calculus than in the $\pi$-calculus where the distinction between input and output is a complication. It should be interesting to look at the relation between fusion calculus and the $\pi$I-calculus, and to study an asynchronous version of the fusion calculus. Type systems and modal logics should also be developed, based on existing work for the $\pi$-calculus. In the case of modal logics we expect a simplification. Finally, analysis algorithms and tools such as the MWB [20] are being adapted and extended to handle the fusion calculus.

**Main results and overview of the rest of the paper:** In Section 2 we define the syntax and semantics of the fusion calculus in a way that has now become standard: first we give a structural congruence saying which agents are declared to have the same transitions, and then a labelled transition system where the labels are either communication actions or fusions. We also formally compare the fusion calculus with the $\pi$-calculus. In Section 3 we define bisimulation equivalence as the largest sensible (strong) bisimulation equivalence, and show that the largest congruence contained in it has a pleasant characterization through a kind of substitution-closed bisimulations. The resulting congruence, called hyperequivalence, is stronger than any of the equivalences which have been suggested for the $\pi$-calculus. In Section 4 we proceed to give complete axiomatizations in two versions, with and without the mismatch operator. Finally in Section 5 we demonstrate how the strong lazy reduction strategy in the $\lambda$-calculus has a straightforward encoding. The proofs are presented in the full version of this paper [12] and in the second author's PhD thesis [19].

## 2 Syntax and Semantics

We assume an infinite set $\mathcal{N}$ of *names* ranged over by $u, v, \ldots, z$. Like in the $\pi$-calculus, names represent communication ports, which are also the values transmitted. We write $\tilde{x}$ for a (possibly empty) finite sequence $x_1 \cdots x_n$ of names. $\varphi$ ranges over total equivalence relations over $\mathcal{N}$ (i.e. equivalence relations with $\mathrm{dom}(\varphi) = \mathcal{N}$) with only finitely many non-singular equivalence classes. We write $\{\tilde{x} = \tilde{y}\}$ to mean the smallest such equivalence relation relating each $x_i$ with $y_i$, and write $\mathbf{1}$ for the identity relation. As a consequence, a fusion written $\{x = x\}$ is the same as $\{y = y\}$, namely $\mathbf{1}$.

**Definition 1** *The* free actions*, ranged over by $\alpha$, and the* agents*, ranged over by $P, Q, \ldots$, are defined by*

$$
\begin{array}{lll}
\alpha ::= & u\tilde{x} & \text{(Input)} \\
 & \overline{u}\tilde{x} & \text{(Output)} \\
 & \varphi & \text{(Fusion)} \\
 & & \\
P ::= & \mathbf{0} & \text{(Inaction)} \\
 & \alpha \,.\, Q & \text{(Prefix)} \\
 & Q + R & \text{(Summation)} \\
 & Q \mid R & \text{(Composition)} \\
 & (x)Q & \text{(Scope)}
\end{array}
$$

Input and output actions are collectively called *free communication* actions. In these, the names $\tilde{x}$ are the *objects* of the action, and the name $u$ is the *subject*. We write $a$ to stand for either $u$ or $\overline{u}$, thus $a\tilde{x}$ is the general form of a communication action. Fusion actions have neither subject nor objects.

Prefixing an agent $Q$ means that the agent must perform the action $\alpha$ before acting like $Q$. We often omit a trailing $\mathbf{0}$ and write $\alpha$ for $\alpha \,.\, \mathbf{0}$ if no confusion can arise. Like in the $\pi$-calculus, Summation is alternative choice and Composition lets agents act in parallel.

The Scope $(x)Q$ limits the scope of $x$ to $Q$; no visible communication action of $(x)Q$ can have $x$ as its subject, and fusion effects with respect to $x$ are limited to $Q$. Restriction and input binding of $\pi$-calculus can be recovered as special cases of Scope, as we will see below.

The name $x$ is said to be *bound* in $(x)P$. We write $(\tilde{x})P$ for $(x_1) \cdots (x_n)P$. The *free names* in $P$, denoted $\mathrm{fn}(P)$, are the names in $P$ with a non-bound occurrence; here the names occurring in the fusion $\varphi$ is defined to be the names in the non-singular equivalence classes, i.e. in the relation $\varphi - \mathbf{1}$. As usual we will not distinguish between alpha-variants of agents, i.e., agents differing only in the choice of bound names.

The action of a transition may be free or bound:

**Definition 2** *The* actions*, ranged over by $\gamma$, consist of the fusion actions and of communication actions of the form $(z_1) \cdots (z_n)a\tilde{x}$ (written $(\tilde{z})a\tilde{x}$), where $n \geq 0$ and all elements in $\tilde{z}$ are also in $\tilde{x}$. If $n > 0$ we say it is a* bound *action.*

Note that there are no bound fusion actions. In the communication actions above, $\tilde{z}$ are the *bound objects* and the elements in $\tilde{x}$ that are not in $\tilde{z}$ are the *free objects*. Free actions have no bound objects. We further write $\mathrm{n}(\gamma)$ to mean all names occurring in $\gamma$ (i.e., also including the subject of communication actions and the names in non-singular equivalence classes in fusion actions).

For convenience we define $\varphi \backslash z$ to mean

$$\varphi \cap (\mathcal{N} - \{z\})^2 \cup \{(z, z)\}$$

i.e., the equivalence relation $\varphi$ 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 now define a structural congruence which equates all agents we will never want to distinguish for any semantic reason, and then use this when giving the transitional semantics.

**Definition 3** *The* structural congruence*, $\equiv$, between agents is the least congruence satisfying the abelian monoid laws for Summation and Composition (associativity, commutativity and $\mathbf{0}$ as identity), and the scope laws*

$$(x)\mathbf{0} \equiv \mathbf{0}, \quad (x)(y)P \equiv (y)(x)P,$$
$$(x)(P + Q) \equiv (x)P + (x)Q$$

*and also the* scope extension *law*
$P \mid (z)Q \equiv (z)(P \mid Q)$ *where* $z \notin \mathrm{fn}(P)$.

**Definition 4** *The family of* transitions $P \xrightarrow{\gamma} Q$ *is the least family satisfying the laws in Table 1. In this definition structurally equivalent agents are considered the same, i.e., if* $P \equiv P'$ *and* $Q \equiv Q'$ *and* $P \xrightarrow{\gamma} Q$ *then also* $P' \xrightarrow{\gamma} Q'$.

Use of the SCOPE rule entails a substitution of the scoped name $z$ for a nondeterministically chosen name $x$ related to it by $\varphi$. For the purpose of the equivalence defined below in Section 3 it will not matter which such $x$ replaces $z$. The only rule dealing with bound actions is OPEN. This simplifies proofs, e.g., by induction on the depth of transition inference. Using structural congruence, pulling the relevant scope to the top level, we can still infer e.g. $P \mid (x)ayx\,.\,Q \xrightarrow{(x)ayx} P \mid Q$ using PREF and OPEN (provided $x \notin \mathrm{fn}(P)$, otherwise an alpha-conversion is necessary). It is clear that for the purpose of the semantics, fusion prefixes can be regarded as derived forms since $\{\tilde{y} = \tilde{z}\}\,.\,P$ has exactly the same transitions as $(u)(\overline{u}\tilde{y}\,.\,\mathbf{0} \mid u\tilde{z}\,.\,P)$ when $u \notin \mathrm{fn}(P)$. (In the same way the $\tau$ prefix can be regarded as derived in CCS and in the $\pi$-calculus.)

As an example of the expressiveness it is now easy to define a *delayed input* construct $u(x)\colon P$, which is input without the temporal precedence. Here $P$ may execute and can also at any time receive something for $x$ along $u$. The binding of $x$ extends into $P$, which is the reason this is different from $u(x)\,.\,\mathbf{0} \mid P$. One of the reasons why this operator has only been suggested but not formally developed in the $\pi$-calculus is that it would have quite complicated effects on scope extrusion and intrusion. In the fusion calculus on the other hand, the delayed input can without complications be defined as $(x)(ux \mid P)$; here it is possible for $P$ to execute before $ux$ interacts with an action of kind $\overline{u}v$, fusing $x$ in $P$ with $v$. Another example of expressiveness is the polyadic input of the same name twice: $(z)u\,zz \mid P$. Here any two names emitted by $P$ along $u$ will be fused, even if they are bound in $P$, for example

$$(z)u\,zz \mid \overline{u}\,vw\,.\,P \xrightarrow{\{v=w\}} P$$

A similar effect is obtained by two monadic inputs in $(z)uz\,.\,uz \mid P$. For more examples see the introduction.

The polyadic $\pi$-calculus [5] is isomorphic to a subcalculus of the fusion calculus, formed by omitting fusion prefixes and by requiring all input prefixes $u\tilde{x}\,.\,P$ to occur immediately under a scope $(\tilde{x})$ of the objects in the input prefix; the objects must be pairwise distinct. This way, no fusion actions except $\mathbf{1}$ are possible, since the objects are always sufficiently scoped. Call this subcalculus $f_\pi$. The isomorphism $\leftrightarrow$ between $f_\pi$ and $\pi$ is simply the homomorphic extension of

$$\begin{array}{rcll} (\tilde{x})u\tilde{x}\,.\,P & \leftrightarrow & u(\tilde{x})\,.\,P & \\ (z)P & \leftrightarrow & (\boldsymbol{\nu} z)P & \text{if } P \text{ not a free input prefix} \end{array}$$

The rationale behind this is as follows. Consider a scope $(x)Q$ in $f_\pi$. If it is immediately over an input prefix, $Q = ux\,.\,P$, then before anything happens in $P$ something must be received along $u$ to replace $x$ in $P$, precisely as the effect of a $\pi$-calculus input prefix $u(x)\,.\,P$. On the other hand, if it is not immediately over an input prefix, $Q \neq ux\,.\,P$, then there cannot be any input prefix inside $Q$ for that particular $x$. Because of the syntactic restrictions in $f_\pi$, a free input $ux$ not at the top level of $Q$ must be enclosed by another scope $(x)$, making the object different from the $x$ in $(x)Q$. So this $x$ cannot occur in an input and there are no fusion prefixes, therefore $x$ will remain distinct from other names throughout execution of the agent. This is precisely the effect of $(\boldsymbol{\nu} x)Q$.

We identify actions of the $\pi$-calculus and of $f_\pi$ as follows: $u(\tilde{x}) = (\tilde{x})u\tilde{x}$, $(\boldsymbol{\nu}\tilde{z})\overline{u}\,\tilde{y} = (\tilde{z})\overline{u}\,\tilde{y}$ and $\tau = \mathbf{1}$. Decorating the transition arrows with their respective calculus, we can then formalize the result as

**Theorem 1** *For any* $\pi$*-calculus agent* $P$ *and* $Q$ *an* $f_\pi$*-calculus agent such that* $P \leftrightarrow Q$, $P \xrightarrow{\gamma}_\pi P'$ *implies* $Q \xrightarrow{\gamma}_f Q'$ *with* $P' \leftrightarrow Q'$, *and* $Q \xrightarrow{\gamma}_f Q'$ *implies* $P \xrightarrow{\gamma}_\pi P'$ *with* $P' \leftrightarrow Q'$.

Because of this isomorphism of transitions, the theory of the $\pi$-calculus holds for $f_\pi$, and vice versa.

The $\pi$I-calculus by Sangiorgi [16] is a subset of the $\pi$-calculus formed by requiring all output prefixes $\overline{u}\tilde{x}$ to occur immediately below a restriction $(\boldsymbol{\nu}\tilde{x})$ of the objects. Therefore that calculus is isomorphic to the subcalculus of $f_\pi$ requiring that *all* prefixes — not merely input prefixes — occur under a scope of the object. As Sangiorgi has remarked the $\pi$I-calculus has a pleasing symmetry between input and output actions. The fusion calculus has the same kind of symmetry, achieved by extending rather than restricting the $\pi$-calculus, as implied by the following theorem where $\overline{P}$ ($\overline{\gamma}$) stands for $P$ ($\gamma$) with all input/output polarities inverted:

**Theorem 2** $P \xrightarrow{\gamma} Q$ *iff* $\overline{P} \xrightarrow{\overline{\gamma}} \overline{Q}$.

## 3 Equivalence and Congruence

Our notion of equivalence uses the well known idea of bisimulation. In essence, for two agents to be bisimilar, each transition from one agent must be mimicked

| | | | | | |
|---|---|---|---|---|---|
| PREF | $\dfrac{-}{\alpha\,.\,P \xrightarrow{\alpha} P}$ | SUM | $\dfrac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'}$ | PAR | $\dfrac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q}$ |
| COM | $\dfrac{P \xrightarrow{u\tilde{x}} P',\ Q \xrightarrow{\overline{u}\tilde{y}} Q',\ \lvert\tilde{x}\rvert = \lvert\tilde{y}\rvert}{P \mid Q \xrightarrow{\{\tilde{x}=\tilde{y}\}} P' \mid Q'}$ | | | SCOPE | $\dfrac{P \xrightarrow{\varphi} P',\ z\,\varphi\,x,\ z \neq x}{(z)P \xrightarrow{\varphi\backslash z} P'\{x/z\}}$ |
| PASS | $\dfrac{P \xrightarrow{\alpha} P',\ z \notin \mathrm{n}(\alpha)}{(z)P \xrightarrow{\alpha} (z)P'}$ | OPEN | $\dfrac{P \xrightarrow{(\tilde{y})a\,\tilde{x}} P',\ z \in \tilde{x} - \tilde{y},\ a \notin \{z, \overline{z}\}}{(z)P \xrightarrow{(z\tilde{y})a\,\tilde{x}} P'}$ | | |

Table 1: The Fusion Calculus: Laws of action.

from the other, reaching agents which are again bisimilar. In calculi where transitions can contain placeholders (such as variables) for values, care has been taken to properly define what "again bisimilar" means. For example, in the $\pi$-calculus there are varieties such as *early*, *late* [8], and *open* [17] bisimulation. These differ in the use of universal quantification over the names, and each leads to a different congruence.

In the fusion calculus this diversity is mercifully absent; there is only one sensible bisimulation congruence. The reason is that the fusion actions allow us to formulate more powerful contexts than what is possible in the $\pi$-calculus. Fusion actions can enforce a global substitution of *any* name, whereas in the $\pi$-calculus no action can enforce a substitution on a name bound by a restriction. Therefore a congruence for the fusion calculus must be closed under arbitrary substitutions in a way which will be made precise below. We use $\sigma$ to range over substitutions of names.

**Definition 5** *A substitution $\sigma$* agrees with *the fusion $\varphi$ if $\forall x, y : x\,\varphi\,y \Leftrightarrow \sigma(x) = \sigma(y)$. A* substitutive effect *of a fusion $\varphi$ is a substitution $\sigma$ agreeing with $\varphi$ such that $\forall x, y : \sigma(x) = y \Rightarrow x\,\varphi\,y$ (i.e., $\sigma$ sends all members of the equivalence class to one representative of the class). The only substitutive effect of a communication action is the identity substitution.*

For example, the substitutive effects of $\{x = y\}$ are $\{x/y\}$ and $\{y/x\}$, and the only substitutive effect of $\mathbf{1}$ is the identity substitution.

**Definition 6** *A* bisimulation *is a binary symmetric relation $\mathcal{S}$ between agents such that $P\ \mathcal{S}\ Q$ implies:*

*If $P \xrightarrow{\gamma} P'$ with $\mathrm{bn}(\gamma) \cap \mathrm{fn}(Q) = \emptyset$ then*
*$Q \xrightarrow{\gamma} Q'$ and $P'\sigma\ \mathcal{S}\ Q'\sigma$*
*for some substitutive effect $\sigma$ of $\gamma$.*

*$P$ is bisimilar to $Q$, written $P \dot\sim Q$, if $P\ \mathcal{S}\ Q$ for some bisimulation $\mathcal{S}$.*

This definition differs from *ground* bisimulation of [17] only in the treatment of fusion actions. A fusion $\{x = y\}$ represents an obligation to make $x$ and $y$ equal everywhere. Therefore, if $\gamma$ above is such a fusion, it only makes sense to relate $P'$ and $Q'$ when a substitution $\{y/x\}$ or $\{x/y\}$ has been performed. Note that it does not matter which substitution we choose, since $P\{x/y\} \dot\sim Q\{x/y\}$ implies $P\{y/x\} \dot\sim Q\{y/x\}$, by the simple fact that $P\{x/y\}\{y/x\} \equiv P\{y/x\}$ and that bisimulation is closed under injective substitutions.

Thus we argue that any sensible bisimulation-like equivalence must contain bisimilarity. But bisimilarity is not a congruence. An easy example is that $x \mid \overline{y}$ is bisimilar to $x\,.\,\overline{y} + \overline{y}\,.\,x$, but prefixing both with $\{x = y\}$ will result in non bisimilar agents. We therefore look for the largest congruence included in bisimilarity. This is achieved by closing the definition of bisimulation under arbitrary substitutions.

**Definition 7** *A* hyperbisimulation *is a substitution closed bisimulation, i.e., a bisimulation $\mathcal{S}$ with the property that $P\ \mathcal{S}\ Q$ implies $P\sigma\ \mathcal{S}\ Q\sigma$ for any substitution $\sigma$. Two agents $P$ and $Q$ are* hyperequivalent, *written $P \sim Q$, if they are related by a hyperbisimulation.*

**Theorem 3** *Hyperequivalence is the largest congruence in bisimilarity.*

The proof that it is really the largest of all congruences in bisimilarity goes by constructing a context which can perform any relevant substitutions at any time, and by showing that if two agents are equivalent in that context, which they must be if they are congruent, then they are also hyperequivalent. The context is of type $(\cdot \mid S_a(N))$, where $N$ contains at least the free names of the agents in the context, and $a$ is the maximal arity of communication actions in the agents. We give the definition of $S_a(N)$ below, where we assume $\{s, d, r_x, e : x \in N\}$ are fresh names. The $s$-prefixed terms perform arbitrary fusions, and the other terms collect new names from the agents.

$$\begin{array}{rcl} S_a(N) & \stackrel{def}{=} & \sum_{x,y\in N} s\,.\,\{x=y\}\,.\,d\,.\,S_a(N) \\ & + & \sum_{x\in N} r_x\,.\sum_{|\tilde{u}|<a,\tilde{u}\cap N=\emptyset}(\tilde{u})x\tilde{u}\,.\,\overline{e}\,\tilde{u}\,.\,S_a(N\tilde{u}) \\ & + & \sum_{x\in N} \overline{r_x}\,.\sum_{|\tilde{u}|<a,\tilde{u}\cap N=\emptyset}(\tilde{u})\overline{x}\,\tilde{u}\,.\,e\tilde{u}\,.\,S_a(N\tilde{u}) \end{array}$$

It is interesting to relate hyperequivalence to open equivalence, which is a congruence in the $\pi$-calculus and hence also in $f_\pi$. For a definition of open we refer to [17], but even without the details of the definition a reader might appreciate the following arguments. The $f_\pi$-calculus agents

$$P \equiv (xy)\overline{u}\,xy\,.\,(x \mid \overline{y})$$

and

$$Q \equiv (xy)\overline{u}\,xy\,.\,(x\,.\,\overline{y} + \overline{y}\,.\,x)$$

are open equivalent but not hyperequivalent. To see this, after a bound output transition the agents become $x \mid \overline{y}$ and $x\,.\,\overline{y} + \overline{y}\,.\,x$ and these are not bisimilar under the substitution $\{x/y\}$. This substitution, which affects names previously emitted as bound objects, is not relevant for open equivalence. Now put the two agents in parallel with $(z)u\,zz$ (note that this agent is not in $f_\pi$ because the two input objects are not distinct). There is then one more $\mathbf{1}$ transition from $(z)u\,zz \mid P$ than from $(z)u\,zz \mid Q$. In essence, the parallel context $(z)u\,zz$ will enforce a fusion of any two names emitted along $u$. This demonstrates that open equivalence is not a congruence in the fusion calculus, and that this context cannot be encoded compositionally in the $\pi$-calculus. A similar argument holds for the monadic $(z)uz\,.\,uz$.

Finally, to relate the fusion calculus to its monadic predecessor, the *update calculus* [11], we regard both a monadic fusion prefix $\{x = y\}\,.\,P$ and an update prefix $[x/y]\,.\,P$ as defined by $(u)(\overline{u}\,x \mid uy\,.\,P)$ (where $u$ is fresh).

**Theorem 4** *For $P$ and $Q$ monadic fusion calculus processes, it holds that if $P \sim Q$ in the update calculus, then $P \sim Q$ in the fusion calculus.*

The reverse does not hold, since $[x/y]P \sim [y/x]P$ in the fusion calculus but not in the update calculus; we disregard the direction of the substitution when moving from update to fusion actions.

| | | |
|---|---|---|
| **Summation** | | |
| **S1** | $P + \mathbf{0} = P$ | |
| **S2** | $P + Q = Q + P$ | |
| **S3** | $P + (Q + R) = (P + Q) + R$ | |
| **Scope** | | |
| **R0** | $(x)\mathbf{0} = \mathbf{0}$ | |
| **R1** | $(x)(y)P = (y)(x)P$ | |
| **R2** | $(x)(P + Q) = (x)P + (x)Q$ | |
| **Match and Scope** | | |
| **RM1** | $(x)[y = z]P = [y = z](x)P$ | if $x \neq y, x \neq z$ |

Table 2: Axioms from structural congruence.

## 4 Axiomatization

We now introduce two additional operators in the fusion calculus: if $P$ is an agent then so are $[x = y]P$, a *match*, and $[x \neq y]P$, a *mismatch*, with the laws of action

$$\text{MATCH}\ \frac{P \xrightarrow{\alpha} P'}{[x = x]P \xrightarrow{\alpha} P'} \qquad \text{MISM}\ \frac{P \xrightarrow{\alpha} P',\, x \neq y}{[x \neq y]P \xrightarrow{\alpha} P'}$$

These operators are found in earlier work on the $\pi$-calculus; match has been used for a sound axiomatization of late congruence [8] as well as a complete axiomatization of open equivalence [17], and mismatch has been used for complete axiomatizations of the late and early congruence [10]. Below we give complete axiomatizations of hyperequivalence, both with and without mismatch.

Here and in the following we use $M, N$ to stand for a match or a mismatch operator, and define $\mathrm{n}([x = y]) = \mathrm{n}([x \neq y]) = \{x, y\}$. We add the scope law $(x)MP \equiv M(x)P$, *if* $x \notin \mathrm{n}(M)$ to the structural congruence. We write "match sequence" for a sequence of match *and* mismatch operators, ranged over by $\tilde{M}$, $\tilde{N}$, and we say that $\tilde{M}$ *implies* $\tilde{N}$, written $\tilde{M} \Rightarrow \tilde{N}$, if the conjunction of all matches and mismatches in $\tilde{M}$ logically implies all elements in $\tilde{N}$, and that $\tilde{M} \Leftrightarrow \tilde{N}$ if $\tilde{M}$ and $\tilde{N}$ imply each other. We write $\sum_{i\in I} P_i$ for finite general summation, $P_1 + \cdots + P_n$.

For the axiomatization of hyperequivalence we subsume the fact that the equivalence is a congruence. We also use some of the laws for structural congruence (see Table 2). The axioms of the fusion calculus are given in Table 3.

It is interesting to note that all axioms except the two for fusion are well known from previous axiomatizations of the $\pi$-calculus. For example, all the axioms from Sangiorgi's axiomatization of open bisimulation

| | | |
|---|---|---|
| **Summation** | | |
| **S4** | $P + P = P$ | |
| **Match** | | |
| **M1** | $\tilde{M}P = \tilde{N}P$ | if $\tilde{M} \Leftrightarrow \tilde{N}$ |
| **M2** | $[x = y]P = [x = y](P\{x/y\})$ | |
| **M3** | $MP + MQ = M(P + Q)$ | |
| **M4** | $[x \neq x]P = \mathbf{0}$ | |
| **M5** | $P = [x = y]P + [x \neq y]P$ | |
| **Scope** | | |
| **R3** | $(x)\alpha \,.\, P = \alpha \,.\, (x)P$ | if $x \notin \mathrm{n}(\alpha)$ |
| **R4** | $(x)\alpha \,.\, P = \mathbf{0}$ | if $x$ is the subject of $\alpha$ |
| **Match and Scope** | | |
| **RM2** | $(x)[x = y]P = \mathbf{0}$ | if $x \neq y$ |
| **Fusion** | | |
| **F1** | $\varphi \,.\, P = \varphi \,.\, [x = y]P$ | if $x \,\varphi\, y$ |
| **F2** | $(z)\varphi \,.\, P = \varphi \backslash z \,.\, P$ | if $z \notin \mathrm{fn}(P)$ |
| **Expansion** | | |
| **E** | for $P \equiv \Sigma_i M_i(\tilde{x}_i)\alpha_i.P_i$, $Q \equiv \Sigma_j N_j(\tilde{y}_j)\beta_j.Q_j$, $P \mid Q \;=\; \sum_i M_i(\tilde{x}_i)\alpha_i \,.\, (P_i \mid Q) + \sum_j N_j(\tilde{y}_j)\beta_j \,.\, (P \mid Q_j) + \sum_{\alpha_i \mathrm{opp} \beta_j} M_i N_j(\tilde{x}_i, \tilde{y}_j)[u_i = v_j]\{\tilde{z}_i = \tilde{w}_j\} \,.\, (P_i \mid Q_j)$ where $\alpha_i \mathrm{opp} \beta_j$ means $\alpha_i \equiv \overline{u_i}\tilde{z}_i$ and $\beta_j \equiv v_j \tilde{w}_j$. | |

Table 3: Axioms for hyperequivalence.

equivalence for the $\pi$-calculus [17] hold for hyperequivalence in the fusion calculus, except the axiom treating restriction and the weaker form of congruence wrt input prefix. Please note also that our expansion law **E** is syntactically simpler than the corresponding laws for $\pi$-calculus equivalences.

Distinction relations between names have been used in axiomatizing $\pi$-calculus equivalences in [8, 17], but are not needed in our axiomatization of fusion calculus. A notable difference is also that in the $\pi$-calculus the law $[x \neq y]\alpha \,.\, P \;=\; [x \neq y]\alpha \,.\, [x \neq y]P$ holds for early and late congruence [10], while it is not valid for hyperequivalence.

**Definition 8** *A match sequence $\tilde{M}$ is* complete *on a set of names $V$ if for some equivalence relation $\mathcal{R}$ on $V$, called the* equivalence relation corresponding to *$\tilde{M}$, it holds that $\tilde{M} \Rightarrow [x = y]$ iff $x \mathcal{R} y$, and $\tilde{M} \Rightarrow [x \neq y]$ iff $\neg(x \mathcal{R} y)$*

**Definition 9** *The* depth *of an agent $P$, $\mathrm{d}(P)$, is defined inductively as follows:*
$\mathrm{d}(\mathbf{0}) = 0$, $\mathrm{d}(\alpha \,.\, P) = 1 + \mathrm{d}(P)$, $\mathrm{d}((\tilde{x})P) = \mathrm{d}(MP) = \mathrm{d}(P)$, $\mathrm{d}(P \mid Q) = \mathrm{d}(P) + \mathrm{d}(Q)$, $\mathrm{d}(P + Q) = \max(\mathrm{d}(P), \mathrm{d}(Q))$.

We prove the completeness of the axiom system in a standard way: first we define a head normal form for agents, and show that any agent can be written on this form. We can then find exactly which part of one agent that was used to simulate the transition of the other.

**Definition 10** *An agent $P$ is in* head normal form *(HNF) on $V$ (a finite set of names) if $P$ is on the form*

$$\sum_{i \in I} \tilde{M}_i(\tilde{x}_i)\alpha_i \,.\, P_i$$

*where for all $i$, $\tilde{x}_i \cap V = \emptyset$, $\tilde{x}_i \subseteq \mathrm{obj}(\alpha_i)$ and $\tilde{M}_i$ is complete on $V$.*

**Lemma 5** *For all agents $P$ and finite $V$ such that $\mathrm{fn}(P) \subseteq V$, there is an agent $H$ such that $\mathrm{d}(H) \leq \mathrm{d}(P)$, $H$ is in HNF on $V$, and $\vdash P = H$ from the axioms of Tables 2 and 3.*

**Theorem 6** *$P \sim Q$ iff $\vdash P = Q$ from the axioms of Tables 2 and 3.*

For the subcalculus without mismatches another head normal form is needed:

**Definition 11** *An agent $P$ is in* mismatch-free head normal form (mHNF) *if $P$ is on the form $\sum_{i\in I}\tilde{M}_i(\tilde{x}_i)\alpha_i\,.\,P_i$, where*

1. $\forall i : \tilde{x}_i \cap \mathrm{fn}(P) = \emptyset$, *and* $\tilde{x}_i \subseteq \mathrm{obj}(\alpha_i)$
2. *if* $i \neq j$ *then* $\tilde{M}_i(\tilde{x}_i)\alpha_i\,.\,P_i \not\succ \tilde{M}_i(\tilde{x}_i)\alpha_i\,.\,P_i + \tilde{M}_j(\tilde{x_j})\alpha_j\,.\,P_j$

**Lemma 7** *For all mismatch-free agents $P$ there is an agent $H$ such that $\mathrm{d}(H) \leq \mathrm{d}(P)$, $H$ is in mHNF, and $\vdash P = H$ from the axioms of $\mathcal{E}$.*

**Theorem 8** *If $P$ and $Q$ contain no mismatch operators, then $P \sim Q$ iff $\vdash P = Q$ from the axioms of Tables 2 and 3, dropping axioms* **M4** *and* **M5** *and adding the axiom $MP + P = P$.*

($MP + P = P$ of course also holds in the presence of mismatch but it is derivable in the axiomatization in Theorem 6.) The proofs of Lemma 7 and Theorem 8 are by simultaneous induction over the depth of agents.

## 5 Encoding the $\lambda$-Calculus

Most process calculi with the ability of treating processes as first class objects (directly or indirectly) have attempted to embed or encode the $\lambda$-calculus, e.g. [2, 18, 6, 13, 14, 15, 3].

The lazy $\lambda$-calculus [1] has been modelled and analysed in the $\pi$-calculus by Milner [6] and Sangiorgi [13, 15]. The *strong* lazy reduction strategy, where reduction is possible under abstraction, is more difficult to encode. The problem is that input binding in $\pi$-calculus can only be done by temporal guarding, $u(x)\,.\,P$, while the $\lambda$-calculus reduction rule allows the body of an abstraction to reduce *before* the abstracted variable is instantiated. In the fusion calculus, by decoupling the input action from the binding of the object, we can define a *delayed* input $(x)(u\,x \mid P)$ with the desired property. Using other calculi where variants of delayed input can be defined, Sangiorgi [16] and Fu [4] have presented encodings (into $\pi$I and $\chi$-calculus, respectively). However they have not published any formal analysis of these encodings. In this section we present an encoding into the fusion calculus, and prove its accuracy. Our encoding is simpler than Sangiorgi's, and our calculus is more symmetric than that of Fu.

The strong lazy reduction strategy has the following reduction rules, where $M$ and $N$ range over $\lambda$ terms:

$$(\lambda x\,.\,M)N \rightarrow_\lambda M\{N/x\} \qquad \frac{M \rightarrow_\lambda M'}{MN \rightarrow_\lambda M'N}$$

$$\frac{M \rightarrow_\lambda M'}{\lambda x\,.\,M \rightarrow_\lambda \lambda x\,.\,M'}$$

To model persistent terms, we now add the usual replication operator of the $\pi$-calculus (first defined in [6]) to our calculus. Essentially a replication $!\,P$ can be seen as "any number of copies of $P$ in parallel". It has one transition rule: $!\,P \xrightarrow{\gamma} P'$ if $P \mid !\,P \xrightarrow{\gamma} P'$.

We only need to change Milner's encoding of the (non-strong) lazy reduction strategy [6] minutely in the case of an abstraction, replacing an input prefix by a delayed input:

$$\begin{aligned}
[\![x]\!]u &= \overline{x}\,u \\
[\![\lambda x\,.\,M]\!]u &= (x\,v)(u\,x v \mid [\![M]\!]v) \\
[\![MN]\!]u &= (v)([\![M]\!]v \mid (x)\overline{v}\,xu\,.\,[\![x := N]\!])
\end{aligned}$$

where $[\![x := N]\!]$ stands for $!\,(w)xw\,.\,[\![N]\!]w$.

We now show that the encoding is accurate, in the sense that there is a strong operational correspondence between reduction in the $\lambda$-calculus term and its encoding into the fusion calculus, and that convergence and divergence properties are preserved by the encoding. Below, we write $\tilde{N}$ for $N_1, \ldots, N_k$ and $[\![\tilde{x} := \tilde{N}]\!]$ for $[\![x_1 := N_1]\!] \mid \cdots \mid [\![x_n := N_n]\!], n \geq 0$. Let $\mathcal{L}$ stand for the family of $\lambda$-calculus terms, and $\mathcal{F}$ for the family of fusion calculus agents.

We define a correspondence between a $\lambda$ term and its encoding by handling substitutions explicitly: a substitution $\{N/x\}$ on a $\lambda$ term is represented by a fusion agent $[\![x := N]\!]$, which acts as an environment binding $x$ to $N$.

**Definition 12** *The relation $\mathcal{S} \subseteq \mathcal{L} \times \mathcal{F}$ contains all pairs $(L, P)$ such that for some $k \geq 0$, $M, N_1, \ldots, N_k \in \mathcal{L}$, the following holds: $\mathrm{fv}(N_i) \cap \{x_1, \ldots, x_i\} = \emptyset$ for $1 \leq i \leq k$, $L \equiv M\{\tilde{N}/\tilde{x}\}$, and $P \equiv (\tilde{x})([\![M]\!]u \mid [\![\tilde{x} := \tilde{N}]\!])$ for any fresh $u$.*

So especially, for $k = 0$, $(M, [\![M]\!]u) \in \mathcal{S}$.

**Theorem 9** *For any $(L, P) \in \mathcal{S}$,*
*1. if $L \rightarrow_\lambda L'$, then $P \xrightarrow{1}{}^{+} P'$ such that $(L', P') \in \mathcal{S}$.*
*2. if $P \xrightarrow{1} P'$, then either $(L, P') \in \mathcal{S}$ or $L \rightarrow_\lambda L'$ such that $(L', P') \in \mathcal{S}$.*

The proof goes by induction on the depth of the deduction of the reduction of $L$ and transition of $P$, respectively, and induction on $k - i$.

The preservation of convergence (and thereby divergence) properties now follows directly from Theorem 9 and the fact that for $(L, P) \in \mathcal{S}$, if $L \not\rightarrow_\lambda$, then $P \xrightarrow{1}{}^{*} P'$ such that $P' \not\xrightarrow{1}$ and $(L, P') \in \mathcal{S}$ (this can be shown by structural induction on $M$, where $L \equiv M\{\tilde{N}/\tilde{x}\})$.

**Theorem 10** *For any* $(L, P) \in \mathcal{S}$*, if* $L \downarrow L'$ *then* $P \downarrow P'$ *with* $(L', P') \in \mathcal{S}$*, and if* $P \downarrow P'$ *then* $L \downarrow L'$ *with* $(L', P') \in \mathcal{S}$.

## References


[1] S. Abramsky. The lazy lambda calculus. In D. Turner, editor, *Research Topics in Functional Programming*. Addison-Wesley, 1989.

[2] G. Boudol. Towards a lambda-calculus for concurrent and communicating systems. In J. Díaz and F. Orejas, editors, *Proceedings of TAPSOFT '89, Volume 1*, volume 351 of *LNCS*, pages 149–161. Springer, 1989.

[3] G. Boudol. The $\pi$-calculus in direct style. In *Proceedings of POPL '97*, pages 228–241. ACM, Jan. 1997.

[4] 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.

[5] R. Milner. The polyadic $\pi$-calculus: a tutorial. Technical Report ECS-LFCS-91-180, LFCS, University of Edinburgh, Oct. 1991. Also in *Logic and Algebra of Specification*, ed. F. L. Bauer, W. Brauer and H. Schwichtenberg, Springer, 1993.

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

[7] R. Milner. Calculi for interaction. *Acta Informatica*, 33(8):707–737, Nov. 1996.

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

[9] J. Niehren and M. Müller. Constraints for free in concurrent computation. In K. Kanchanasut and J.-J. Lévy, editors, *Asian Computer Science Conference*, volume 1023 of *LNCS*, pages 171–186, Pathumthani, Thailand, 11–13 Dec. 1995. Springer.

[10] J. Parrow and D. Sangiorgi. Algebraic theories for name-passing calculi. *Journal of Information and Computation*, 120(2):174–197, 1995.

[11] J. Parrow and B. Victor. The update calculus. In *Proceedings of AMAST'97*, volume 1349 of *LNCS*. Springer, 1997.

[12] J. Parrow and B. Victor. The fusion calculus: Expressiveness and symmetry in mobile processes. Technical Report DoCS 97/96, Dept. of Computer Systems, Uppsala University, Sweden, Dec. 1997. URL: `http://www.docs.uu.se/~victor/tr/fusion.html`.

[13] D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms*. PhD thesis, LFCS, University of Edinburgh, 1993.

[14] D. Sangiorgi. The lazy lambda calculus in a concurrency scenario. *Journal of Information and Computation*, 111(1):120–131, 1994.

[15] D. Sangiorgi. Lazy functions and mobile processes. Rapport de Recherche RR-2515, INRIA Sophia-Antipolis, 1995.

[16] D. Sangiorgi. $\pi$-calculus, internal mobility and agent-passing calculi. *Theoretical Computer Science*, 167(1,2):235–274, 1996.

[17] D. Sangiorgi. A theory of bisimulation for the $\pi$-calculus. *Acta Informatica*, 33:69–97, 1996.

[18] B. Thomsen. *Calculi for Higher Order Communicating Systems*. PhD thesis, Imperial College, University of London, Sept. 1990.

[19] B. Victor. *The Fusion Calculus: Expressiveness and Symmetry in Mobile Processes*. PhD thesis, Dept. of Computer Systems, Uppsala University, Sweden, June 1998. URL: `http://www.docs.uu.se/~victor/thesis.shtml`.

[20] B. Victor and F. Moller. The Mobility Workbench — a tool for the $\pi$-calculus. In D. Dill, editor, *Proceedings of CAV '94*, volume 818 of *LNCS*, pages 428–440. Springer, 1994.

[21] B. Victor and J. Parrow. Concurrent constraints in the fusion calculus. In *Proceedings of ICALP'98*, LNCS. Springer, 1998. URL: `http://www.docs.uu.se/~victor/tr/ccfc.html`.