Symbolic semantics for multiparty interactions in the link-calculus

Linda Brodo[1] and Carlos Olarte[2]

[1] Dipartimento di Scienze Politiche, Scienze della Comunicazione e Ingegneria dell'Informazione, Università di Sassari, Italy
[2] ECT - Universidade Federal do Rio Grande do Norte, Brazil

Abstract. The `link`-calculus is a model for concurrency that extends the point-to-point communication discipline of Milner's CCS with multiparty interactions. Links are used to build chains describing how information flows among the different agents participating in a multiparty interaction. The inherent non-determinism in deciding both, the number of participants in an interaction and how they synchronize, makes it difficult to devise efficient verification techniques for it. In this paper we propose a symbolic semantics and a symbolic bisimulation for the `link`-calculus which are more amenable to automating reasoning. Unlike the operational semantics of the `link`-calculus, the symbolic semantics is finitely branching and it represents, compactly, a possibly infinite number of transitions. We give necessary and sufficient conditions to efficiently check the validity of symbolic configurations. We also implement an interpreter based on this semantics and we show how to use such implementation for verification.

1 Introduction

Distributed systems are evolving in complex ways and adequate modeling languages are needed to specify and verify properties such as resources consuming, security, privacy, among several others. Multiparty interactions are commonplace in this new era of distributed systems. Take for instance an on-line payment service that can involve the shopper, the merchant's website, a cashier service and a bank. In order to have a more comprehensive representation of the system's dynamics, it would be convenient to consider multiparty interactions instead of binary ones. In the literature there are multi-way synchronization calculi [10, 6, 11] that seem to be adequate to be applied in different areas such as distributed computing, web applications and Systems Biology. Here we shall focus on the `link`-calculus [1, 2] to model multiparty communications.

The `link`-calculus is a new multiparty process algebra where the number of participants in each synchronization is not fixed a priori. It extends the binary communication discipline of CCS [9] with *links*, e.g., ${}^{a}\backslash_{b}$, that can be thought of as the forwarding of a message received on channel a (the input channel) to another channel b (the output channel). It could be the case that a link exposes only an output (${}^{\tau}\backslash_{b}$), or an input (${}^{a}\backslash_{\tau}$); these particular actions are the ends of a *link chain*.

A link chain is the mechanism by which $n \geq 2$ entities can synchronize. Each entity must offer a link that have to match with an adjacent link offered by another entity. For instance, if three processes offer, respectively, the links ${}^{a}\backslash_{b}$, ${}^{b}\backslash_{c}$ and ${}^{c}\backslash_{d}$, they can synchronize and produce the link chain ${}^{a}\backslash_{b}^{b}\backslash_{c}^{c}\backslash_{d}$, where information flow from a to d through b and c.

The multiparty synchronization mechanism of the `link`-calculus brings interesting challenges for devising automatic reasoning tools. The main technical problem is that the number of participants in an interaction is not known a priori. Then, the operational semantics must consider all the possible synchronizations among the agents running in parallel. For instance, consider two processes offering, respectively, the links ${}^{a}\backslash_{b}$ and ${}^{b}\backslash_{a}$. They may synchronize and produce the link chain ${}^{a}\backslash_{b}^{b}\backslash_{a}$, but also ${}^{b}\backslash_{a}^{a}\backslash_{b}$. Moreover, they may also produce the chain ${}^{a}\backslash_{b}^{\square}\backslash_{\square}^{b}\backslash_{a}$ where the virtual link ${}^{\square}\backslash_{\square}$ allows another process to participate in the interaction.

We propose a symbolic semantics which is more amenable for reasoning about `link` processes. The semantics collects together all the possible synchronizations that can be composed with a multiset of links (e.g., $\langle {}^{a}\backslash_{b}, {}^{b}\backslash_{a} \rangle$ for the example above). We thus abstract from the order of the links and we represent, compactly, a possibly infinite number of transitions in the operational semantics. Moreover, unlike the operational semantics, the proposed semantics is finitely branching.

The presence of restricted names makes more interesting the definition of symbolic configurations. In fact, internal (multiparty) synchronizations play an important role in the definition of network bisimulation [1, 2]. We give a symbolic representation of transitions involving restricted names and we give efficient procedures to check the validity of such configurations. Furthermore, we define a symbolic bisimulation and we show that it is a congruence and it coincides with network bisimulation.

We describe a prototypical implementation in Maude (http://maude.cs.illinois.edu) of the symbolic semantics, available at http://subsell.logic.at/links. In order to illustrate the semantics and the tool, we consider the classical problem of the dining philosophers. We show that this problem has a simple implementation in the `link`-calculus. Furthermore, we use our tool to show that the model is deadlock free. We then contribute with a theoretical framework, that may help to better understand multiparty interactions, and a tool to enact it.

Contributions and Plan of the paper. Section 2 recalls the theory of the `link`-calculus. In Section 3 we define our symbolic semantics and we give polynomial procedures to check whether a symbolic configuration is valid or not (Propositions 1 and 2) . We show that the symbolic semantics is sound and complete wrt the classical one (Corollary 1). We define a procedure to extract a symbolic configuration from a trace in the operational semantics and we show that the resulting configuration is an upper bound for the symbolic semantics (Theorem 3). In Section 3.3 we define a symbolic bisimulation that coincides with network bisimulation (Theorem 5) and has the property to be a congruence (Corollary 2). In Section 3.4 we present the implementation of simulation and verification techniques for the `link`-calculus based on the symbolic semantics. Section 4 concludes the paper with future and related work. Due to space restrictions, auxiliary results and the detailed proofs are given in the Appendix.

2 Background on `link`-calculus

A *link* is a pair ${}^{\alpha}\backslash_{\beta}$ where $\alpha, \beta \in \mathcal{C} \cup \{\tau, \square\}$. $\mathcal{C}$ denotes the set of channels, ranged over by $a, b, c, ...$; τ is the silent action and $\square$ is a virtual action. Intuitively, ${}^{a}\backslash_{b}$ is a prefix that executes an input on channel a and an output on b. The τ action is used to represent a link where no interaction is required (on the left or on the right) as in ${}^{a}\backslash_{\tau}$. A virtual link ${}^{\square}\backslash_{\square}$ represents a non specified interaction that will be later completed. The link ${}^{\alpha}\backslash_{\beta}$ is *solid* if $\alpha, \beta \neq \square$, and it is virtual if $\alpha, \beta = \square$. A link is *valid* if it is solid or virtual. For instance, ${}^{\square}\backslash_{\square}$, ${}^{a}\backslash_{a}$, ${}^{\tau}\backslash_{a}$, ${}^{b}\backslash_{a}$ are valid links whereas ${}^{\square}\backslash_{a}$, ${}^{\tau}\backslash_{\square}$ are not.

Links can be combined in *link chains* that record the source and the target sites of each hop of the interaction. Formally, a link chain is a non-empty finite sequence $s = \ell_1 ... \ell_n$ of valid links $\ell_i =^{\alpha_i} \backslash_{\beta_i}$ such that:

1. for any $i \in 1..n-1$, $\begin{cases} \beta_i, \alpha_{i+1} \in \mathcal{C} & \text{implies } \beta_i = \alpha_{i+1} \\ \beta_i = \tau & \text{iff } \alpha_{i+1} = \tau \end{cases}$
2. $\exists i \in 1..n.\ \ell_i \neq^{\square} \backslash_{\square}$.

The first condition says that two adjacent solid links must match on their adjacent sites. Moreover, the silent action τ can not be matched by a virtual action $\square$. This last condition is required since, as we shall see, a τ action can be only matched with τ when processes synchronize on restricted channels. The second condition says that a valid link must have at least one solid link. We shall use VC to denote the set of valid chains and we write $|s|$ to denote the *length* of the chain s.

Some examples of valid link chains are: ${}^{\square}\backslash^{a}_{\square}\backslash^{b}_{b}\backslash_{\tau}$, ${}^{a}\backslash^{\square}_{b}\backslash^{c}_{\square}\backslash_{d}$, and ${}^{\tau}\backslash^{a}_{a}\backslash_{\tau}$. The first chain represents an interaction where there is a pending synchronization on the left of ${}^{a}\backslash_{b}$; similarly, the second chain represents an interaction where a third-party process must offer a link joining b and c (i.e., ${}^{b}\backslash_{c}$). Finally, the last chain is the result of a binary interaction between a process performing the output ${}^{\tau}\backslash_{a}$ and a process performing the input ${}^{a}\backslash_{\tau}$. Examples of non valid link chains are: ${}^{a}\backslash^{c}_{b}\backslash_{d}$, ${}^{\square}\backslash^{\tau}_{\square}\backslash_{a}$, and ${}^{a}\backslash^{c}_{\tau}\backslash_{d}$.

Processes in the `link`-calculus are built from the syntax

$$P, Q ::= \mathbf{0} \mid \ell.P \mid P + Q \mid P|Q \mid (\nu\, a)P \mid A$$

where ℓ is a solid link (i.e. $\ell =^{\alpha} \backslash_{\beta}$ with $\alpha, \beta \neq \square$) and A is a process identifier for which we assume a (possibly recursive) definition $A \triangleq P$.

The nil process $\mathbf{0}$ does nothing. The process $\ell.P$ first performs ℓ and then behaves as P. The non-deterministic process $P + Q$ can either behave as P or Q. Parallel composition is denoted as $P \mid Q$. The process $(\nu\, a)P$ behaves as P but it cannot exhibit any output where the name a is not matched. Finally, A behaves as P if $A \triangleq P$.

As usual, $(\nu\, a)P$ binds the occurrences of a in P. The sets of free and of bound names of a process P are defined in the obvious way and denoted, respectively, by $fn(P)$ and $bn(P)$. Processes are taken up to alpha-conversion of bound names. We shall often omit a trailing $\mathbf{0}$, e.g. by writing ${}^{a}\backslash_{b}$ instead of ${}^{a} \backslash_{b}\ .\mathbf{0}$.

Operational Semantics. The operational semantics is given by the labeled transition system $(\mathcal{P}, \mathcal{L}, \longrightarrow)$ where states $\mathcal{P}$ are `link`-processes, labels $\mathcal{L}$ are valid chains (i.e., $\mathcal{L} = VC$) and the transition relation $\longrightarrow$ is the minimal transition relation generated by the rules in Figure 1. In the following we explain the rules.

The presence of virtual links in a link chain suggests that an interaction is not completed and it allows for more processes to synchronize by offering the correct links. A process $\ell.P$ can take part in any interaction where ℓ can be placed in an admissible position of a (larger) chain. Hence, in order to join in a communication, $\ell.P$ should suitably enlarge its link ℓ to a link chain s including ℓ and some virtual links. Formally, Rule *Act* says that $\ell.P \xrightarrow{s} P$ for any link chain s such that $s \bowtie \ell$ where $\bowtie$ is the least equivalence relation on valid link chains closed under the following axioms:

$$\begin{array}{rclcrcl} s^{\square}\backslash_{\square} & \bowtie & s & \qquad & s_1{}^{\square}\backslash^{\square}_{\square}\backslash_{\square}s_2 & \bowtie & s_1^{\square}\backslash_{\square}\,s_2 \\ {}^{\square}\backslash_{\square}\,s & \bowtie & s & \qquad & s_1{}^{\alpha}\backslash^{\square}_{a}\backslash^{a}_{\square}\backslash_{\beta}s_2 & \bowtie & s_1{}^{\alpha}\backslash^{a}_{a}\backslash_{\beta}s_2 \end{array}$$

Note that the link ${}^{\tau}\backslash_{a}$ (resp. ${}^{a}\backslash_{\tau}$) can be only enlarged with virtual links on the right (resp. left). Moreover, if $s \bowtie {}^{\tau}\backslash_{\tau}$ then $s ={}^{\tau}\backslash_{\tau}$.

Rules $Lsum$, $Lpar$ and Ide are standard. If P is able to exhibit a transition to P' with label s, then $P + Q \xrightarrow{s} P'$ (Rule $Lsum$). Similarly for Q with Rule $Rsum$ omitted in Figure 1. If P can exhibit a transition, it can also exhibit the same transition when running in parallel with Q (Rules $Lpar$ and $Rpar$). Finally, A moves to P' if its body definition P can move to P' (Rule Ide).

The synchronization mechanism (Rule Com) works by merging two link chains, say s and s', and requires that $|s| = |s'|$. It also requires that every solid link of s must correspond to a virtual link in s' in the same position, and vice versa. Then we make the two link chains collapse in one link chain where the virtual links have been substitute with the corresponding solid links. More precisely, let α, β be actions. We define

$$\alpha \bullet \beta = \alpha \text{ if } \beta = \square \qquad \alpha \bullet \beta = \beta \text{ if } \alpha = \square \qquad \alpha \bullet \beta = \bot \text{ otherwise}$$

Let $l_1 ={}^{\alpha_1}\backslash_{\beta_1}$ and $l_2 ={}^{\alpha_2}\backslash_{\beta_2}$ be valid links and $\alpha_1 \bullet \alpha_2 = \alpha$, $\beta_1 \bullet \beta_2 = \beta$. If $\alpha, \beta \neq \bot$, then $l_1 \bullet l_2 ={}^{\alpha}\backslash_{\beta}$. Otherwise, $l_1 \bullet l_2 = \bot$. Let $s = \ell_1 ... \ell_n$ and $s' = \ell'_1 ... \ell'_n$ be valid chains with $\ell_i ={}^{\alpha_i}\backslash_{\beta_i}$ and $\ell'_i ={}^{\alpha'_i}\backslash_{\beta'_i}$. If $l_i \bullet l'_i \neq \bot$ for all $i \in 1..n$ and $(l_1 \bullet l'_1)...(l_n \bullet l_n)$ is a valid chain, then $s \bullet s' = (l_1 \bullet l'_1)...(l_n \bullet l_n)$. Otherwise, $s \bullet s' = \bot$.

As an example, the chains ${}^{\square}\backslash^{\square}_{\square}\backslash^{a}_{\square}\backslash_{b}$ and ${}^{c}\backslash^{\square}_{a}\backslash_{\square}$ cannot merge, as they have different length; ${}^{a}\backslash^{\square}_{b}\backslash_{\square}$ and ${}^{\square}\backslash^{c}_{\square}\backslash_{d}$ cannot merge since ${}^{a}\backslash^{c}_{b}\backslash_{d}$ is not a valid chain; a chain s cannot merge with itself; finally, ${}^{c}\backslash^{\square}_{a}\backslash^{b}_{\square}\backslash_{d}$ and ${}^{\square}\backslash^{a}_{\square}\backslash^{\square}_{b}\backslash_{\square}$ merges into ${}^{c}\backslash^{a}_{a}\backslash^{b}_{b}\backslash_{d}$.

We note that, contrary to CCS, the Rule Com can appear several times in the proof tree of a transition since $s \bullet s'$ can still contain virtual links (if s and s' have a virtual link in the same position). Hence, $s \bullet s'$ can possibly be merged with other link chains. However, when $s \bullet s'$ is solid, no further synchronization is possible (since $s \bullet s' = \bot$ whenever s is a chain without virtual links).

As usual in process calculi, names are restricted in order to force an interaction. Let α be an action and $a \in C$. Then,

$$(\nu\, a)\alpha = \begin{cases} \tau \text{ if } \alpha = a \\ \alpha \text{ otherwise} \end{cases} \qquad \text{and} \qquad (\nu\, a)^{\alpha}\backslash_{\beta} ={}^{((\nu\, a)\alpha)}\backslash_{((\nu\, a)\beta)}$$

Let $s = \ell_1 ... \ell_n$, with $\ell_i ={}^{\alpha_i}\backslash_{\beta_i}$ and $i \in 1..n$. We say that a is *matched* in s if:

1. $a \neq \alpha_1, \beta_n$ (i.e., a cannot occur in the extremes of the chain), and
2. for any $i \in 1..n-1$, either $\beta_i = \alpha_{i+1} = a$ or $\beta_i, \alpha_{i+1} \neq a$.

Otherwise, we say that a is *unmatched* (or *pending*) in s. We define,

$$(\nu\, a)s = \begin{cases} ((\nu\, a)\ell_1) \dots ((\nu\, a)\ell_n) & \text{if } a \text{ is } \textit{matched} \text{ in } s \\ \bot & \text{otherwise} \end{cases}$$

$$\frac{s \blacktriangleright\!\!\blacktriangleleft \ell}{\ell.P \xrightarrow{s} P}\ Act \qquad \frac{P \xrightarrow{s} P'}{P + Q \xrightarrow{s} P'}\ Lsum \qquad \frac{P \xrightarrow{s} P'}{P \mid Q \xrightarrow{s} P' \mid Q}\ Lpar \qquad \frac{P \xrightarrow{s} P' \quad A \triangleq P}{A \xrightarrow{s} P'}\ Ide$$

$$\frac{P \xrightarrow{s} P'}{(\nu a)P \xrightarrow{(\nu a)s} (\nu a)P'}\ Res \qquad \frac{P \xrightarrow{s} P' \quad Q \xrightarrow{s'} Q'}{P \mid Q \xrightarrow{s \bullet s'} P' \mid Q'}\ Com$$

Fig. 1. SOS semantic rules. Rules $Rsum$ and $Rpar$ are omitted. All the rules have, as a side condition, that the link chains in the conclusion and premises are valid (i.e., different from $\perp$).

As an example, all the names are matched in the valid link chain ${}^{\tau}\backslash_{\tau}$. Instead, neither a nor b are matched in ${}^{a}\backslash^{a}_{a}\backslash_{b}$. In $s = {}^{\tau}\backslash^{a}_{a}\backslash^{\square}_{b}\backslash_{\square}$, the name a can be restricted and $(\nu a)s =^{\tau} \backslash^{\tau}_{\tau} \backslash^{\square}_{b} \backslash_{\square}$; whereas $(\nu b)s$ is undefined since b is pending in s.

The Rule Res can serve different aims: (i) *floating*, if a does not occur in s, then $(\nu\, a)s = s$ and $(\nu\, a)P \xrightarrow{s} (\nu\, a)P'$; (ii) *hiding*, if a is matched in s, then all occurrences of a in s are replaced with τ in $(\nu\, a)s$; (iii) *blocking*, if a is pending in s (i.e., there is some unmatched occurrence of a in s), then $(\nu\, a)s = \perp$ and the rule cannot be applied.

3 Symbolic Semantics

As mentioned in the introduction, the system ${}^{a}\backslash_{b}\,\mathbf{0} \mid {}^{b}\backslash_{a}\,.\mathbf{0}$ can synchronize in different ways, i.e, we can use the rule Com to observe different link chains such as ${}^{a}\backslash^{b}_{b}\backslash_{a}$, ${}^{b}\backslash^{a}_{a}\backslash_{b}$, ${}^{\square}\backslash^{a}_{\square}\backslash^{\square}_{b}\backslash^{b}_{\square}\backslash_{a}$, etc. In this section we propose a novel symbolic semantics that represents, in a unique configuration, all these link-chains. Hence, the non-determinism of the operational semantics (due to Com and Act) is completely replaced with a deterministic transition collecting all the possible interactions the process may engage. We also give sufficient and necessary conditions for testing the validity on configuration.

3.1 Symbolic Configurations

Definition 1 (Link configurations). *Let L be a multiset of solid links. We define the (symbolic) configuration $\langle L\rangle$ as the set*

$$\langle L\rangle = \{s \in VC \mid \textit{ there exists } s_i \blacktriangleright\!\!\blacktriangleleft l_i \textit{ for all } l_i \in L \textit{ s.t. } s = s_1 \bullet s_2 \bullet \cdots \bullet s_n\}$$

We say that $\langle L\rangle$ is a valid configuration if the set above is not empty.

Intuitively, the configuration $\langle L\rangle$ accumulates the links that can be merged in an application of the rule Com. As an example, the configuration $\langle {}^{a}\backslash_{b}\rangle$ represents, for instance, ${}^{a}\backslash_{b}$ (and the process does not interact any more), ${}^{\square}\backslash^{a}_{\square}\backslash_{b}$ where there are no further interaction on b and a is still pending, ${}^{\square}\backslash^{a}_{\square}\backslash^{\square}_{b}\backslash_{\square}$ where both a and b are pending. The configuration $\langle {}^{a}\backslash_{b}, {}^{b}\backslash_{a}\rangle$ represent, e.g., the following chains: ${}^{a}\backslash^{b}_{b}\backslash_{a}$, ${}^{b}\backslash^{a}_{a}\backslash_{b}$, ${}^{b}\backslash^{\square}_{a}\backslash^{\square}_{\square}\backslash^{a}_{\square}\backslash_{b}$, ${}^{\square}\backslash^{b}_{\square}\backslash^{a}_{a}\backslash^{\square}_{b}\backslash_{\square}$, etc. Finally, the configuration $\langle {}^{\tau}\backslash_{a}, {}^{a}\backslash_{\tau}\rangle$ contains the chains ${}^{\tau}\backslash^{a}_{a}\backslash_{\tau}$, ${}^{\tau}\backslash^{\square}_{a}\backslash^{a}_{\square}\backslash_{\tau}$, ${}^{\tau}\backslash^{\square}_{a}\backslash^{\square}_{\square}\backslash^{a}_{\square}\backslash_{\tau}$, etc. (recall that τ cannot match $\square$).

Next proposition (see the proof in Appendix A.1) allows us to test whether a configuration $\langle L\rangle$ is valid without checking the existence of a chain s s.t. $s \in \langle L\rangle$. This proposition gives an algorithm linear on the number of elements in L.

Proposition 1 (Valid Configurations). *Let L be a non-empty multiset of solid links. Then, $\langle L\rangle$ is valid iff τ appears at most once in L as input and at most once as output.*

In order to define the behavior of the restriction operator in the symbolic semantics, we have to give also a definition of restriction on configurations.

Definition 2 (Hiding). *Let γ be a configuration and $a \in C$. We define the configuration*

$$(\nu a)\gamma = \{s \in VC \mid \textit{ there exists } s' \in \gamma \textit{ and } s = (\nu a)s'\}$$

We say that $(\nu a)\gamma$ is valid if the set above is not empty.

If γ is not valid, by definition, $(\nu a)\gamma$ is not valid. The other direction is not necessarily true. For instance, $L_1 = \langle {}^{a}\backslash_{a}\rangle$ and $L_2 = \langle {}^{\tau}\backslash_{a}, {}^{a}\backslash_{\tau}, {}^{b}\backslash_{c}\rangle$ are valid configurations but neither $(\nu a)\langle L_1\rangle$ nor $(\nu a)\langle L_2\rangle$ are valid. In the first case, observe that $(\nu a)(s)$ is not valid for any $s \bowtie {}^{a}\backslash_{a}$ (since a cannot appear in the extremes and it must be matched). In the second case, if $s \in \langle L_2\rangle$, then s must be of the shape ${}^{\tau}\backslash^{\Box}_{a}\backslash_{\Box} \ldots {}^{\Box}\backslash^{b}_{\Box}\backslash^{\Box}_{c}\backslash_{\Box} \ldots {}^{\Box}\backslash^{a}_{\Box}\backslash_{\tau}$. Since a is not matched, $(\nu a)s = \bot$ and $(\nu a)\langle L_2\rangle$ is empty.

We shall use $\gamma, \gamma', \psi, \psi'$ to denote configurations (with and without restricted names). Given a multiset L of solid links, we shall use $names(L)$ to denote the set of names occurring in the links in L. Let $\gamma = (\nu a_1)...(\nu a_n)\langle L\rangle$. We define the free names of γ as $fn(\gamma) = names(L)\backslash\{a_1, \ldots, a_n\}$ and its bound names as $bn(\gamma) = \{a_1, \ldots, a_n\}$. Given a sequence of distinct names $\boldsymbol{a} = a_1, ..., a_n$, we shall use $(\nu a_1, ..., a_n)\langle L\rangle$ to denote the configuration $(\nu a_1)...(\nu a_n)\langle L\rangle$. If $\boldsymbol{a}$ is empty, then we write $\langle L\rangle$ instead of $(\nu \boldsymbol{a})\langle L\rangle$. Finally, we shall write $\gamma \equiv_s \gamma'$ when $\gamma = \gamma'$ (i.e., $\gamma \subseteq \gamma'$ and $\gamma' \subseteq \gamma$) .

As a direct consequence of the corresponding equivalences on chains [2], we can show that (1) $(\nu a)\gamma \equiv_s \gamma$ if $a \notin fn(\gamma)$; (2) $(\nu a)(\nu b)\gamma \equiv_s (\nu b)(\nu a)\gamma$; (3) $(\nu a)\gamma \equiv_s (\nu b)\gamma[b/a]$ is $b \notin names(\gamma)$ (α-conversion).

Now we give necessary and sufficient conditions for testing if a configuration of the shape $(\nu a)\gamma$ is valid or not (see the proof in Appendix A.1). Such checking can be performed in linear time on the number of links in the configuration γ.

Proposition 2 (Valid Configuration). *Let $\gamma = (\nu \boldsymbol{x})\langle L\rangle$ be a valid configuration and $a \in fn(\gamma)$. $(\nu a)\gamma$ is valid iff the three conditions below hold:*

1. **Matched***: a occurs the same number of times as input and as output in γ.*
2. **Extremes***: there exist two links ${}^{\alpha}\backslash_{\beta}, {}^{\alpha'}\backslash_{\beta'}$ in γ where $\alpha, \beta' \neq a$.*
3. **Synchronizations***: if both ${}^{\tau}\backslash_{a}$ and ${}^{a}\backslash_{\tau}$ occur in L, then either $names(L) = \{a, \tau\}$ or there exist two links ${}^{a}\backslash_{\beta}, {}^{\beta'}\backslash_{a}$ in L s.t. $\beta, \beta' \notin \{a, \tau\}$.*

The following definition shows how to merge two valid configurations. This definition will be useful to define the rule Com in the symbolic semantics.

Definition 3 (Merging). *Let $(\nu a_1, ..., a_n)\langle L\rangle$ and $(\nu b_1, ...b_m)\langle L'\rangle$ be two valid configurations. By alpha conversion, we assume that the names $a_1, ..., a_n$ (resp. $b_1, ..., b_m$) do not occur in L' (resp. L). We define*

$$(\nu a_1, ..., a_n)\langle L\rangle \bullet (\nu b_1, ...b_m)\langle L'\rangle = (\nu a_1, ..., a_n, b_1, ..., b_m)\langle L \uplus L'\rangle$$

where $\uplus$ denotes multiset union.

It is easy to see that $\bullet$ is a commutative and associative operator on configurations.

$$\frac{P \xRightarrow{\gamma} P'}{P + Q \xRightarrow{\gamma} P'}\ Lsum_s \qquad \frac{P \xRightarrow{\gamma} P'}{P \mid Q \xRightarrow{\gamma} P' \mid Q}\ Lpar_s \qquad \frac{P \xRightarrow{\gamma} P' \quad A \triangleq P}{A \xRightarrow{\gamma} P'}\ Ide_s$$

$$\frac{}{\ell.P \xRightarrow{\langle\{\ell\}\rangle} P}\ Act_s \qquad \frac{P \xRightarrow{\gamma} P'}{(\nu a)P \xRightarrow{(\nu a)\gamma} (\nu a)P'}\ Res_s \qquad \frac{P \xRightarrow{\gamma} P' \quad Q \xRightarrow{\gamma'} Q'}{P \mid Q \xRightarrow{\gamma \bullet \gamma'} P' \mid Q'}\ Com_s$$

Fig. 2. Symbolic semantics for the `link`-calculus. All the rules have, as a side condition, that the configurations in the conclusion and premises are valid. Rules $Rpar_s$ and $Rsum_s$ are omitted.

3.2 Semantic Rules

The rules of the symbolic semantics are given in Figure 2 and explained below.

We note that the equivalence relation $\bowtie$ relates two valid link chains when they only differ on the number of virtual links. This relation is central to the definition of configurations. In fact, it is easy to see that if $s \in \gamma$, then $s' \bowtie s$ iff $s' \in \gamma$ (see Lemma 6 in Appendix A.1). Rule Act_s builds a configuration containing only the solid link l. Then, as we shall see, any move of the operational rule Act can be mimicked by Act_s.

Rules $Lsum_s$, $Lpar_s$ and Ide_s are self-explanatory.

Rules Res_s, as expected, makes use of the restriction operator on configurations. From the definition of restriction on configurations, it is easy to see that:

1. if $s \in \gamma$ and $(\nu a)s$ is valid configuration, then, $(\nu a)s \in (\nu a)\gamma$ (see Lemma 7 in Appendix A.1); and
2. if $s \in (\nu a)\gamma$, then, by definition, there exists $s' \in \gamma$ s.t. $s = (\nu a)s'$.

Hence, as we prove below, if γ captures all the (operational) transitions of P, $(\nu a)\gamma$ captures correctly all the transitions of $(\nu a)P$.

Rule Com_s merges the symbolic configurations γ and γ'. Recall that the merge operator simply computes the union (resp. multiset union) of the bounded names (resp. links) in γ and γ'. Unlike the operational rule, Com_s does not need to know in advance the length of the chains to be merged. Instead, it only checks if $\gamma \bullet \gamma'$ is valid (by using the algorithms in Propositions 1 and 2). Moreover, from the definition of the merge operator, we can show that,

1. **Composition**: if $s \in \gamma$, $s' \in \gamma'$ and $s \bullet s'$ is defined then $s \bullet s' \in \gamma \bullet \gamma'$ (see Lemma 8 in Appendix A.1).
2. **Splitting**: if $w \in \gamma \bullet \gamma'$ then there exist s, s' s.t. $w = s \bullet s'$ and $s \in \gamma$ and $s' \in \gamma'$ (see Lemma 9 in the Appendix A.1).

Now we are ready to show the desired adequacy results (proofs in Appendix A.2).

Theorem 1 (Soundness). *Let P be a process and assume that $P \xrightarrow{s} P'$. Then, there exists γ s.t. $P \xRightarrow{\gamma} P'$ and $s \in \gamma$.*

Theorem 2 (Completeness). *Let P be a process and assume that $P \xRightarrow{\gamma} P'$. Then, for all $s \in \gamma$, $P \xrightarrow{s} P'$.*

The above results can be easily extended to sequences of transitions. Given a sequence of symbolic configurations $\Gamma = \gamma_1, ..., \gamma_n$, we say that the sequence of chains $s_1, ..., s_n$ is an instance of Γ if $s_i \in \gamma_i$ for all $i \in 1..n$.

Corollary 1 (Adequacy). *Let P be a process. Then,*

1. *if $P \xrightarrow{s_1} P_1 \xrightarrow{s_2} P_2 \cdots \xrightarrow{s_n} P_n$ then there exists $\gamma_1, ..., \gamma_n$ s.t. $P \xRightarrow{\gamma_1} P_1 \cdots \xRightarrow{\gamma_n} P_n$ and for all $i \in 1..n$, $s_i \in \gamma_i$.*
2. *if $P \xRightarrow{\gamma_1} P_1 \cdots \xRightarrow{\gamma_n} P_n$. Then, for all instance $s_1, ..., s_n$ of $\gamma_1, ..., \gamma_n$, we have $P_1 \xrightarrow{s_1} P_2 \cdots \xrightarrow{s_n} P_n$.*

Extraction and Soundness We can strength Theorem 1 and give an upper bound to γ. If $P \xrightarrow{s} P'$, one may be tempted to think that such upper bound is $\gamma = solid(s)$ where $solid(s)$ denotes the multiset of solid links in s. We note that this does not work under the presence of restriction. For instance, $s = (\nu a)({}^{\tau}\backslash^{a}_{a}\backslash_{\tau}) = {}^{\tau}\backslash^{\tau}_{\tau}\backslash_{\tau}$ if a valid label for a transition $P \xrightarrow{s} P'$ but $\langle {}^{\tau}\backslash_{\tau}, {}^{\tau}\backslash_{\tau} \rangle$ is not a valid configuration.

Next definition shows how to extract a valid configuration from a link chain, that we later show to be a suitable over approximation of the symbolic semantics.

Definition 4 (Extraction). *Let $s = {}^{x_1}\backslash^{x_2}_{x'_1}\backslash_{x'_2} \cdots {}^{x_n}\backslash_{x'_n}$ be a valid chain and $\alpha \in C$ be a name not occurring in s. We define $\mathtt{ext}(s) = (\nu\ \alpha)\langle L \rangle$ where L is the multiset of solid links of s subject to the following substitutions:*
$\forall\ i \in 1 \ldots n-1$, substitute x'_i, x_{i+1} with α if $x'_i = x_{i+1} = \tau$.

For instance, if $s = {}^{a}\backslash^{\tau}_{\tau}\backslash^{c}_{c}\backslash_{d}$ then $\mathtt{ext}(s) = (\nu x)\langle {}^{a}\backslash_{x}, {}^{x}\backslash_{c}, {}^{c}\backslash_{d} \rangle$. We note that if s is a valid chain without occurrences of matched τ's, then $\mathtt{ext}(s) = (\nu a)\langle solid(s) \rangle \equiv_s \langle solid(s) \rangle$. Moreover, if $|s| = 1$, i.e., $s = \ell$ for some solid link ℓ, then $\mathtt{ext}(s) \equiv \langle \{\ell\} \rangle$. Finally, for any valid chain s, $s \in \mathtt{ext}(s)$.

In the following we state some lemmas needed to prove the desired adequacy result in Theorem 3. The proofs are in Appendix A.3.

First, we show that $\mathtt{ext}(s \bullet s')$ approximates $\mathtt{ext}(s) \bullet \mathtt{ext}(s')$. More precisely,

Lemma 1. *Let $s \bullet s'$ be a valid chain. Then $\mathtt{ext}(s) \bullet \mathtt{ext}(s') \subseteq \mathtt{ext}(s \bullet s')$.*

Next, we relate restrictions and the extraction operator.

Lemma 2. *Let $(\nu a)s$ be a valid chain. Then,*

1. *if $\mathtt{ext}(s) = (\nu\beta)\langle L \rangle$ then $\mathtt{ext}((\nu a)s) \equiv_s (\nu\beta)\langle L[\beta/a] \rangle$; and*
2. *$(\nu a)\mathtt{ext}(s) \subseteq \mathtt{ext}((\nu a)s)$.*

Theorem 3 (Soundness). *Let P be a process and assume that $P \xrightarrow{s} P'$. Then, there exists $\gamma \subseteq \mathtt{ext}(s)$ s.t. $P \xRightarrow{\gamma} P'$.*

We note that $\mathtt{ext}(s)$ over approximates the output of the symbolic semantics since $\mathtt{ext}(s)$ identifies τ actions that may come from different synchronizations. For instance, consider the operational transition $(\nu a)({}^{b}\backslash_{a}|{}^{a}\backslash_{b}) \mid (\nu c)({}^{d}\backslash_{c}|{}^{c}\backslash_{d}) \xrightarrow{s \bullet s'} \mathbf{0}$ where

$$s = {}^{\square}\backslash^{\square}_{\square}\backslash^{\square}_{\square}\backslash^{b}_{\square}\backslash^{\tau}_{\tau}\backslash_{b} \qquad s' = {}^{d}\backslash^{\tau}_{\tau}\backslash^{\square}_{d}\backslash^{\square}_{\square}\backslash^{\square}_{\square}\backslash_{\square} \qquad w = s \bullet s' = {}^{d}\backslash^{\tau}_{\tau}\backslash^{\square}_{d}\backslash^{b}_{\square}\backslash^{\tau}_{\tau}\backslash_{b}$$

In the symbolic semantics we have $(\nu a)({}^{b}\backslash_{a}|{}^{a}\backslash_{b}) \mid (\nu c)({}^{d}\backslash_{c}|{}^{c}\backslash_{d}) \xRightarrow{\gamma \bullet \gamma'} \mathbf{0}$ where

$$\gamma = (\nu a)\langle {}^{b}\backslash_{a}, {}^{a}\backslash_{b} \rangle \qquad \gamma' = (\nu c)\langle {}^{d}\backslash_{c}, {}^{c}\backslash_{d} \rangle \qquad \psi = \gamma \bullet \gamma' = (\nu\ a, c)\langle {}^{b}\backslash_{a}, {}^{a}\backslash_{b}, {}^{d}\backslash_{c}, {}^{c}\backslash_{d} \rangle$$

Note that $\mathtt{ext}(w) = (\nu x)\langle {}^{b}\backslash_{x}, {}^{x}\backslash_{b}, {}^{d}\backslash_{x}, {}^{x}\backslash_{d} \rangle$ and $w' = {}^{b}\backslash^{\tau}_{\tau}\backslash^{d}_{d}\backslash^{\tau}_{\tau}\backslash_{b} \in \mathtt{ext}(w)$. Note also that w' is not part of the operational semantics and $w' \notin \psi$.

3.3 Symbolic Bisimulation

In this section we show that network bisimulation, [1, 2] coincides with the symbolic bisimulation as defined below in Definition 7. Let us recall some definitions from [1].

Let $\bowtie$ be the least equivalence relation over VC closed under the inference rules:

$$\frac{s \blacktriangleright\!\!\blacktriangleleft s'}{s \bowtie s'} \qquad\qquad s_1{}^{\alpha}\backslash^{\tau}_{\tau}\backslash_{\beta} s_2 \bowtie s_1^{\alpha} \backslash_{\beta} s_2$$

The relation $\bowtie$ allows us to enlarge/contract chains by adding/removing matched τ actions (similar to $\blacktriangleright\!\!\blacktriangleleft$ for virtual actions). This means that $\bowtie$ abstracts away also from internal (restricted) communications. A link chain is *essential* if it is composed by alternating solid and virtual links, and has solid links at its extremes. It is immediate to check that, by orienting the axioms of $\blacktriangleright\!\!\blacktriangleleft$ and $\bowtie$ from left to right, we have a procedure to transform any link chain s to a unique essential link chain s' such that $s \bowtie s'$. We write $\mathsf{e}(s)$ to denote such unique representative.

Lemma 3 ([1]). *For any link chains s, s' we have $s \bowtie s'$ iff $\mathsf{e}(s) = \mathsf{e}(s')$.*

Definition 5. *A* network bisimulation *[1] $\mathbf{R}$ is a binary relation over* `link` *processes such that, if $P\ \mathbf{R}\ Q$ then:*

- *if $P \xrightarrow{s} P'$, then $\exists\ s'$, Q' such that $\mathsf{e}(s) = \mathsf{e}(s')$, $Q \xrightarrow{s'} Q'$, and $P'\ \mathbf{R}\ Q'$;*
- *if $Q \xrightarrow{s} Q'$, then $\exists\ s'$, P' such that $\mathsf{e}(s) = \mathsf{e}(s')$, $P \xrightarrow{s'} P'$, and $P'\ \mathbf{R}\ Q'$.*

We let $\sim_n$ denote the largest network bisimulation and we say that P is *network bisimilar* to Q if $P \sim_n Q$.

Theorem 4 (Congruence [1]). *Network bisimilarity is a congruence.*

Symbolic Bisimulation. Let $s = {}^{a}\backslash^{\tau}_{\tau}\backslash_{a}$ and $s' = {}^{a}\backslash_{a}$. We know that $s \bowtie s'$. However, there is no a symbolic configuration γ such that $s \in \gamma$ and also $s' \in \gamma$. On the other side, let $\gamma = \langle {}^{a}\backslash_{a} \rangle$ and $\gamma' = (\nu b)\langle {}^{a}\backslash_{b}, {}^{b}\backslash_{a} \rangle$. We know that $\gamma \not\equiv_s \gamma'$ but, if $w \in \gamma$ and $w' \in \gamma'$, it must be the case that $w \bowtie w'$.

Next definition introduces the relation $\bowtie$ on configurations.

Definition 6. *Let $\bowtie$ be the least symmetric relation on valid configurations s.t. $\gamma \bowtie \gamma'$ iff for all $s \in \gamma$ there exists $s' \in \gamma'$ s.t. $s' \bowtie s$.*

Note that $\gamma \equiv_s \gamma'$ implies, of course, that $\gamma \bowtie \gamma'$. Moreover, it is easy to see that $\bowtie$ is an equivalence relation (see Lemma 14 in Appendix A.4).

Intuitively, if $\gamma \bowtie \gamma'$, then from γ we can build the same chains as in γ' but adding/removing τ synchronizations. For instance, let $\gamma = (\nu x)\langle {}^{a}\backslash_{x}, {}^{x}\backslash_{b} \rangle$ and $\gamma' = \langle {}^{a}\backslash_{b} \rangle$. If $s \in \gamma$ (resp. $s' \in \gamma'$) then s must be of the shape $\ldots{}^{\Box}\backslash^{a}_{\Box}\backslash^{\tau}_{\tau}\backslash^{\Box}_{b}\backslash_{\Box}\ldots$ (resp. s' must be of the shape $\ldots{}^{\Box}\backslash^{a}_{\Box}\backslash^{\Box}_{b}\backslash_{\Box}\ldots$). Hence, $\gamma \bowtie \gamma$.

Definition 7 (Symbolic Bisimulation). *A symbolic network bisimulation $\mathbf{R}$ is a binary relation over* `link` *processes such that, if $P\mathbf{R}Q$ then:*

- *If $P \xRightarrow{\ \gamma\ } P'$, then, there exists $\gamma' \bowtie \gamma$ s.t. $Q \xRightarrow{\ \gamma'\ } Q'$ and $P'\mathbf{R}Q'$.*

– *If $Q \xRightarrow{\gamma} Q'$, then, there exists $\gamma' \bowtie \gamma$ s.t. $P \xRightarrow{\gamma'} P'$ and $Q'\mathbf{R}P'$.*

We let $\sim_s$ be the largest symbolic network bisimulation and we say that P and Q are bisimilar if $P \sim_s Q$.

Testing whether $\gamma \bowtie \gamma'$, according to Definition 6, requires to check for every sequence $s \in \gamma$ the existence of $s' \in \gamma'$ s.t. $s' \bowtie s$ and vice versa. It turns out that there is a more efficient procedure to decide $\gamma \bowtie \gamma'$ using the next definition and lemma.

Definition 8 (Capabilities). *Let $\gamma = (\nu \boldsymbol{x})\langle L \rangle$ be a valid configuration. Let $a, b \notin \boldsymbol{x}$. We say that $[a{\cdot}b]$ is a capability of γ, notation $[a{\cdot}b] \in \gamma$, if ${}^{a}\backslash_{b} \in L$ or, it is possible to use the links in L to form a chain of the shape ${}^{a}\backslash^{x_1}_{x_1}\backslash_{x_2} \cdots {}^{x_{n-1}}\backslash^{x_n}_{x_n}\backslash_{b}$ where $x_1, ..., x_n \in \boldsymbol{x}$. We shall use $\mathtt{cap}(\gamma)$ to denote the multiset of capabilities in γ.*

Lemma 4. *Let $s \in \gamma$. For all solid link ${}^{a}\backslash_{b}$, ${}^{a}\backslash_{b} \in \mathsf{e}(s)$ iff $[a \cdot b] \in \mathtt{cap}(\gamma)$ (Lemma 15 in Appendix A.4). Moreover, let γ, γ' be valid configurations. Then, $\gamma \bowtie \gamma'$ iff $\mathtt{cap}(\gamma) = \mathtt{cap}(\gamma')$ (Lemma 16 in Appendix A.4).*

Therefore, checking $\gamma \bowtie \gamma'$ can be done in polynomial time by extracting and comparing the capabilities of the configurations (see Algorithm 1 in Appendix A.5).

Next theorem (see the proof in Appendix A.4) shows that network and symbolic bisimulations coincides. Moreover, since network bisimulation is a congruence [1], so the symbolic bisimulation.

Theorem 5. *Let P and Q be processes. Then, $P \sim_n Q$ iff $P \sim_s Q$.*

Corollary 2. *$\sim_s$ is a congruence.*

3.4 Implementation

As we saw in the previous sections, the symbolic semantics allows for simple mechanisms to generate traces and check whether a configuration is valid or not. Moreover, it is finitely branching unlike the operational semantics. We have implemented the symbolic semantics in Maude (http://maude.cs.illinois.edu) and it is available at http://subsell.logic.at/links. In this section, relaying on the multiparty synchronization mechanism of the `link`-calculus, we model the classical problem of dining philosophers. We show how the semantics, and our tool, allow for the verification of such system.

The dining philosophers is a classical example introduced to study interactions between independent and distributed entities that want to share resources. The problem relates n philosopher sitting around a table, where each one has its own dish, and they can only eat or think. When they, independently, decide to eat, they need two forks. On the table, there is only one fork between two dishes, i.e. exactly n forks.

A solution to this problem in a binary synchronization calculus such as CCS leads to a deadlock exactly when all the philosophers take the fork at their left at the same time [8]. Hence, the system reaches a state where no further transition is possible. The multiparty synchronization mechanism of the `link`-calculus allows us to overcome this

problem. The idea is that, atomically, the philosopher willing to eat has to synchronize with both, the fork on his right and the one on his left. Then he can eat. The `link`-calculus model is:
$(\nu\, dw_0, \ldots, dw_{n-1}, up_0, \ldots, up_{n-1})(Phil_0 \mid \cdots \mid Phil_{n-1} \mid Fork_0 \mid \cdots \mid Fork_{n-1})$ where processes $Phil_i$ and $Fork_i$ are defined as:

$$\begin{array}{ll} Phil_i & \triangleq {}^{\tau}\backslash_{think_i}.Phil_i + {}^{up_i}\backslash_{up_{(i+1)\mathbf{mod}n}}.PhilEat_i \\ PhilEat_i & \triangleq {}^{\tau}\backslash_{eat_i}.{}^{dw_i}\backslash_{dw_{(i+1)\mathbf{mod}n}}.Phil_i \\ Fork_i & \triangleq {}^{\tau}\backslash_{up_i}.{}^{\tau}\backslash_{dw_i}.Fork_i + {}^{up_i}\backslash_{\tau}.{}^{dw_i}\backslash_{\tau}.Fork_i \end{array}$$

Let us show a trace generated with our tool for the system with $n = 2$ philosophers:

```
(tau \ 'tk_1) --> (tau \ 'tk_0) --> ('up_0 \ 'up_1 ; 'up_1 \ tau ; tau \ 'up_0)  -->
(tau \ 'eat_0) --> (tau \ 'tk_1) --> ('dw_0 \ 'dw_1 ; 'dw_1 \ tau ; tau \ 'dw_0) -->
('up_0 \ tau ; 'up_1 \ 'up_0 ; tau \ 'up_1) --> (tau \ 'eat_1) --> (tau \ 'tk_0) -->
('dw_0 \ tau ; 'dw_1 \ 'dw_0 ; tau \ 'dw_1)
```

In the first line, $Phil_1$ thinks and then $Phil_0$ thinks. Later, $Phil_0$ grabs the two forks, as shown in the last configuration of the first line. Such output represents the symbolic configuration $(\nu up_0, up_1)\langle L\rangle$ where $L = \{{}^{up_0}\backslash_{up_1}, {}^{up_1}\backslash_{\tau}, {}^{\tau}\backslash_{up_0}\}$. This configuration is a three-party interaction involving $Phil_0$ and the two forks. Note that the chain $(\nu\, up_0, up_1)^{\tau}\backslash^{up_0}_{up_0}\backslash^{up_1}_{up_1}\backslash_{\tau} = {}^{\tau}\backslash^{\tau}_{\tau}\backslash^{\tau}_{\tau}\backslash_{\tau}$ is the only chain that belongs to the configuration (due to the restriction on up_i). Hence, in one transition, we observe the atomic action of grabbing the two forks. In the second line, we observe $Phil_0$ eating, then $Phil_1$ thinking again and, in the end of the line, $Phil_0$ releases the two forks with a multiparty synchronization. The third and forth lines represent the transitions where $Phil_1$ grabs the forks, eat and then releases the forks.

Our tool can also compute the label transition system with all the reachable states that, in the case of the dinning philosophers, is finite (note that this is not always the case since the `link`-calculus is a conservative extension of CCS where Turing Machines can be encoded [4]). The output of the tool and the resulting graph can be found at the tool's site. The transition system is deadlock-free, i.e., all the states have at least one transition. Moreover, using the search procedures in Maude, we can verify that the system cannot reach a configuration containing both ${}^{\tau}\backslash_{eat_0}$ and ${}^{\tau}\backslash_{eat_1}$.

4 Concluding Remarks

We proposed a symbolic semantics and bisimulation for an open and multiparty interaction process calculus. We gave efficient procedures to check whether a symbolic configuration is valid or not and proved adequate our semantics wrt the operational semantics. We implemented also a tool based on this semantics to simulate and verify systems modeled in the calculus. We are currently implementing a procedure to check (symbolic) bisimulation in the `link`-calculus. We are also planning to use the extraction procedure (`ext`(s)), that over approximates the semantics, as basis for abstract debugging and analysis of `link`-calculus specifications.
Related Work. Multiparty calculi with different synchronization mechanisms have been proposed, e.g., in CSP [7], PEPA [6], full Lotos [3]. These calculi offer parallel operators that exhibit a set of action names (or channel names), and all the parallel processes

offering that action (or an input/output action along that channel) can synchronize by executing it. In [11], a binary form of input allows for a three-way communication. MultiCCS [4] is equipped with a new form of prefix to execute atomic sequences of actions and the resulting parallel operator allows for multi-synchronizations. The multiparty calculus most related to the `link`-calculus is in [10], where links are named and are distinct from usual input/output actions: there is one sender and one receiver (the output includes the final receiver name).

Symbolic semantics in processes calculi are used to represent compactly the possibly infinitely many transitions a process may exhibit. For instance, [5] proposes a symbolic semantics for the π-calculus to avoid the problem of considering the possibly infinite number of values a process can send/receive along a channel. We are currently considering such techniques to give a symbolic semantics for the `link`-calculus with value-passing [1]. The only symbolic semantics for a multiparty calculus we are aware of is [3, 12] where the authors present the definition of a symbolic semantics for the full Lotos language and its implementation.

References

1. Chiara Bodei, Linda Brodo, and Roberto Bruni. Open multiparty interaction. In Narciso Martí-Oliet and Miguel Palomino, editors, *WADT 2012, Revised Selected Papers*, volume 7841 of *LNCS*, pages 1–23. Springer, 2012.
2. Chiara Bodei, Linda Brodo, Roberto Bruni, and Davide Chiarugi. A flat process calculus for nested membrane interactions. *Sci. Ann. Comp. Sci.*, 24(1):91–136, 2014.
3. Muffy Calder and Carron Shankland. A symbolic semantics and bisimulation for full LOTOS. In Myungchul Kim, Byoungmoon Chin, Sungwon Kang, and Danhyung Lee, editors, *FORTE 2001*, volume 197 of *IFIP Conference Proceedings*, pages 185–200. Kluwer, 2001.
4. Roberto Gorrieri and Cristian Versari. *Introduction to Concurrency Theory - Transition Systems and CCS*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2015.
5. Matthew Hennessy and Huimin Lin. Symbolic bisimulations. *Theor. Comput. Sci.*, 138(2):353–389, 1995.
6. Jane Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.
7. C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, Inc., 1985.
8. Daniel J. Lehmann and Michael O. Rabin. On the advantages of free choice: A symmetric and fully distributed solution to the dining philosophers problem. In John White, Richard J. Lipton, and Patricia C. Goldberg, editors, *POPL*, pages 133–138. ACM Press, 1981.
9. Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980.
10. Ugo Montanari and Matteo Sammartino. Network conscious pi-calculus: a concurrent semantics. In *Proc. of Mathematical Foundations of Programming Semantics (MFPS 2012)*, Electronic Notes in Theoretical Computer Science 286, pages 291–306. Elsevier, 2012.
11. Uwe Nestmann. On the expressive power of joint input. *Electronic Notes in Theoretical Computer Science*, 16(2), 1998.
12. Alberto Verdejo. Building tools for LOTOS symbolic semantics in maude. In Doron A. Peled and Moshe Y. Vardi, editors, *FORTE 2002*, volume 2529 of *LNCS*, pages 292–307. Springer, 2002.

A Auxiliary Results and Proofs

A.1 Symbolic Semantics

Proof of Proposition 1: Let L be a non-empty multiset of solid links. Then, $\langle L\rangle$ is valid iff τ appears at most once in L as input and at most once as output.

Proof. ($\Rightarrow$). Assume that there exists a valid chain $s = s_1 \bullet s_2 \bullet \cdots \bullet s_n \in \gamma$. This is possible only if τ appears only in the extremes (for that note that ${}^{\square}\backslash^{\tau}_{\square}\backslash_{\alpha}$ and ${}^{\alpha}\backslash^{\square}_{\tau}\backslash_{\square}$ are both not valid). Hence, it most be the case that τ occurs at most once as input and at most once as output in L. ($\Rightarrow$). It suffices to leave the τ actions (if any) to the extremes (as input in s_1 and as output in s_n) and complete the chains with virtual links such that all s_i are of the same length and they can be merged into a valid chain.

Proof of Proposition 2: Let $\gamma = (\nu \boldsymbol{x})\langle L\rangle$ be a valid configuration and $a \in C$. $(\nu a)\gamma$ is valid iff the three conditions below hold:

1. **Matched**: a occurs the same number of times as input and as output in γ.
2. **Extremes**: there exist two links ${}^{\alpha}\backslash_{\beta}, {}^{\alpha'}\backslash_{\beta'}$ in γ where $\alpha, \beta' \neq a$.
3. **Synchronizations**: if both ${}^{\tau}\backslash_{a}$ and ${}^{a}\backslash_{\tau}$ occur in L, then either $names(L) = \{a, \tau\}$ or there exist two links ${}^{a}\backslash_{\beta}, {}^{\beta'}\backslash_{a}$ in L s.t. $\beta, \beta' \notin \{a, \tau\}$.

Proof. ($\Rightarrow$) Assume that there exists $s \in (\nu a)(\nu \boldsymbol{x})\langle L\rangle$ and $s = (\nu a)(\nu \boldsymbol{x})s'$ for some $s' \in \gamma = (\nu \boldsymbol{x})\langle L\rangle$. By definition of (νa) (in chains), we know that a does not occur in the extremes of s' (and then (2) holds) and all the occurrences of a are matched in s' (and then (1) holds). Now assume that both ${}^{\tau}\backslash_{a}$ and ${}^{a}\backslash_{\tau}$ occur in γ. By definition, we know that there exists $s_i \bowtie l_i$ for all $l_i \in L$ and $s' = s_1 \bullet s_2 \bullet \cdots \bullet s_n$. Since $s = (\nu a)(\nu \boldsymbol{x})s'$ and s is valid, we also know that $s_1 = {}^{\tau}\backslash^{\square}_{a}\backslash_{\square} \cdots {}^{\square}\backslash_{\square}$ and $s_n = {}^{\square}\backslash^{\square}_{\square}\backslash_{\square} \cdots {}^{a}\backslash_{\tau}$. Hence, either all the other links are ${}^{a}\backslash_{a}$ and we obtain

$$s = (\nu a)(\nu \boldsymbol{x})^{\tau}\backslash^{a}_{a}\backslash_{a} \ldots {}^{a}\backslash^{a}_{a}\backslash_{\tau} = {}^{\tau}\backslash_{\tau} \cdots {}^{\tau}\backslash_{\tau}$$

or, there must be $\beta, \beta' \neq a$ that allow us to complete the chain:

$$s = (\nu a)(\nu \boldsymbol{x})^{\tau}\backslash^{a}_{a}\backslash_{a} \cdots {}^{a}\backslash^{\square}_{\beta}\backslash_{\square} \cdots {}^{\square}\backslash^{\beta'}_{\square}\backslash^{a}_{a}\backslash_{a} \cdots {}^{a}\backslash_{\tau} = {}^{\tau}\backslash_{\tau} \cdots {}^{\tau}\backslash^{\square}_{\beta}\backslash_{\square} \cdots {}^{\square}\backslash^{\beta'}_{\square}\backslash_{\tau} \cdots {}^{\tau}\backslash_{\tau} \cdots {}^{\tau}\backslash_{\tau}$$

We then conclude that (3) holds.

($\Leftarrow$) If the three conditions hold, by using a similar reasoning as above, we can always build a $s \in (\nu a)\gamma$.

Lemma 5 (Congruences). *Let γ be a valid configuration. Then, (1) $(\nu a)\gamma \equiv_s \gamma$ if $a \notin fn(\gamma)$; (2) $(\nu a)(\nu b)\gamma \equiv_s (\nu b)(\nu a)\gamma$; (3) $(\nu a)\gamma \equiv_s (\nu b)\gamma[b/a]$ is $b \notin names(\gamma)$, up to α-conversion.*

Proof. The proof is immediate from the corresponding equivalences on link chains.

Lemma 6. *Let s be a valid chain and γ a configuration s.t. $s \in \gamma$. Then, $s' \bowtie s$ iff $s' \in \gamma$.*

Proof. Immediate from the definition of configuration.

Lemma 7. *Let $s \in \gamma$ and assume that $(\nu a)s$ is valid. Then $(\nu a)s \in (\nu a)\gamma$*

Proof. Straightforward from definition of $(\nu a)\langle L\rangle$.

Lemma 8. *Assume that $s \in \gamma$ and $s' \in \gamma'$. If $s \bullet s'$ is defined then $s \bullet s' \in \gamma \bullet \gamma'$.*

Proof. Straightforward from definitions of configurations and $\bullet$.

Lemma 9. *Let w be a valid chain s.t. $w \in \gamma \bullet \gamma'$. Then, there exist s, s' s.t. $w = s \bullet s'$ and $s \in \gamma$ and $s' \in \gamma'$.*

Proof. Straightforward from definitions of configurations and $\bullet$.

A.2 Adequacy Results

Theorem 6 (Soundness). *Let P be a process and assume that $P \xrightarrow{s} P'$. Then, there exists γ s.t. $P \xRightarrow{\gamma} P'$ and $s \in \gamma$.*

Proof. We proceed by induction on the (height of) derivation $P \xrightarrow{s} P'$:

- Case Act. By Rule Act we know that $s \bowtie l$ and, by Rule Act_s, we also know that $l.P \xRightarrow{\langle\{l\}\rangle} P'$. We conclude by noticing that $s \in \langle\{l\}\rangle$.
- The cases Sum, Par and Ide are easy consequences of induction (since we have shorter derivations on the premises and the rules do not modify the label s).
- Case Com. We know that $P \mid Q \xrightarrow{s \bullet s'} P' \mid Q'$ and $P \xrightarrow{s} P'$ and $Q \xrightarrow{s'} Q'$. By induction we know that there exists γ, γ' such that $P \xRightarrow{\gamma} P'$, $Q \xRightarrow{\gamma'} Q'$ and $s \in \gamma$ and $s' \in \gamma'$. By Lemma 8 we know that $s \bullet s' \in \gamma \bullet \gamma'$ and then, $\gamma \bullet \gamma'$ is valid. We conclude by using Com_s to show that $P \mid Q \xRightarrow{\gamma \bullet \gamma'} P' \mid Q'$ as needed.
- Case Res. Let $P = (\nu a)Q$. We know that $P \xrightarrow{(\nu a)s} Q'$ and $Q \xrightarrow{s} Q'$. By induction we know that there exists γ s.t. $Q \xRightarrow{\gamma} Q'$ and $s \in \gamma$. By Lemma 7 we know that $(\nu a)s \in (\nu a)\gamma$ and hence, $(\nu a)\gamma$ is a valid configuration. By using Res_s, we conclude $P \xRightarrow{(\nu a)\gamma} (\nu a)Q'$ as needed.

Theorem 7 (Completeness). *Let P be a process and assume that $P \xRightarrow{\gamma} P'$. Then, for all $s \in \gamma$, $P \xrightarrow{s} P'$.*

Proof. We proceed by induction on the (height of) derivation $P \xRightarrow{\gamma} P'$.

- Case Act_s. It is easy to show that $s \in \gamma = \langle\{l\}\rangle$ iff $s \bowtie l$. Then, for any $s \in \gamma$, $P \xrightarrow{s} P'$.
- The cases Sum_s, Par_s and Ide_s are easy consequences of induction.
- Case Com_s. We know that $P \mid Q \xRightarrow{\gamma \bullet \gamma'} P' \mid Q'$ and $P \xRightarrow{\gamma} P'$ and $Q \xRightarrow{\gamma'} Q'$. Let $w \in \gamma \bullet \gamma'$. By Lemma 9, there exist $s \in \gamma$ and $s' \in \gamma'$ s.t. $w = s \bullet s'$. By induction we know that $P \xrightarrow{s} P'$ and $Q \xrightarrow{s'} Q'$. We conclude by using the rule Com to show that $P \mid Q \xrightarrow{s \bullet s'} P' \mid Q'$ as wanted.

– Case Res_s. Let $P = (\nu a)Q$. We know that $P \xRightarrow{(\nu a)\gamma} (\nu a)Q'$ and $Q \xRightarrow{\gamma} Q'$. Let $s \in (\nu a)\gamma$. By definition, there exists $s' \in \gamma$ s.t. $s = (\nu a)s'$. By induction we know that $Q \xrightarrow{s'} Q'$. We conclude by using the rule Res to show that $P \xrightarrow{s} (\nu a)Q'$.

Corollary 3 (Soundness). *Let P be a process and assume that $P \xrightarrow{s_1} P_1 \xrightarrow{s_2} P_2 \cdots \xrightarrow{s_n} P_n$. Then, there exists $\gamma_1, ..., \gamma_n$ s.t. $P \xRightarrow{\gamma_1} P_1 \cdots \xRightarrow{\gamma_n} P_n$ and for all $i \in 1..n$, $s_i \in \gamma_i$.*

Proof. By induction on the length of the derivation (n) and using Theorem 1.

Definition 9 (Instance). *Given a sequence of symbolic configurations $\Gamma = \gamma_1, ..., \gamma_n$, we say that the sequence of chains $s_1, ..., s_n$ is an instance of Γ if $s_i \in \gamma_i$ for all $i \in 1..n$.*

Corollary 4 (Completeness). *Let P be a process and assume that $P \xRightarrow{\gamma_1} P_1 \cdots \xRightarrow{\gamma_n} P_n$. Then, for all instance $s_1, ..., s_n$ of $\gamma_1, ..., \gamma_n$, we have $P_1 \xrightarrow{s_1} P_2 \cdots \xrightarrow{s_n} P_n$.*

Proof. By induction on the length of the derivation (n) and using Theorem 2.

A.3 Extraction

Lemma 10. *Let $s \bullet s'$ be a valid chain. Then $\texttt{ext}(s) \bullet \texttt{ext}(s') \subseteq \texttt{ext}(s \bullet s')$.*

Proof. Let $w \in \texttt{ext}(s) \bullet \texttt{ext}(s')$. By Lemma 9, we know that there exist w_1, w_2 s.t. $w_1 \in \texttt{ext}(s)$ and $w_2 \in \texttt{ext}(s')$. Let $\texttt{ext}(s) = (\nu\alpha)\langle L\rangle$ and $\texttt{ext}(s') = (\nu\alpha')\langle L'\rangle$. Note that α (resp. α') may be "empty" if s (resp. s') does not have occurrences of matched τ's. Let w_1' (resp. w_2') be as w_1 (resp. w_2) but replacing the matched τ's (if any) by α (resp. α'). It is easy to see that $w_1' \in \langle L\rangle$ and $w_2' \in \langle L'\rangle$ and also that $w_1' \bullet w_2'$ is valid. By Lemma 8, $w_1' \bullet w_2' \in \langle L_1\rangle \bullet \langle L_2\rangle$. Let $\texttt{ext}(s \bullet s') = (\nu\beta)\langle M\rangle$. Note that it must be the case that $M = L_1 \uplus L_2$ (since $s \bullet s'$ cannot add new matched τ's wrt s and s'). Let $w_\beta = (w_1' \bullet w_2')[\beta/\alpha][\beta/\alpha']$. We conclude by noticing that $w = (\nu\beta)w_\beta \in (\nu\beta)\langle M\rangle$.

Lemma 11. *Let $(\nu a)s$ be a valid chain and let $\texttt{ext}(s) = (\nu\beta)\langle L\rangle$. Then, $\texttt{ext}((\nu a)s) \equiv (\nu\beta)\langle L[\beta/a]\rangle$.*

Proof. Since $(\nu a)s$ is defined, we know that all a's in s are matched. Moreover, $(\nu a)s$ adds to s some extra τ synchronization (those caused by matched a's). Then, $\texttt{ext}((\nu a)s)$ must also abstract away (using the fresh name β) all the matched a.

Lemma 12. *Let $(\nu a)s$ be a valid chain. Then $(\nu a)\texttt{ext}(s) \subseteq \texttt{ext}((\nu a)s)$.*

Proof. Let $\texttt{ext}(s)$ be of the shape $(\nu b)\langle L\rangle$ and $w \in (\nu a)\texttt{ext}(s)$. Then, there exists $w' \in \langle L\rangle$ s.t. $w = (\nu a)(\nu b)w'$ (where all the a's and b's in w' are matched and they do not occur in the extremes). Let $\texttt{ext}((\nu a)s) = (\nu\beta)\langle L'\rangle$ and $w_\beta = w'[\beta/a][\beta/b]$. It is easy to see that $w = (\nu\beta)w_\beta$. Using Lemma 11, we know that $L' = L[\beta/a]$ and then, $w_\beta \in \langle L'\rangle$ as needed.

Lemma 13 (Ordering preservation). *Let $\gamma, \gamma', \psi, \psi'$ be configurations. Then,*

1. *If $\gamma \subseteq \gamma'$ then $(\nu a)\gamma \subseteq (\nu a)\gamma'$.*
2. *If $\gamma \subseteq \gamma'$ and $\psi \subseteq \psi'$ then $\gamma \bullet \psi \subseteq \gamma' \bullet \psi'$.*

Proof. (1) Let $s \in (\nu a)\gamma$. We know that there exists $s' \in \gamma$ s.t. $s = (\nu a)s'$. Hence, $s' \in \gamma'$ and then $s \in (\nu a)\gamma'$.

(2) If $w \in \gamma \bullet \psi$, by Lemma 9, there exists $s \in \gamma$ and $s' \in \psi$ s.t $w = s \bullet s'$. Since $s \in \gamma'$ and $s' \in \psi'$, we use Lemma 8 to conclude that $s \bullet s' \in \gamma' \bullet \psi'$.

Theorem 8 (Soundness). *Let P be a process and assume that $P \xrightarrow{s} P'$. Then, there exists $\gamma \subseteq \mathtt{ext}(s)$ s.t. $P \overset{\gamma}{\Longrightarrow} P'$.*

Proof. We proceed by induction on the (height of) derivation $P \xrightarrow{s} P'$:

- Case Act. This case is easy by noticing that for any $s \blacktriangleright\!\blacktriangleleft l$, $\mathtt{ext}(s) = \langle\{l\}\rangle$.
- The cases Sum, Par and Ide are easy consequences of induction (since we have shorter derivations on the premises and the rules do not modify the label s).
- Case Com. We know that $P \mid Q \xrightarrow{s \bullet s'} P' \mid Q'$ and $P \xrightarrow{s} P'$ and $Q \xrightarrow{s'} Q'$. By induction we know that $P \overset{\gamma}{\Longrightarrow} P'$, $Q \overset{\gamma'}{\Longrightarrow} Q'$ and $\gamma \subseteq \mathtt{ext}(s)$ and $\gamma' \subseteq \mathtt{ext}(s')$. Applying Rule Com_s, we know that $P \mid Q \overset{\gamma \bullet \gamma'}{\Longrightarrow} P' \mid Q'$. The result follows by using Lemmas 10 and 13 to show that $\gamma \bullet \gamma' \subseteq \mathtt{ext}(s) \bullet \mathtt{ext}(s') \subseteq \mathtt{ext}(s \bullet s')$.
- Case Res. Let $P = (\nu a)Q$. We know that $P \xrightarrow{(\nu a)s} Q'$ and $Q \xrightarrow{s} Q'$. By induction we know that $Q \overset{\gamma}{\Longrightarrow} Q'$ and $\gamma \subseteq \mathtt{ext}(s)$. Applying Rule Res_s, we know that $P \overset{(\nu a)\gamma}{\Longrightarrow} Q'$. The result follows by using Lemmas 12 and 13 to show that $(\nu a)\gamma \subseteq (\nu a)\mathtt{ext}(s) \subseteq \mathtt{ext}((\nu a)s)$ as wanted.

A.4 Bisimulation Results

Lemma 14. *$\bowtie$ is an equivalence relation.*

Proof. Reflexivity and transitivity are easy and symmetry holds by definition.

Lemma 15. *Let $s \in \gamma$. For all solid link ${}^{a}\backslash_{b}$, ${}^{a}\backslash_{b} \in \mathsf{e}(s)$ iff $[a \cdot b] \in \mathit{cap}(\gamma)$.*

Proof. Straightforward from the definition of configuration and capabilities.

Lemma 16. *Let γ, γ' be valid configurations. Then, $(\gamma, \gamma') \in \bowtie$ iff $\mathit{cap}(\gamma) = \mathit{cap}(\gamma')$*

Proof. ($\Rightarrow$) Let s be a chain s.t. $s \in \gamma$. We know that there exists $s' \in \gamma'$ s.t. $s \bowtie s'$. By Corollary 3, we know that $\mathsf{e}(s) = \mathsf{e}(s')$. Then, we can use Lemma 15 to show that $\mathtt{cap}(\gamma) = \mathtt{cap}(\gamma')$.

($\Leftarrow$) Let $s \in \gamma$ and assume that $\mathtt{cap}(\gamma) = \mathtt{cap}(\gamma')$. Let s' be as $\mathsf{e}(s)$ but adding/removing some τ transitions so that $s' \in \gamma'$. Note that such s' exists since $\mathtt{cap}(\gamma) = \mathtt{cap}(\gamma')$. We have $\mathsf{e}(s) = \mathsf{e}(s')$ and, by Corollary 3, $s \bowtie s'$. Hence, $(\gamma, \gamma') \in \bowtie$.

Theorem 9. *Let P and Q be processes. Then, $P \sim_n Q$ iff $P \sim_s Q$.*

Proof. ($\Rightarrow$) We shall show that $\mathcal{R} = \{(P, Q) \mid P \sim_n Q\}$ is a symbolic bisimulation. If $P \xRightarrow{\gamma} P'$, γ is a valid configuration and then, there exists $s \in \gamma$. By completeness, we know that $P \xrightarrow{s} P'$. Hence, there exists s' s.t. $\mathsf{e}(s) = \mathsf{e}(s')$ and $Q \xrightarrow{s'} Q'$. By soundness, there exists γ' s.t. $s' \in \gamma'$ and $Q \xRightarrow{\gamma'} Q'$. By Lemma 15 we know that, for all solid link ${}^{a}\backslash_{b} \in \mathsf{e}(s)$ (resp. $\in \mathsf{e}(s')$), $[a \cdot b] \in \mathtt{cap}(\gamma)$ (resp. $\mathtt{cap}(\gamma')$). Since $\mathsf{e}(s) = \mathsf{e}(s')$, $\mathtt{cap}(\gamma) = \mathtt{cap}(\gamma')$ and, by Lemma 16, $\gamma \bowtie \gamma'$ as needed.

($\Leftarrow$) We shall show that $\mathcal{R} = \{(P, Q) \mid P \sim_s Q\}$ is a network bisimulation. Assume that $P \xrightarrow{s} P'$. By soundness, we know that there exists γ s.t. $s \in \gamma$ and $P \xRightarrow{\gamma} P'$. Hence, there exists $\gamma' \bowtie \gamma$ s.t. $Q \xRightarrow{\gamma'} Q'$. By completeness, for all $s' \in \gamma'$, $Q \xrightarrow{s'} Q'$. Pick one s' to have the same order in the solid links as s (possibly with different τ synchronizations). Then, it must be the case that $\mathsf{e}(s) = \mathsf{e}(s')$ as needed.

Corollary 5. *$\sim_s$ is a congruence.*

Proof. Directly from Theorem 9 and Theorem 4

A.5 Algorithm for Checking $\gamma \bowtie \gamma'$.

Algorithm 1 gives us a procedure to check whether two configurations are related via $\bowtie$. More precisely, it checks whether $\mathtt{cap}(\gamma) = \mathtt{cap}(\gamma')$. The function $F_{\bowtie}$ works as follows. Let $\gamma = (\nu \boldsymbol{x})\langle L\rangle$ and $\gamma' = (\nu \boldsymbol{x'})\langle L'\rangle$ be two valid configurations. First, we eliminate from L (resp. L') all the links of the shape ${}^{x}\backslash_{x}$, where $x \in \boldsymbol{x}$ (resp $x \in \boldsymbol{x'}$). This corresponds to eliminate all the "intermediate" ${}^{\tau}\backslash_{\tau}$ synchronization as in ${}^{a}\backslash^{\tau}_{\tau}\backslash^{\tau}_{\tau}\backslash_{b} \rightsquigarrow {}^{a}\backslash^{\tau}_{\tau}\backslash_{b}$. Then, we test whether the capabilities of the configurations are the same. This is done in the function $simulates$: we pick a capability $[a \cdot b]$ in the first configuration. Then, we try to remove the same capability from γ'. This can be done by either, finding exactly the same link on L' or by finding a path of links built from restricted names where a and b appear on the extremes. In the second case, note that from γ' we cannot build the chain ${}^{\square}\backslash_{\square} \ldots {}^{\square}\backslash^{a}_{\square}\backslash^{\square}_{b}\backslash_{\square} \ldots {}^{\square}\backslash_{\square}$ but we can build ${}^{\square}\backslash_{\square} \ldots {}^{\square}\backslash^{a}_{\square}\backslash^{\tau}_{\tau}\backslash_{\tau} \cdots {}^{\tau}\backslash^{\tau}_{\tau}\backslash^{\square}_{b}\backslash_{\square} \ldots {}^{\square}\backslash_{\square}$ which is equivalent to the former under $\bowtie$. Finally, in the end, if γ' has no more capabilities, L' is empty and it means that all the capabilities of γ are in γ' too.

An invariant of $simulates$ is that either L' is empty or $(\nu \boldsymbol{x'})\langle L'\rangle$ is a valid configuration. To see that, note that in each step, we eliminate from L' a balanced number of restricted names as input and as output.

```
Precondition: $a, b \notin \boldsymbol{x}$
Function take($\boldsymbol{x}$, $L$, ${}^a\backslash_b$)
    if ${}^a\backslash_b \in L$ then return $L \setminus \{{}^a\backslash_b\}$;
    if $\exists x_1, ..., x_n \in \boldsymbol{x}$ $s.t.$ ${}^a\backslash_{x_1}, {}^{x_1}\backslash_{x_2}, ..., {}^{x_n}\backslash_b \in L$ then
        return $L \setminus \{{}^a\backslash_{x_1}, {}^{x_1}\backslash_{x_2}, ..., {}^{x_n}\backslash_b\}$
    end
    return $L$
end
Function simulates($\boldsymbol{x}, L, \boldsymbol{x'}, L'$)
    foreach $[a \cdot b] \in (\nu\boldsymbol{x})\langle L\rangle$ do
        let $L'' := take(\boldsymbol{x'}, L', {}^a\backslash_b)$
        if $L'' == L'$ then return false;
        $L' := L''$
    end
    return $L' == \emptyset$
end
Precondition: $(\nu\boldsymbol{x})\langle L\rangle$ and $(\nu\boldsymbol{x'})\langle L'\rangle$ are valid configurations.
Function $F_{\bowtie}$($\boldsymbol{x}, L, \boldsymbol{x'}, L'$)
    foreach $x \in \boldsymbol{x}$ do $L := L \setminus\!\!\setminus {}^x\backslash_x$ ;
    foreach $x' \in \boldsymbol{x'}$ do $L' := L' \setminus\!\!\setminus {}^{x'}\backslash_{x'}$ ;
    return $simulates(\boldsymbol{x}, L, \boldsymbol{x'}, L')$ and $simulates(\boldsymbol{x'}, L', \boldsymbol{x}, L)$
end
```

Algorithm 1: Algorithm to decide whether $(\nu\boldsymbol{x})\langle L\rangle \bowtie (\nu\boldsymbol{x'})\langle L'\rangle$. In $L \setminus\!\!\setminus a$ removes all the copies of a in L. With aTb we mean either the chain with a unique element ${}^a\backslash_b$ or a chain of the shape ${}^a\backslash^{x_1}_{x_1}\backslash_{x_2}\cdots{}^{x_n}\backslash_b$.