Open access • Journal Article • DOI:10.1145/2682570

The Psi-Calculi Workbench: A Generic Tool for Applied Process Calculi

Johannes Borgström, Ramūnas Gutkovas, Ioana Rodhe, Björn Victor

Institutions: Uppsala University

Published on: 21 Jan 2015 - ACM Transactions in Embedded Computing Systems (ACM)

Topics: Symbolic execution, Operational semantics, Process calculus, Broadcast communication network and Bisimulation

These features of psi-calculi save a lot of effort for anyone using it — psi-calculi is a reusable framework.

This paper describes the Psi-Calculi Workbench (PWB), a generic tool for implementing psi-calculus instances, and for analysing processes in the resulting instances. While there are several other tools, specialised for particular process calculi and particular application areas, our tool is generic and reusable. It has a wider scope than previous works, and also allows experimentation with new process calculi with a relatively low effort. Like psi-calculi, our tool is parametric: it provides functionality for bisimulation equivalence checking and symbolic simulation (or execution) of processes in any psi instance, and a base library for implementing new psi-calculi instances. PWB thus has two types of users: the user analysing systems in an existing instance of the framework, and the instance implementor.

We illustrate both uses of the tool in three steps: In Section 2 we introduce the framework of psi-calculi semiformally, relating an instance corresponding to the pi-calculus and showing symbolic simulation of agents. After describing the design of PWB and how to implement an instance in Section 3, we show how to add data and computation in Section 4 by modelling the traditional alternating bit protocol for reliable communication. In Section 5 we model a data aggregation protocol for wireless sensor networks, incorporating specialised data structures and logics, and both unicast and broadcast communication. Section 6 extends the previous example with a dynamic topology.

In Section 7 we describe the symbolic semantics implemented in PWB. The symbolic operational semantics of Section 7.1 simplifies previous symbolic semantics for psi-calculi [Johansson et al. 2012], and adds rules for wireless (synchronous and unreliable) broadcast [Borgström et al. 2011]. To our knowledge, this is the first symbolic semantics for lexically scoped broadcast communication.

In Section 8 we discuss related work. An abridged version of this article was published as [Borgström et al. 2013].

2. INTRODUCING PSI-CALCULI

In this section we introduce the psi-calculi parametric semantic framework semiformally, and defer some precise definitions and the operational semantics to Section 7. For a more extensive treatment of psi-calculi, including motivations of the requisites and examples of other instances see [Bengtson et al. 2011; Borgström et al. 2011; Johansson et al. 2012; Johansson et al. 2010]. We show more complex examples in Sections 4, 5 and 6.

A psi-calculus instance is specified by three data types: the (data) terms $\mathbf{T}$, ranged over by M, N, the conditions $\mathbf{C}$, ranged over by φ, and the assertions $\mathbf{A}$, ranged over by Ψ. The terms, conditions and assertions can be any sets where the elements may contain names (from the set $\mathcal{N}$ of names) and name permutations are admitted (so-called *nominal sets* [Pitts 2003]). In particular, every element X has a finite set of free names $\mathsf{n}(X) \subseteq \mathcal{N}$, and we write $a\#X$ for $a \notin \mathsf{n}(X)$.

Terms are used both as communication channels, and for the data sent and received in communication. They can be structured, and so permit standard constructs as lists and sets, numbers and booleans, as well as more advanced structures. Assertions are used to model "facts" about terms and relations between them, for instance by giving values to variables or by constraining their values. The minimal assertion is the unit, written $\mathbf{1}$, and assertions are composed by the $\otimes$ operator. Conditions are used to perform tests on terms. Their outcome depends on the current assertion environment, through an entailment relation (Ψ entails φ, written $\Psi \vdash \varphi$) which is also part of the psi instance specification.

In the **Pi** instance, corresponding to the polyadic pi-calculus, terms are simply names $a, b, c \ldots$ and the conditions are equality tests on names. (Name equality is used in

the match construct $[a = b]P$, which behaves as P if $a = b$ holds.) In the pi-calculus there are no assertions, but the psi-calculi framework requires at least the trivial unit assertion. Later examples will show how assertions can be exploited for modelling advanced features.

Given the psi-calculus parameters $\mathbf{T}, \mathbf{C}, \mathbf{A}$, the *agents*, ranged over by $P, Q, \ldots$, are of the following forms:

$\overline{M}\,\tilde{N}\,.\,P$	Output prefix
$\underline{M}(\tilde{x})\,.\,P$	Input prefix
$\overline{\overline{M}}!\tilde{N}\,.\,P$	Broadcast output prefix
$\underline{\underline{M}}?\,(\tilde{x})\,.\,P$	Broadcast input prefix
case $\varphi_1 : P_1 \,[]\, \cdots \,[]\, \varphi_n : P_n$	Case
$(\nu a)P$	Restriction
$P \mid Q$	Parallel
$!\,P$	Replication
$(\!\lvert \Psi \rvert\!)$	Assertion
$A\langle \tilde{M} \rangle$	Invocation

We write $\widetilde{M}$ for the tuple $M_1, \ldots M_n$. The output and input prefixes denote polyadic (unicast) output and input, while the broadcast prefixes denote (synchronous) broadcast output and input, which is unreliable (as in wireless systems) in the sense that transmissions might not be received. The case construct can act as any P_i such that the corresponding condition φ_i is true; the other cases are discarded. Restriction binds a in P and input prefixes bind $\tilde{x}$ in the suffix; we identify alpha-equivalent agents. The Invocation form invokes a process A, defined by the form $A(\tilde{y}) \Leftarrow P$; the behaviour is that of $P\{\tilde{M}/\tilde{y}\}$.

In the **Pi** instance, the output and input prefixes are the usual $\overline{a}\,\tilde{x}\,.\,P$ and $\underline{a}(\tilde{x})\,.\,P$; the match construct $[a = b]P$ corresponds to **case** $a = b : P$. If we have a condition $\mathbf{true}$ which is always true, we can model nondeterministic choice (traditionally written $P + Q$) as **case** $\mathbf{true} : P \,[]\, \mathbf{true} : Q$.

The semantics for psi-calculi is defined by a labelled transition relation written $\Psi \rhd P \xrightarrow{\alpha} P'$, meaning that in environment Ψ agent P can do an action α to become P'. In the pi-calculus instance, the environment Ψ is always the trivial $\mathbf{1}$, but in general it represents the assertions of the environment, including parallel agents.

The semantics is defined only for well-formed agents. An occurrence of a subterm in an agent is *guarded* if it is a proper subterm of a prefix form. An agent is *well-formed* if in $\underline{M}(\widetilde{x}).P$ and $\underline{\underline{M}}?\,(\widetilde{x}).P$ it holds that $\widetilde{x}$ is a sequence without duplicates, that in **case** $\varphi_1 : P_1 \,[]\, \cdots \,[]\, \varphi_n : P_n$ the agents P_i have no unguarded assertions, and that in a replication $!P$ the agent P has no unguarded assertions or broadcast input prefixes. For process definitions a similar requirement as for replication applies.

The actions are input $\underline{M}\ (\tilde{x})$ denoting the reception of data bound to $\tilde{x}$ over the channel denoted by M, output $\overline{M}\ (\nu\tilde{x})\tilde{N}$ denoting the sending of $\tilde{N}$ over M and additionally opening the scopes of the names $\tilde{x}$, the corresponding broadcast actions $\underline{\underline{M}}?(\tilde{x})$ and $\overline{\overline{M}}!\,(\nu\tilde{x})\tilde{N}$, and the silent action τ which is the result of communication between an input and an output. When $\widetilde{x}$ is empty, we often omit $(\nu\widetilde{x})$ and $(\widetilde{x})$.

The connectivity predicates used for communication are also defined by the instantiation. The conditions include the *channel equivalence* predicate $M \leftrightarrow N$ which is used to define which terms denote the same unicast channel, and the broadcast connectivity predicates $M \dot{\prec} K$ and $K \dot{\succ} M$ for sending and receiving on broadcast channels: a term M can be used to send a broadcast message on the channel K only if $M \dot{\prec} K$ in

the current assertion environment, and similar for broadcast reception (see Section 5 for an example).

As an example, the **Pi** agent

$$\overline{b}\,c.Q \mid \underline{b}(a).\mathbf{case}\ a = b : \underline{a}(z).R$$

has transitions labelled $\overline{b}c$, $\underline{b}\ (x)$ for all names x, and τ. The input prefix can generate infinitely many input actions (here one for each x). To avoid this infinite branching, we use a *symbolic* semantics in the tool (see Section 7.1), where the actual values are abstracted by variables. Instead each transition has a *transition constraint*, which must be satisfied for the corresponding non-symbolic transitions to be possible. Formally these transitions are written $P \xrightarrow[C]{\alpha} P'$ where C is a transition constraint.

The input transitions of the agent above can be represented by a single transition in the symbolic semantics. For simplicity we show the first two transitions of the input prefix subagent:

$$P = \underline{b}(a)\,.\,\mathbf{case}\ a = b : \underline{a}(z).R \xrightarrow[\{\!|\mathbf{1} \vdash b \dot{\leftrightarrow} w|\!\}]{\underline{w}\ (a)} \mathbf{case}\ a = b : \underline{a}(z).R \xrightarrow[\{\!|\mathbf{1} \vdash a \dot{\leftrightarrow} v|\!\} \wedge \{\!|\mathbf{1} \vdash a = b|\!\}]{\underline{v}\ (z)} R$$

where w and v are fresh (see Section 7 for the formal semantics). The constraint of the first transition intuitively says that the channel w is equivalent to b (there may not always be such a w!); for the second transition a similar constraint appears in addition to the condition of the case construct.

We can use the PWB to simulate the transitions of P. The tool uses an ASCII representation of agents, where non-alphanumeric terms and conditions must be in double quotes, ν is written new, output objects are written between angular brackets and the overline in outputs is written by a preceding single quote. For example, $\overline{b}\,\mathrm{f}(a,c)\,.\,(\nu x)Q$ is written 'b<"f(a,c)">.(new x)Q.

The first transition of the agent P above:

```
--|gna(a)|-->

  Source:
    b(a). case "a = b" : a(x). R<>
  Constraint:
    {| "b = gna" |}
  Solution:
    ([gna := b], 1)
  Derivative:
    case "a = b" : a(x). R<>
```

When printing the constraint, the trivial $\mathbf{1} \vdash$ is elided. The "gna" here represents a fresh name, corresponding to w above: the subject of the symbolic input action.

The derivative **case** "a = b" : a(x).R does not have a non-symbolic transition since a is not the same name as b, but the symbolic semantics does have a transition *under the constraint* that a=b.

```
--|gnb(x)|-->

  Source:
    case "a = b" : a(x). R<>
  Constraint:
    {| "a = gnb" |} ∧   {| "a = b" |}
  Solution:
    ([b := a, gnb := a], 1)
  Derivative:
    R<>
```

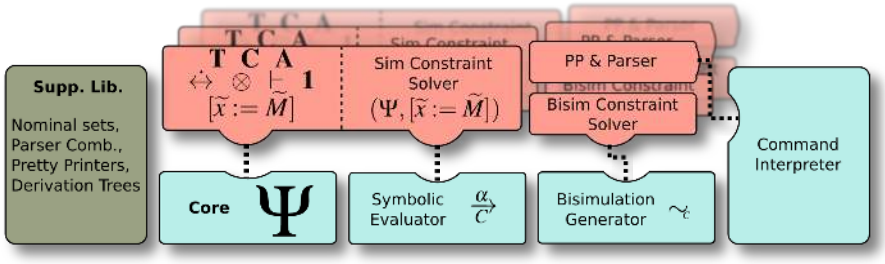

Fig. 1: Psi-Calculi Workbench Architecture

The constraint {| "a = b" |} can be solved by substituting a for b, as stated by the Solution line above. The solution is generated by a constraint solver module in the PWB, which for the pi-calculus instance performs name unification (see Section 3.2), similar to earlier tools for pi-related calculi (e.g. MWB). After applying the solution to the agent, there is a corresponding non-symbolic transition.

In addition to symbolic execution, the PWB also includes a symbolic checker that computes a minimal sufficient constraint for one agent to be (bi)similar to another, plus a witnessing relation. The two agents are non-symbolically related after applying a solution to the constraint (if there is one).

3. IMPLEMENTATION

The Psi-Calculi Workbench (PWB) is implemented in the Standard ML programming language and compiles under the Poly/ML compiler [PolyML 2013] version 5.4. PWB is open source and freely available online from [Gutkovas and Borgström 2013].

PWB is a modular implementation of psi-calculi, and can be viewed both as a modelling tool and as a library for building tools for particular instances of psi-calculi. Used as a modelling tool, the user interacts with a command interpreter that provides commands for process definitions (manually or from files), manipulation of the process environment, stepping through symbolic (strong and weak) transitions of a process, and symbolic bisimilarity checking (strong and weak). Examples of such use are given in sections 4 and 5. Below we describe the implementation of PWB and the modules which need to be provided when creating an instance of psi-calculi.

3.1. Psi-Calculus instantiation

PWB implements a number of helper libraries for the instance implementor. We show the architecture of PWB in Figure 1. In this figure, dependencies between components go from right to left: each component may depend only on components that are above it or to its left. All components build on the supporting library that provides the basic data structures and core algorithms for psi-calculi. The instance implementor provides definitions for the parameters of an instance, constraint solvers, and parsing and pretty-printing code. These user-implemented components are then called by the different algorithms implemented by the tool and by the command interpreter. Not all components are required to be implemented: for instance, the bisimulation constraint solver is only needed for bisimilarity checking.

The parameters of an instance consist of the types name, term, condition and assertion, and three classes of functions: those defining the logics, the substitutions, and the connectivity. As an example of the types, here are the declarations for the pi-calculus

instance mentioned in Section 2. All SML code presented is written by the instance implementor.

```
type term           = name
datatype condition = Eq of term * term | T
datatype assertion = Unit
```

We need three functions to define the logic of the instance: entailment (entails, or $\vdash$) that describes which conditions are true given an assertion, a composition operator (compose, or $\otimes$) that composes two assertions, and a unit assertion (unit, or $\mathbf{1}$). We require that assertion composition forms a commutative monoid (modulo entailment), and that all functions are equivariant, meaning that they treat all names equally. The bisimulation algorithm and the weak symbolic semantics also require weakening to hold, meaning that $\Psi \vdash \varphi$ implies $\Psi \otimes \Psi' \vdash \varphi$ for all Ψ'.

```
val entails : assertion * condition -> bool
val compose : assertion * assertion -> assertion
val unit    : assertion
```

We also need equivariant substitution functions, substituting terms for names in each of term, condition and assertion.

```
type subst = (name * term) list
val substT : subst -> term      -> term
val substC : subst -> condition -> condition
val substA : subst -> assertion -> assertion
```

Finally, we have three equivariant functions that describe the connectivity of the calculus: chaneq (for unicast connectivity), brTransmit and brReceive (for broadcast). Typically, these functions are simple injections into the conditions type (e.g., **fun** chaneq (M,N) = ChanEq (M,N) where ChanEq is a data constructor of condition) leaving the definition of connectivity to either the entailment relation or the constraint solver.

Channel equivalence chaneq is required to be commutative and transitive (for every Ψ). brTransmit is broadcast output connectivity $\dot{\prec}$ and brReceive is broadcast input connectivity $\dot{\succ}$; these functions are exemplified in Section 5. If Ψ entails $M \dot{\prec} K$ or Ψ entails $K \dot{\succ} M$, then we require all names that occur in K to also occur in M.

```
val chaneq      : term * term -> condition
val brTransmit : term * term -> condition
val brReceive   : term * term -> condition
```

All of the functions above are further required to commute with substitution, in the sense that $f(X\sigma) = f(X)\sigma$.

The user also needs to implement parsers for each of the data types, that are called by the parser for process terms.

3.2. Symbolic execution

PWB provides symbolic execution of processes by the sstep command. This is a useful tool to explore the properties of a process, or indeed the model itself. Here values input by the process are represented by variables, and constraints are collected along the derivation of a transition. The constraints show under which conditions transitions are possible, deferring instantiation of variables as long as possible. Both strong and weak (ignoring τ-transitions) symbolic semantics are available (presented in Section 7).

In psi-calculi, parallel contexts that contain an assertion, such as $(\!|x = 3|\!)$, can enable additional transitions. Therefore, a solution (σ, Ψ) to a constraint consists of a

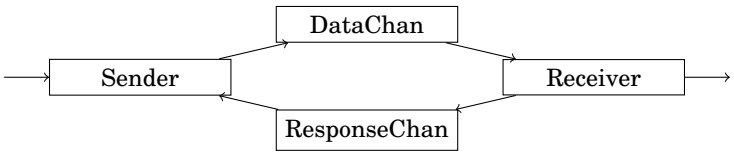

Fig. 2: Alternating Bit Protocol scheme

substitution σ (representing earlier inputs) and an assertion Ψ (representing the parallel context). Intuitively, every solution (σ, Ψ) solves true, there is no solution to false, every solution to both C and C' is a solution to $C \wedge C'$, and the solutions to $(\nu\tilde{a})\{\!|\Psi' \vdash \varphi|\!\}$ are the pairs (σ, Ψ) where $\Psi \otimes \Psi'\sigma \vdash \varphi\sigma$ and the names in $\tilde{a}$ do not occur in Ψ, σ.

The instance implementor may provide a constraint solver for the transition constraints. The solver should return either a string describing the unsatisfiability of a constraint, or a solution consisting of a substitution and assertion. Since transition constraints are simply a conjunction of atomic constraints, a simple unification-based solver often suffices. The type of the solver is the following:

```
val solve : constraint -> (string, (name * term) list * assertion) either
```

As an example, the solver for the pi-calculus instance of Section 2 performs unification, implemented by the transition relation below. The nodes in the transition system are either a pair (C, σ), or the failed state $\varnothing$.

$$\begin{array}{rcll}
(\nu\tilde{a})\{\!|\mathbf{1} \vdash \mathrm{T}\ |\!\} \wedge C, \sigma & \rightarrow & C, \sigma & \\
(\nu\tilde{a})\{\!|\mathbf{1} \vdash a = a|\!\} \wedge C, \sigma & \rightarrow & C, \sigma & \\
(\nu\tilde{a})\{\!|\mathbf{1} \vdash a = b|\!\} \wedge C, \sigma & \rightarrow & \varnothing & \text{if } a \neq b \wedge (a \in \tilde{a} \vee b \in \tilde{a}) \\
(\nu\tilde{a})\{\!|\mathbf{1} \vdash a = b|\!\} \wedge C, \sigma & \rightarrow & C[b := a], \sigma[b := a] & \text{otherwise}
\end{array}$$

3.3. Symbolic (bi)simulation

PWB can also be used to check simulation relations on processes. As an example, the command `P ~ Q` attempts to construct a bisimulation relation relating agents P and Q. To this end, we implement a symbolic bisimulation algorithm based on [Johansson et al. 2012] (with some corrections and optimisations). This algorithm takes two processes and yields a constraint in an extended constraint language; the two processes are bisimilar under all solutions to the constraint. A simple variation of the algorithm is used for simulation checking.

The language for bisimulation constraints additionally includes conjunction, disjunction and implication, as well as constraints for term equality $\{\!|M = N|\!\}$, freshness $\{\!|a\#X|\!\}$ (with the intuition "a is not free in X"), and static implication. In order to simplify the development of a constraint solver for this richer language, PWB contains an SMT solver library with suitable helper functions. Unless the assertion language is trivial (only the unit assertion), most of the additional effort in extending a solver for transition constraints to one for bisimulation constraints lies in properly treating static implication constraints.

4. THE ALTERNATING BIT PROTOCOL

In this section, we describe the modelling in PWB of the classical Alternating Bit Protocol. This demonstrates the use of PWB to define a tailor-made process calculus for a particular problem or problem domain. We also give an example of symbolic weak transition generation in PWB.

4.1. Introduction to the Alternating Bit Protocol

The Alternating Bit Protocol (ABP) [Bartlett et al. 1969] is a simple network protocol for reliable data transmission through lossy channels. Reliable here means that all data fragments are received exactly once and in the right order at the receiver. Consider a sender Sender, a receiver Receiver and two communication channels between them: DataChan, over which data fragments are sent, and ResponseChan, over which acknowledgements are sent. We show this situation in Figure 2: the arrows denote the direction of the data being transmitted. ABP assumes reliable error detection, but no error correction.

To ensure that Receiver receives every fragment despite lossy communication channels, Sender repeatedly sends the same fragment until it receives a corresponding acknowledgment, at which point the sender starts transmitting the next fragment. Since the receiver should not accept the same fragment twice, a protocol is needed for distinguishing between packets. In ABP, each data packet has a one-bit flag attached to it. The flag 0 is attached to the first packet sent; the acknowledgment of the receiver for this packet will also have flag bit 0. When Sender receives an acknowledgment with flag 0, it knows that Receiver has correctly received the fragment, and Sender will then start sending the next packet with flag bit 1, and so on. Thus, sequences of sent or received packages resp. acknowledgments with the same flag bit all refer to the same data fragment.

4.2. A Psi-calculus Instance for ABP

To define a psi-calculus instance where ABP can be expressed, we start with the data terms. Since the behaviour of the protocol does not depend on the data being transmitted, we simply represent each fragment as a name. However, the protocol itself needs some data values and structures.

In the set of terms we include the channels DataChan and ResponseChan, and the value ERR to signify that an error has been detected. We also have 0 and 1 bits and a negation operation $\sim\cdot$ on them with the expected equalities $\sim 0 = 1$ and $\sim 1 = 0$.

Our account of ABP is untyped, so these term constructors yield terms which are not intended to be part of the model, such as $\sim$ERR. Such spurious terms yield the invalid value $\bot$. In summary, we define the data terms $\mathbf{T}$ as follows:

Notation	*SML*	PWB
$Val \triangleq \{\mathrm{ERR}, 0, 1\}$ $\mathbf{T} \triangleq Val \cup \{\bot\} \cup \mathcal{N}$ $\cup \{\sim M : M \in \mathbf{T}\}$	**datatype** term = Error | Zero | One | Bottom | Name **of** name | Neg **of** term	M ::= ERR | 0 | 1 | -|- | *Name* | ~M

Here and in subsequent displays, the column *Notation* is the mathematical notation, *SML* is the code written by the instance implementor, and PWB is the ASCII syntax used in the tool by the user of the instance.

Next we define the conditions. In the protocol, we need to compare the sender's or receiver's bit with a transmitted bit, and to see whether an error occurred while transmitting data. To do this, we use equality $=$ on values.

We add a condition True which always holds, and a False condition that never holds. Lastly, we include a channel equivalence condition for unicast communication (ABP does not use broadcast, so we let the broadcast connectivity predicates yield False).

Notation	*SML*	PWB
$\mathbf{C} \triangleq \{\text{True}, \text{False}\}$ $\cup \{M = N : M, N \in \mathbf{T}\}$ $\cup \{M \leftrightarrows N : M, N \in \mathbf{T}\}$	**datatype** condition = CTrue | CFalse | Equal **of** term * term | ChEq **of** term * term	$\varphi ::=$ True | False | M = M | M <-> M

We do not need assertions to model the ABP, so we let $\mathbf{A} = \{\text{Unit}\}$ as in Section 2.

As the last step we define the substitution functions on terms and conditions. They are standard capture avoiding substitutions, followed by normalisation with respect to a term rewriting system given below. We use rewriting after substitutions in order to accurately detect loops of τ-transitions when computing weak transitions. This also significantly simplifies the constraint solver, since the normal forms are simpler to handle than arbitrary terms.

Below, we give the rewrite system for terms for reduction context $\mathcal{R} ::= []|\sim\mathcal{R}$. It evaluates the $\sim\cdot$ operator, cancels out double negation of variables, and identifies the spurious terms. In particular, the term $\sim\sim$ERR is spurious, and is rewritten to $\bot$.

$$\begin{array}{rlrlrl} \sim\text{ERR} & \to \bot & \sim 0 & \to 1 & \sim\sim x & \to x \text{ if } x \in \mathcal{N} \\ \sim\bot & \to \bot & \sim 1 & \to 0 & & \end{array}$$

The following is the term rewriting system for the conditions. Equalities involving spurious terms $\bot$ are rewritten to False. Note that we only consider equality conditions where the constituent terms are already in normal form; this suffices since the substitution function on conditions is defined in terms of substitution function on terms.

$$\begin{array}{rlrl} \sim x = \sim y & \to x = y & M = N & \to \text{True} \text{ if } M = N \text{ and } \{M, N\} \subseteq \mathit{Val} \cup \mathcal{N} \\ \sim x = x & \to \text{False} & M = N & \to \text{False} \text{ if } M \neq N \text{ and } \{M, N\} \subseteq \mathit{Val} \\ x = \sim x & \to \text{False} & M = N & \to \text{False} \text{ if } \bot \in \{M, N\} \end{array}$$

Finally, we need to define entailment. For conditions in normal form we define

$$\text{Unit} \vdash a \leftrightarrows b \text{ iff } a = b \qquad \text{Unit} \vdash M = N \text{ iff } M = N \qquad \text{Unit} \vdash \text{True},$$

and otherwise we let $\text{Unit} \vdash \varphi$ iff $\varphi \to^+ \varphi' \not\to$ and $\text{Unit} \vdash \varphi'$

4.3. Constraint Solver for ABP Transition Constraints

The ABP constraint solver is a standard unification algorithm defined as a transition system. The design is greatly simplified by the fact that the conditions in the constraints are in normal form.

The following is the unification transition system. The first two rules are trivial. The rules concerning the channel equivalence $\leftrightarrows$ condition are the classic unification on names as seen in the pi-calculus solver. The last rules concern the equality condition $=$. Because the terms are in the normal form, we know that one of the sides is a name, and thus we do elimination, or swapping in order to allow elimination. Below, $\tilde{a}\#X$ denotes that names $\tilde{a}$ don't occur freely in X; we omit $\mathbf{1} \vdash$ in front of every condition.

$$\begin{array}{rcll} (\nu\tilde{a})\{\!|\text{True}|\!\} \wedge C, \sigma & \to & C, \sigma & \\ (\nu\tilde{a})\{\!|\text{False}|\!\} \wedge C, \sigma & \to & \varnothing & \\ (\nu\tilde{a})\{\!|a \leftrightarrows b|\!\} \wedge C, \sigma & \to & C, \sigma & \text{if } a = b \text{ and } a, b \in \mathcal{N} \\ (\nu\tilde{a})\{\!|a \leftrightarrows b|\!\} \wedge C, \sigma & \to & C[b := a], \sigma[b := a] & \text{if } \tilde{a}\#a, b \text{ and } a \neq b \\ (\nu\tilde{a})\{\!|a \leftrightarrows b|\!\} \wedge C, \sigma & \to & \varnothing & \text{otherwise} \\ (\nu\tilde{a})\{\!|a = M|\!\} \wedge C, \sigma & \to & C[a := M], \sigma[a := M] & \text{if } \tilde{a}\#a, M \text{ and } a \in \mathcal{N} \\ (\nu\tilde{a})\{\!|M = N|\!\} \wedge C, \sigma & \to & (\nu\tilde{a})\{\!|N = M|\!\} \wedge C, \sigma & \text{otherwise} \end{array}$$

4.4. The ABP as a Process

Here we present the process modelling the ABP in the ABP psi-calculus instance defined above. We give the definition in PWB syntax, which is used by the user of the psi instance.

We model the components Sender and Receiver of ABP shown in Figure 2 as psi-calculus processes. The behaviour of components DataChan and ResponseChan are captured implicitly in our model. For composing the system, components have input and output channels inp and out, respectively. The Receiver and Sender each have one additional channel for output o resp. input i to the application that uses the protocol.

The sender is modelled as follows: first it inputs data on input channel i and then recursively outputs the data together with the current bit b on the channel out. Then the sender receives the acknowledgment bit on input channel inp: if it matches b, the sender flips b and returns to waiting for data, otherwise (if the bit did not match or an error occurred) the sender attempts to send the data and b until it receives an acknowledgment with flag b.

```
Sender(i,inp,out,b) <= i(data).SenderSend<i,inp,out,data,b>;

SenderSend(i,inp,out,data,b) <= 'out<data, b>. inp(ackBit).
    case "b = ackBit"    : Sender<i,inp,out,"~b">
      [] "b = ~ackBit"   : SenderSend<i,inp,out,data,b>
      [] "ERR = ackBit"  : SenderSend<i,inp,out,data,b> ;
```

The receiver works in a dual fashion.

```
Receiver(o, inp, out, b) <= inp(data, bit).
    case "b = bit"    : 'o<data>.'out<b>.Receiver<o,inp,out,"~b">
      [] "b = ~bit"   : 'out<"~bit"> . Receiver<o,inp,out,b>
      [] "ERR = bit"  : 'out<"~b"> . Receiver<o,inp,out,b> ;
```

An error might occur at any time on each of the channels. This kind of unreliable process is modelled implicitly by treating names (representing bits) as variables. Since transmitted names are variables the constraint solver may enable any **case** clause in either Sender or Receiver by finding a suitable term to substitute them for.

Hiding the internal channels, the ABP system can be described as follows:

```
ABP(i,o,sb,rb) <= (new RcSn, SnRc) (
    Sender<i,RcSn,SnRc,sb> | Receiver<o,SnRc,RcSn,rb>);
```

4.5. A Sample Weak Transition

When studying the ABP, it is interesting to see when the protocol communicates with the outside system, ignoring τ-transitions. We here show such a "weak" transition, where the sender receives data and transmits it to the receiver via the data channel. We use the wsstep command on ABP<i,o,sb,rb> to obtain the following transition, among others.

```
1  ==|gen2(data1)|==>
2    Source:
3      ABP<i, o, sb, rb>
4    Constraint:
5      (new RcSn, SnRc){| "i <-> gen2" |} ∧
6      (new RcSn, SnRc){| "SnRc <-> SnRc" |} ∧
7      (new RcSn, SnRc){| "RcSn <-> RcSn" |} ∧
8      (new RcSn, SnRc){| "rb = ~sb" |}
9    Solution:
10     ([rb := "~sb", gen2 := i], 1)
```

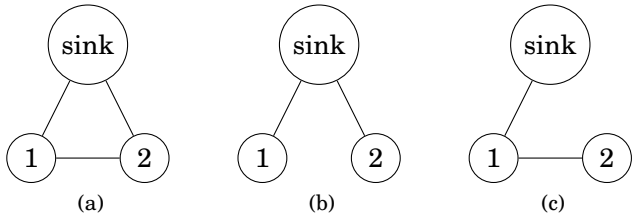

Fig. 3: A simple topology with a sink and two sensor nodes where (a) shows the connectivity and (b)-(c) show some possible routing trees.

```
11    Derivative:
12      (new SnRc, RcSn)(
13        (case
14          False : Sender<i, RcSn, SnRc, "~sb"> []
15          True  : SenderSend<i, RcSn, SnRc, data1, sb> []
16          False : SenderSend<i, RcSn, SnRc, data1, sb>
17        ) |
18          (Receiver<o, SnRc, RcSn, rb>)
19      )
```

After the transition, the sender (lines 13-16) is in a state where it has received an acknowledgment bit which does not match its own bit (constraint on line 8) reducing the condition "b = ~ackBit" (at this state it is "sb = ~rb") of SenderSend to true (on line 15).

This transition is among the seven transitions produced by PWB. Since there is always a possibility that both sender and receiver will detect an error ERR, there are infinitely many weak transitions following a cycle between them. The occurrence of such cycles are detected (modulo alpha-equivalence) by the wsstep command. Since the terms occurring in agents are in normal form, wsstep terminates on ABP.

We have shown the development of a tailor-made psi-calculus instance in PWB. (The full code listing is available online [Gutkovas and Borgström 2013].) Doing so, we have expressed bits and bit operations directly, and we have shown that it is possible and useful to use computation in the substitution functions, which departs from traditional calculi. We have also shown the symbolic simulation of a weak transition, which is useful for applications.

5. DATA COLLECTION IN A WIRELESS SENSOR NETWORK

In this example we study a data collection protocol for wireless sensor networks (WSNs) by modelling it in a custom psi-calculus that we implement in PWB.

A wireless sensor network consists of numerous sensor nodes that sense environmental data. A special node, called the sink, is used to collect data from the network. Collection often uses multi-hop communication, building a routing tree rooted at the sink [Madden et al. 2002]. As wireless communication is unreliable, different trees may be built in each protocol run.

We present a simple algorithm to build a routing tree: the sink starts the tree building by broadcasting a special init message containing its identifier $Sink$. When a node n first receives an init message, it sets its parent $parent_n$ to the sender of the message, and broadcasts a new init message containing its own identifier to continue building the next level of the tree. After the building of a tree is complete, each node sends a data message containing its data to its parent. Moreover, each node forwards received data messages to its parent, ensuring that it eventually reaches the sink.

5.1. Psi-calculus instance for WSN data collection

We first define and implement a custom Psi-calculus instance suitable for modelling the tree building and data collection protocol described above. We use structured channels, of two kinds: broadcast channels $\mathsf{init}(M)$ and unicast channels $\mathsf{data}(M)$. The broadcast connectivity between nodes is given by an undirected topology graph. We first assume a static topology top; the topology in Figure 3(a) would be represented by $\mathrm{top} = \{(0,1),(0,2),(1,2)\}$ where the sink has id 0. The corresponding psi-calculi parameters are defined as follows.

Notation	*SML*	PWB
$\mathbf{T} \triangleq \{\mathsf{init}(M), \mathsf{data}(N) : M, N \in \mathbf{T}\} \cup \mathcal{N} \cup \mathbb{N}$ $\mathbf{C} \triangleq \{M \dot{\prec} N, M \dot{\succ} N, M \dot{\leftrightarrow} N : M, N \in \mathbf{T}\}$ $\mathbf{A} \triangleq \{\mathrm{top}\}$ $\mathbf{1} \triangleq \mathrm{top}$	**datatype** term = Init **of** term | Data **of** term | Name **of** name | Int **of int** **datatype** condition = OutputConn **of** term*term | InputConn **of** term*term | ChEq **of** term*term **datatype** assertion = Unit **val** unit = Unit	M ::= init(M) | data(M) | *Name* | *N* φ ::= M < M | M > M | M <-> M Ψ ::= 1 N ::= $[0-9]^+$

Since we consider a static topology, we implement assertions as a unit type. A broadcast output prefix with subject $\mathsf{init}(i)$ can broadcast on the broadcast channel $\mathsf{init}(i)$, while an input prefix with the same subject can receive from any connected broadcast channel as given by the topology. Two unicast prefixes may communicate iff their subjects are the same name. Thus, we define $\vdash$ as follows.

$$\begin{array}{ll}
\Psi \vdash \mathsf{init}(M) \dot{\prec} \mathsf{init}(N) & \text{iff } M = N \in \mathbb{N} \\
\Psi \vdash \mathsf{init}(M) \dot{\succ} \mathsf{init}(N) & \text{iff } M, N \in \mathbb{N} \text{ and either } (M,N) \in \Psi \text{ or } (N,M) \in \Psi \\
\Psi \vdash \mathsf{data}(a) \dot{\leftrightarrow} \mathsf{data}(b) & \text{iff } a = b \in \mathcal{N}
\end{array}$$

5.2. Constraint Solver for Symbolic Transitions

We describe the implementation of the transition constraint solver. We write $\varnothing$ for no solution. Transition constraints are conjunctions of conditions. The constraints are solved in two phases, corresponding to the unicast connectivity constraints and the broadcast connectivity constraints, respectively. To simplify the solver, we treat all free names in the processes as distinct (cf. distinctions [Milner et al. 1992b]). For unicast constraints, the solver thus fails (returning $\varnothing$) if the constraint is not satisfied.

$$\begin{array}{ll}
(\nu\tilde{a})\{\!|\mathsf{data}(a) \dot{\leftrightarrow} \mathsf{data}(b)|\!\} \wedge C \rightarrow C & \text{if } a = b \\
(\nu\tilde{a})\{\!|\mathsf{data}(a) \dot{\leftrightarrow} \mathsf{data}(b)|\!\} \wedge C \rightarrow \varnothing & \text{otherwise}
\end{array}$$

The constraint solver then checks for broadcast connectivity in the given topology. Let O be the output constraints $\{\!|\mathsf{init}(n) \dot{\prec} a|\!\}$ and I the input constraints $\{\!|a \dot{\succ} \mathsf{init}(n)|\!\}$. We distinguish four different cases:

(1) if $I = \emptyset$ and $O = \{\{\!|\mathsf{init}(n) \dot{\prec} a|\!\}\}$, then the solution is $[a := \mathsf{init}(n)]$.
(2) if $I \neq \emptyset$ and $O = \{\{\!|\mathsf{init}(n) \dot{\prec} a|\!\}\}$, and we have $(n,m) \in \mathrm{top}$ for every constraint $\{\!|a \dot{\succ} \mathsf{init}(m)|\!\}$ in I, then the solution is $[a := \mathsf{init}(n)]$. Otherwise the constraint is unsatisfiable, i.e. $\varnothing$.
(3) if $I \neq \emptyset$ and $O = \emptyset$, then the constraint solver finds n such that for every $\{\!|a \dot{\succ} \mathsf{init}(m)|\!\} \in I$ we have $(n,m) \in \mathrm{top}$. For each such n, $[a := \mathsf{init}(n)]$ is a possible solution.
(4) if $I = \emptyset$ and $O = \emptyset$, then the broadcast part of the constraint is trivially true.

5.3. Tree building model

Once the instance is implemented, we can define processes modelling the tree building algorithm in PWB syntax. The sink broadcasts its own channel and then goes into data collection mode, that is, it listens on its unicast channel repeatedly.

```
Sink(nodeId, bsChan) <=
    '"init(nodeId)"!<bsChan> .
    ! "data(bsChan)"(x) ;
```

A node listens on its broadcast channel for a channel of a parent to which it will send data to. Then, similarly to the sink, it broadcasts its own unicast channel on which it expects data to receive in order to forward it to the parent. After completing the broadcast, it sends its data to the parent and goes into mode of forwarding data.

```
Node(nodeId, nodeChan, datum) <=
    "init(nodeId)"?(pChan) .
    '"init(nodeId)"!<nodeChan> .
    '"data(pChan)"<datum> .
    NodeForwardData<nodeChan, pChan> ;

NodeForwardData(nodeChan, pChan) <=
    ! "data(nodeChan)"(x). '"data(pChan)"<x>  ;
```

5.4. Example Strong Transitions

We here study the (symbolic) transition system generated by a small WSN with a sink and two sensor nodes. Each node has a unique channel for response messages.

```
System3(d1,d2) <=
    (new chanS) Sink<0,chanS>      |
    (new chan1)  Node<1, chan1, d1>  |
    (new chan2)  Node<2, chan2, d2>
```

We will show a possible transition sequence in PWB, using the topology shown in Figure 3a. Below, we only consider transitions labelled with broadcast output and unicast communication actions.

The following initial transition is obtained by executing the symbolic simulator of PWB on System3<d1,d2>. The resulting system is in configuration where both sensor nodes have obtained the parent's channel, in this case the sink's. The nodes would then be able to communicate their data to the sink. The unicast channel connectivity corresponds to the routing tree shown in Figure 3b. It is one of seven possible initial transitions produced by PWB, of which three represent broadcast reception from the environment, and the other three situations where not all nodes receive the broadcast message. The transition label gna!(new bsChan)bsChan, represents the channel with a fresh name gna. The generated constraint requires $\{\!|\mathrm{init}(0) \dot{\prec} \mathrm{gna}|\!\} \wedge \{\!|\mathrm{gna} \dot{\succ} \mathrm{init}(1)|\!\} \wedge \{\!|\mathrm{gna} \dot{\succ} \mathrm{init}(2)|\!\}$, meaning node 0 is output connected to some channel gna which is input connected to nodes 1 and 2. The constraint solver finds a solution to the constraint, which substitutes $\mathrm{init}(0)$ for gna.

```
--|gna!(new bsChan)bsChan|-->
Source:
  System3<d1, d2>
Constraint:
  (new chan1, chan2, chanS){| "init(0)<gna" |} ∧
  (new chanS, chan2, chan1){| "gna>init(1)" |} ∧
  (new chanS, chan1, chan2){| "gna>init(2)" |}
Solution:
```

```
  ([gna := "init(0)"], 1)
Derivative:
  (!("data(chanS)"(x))) |
    (((new chan1)(
      '"init(1)"!<chan1>.
        '"data(chanS)"<d1>.
          NodeForwardData<chan1, chanS>
    )) |
      ((new chan2)(
        '"init(2)"!<chan2>.
          '"data(chanS)"<d2>.
            NodeForwardData<chan2, chanS>
      )))
```

In the derivative the Sink successfully communicated its unicast channel chanS to both nodes.

From this point the system can evolve in two symmetrical ways: either of the nodes broadcasts an init message, but since no node in the (closed) system is listening on a broadcast channel, the message is not received. The following transition is for node 1.

```
--|gna!(new chan1)chan1|-->
Source:
  The same as the above derivative
Constraint:
  (new chan2, chan1){| "init(1)<gna" |}
Solution:
  ([gna := "init(1)"], 1)
Derivative:
  (!("data(chanS)"(x))) |
    (('"data(chanS)"<d1>.
      NodeForwardData<chan1, chanS>) |
     ((new chan2)(
       '"init(2)"!<chan2>.
         '"data(chanS)"<d2>.
           NodeForwardData<chan2, chanS>
      )))
```

The system is now in the state where node 1 can send data to the sink. By following the analogous transition for node 2, we get the system where both nodes are ready to communicate the data.

```
--|gna!(new chan2)chan2|-->
Source:
  The same as the above derivative
Constraint:
  (new chan2){| "init(2)<gna" |}
Solution:
  ([gna := "init(2)"], 1)
Derivative:
  (!("data(chanS)"(x))) |
    (('"data(chanS)"<d1>.
      NodeForwardData<chan1, chanS>) |
     ('"data(chanS)"<d2>.
       NodeForwardData<chan2, chanS>))
```

We have demonstrated the use of advanced features in PWB such as the use of structured channels with different modes of communication (point-to point vs broadcast). The broadcast connectivity graph (topology) was formalised as an assertion; this allows

us to potentially extend the model, for instance to dynamic or localised connectivity. We used the symbolic execution to simulate strong symbolic transitions of the system. All of this shows the versatility and utility of PWB for use in modelling and studying WSN algorithms.

6. DYNAMIC TOPOLOGY IN WIRELESS SENSOR NETWORK

We here extend the example of Section 5 with dynamic topology. We first allow adding edges to the connectivity graph, and then add the dual operation of removing edges.

Let the parameters be as in the example in Section 5 except for the assertions, which is now a finite set of tuples representing edges in a topology.

Notation	*SML*	PWB
$\mathbf{A} \triangleq \mathcal{P}_{\text{fin}}(\mathbf{T} \times \mathbf{T})$ $\mathbf{1} \triangleq \emptyset$	**datatype** assertion = Top **of** (term*term)**list** **val** unit = Top []	$\Psi ::= \epsilon$ $\mid \ (\mathbf{M},\mathbf{N})(,(\mathbf{M},\mathbf{N}))^*$

The entailment relation is left unchanged, and the constraint solver for the unicast constraints is the same. To enable broadcast connectivity, if the necessary edge is not present, the solver simply attempts to add it to the solution (as is common in process calculi models for WSNs [Ghassemi et al. 2008; Godskesen 2010]). For example, the solution of the constraint of the first transition in Section 5.4 with an empty topology is ([gna := "init(0)"], "(0,2),(0,1)").

In the following we add the ability for agents to also remove edges from the environment. In the assertions we model edges as binary toggles, so if the same edge occurs twice this is equivalent to it not appearing at all (*i.e.*, $\{(M,N)\} \otimes \{(M,N)\} \simeq \mathbf{1}$). The parameters are extended by adding conditions corresponding to whether an edge is present or not, and the assertions are finite multisets.

Notation	*SML*	PWB
$\mathbf{C} \triangleq \cdots \cup \{\mathsf{conn}(M,N), \mathsf{disconn}(M,N) : M,N \in \mathbf{T}\}$ $\mathbf{A} \triangleq \mathbf{T} \times \mathbf{T} \rightarrow_{\text{fin}} \mathbb{N}$ $\mathbf{1} \triangleq \emptyset$	**datatype** condition = ... | Conn **of** term*term | Disconn **of** term*term **datatype** assertion = Top **of** (term*term)**list** **val** unit = Top []	$\varphi ::= \ldots \mid$ conn(M,N) | disconn(M,N) $\Psi ::= \epsilon$ $\mid \ (\mathbf{M},\mathbf{N})(,(\mathbf{M},\mathbf{N}))^*$

An odd number of edge tuples in the environment denote that the edge is present; an even number denotes absence. Thus adding a tuple to the environment might add or remove an edge. We capture this with the following entailment definition

$$\begin{array}{ll} \Psi \vdash \mathsf{conn}(M,N) & \text{iff } M,N \in \mathbb{N} \text{ and } |\Psi(M,N)| + |\Psi(N,M)| \text{ is odd} \\ \Psi \vdash \mathsf{disconn}(M,N) & \text{iff } M,N \in \mathbb{N} \text{ and } \Psi \not\vdash \mathsf{conn}(M,N) \\ \Psi \vdash \mathsf{init}(M) \dot{\succ} \mathsf{init}(N) & \text{iff } \mathsf{conn}(M,N) \end{array}$$

For the protocol in Section 5 we may reuse the same constraint solver, keeping in mind that it does not handle the case where a disconn condition guards a broadcast input. We can also express the alteration of the topology with the following two agents:

```
Connect(a,b) <=                          | Disconnect(a,b) <=
case "conn(a,b)"    :*tau*.0             | case "conn(a,b)"    :*tau*.(|"(a,b)"|)
  [] "disconn(a,b)":*tau*.(|"(a,b)"|) ;  |   [] "disconn(a,b)":*tau*.0 ;
```

The agent Disconnect<1, 2> | (|"(1,2)"|) has two transitions: first (|"(1,2)" |)|(| "(1,2)"|) with trivially solvable constraint {|"(1,2)" |– "conn(1,2)"|}, and second 0 | (|"(1,2)"|) with the solution ([], "(1,2)"). In both transitions, the environment was extended with an extra tuple $(1,2)$, effectively removing an edge from the topology. Intuitively, the agents Connect and Disconnect can be used to set and unset bits in a global table.

7. SYMBOLIC SEMANTICS

In this section we describe a symbolic operational semantics for broadcast psi-calculi, that is sound (Theorem 7.11) and complete (Theorem 7.12) with respect to the concrete broadcast semantics [Borgström et al. 2011; Borgström et al. 2013]. This semantics is the one that is implemented in the PWB, and it extends, simplifies, and corrects the original symbolic semantics [Johansson et al. 2012].

7.1. Symbolic Operational Semantics

As we have seen, transitions in the symbolic operational semantics are of the form $P \xrightarrow[C]{\alpha} Q$, where C is a constraint that needs to be satisfied for the transition to be enabled. Each PWB instance implements a solver, that computes solutions for the transition constraints of that instance.

Definition 7.1 (*Constraints and Solutions*). A *solution* is a pair (σ, Ψ) where σ is a substitution sequence of terms for names, and Ψ is an assertion. The *transition constraints*, ranged over by C, C_t, and their corresponding solutions $\mathrm{sol}(C)$ are defined by:

	Constraint	Solutions
$C, C' ::=$	$\mathbf{true}$	$\{(\sigma,\Psi) \;:\; \sigma \text{ is a subst. sequence } \wedge\ \Psi \in \mathbf{A}\}$
$\mid$	$\mathbf{false}$	$\emptyset$
$\mid$	$(\nu a)C$	$\{(\sigma,\Psi) \;:\; b\#\sigma,\Psi,C \wedge (\sigma,\Psi) \in \mathrm{sol}((a\ b)\cdot C)\}$
$\mid$	$\{\!\!\{\Psi' \vdash \varphi\}\!\!\}$	$\{(\sigma,\Psi) \;:\; \Psi'\sigma \otimes \Psi \vdash \varphi\sigma\}$
$\mid$	$\exists x.C$	$\{(\sigma,\Psi) \;:\; y\#\sigma,\Psi,C \wedge ([y:=M]\sigma,\Psi) \in \mathrm{sol}((x\ y)\cdot C)\}$
$\mid$	$a \in \mathsf{n}(M)$	$\{(\sigma,\Psi) \;:\; a \in \mathsf{n}(M\sigma)\}$
$\mid$	$C \wedge C'$	$\mathrm{sol}(C) \cap \mathrm{sol}(C')$

Above, $(a\ b)\cdot C$ stands for the simultaneous replacement of a for b and b for a in C ("swapping"). In $(\nu a)C$, a is binding into C; and in $\exists x.C$, x is binding into C. We write $\exists^b x.C$ for $(\nu b)\exists x.(b \in \mathsf{n}(x) \wedge C)$; the only uses of $\exists$ and $\cdot \in \mathsf{n}(\cdot)$ will be in this restricted form (which is itself only used in Rule SBRCLOSE in Table I). We adopt the notation $(\sigma,\Psi) \models C$ to say that $(\sigma,\Psi) \in \mathrm{sol}(C)$, and write $C \leftrightarrow D$ to say that $\mathrm{sol}(C) = \mathrm{sol}(D)$.

A transition constraint C defines a set of solutions $\mathrm{sol}(C)$, namely those where the formula becomes true by applying the substitution and adding the assertion. For example, the transition constraint $\{\!\!\{\mathbf{1} \vdash x = 3\}\!\!\}$ has solutions $([x := 3], \mathbf{1})$ and $([],\ x = 3)$, where $[]$ is the identity substitution.

Restriction distributes over logical conjunction, and logical conjunction has $\mathbf{true}$ as unit and is associative. We thus consider constraints modulo the equations below.

LEMMA 7.2. $(\nu a)(C_1 \wedge C_2) \leftrightarrow (\nu a)C_1 \wedge (\nu a)C_2$ *and* $C_1 \wedge (C_2 \wedge C_3) \leftrightarrow (C_1 \wedge C_2) \wedge C_3$ *and* $C \wedge \mathbf{true} \leftrightarrow C$.

The concept of *frame of an agent* $\mathcal{F}(P)$ is used in the semantics: intuitively it is the top-level assertions of an agent, including the top-level binders. Frames are of the form $F ::= \Psi \mid (\nu\, a)\Psi$ where a is bound in $(\nu\, a)\Psi$. The frame of a process denotes its contribution to parallel agents. For example, the frame $\mathcal{F}((\nu a)(\{\!|\Psi_1|\!\} \mid \overline{M}\,\widetilde{N}\,.\,(\{\!|\Psi_3|\!\} \mid \{\!|\Psi_2|\!\})))$ is $(\nu a)(\Psi_1 \otimes \Psi_2)$. Note that Ψ_3 is not included in the frame, since it occurs under a prefix. In order to define the symbolic operational semantics, we need a way to add the frame of a parallel process to the current transition constraint.

Definition 7.3 (*Adding frames to constraints*). We define $F \otimes C$ as follows:

$$\begin{array}{rcll} F \otimes (\nu a)C & = & (\nu a)(F \otimes C) & \text{where } a\#F \\ (\nu\widetilde{a})\Psi \otimes \{\!|\Psi' \vdash \varphi|\!\} & = & (\nu\widetilde{a})\{\!|\Psi \otimes \Psi' \vdash \varphi|\!\} & \text{where } \widetilde{a}\#\Psi', \varphi \\ (\nu\widetilde{a})\Psi \otimes \exists x.C & = & (\nu\widetilde{a})\exists x.(\Psi \otimes C) & \text{where } \widetilde{a}\#C \text{ and } x\#\widetilde{a}, \Psi \\ F \otimes (C \wedge D) & = & (F \otimes C) \wedge (F \otimes D) & \\ F \otimes C & = & C & \text{otherwise.} \end{array}$$

For the symbolic semantics to be able to pick out the original channel to be used to send a message, we require partial injectivity of channel connectivity in its left argument: we require that for all names a, the function $x \mapsto (x \leftrightarrow a)$ is injective.

A process P is said to be assertion guarded if every occurrence of a $(\!|\Psi|\!)$ in P is a subterm of an input or an output. We require that processes are well-formed: P is well-formed if in every subterm of P of the form $\mathbf{case}\ \widetilde{\varphi} : \widetilde{Q}$ every Q_i is assertion guarded, and in every subterm of P of the form $!Q$ we have that Q is assertion guarded.

We let the subject (or channel) of an action α be $\mathrm{subj}(\underline{x}?(\widetilde{y})) = \mathrm{subj}(\underline{x}(\widetilde{y})) = \mathrm{subj}(\overline{x}!\ (\nu\widetilde{a})\widetilde{N}) = \mathrm{subj}(\overline{x}\ (\nu\widetilde{a})\widetilde{N}) = x$ and $\mathrm{subj}(\tau) = \emptyset$. We also define the bound names (i.e., the private names) of a label as $\mathrm{bn}(\underline{x}?(\widetilde{y})) = \mathrm{bn}(\underline{x}(\widetilde{y})) = \widetilde{y}$ and $\mathrm{bn}(\overline{x}!\ (\nu\widetilde{a})\widetilde{N}) = \mathrm{bn}(\overline{x}\ (\nu\widetilde{a})\widetilde{N}) = \widetilde{a}$ and $\mathrm{bn}(\tau) = \emptyset$.

The structured symbolic operational semantics preserves well-formedness, and is defined in Tables I, II and III. We first describe the broadcast rules in Table I. First consider the sBrOut rule: $\overline{M}\ \widetilde{N}.P \xrightarrow[\{\!|\mathbf{1} \vdash M \dot{\prec} y|\!\}]{\overline{y}!\ \widetilde{N}} P$. The solutions to its transition constraint are those that enable the subject M of the output prefix to broadcast on the fresh channel variable y. Similarly, in sBrIn we can receive a broadcast from any channel x that the subject M of the input prefix can listen to. In sBrMerge, two inputs with the same labels are merged into one. In sBrCom, a broadcast of P is received by Q, substituting the message $\widetilde{N}$ for the input variables $\widetilde{y}$. The names $\widetilde{a}$ are restricted in P, so they must be fresh for Q. In both sBrMerge and sBrCom, each transition constraint is extended with the frame of the other process. In sBrOpen, the scope of the new name b that occurs in the message $\widetilde{N}$ is opened; we remember in the transition constraint that b is fresh. In sBrClose, a broadcast that has reached its lexical scope turns into an internal τ action. The scoping of the new names $\widetilde{a}$ is reestablished.

The other symbolic rules in Tables II and III are similar to the broadcast rules, with two exceptions. In Rule sCase in Table III we add the constraint that φ_i must hold to the transition constraint. In Rule sCom in Table II we partially deconstruct the transition constraints of the input and the output transition, picking out the first conjunct. We then recombine the remainder of the transition constraints, adding the constraint that their channels are equivalent (i.e., $\Psi_1 \otimes \Psi_2 \vdash M_1 \leftrightarrow M_2$), yielding C_{com}. Here the partial injectivity of $\leftrightarrow$ is used to guarantee that M_1 is the channel that originated the transition.

7.2. Comparison with the Original Symbolic Operational Semantics

The symbolic semantics used in this paper differs from the original semantics [Johansson et al. 2012] in four significant ways:

(1) support for broadcast communication (Table I);
(2) support for polyadic communication (sending of multiple message terms at once);
(3) no dependence on an external assertion environment ($\Psi \rhd$ below); and
(4) a new sCom rule, for reasons explained below.

Table I: Symbolic transition rules for broadcast communication. A symmetric version of sBrCom is elided. In sBrOpen the expression $\nu\tilde{a} \cup \{b\}$ means the sequence $\tilde{a}$ with b inserted anywhere.

$$\textsc{sBrOut}\ \frac{x\#\Psi, M, \widetilde{N}, P}{\overline{M}\,\widetilde{N}\,.\,P \xrightarrow[\mathbf{1}\vdash M \dot{\prec} x]{\overline{x}!\ \widetilde{N}} P} \qquad \textsc{sBrIn}\ \frac{x, \widetilde{y}\#\Psi, M, P \quad x\#\widetilde{y}}{\underline{M}(\widetilde{y})\,.\,P \xrightarrow[\mathbf{1}\vdash x \dot{\succ} M]{\underline{x}?(\widetilde{y})} P}$$

$$\textsc{sBrMerge}\ \frac{P \xrightarrow[C_1]{\underline{x}?(\widetilde{y})} P' \qquad Q \xrightarrow[C_2]{\underline{x}?(\widetilde{y})} Q'}{P \mid Q \xrightarrow[(\mathcal{F}(Q)\otimes C_1)\wedge(\mathcal{F}(P)\otimes C_2)]{\underline{x}?(\widetilde{y})} P' \mid Q'}$$

$$\textsc{sBrCom}\ \frac{P \xrightarrow[C_1]{\overline{x}!\ (\nu\widetilde{a})\widetilde{N}} P' \qquad Q \xrightarrow[C_2]{\underline{x}?(\widetilde{y})} Q'}{P \mid Q \xrightarrow[(\mathcal{F}(Q)\otimes C_1)\wedge(\mathcal{F}(P)\otimes C_2)]{\overline{x}!\ (\nu\widetilde{a})\widetilde{N}} P' \mid Q'[\widetilde{y} := \widetilde{N}]}\ \begin{array}{l}|\widetilde{y}| = |\widetilde{N}| \\ \widetilde{a}\#Q\end{array}$$

$$\textsc{sBrOpen}\ \frac{P \xrightarrow[C]{\overline{y}!\ (\nu\widetilde{a})\widetilde{N}} P'}{(\nu b)P \xrightarrow[(\nu b)C]{\overline{y}!\ (\nu\widetilde{a}\cup\{b\})\widetilde{N}} P'}\ \begin{array}{l} b \in \mathsf{n}(\widetilde{N}) \\ b\#\widetilde{a}, y\end{array} \qquad \textsc{sBrClose}\ \frac{P \xrightarrow[C]{\overline{x}!\ (\nu\widetilde{a})\widetilde{N}} P'}{(\nu b)P \xrightarrow[\exists^b x.C]{\tau} (\nu b)(\nu\widetilde{a})P'}$$

Table II: Revised symbolic transition rules for binary communication. The symmetric version of sCom is elided. In sCom, we assume that $\widetilde{c_1}\#y, \widetilde{c_2}, \Psi_2, M_2$ and $\widetilde{c_2}\#z, \Psi_1, M_1$ and let $C_{\text{com}} = ((\nu\widetilde{c_1}\widetilde{c_2})\{\!|\Psi_1 \otimes \Psi_2 \vdash M_1 \leftrightarrow M_2|\!\}) \wedge (((\nu\widetilde{c_2})\Psi_2) \otimes C_1) \wedge (((\nu\widetilde{c_1})\Psi_1) \otimes C_2)$. In sOpen the expression $\nu\tilde{a} \cup \{b\}$ means the sequence $\tilde{a}$ with b inserted anywhere.

$$\textsc{sOut}\ \frac{y\#M, \widetilde{N}, P}{\overline{M}\,\widetilde{N}\,.\,P \xrightarrow[\{\!|\mathbf{1}\vdash M \dot{\leftrightarrow} y|\!\}]{\overline{y}\widetilde{N}} P} \qquad \textsc{sIn}\ \frac{y\#M, P, \widetilde{x}}{\underline{M}(\widetilde{x})\,.\,P \xrightarrow[\{\!|\mathbf{1}\vdash M \dot{\leftrightarrow} y|\!\}]{\underline{y}(\widetilde{x})} P}$$

$$\textsc{sCom}\ \frac{\begin{array}{c}P \xrightarrow[(\nu\widetilde{c_1})\{\!|\Psi_1\vdash M_1 \dot{\leftrightarrow} y|\!\}\wedge C_1]{\overline{y}\ (\nu\widetilde{a})\widetilde{N}} P' \\ Q \xrightarrow[(\nu\widetilde{c_2})\{\!|\Psi_2\vdash M_2 \dot{\leftrightarrow} z|\!\}\wedge C_2]{\underline{z}(\widetilde{x})} Q'\end{array}}{P \mid Q \xrightarrow[C_{\text{com}}]{\tau} (\nu\widetilde{a})(P' \mid Q'[\widetilde{x} := \widetilde{N}])}\ \begin{array}{l}|\widetilde{x}| = |\widetilde{N}| \\ \widetilde{a}\#Q\end{array} \qquad \textsc{sOpen}\ \frac{P \xrightarrow[C]{\overline{y}\ (\nu\widetilde{a})\widetilde{N}} P'}{(\nu b)P \xrightarrow[(\nu b)C]{\overline{y}\ (\nu\widetilde{a}\cup\{b\})\widetilde{N}} P'}\ \begin{array}{l} b \in \mathsf{n}(\widetilde{N}) \\ b\#\widetilde{a}, y\end{array}$$

The original version of the communication rule was as follows (omitting its freshness side conditions). Below, the assertion environment "$\ldots \Psi \rhd$ " collects the assertions of the context of the current process, and can be ignored.

$$\textsc{Old-sCom}\ \frac{\begin{array}{c}\Psi_Q \otimes \Psi \rhd P \xrightarrow[(\nu\widetilde{b_P})\{\!|\Psi_1\vdash M_1 \dot{\leftrightarrow} y|\!\}\wedge C_1]{\overline{y}\ (\nu\widetilde{a})\widetilde{N}} P' \\ \Psi_P \otimes \Psi \rhd Q \xrightarrow[(\nu\widetilde{b_Q})\{\!|\Psi_2\vdash M_2 \dot{\leftrightarrow} z|\!\}\wedge C_2]{\underline{z}(\widetilde{x})} Q'\end{array}}{\Psi \rhd P \mid Q \xrightarrow[C_{\text{old}}]{\tau} (\nu\widetilde{a})(P' \mid Q'[\widetilde{x} := \widetilde{N}])}\ \begin{array}{c}|\widetilde{x}| = |\widetilde{N}| \\ \mathcal{F}(P) = (\nu\widetilde{b_P})\Psi_P \\ \mathcal{F}(Q) = (\nu\widetilde{b_Q})\Psi_Q \\ \Psi_1 = \Psi_2 = \Psi \otimes \Psi_P \otimes \Psi_Q\end{array}$$

In order to derive a transition with Old-sCom, we need to compute the frames of P and Q, equate the bound names in the frames with the ones appearing in the transition constraints such that $\mathcal{F}(P) = (\nu\widetilde{b_P})\Psi_P$ and $\mathcal{F}(Q) = (\nu\widetilde{b_Q})\Psi_Q$, and then check that $\Psi_1 =$

Table III: Revised symbolic transition rules common to broadcast and binary communication. A symmetric version of sPAR is elided.

$$\textsc{sCase}\ \frac{P_i \xrightarrow[C]{\alpha} P'}{\mathbf{case}\ \widetilde{\varphi} : \widetilde{P} \xrightarrow[C \wedge \{\!|\mathbf{1} \vdash \varphi_i|\!\}]{\alpha} P'}\ \mathrm{subj}(\alpha)\#\varphi_i \qquad \textsc{sRep}\ \frac{P \mid !P \xrightarrow[C]{\alpha} P'}{!P \xrightarrow[C]{\alpha} P'}$$

$$\textsc{sPar}\ \frac{P \xrightarrow[C]{\alpha} P'}{P \mid Q \xrightarrow[\mathcal{F}(Q)\otimes C]{\alpha} P' \mid Q}\ \mathrm{bn}(\alpha)\#Q \quad \alpha = \tau \vee \mathrm{subj}(\alpha)\#Q \qquad \textsc{sScope}\ \frac{P \xrightarrow[C]{\alpha} P'}{(\nu b)P \xrightarrow[(\nu b)C]{\alpha} (\nu b)P'}\ b\#\alpha$$

$$\textsc{sInv}\ \frac{P[\widetilde{x} := \widetilde{M}] \xrightarrow[C]{\alpha} P'}{\mathrm{A}\langle \tilde{M} \rangle \xrightarrow[C]{\alpha} P'}\ \mathrm{A}\langle \tilde{x} \rangle \Leftarrow P \quad |\widetilde{x}| = |\widetilde{M}|$$

$\Psi_2 = \Psi \otimes \Psi_P \otimes \Psi_Q$. However, these equalities fail in certain situations where we would expect them to hold.

Example 7.4. This example shows issues related to restrictions under process constructors **case** and replication (!). We use replication as an example; the issue when using **case** is analogous. Consider the process $P = !(\nu\, b)\overline{c}\, b.Q$ in the pi-calculus instance. In the original semantics, the symbolic output transition of P has the constraint $(\nu\, b)\{\!|\mathbf{1} \vdash c \leftrightarrows x|\!\}$ since the frame of $(\nu\, b)\overline{c}\, b.Q$ (which is $(\nu\, b)\mathbf{1}$) is used in the derivation. When attempting to derive a communication between P and the process $\underline{c}(x).R$, the side condition $\mathcal{F}(P) = (\nu\widetilde{b}_P)\Psi_P$ of OLD-sCOM is impossible to satisfy: $\mathcal{F}(P) = (\nu\varepsilon)\mathbf{1}$ while the transition constraint of P is $(\nu\, b)\{\!|\mathbf{1} \vdash c \leftrightarrows x|\!\}$, and the number of bound names thus differ.

A similar issue, related to the ordering of restrictions in the frame, applies when an inactive parallel process has top-level restrictions.

Example 7.5. Let $P = (\nu\, b)\overline{c}\, b.Q \mid (\nu\, a)\underline{c}(x).R$. In the original semantics, the symbolic output transition of P has the constraint $(\nu\, a)(\nu\, b)\{\!|\mathbf{1} \vdash c \leftrightarrows x|\!\}$ but $\mathcal{F}(P) = (\nu\, b)(\nu\, a)\mathbf{1}$ where the order of the bound names is different.

Both these issues could be avoided if the binders of frames were so-called `set+` binders [Huffman and Urban 2010] where order does not matter and redundant binders are ignored. However, such a notion of binders is not available in the version of Nominal Isabelle [Urban and Tasson 2005] that is used for the formalization of psi-calculi [Bengtson and Parrow 2009].

Example 7.6. This example show issues related to situations where assertion composition is non-commutative. Let the assertions be tuples $\widetilde{a}$ of names, composed using concatenation $\widetilde{a};\widetilde{b}$. Consider the premises of OLD-sCOM: in the original semantics Ψ_1 will have a prefix $\Psi_Q;\Psi$ and Ψ_2 will have a prefix $\Psi_P;\Psi$. Since concatenation is non-commutative, the side condition $\Psi_1 = \Psi_2 = \Psi \otimes \Psi_P \otimes \Psi_Q$ of OLD-COM cannot hold if Ψ_P and Ψ_Q are non-empty and $\mathrm{n}(\Psi_P) \neq \mathrm{n}(\Psi_Q)$. This makes it impossible for the two processes $(\!|a|\!) \mid c$ and $(\!|b|\!) \mid \overline{c}$ to communicate using OLD-sCOM.

These examples show that Rule OLD-sCOM makes too strong assumptions on the syntactic form of the constraints of the transitions in its premise. The original symbolic semantics still corresponds to the concrete semantics [Bengtson et al. 2011] in certain instances, such as when communicating processes do not contain restrictions and assertion composition satisfies the commutative monoid laws (not only modulo assertion equivalence). In contrast to OLD-sCOM, Rule sCOM in Table II does not make

Table IV: General requirements on substitution

$$
\begin{array}{rcll}
X[x := x] & = & X & \\
x[x := M] & = & M & \\
X[x := M] & = & X & \text{if } x\#X \\
X[x := L][y := M] & = & X[y := M][x := L] & \text{if } x\#y, M \text{ and } y\#L \\
X[\tilde{x} := \tilde{T}] & = & ((\tilde{y}\ \tilde{x}) \cdot X)[\tilde{y} := \tilde{T}] & \text{if } \tilde{y}\#X, \tilde{x}
\end{array}
$$

Table V: Requirements for specific data types

$$
\begin{array}{rcl}
\mathrm{n}(M\sigma) & \supseteq & \mathrm{n}(M) \setminus \mathrm{n}(\sigma) \\
\mathrm{n}(M[\tilde{a} := \tilde{L}]) & \supseteq & \mathrm{n}(\tilde{L}) \text{ when } \mathrm{n}(M) \supseteq \tilde{a} \\
(M \mathrel{\dot{\prec}} N)\sigma & = & M\sigma \mathrel{\dot{\prec}} N\sigma \\
(N \mathrel{\dot{\succ}} M)\sigma & = & N\sigma \mathrel{\dot{\succ}} M\sigma \\
(M \leftrightarrow N)\sigma & = & M\sigma \leftrightarrow N\sigma \\
\mathbf{1}\sigma & = & \mathbf{1}
\end{array}
\qquad
\begin{array}{rcl}
\Psi \otimes \mathbf{1} & \simeq_{\mathcal{N}} & \Psi \\
\Psi \otimes \Psi' & \simeq_{\mathcal{N}} & \Psi' \otimes \Psi \\
\Psi_1 \otimes (\Psi_2 \otimes \Psi_3) & \simeq_{\mathcal{N}} & (\Psi_1 \otimes \Psi_2) \otimes \Psi_3 \\
(\Psi \otimes \Psi')\sigma & \simeq_{\mathcal{N}} & \Psi\sigma \otimes \Psi'\sigma \\
\Psi \otimes \Psi_1 & \simeq_{\mathcal{N}} & \Psi \otimes \Psi_2 \text{ when } \Psi_1 \simeq_{\mathcal{N}} \Psi_2
\end{array}
$$

any assumptions about the number of bound names nor on the structure of the assertion, and the corresponding broadcast rules sBrCom and sBrMerge in Table I do not make any assumptions at all about the form of their constraints.

7.3. Correctness of the Symbolic Operational Semantics

The proofs for the soundness and completeness of the symbolic semantics with respect to the concrete broadcast semantics [Borgström et al. 2011] mainly follow [Johansson et al. 2012]. The main exception is that their counterpart of Lemma 7.10, which describes the shape of transition constraints, does not hold in all cases, as seen in Examples 7.4, 7.5 and 7.6. We here instead prove a weaker result by considering assertions and frames modulo redundant restrictions (cf. Example 7.4), restriction ordering (cf. Example 7.5) and commutative monoid laws for assertion composition (cf. Example 7.6).

As to technical preliminaries, we assume the general properties of substitution in Table IV, and the homomorphism and name preservation laws in Table V. As an example, the standard notion of substitution in (nominal) term algebras satisfies all of these properties. We write $\Psi \simeq_{\mathcal{N}} \Psi'$ iff $\mathrm{n}(\Psi) = \mathrm{n}(\Psi')$ and for all φ it holds that $\Psi \vdash \varphi$ iff $\Psi' \vdash \varphi$. We then assume the equivalences in Table V. As an example, they are satisfied when assertions are finite sets of equations on terms, with standard substitution.

The main difference to the original proofs is the introduction of an auxiliary relation on frames (Definition 7.7) in order to accurately describe the shape of transition constraints (Lemma 7.10) such that they can always be decomposed in Rule sCom, unlike the case for Old-sCom.

Definition 7.7 (*AC-equivalence*). Associative/Commutative equivalence (AC-equivalence) of assertions is the smallest equivalence relation such that

(1) $\mathbf{1} \otimes \Psi \equiv_{\mathrm{AC}} \Psi$; and
(2) $\Psi_1 \otimes \Psi_2 \equiv_{\mathrm{AC}} \Psi_2 \otimes \Psi_1$; and
(3) $\Psi_1 \otimes (\Psi_2 \otimes \Psi_3) \equiv_{\mathrm{AC}} (\Psi_1 \otimes \Psi_2) \otimes \Psi_3$; and
(4) $\Psi_1 \equiv_{\mathrm{AC}} \Psi_2 \implies \Psi \otimes \Psi_1 \equiv_{\mathrm{AC}} \Psi \otimes \Psi_2$.

Frames $(\nu\widetilde{a})\Psi_1$ and $(\nu\widetilde{c})\Psi_2$ are AC-equivalent, written $(\nu\widetilde{a})\Psi_1 \equiv_{\mathrm{AC}} (\nu\widetilde{c})\Psi_2$, if $\Psi_1 \equiv_{\mathrm{AC}} \Psi_2$ and $\{\widetilde{a}\} \cap \mathrm{n}(\Psi_1) = \{\widetilde{c}\} \cap \mathrm{n}(\Psi_2)$.

Lemma 7.8. *AC-equivalence is an equivalence relation on frames, and whenever* $F_1 \equiv_{\mathrm{AC}} F_2$ *we also have* $\mathrm{n}(F_1) = \mathrm{n}(F_2)$ *and* $(\nu a)F_1 \equiv_{\mathrm{AC}} (\nu a)F_2$ *and* $G \otimes F_1 \equiv_{\mathrm{AC}} G \otimes F_2$.

Proof. Straightforward from the definitions, using the laws in Table V. □

As an example, guarded processes have frames that are AC-equivalent to the unit frame $\mathbf{1}$.

LEMMA 7.9. *If P is assertion guarded, then $\mathcal{F}(P) \equiv_{\mathrm{AC}} \mathbf{1}$.*

PROOF. By induction on P. □

The following lemma characterises the shape of the constraints of point-to-point input and output transitions. The first conjunct in the constraint is always a channel equivalence constraint (between the object M of the original prefix and the transition object variable y) that must hold under a frame $(\nu\widetilde{c})\Psi$ that is AC-equivalent to that of the original process P. The lemma is used in the proof of Theorem 7.12 to show that the precondition on the shape of the transitions in Rule SCOM always holds.

LEMMA 7.10 (FORM OF CONSTRAINT). *Let $\alpha = \overline{y}\,(\nu\widetilde{a})\widetilde{N}$ or $\alpha = \underline{y}(\widetilde{x})$. If $P \xrightarrow[C]{\alpha} P'$ and $y\#P$ then there exist $\widetilde{c}, \Psi, M$ and D such that $\mathcal{F}(P) \equiv_{\mathrm{AC}} (\nu\widetilde{c})\Psi$ and $y\#\widetilde{c}, \Psi, M, D$ and $C = (\nu\widetilde{c})\{\!|\Psi \vdash M \leftrightarrow y|\!\} \wedge D$.*

PROOF. By induction on the derivation of $P \xrightarrow[C]{\alpha} P'$. A base case is as follows.

SOUT. In this case the transition is derived by

$$\textsc{sOut}\ \frac{}{\overline{K}\widetilde{N}\,.\,P \xrightarrow[\{\!|\mathbf{1}\vdash K\leftrightarrow y|\!\}]{\overline{y}\widetilde{N}} P}\ y\#K, \widetilde{N}, P$$

Here $\widetilde{c} = \epsilon$, $\Psi = \mathbf{1}$, $M = K$, and $D = \mathbf{true}$, where $\mathcal{F}(\overline{K}\,\widetilde{N}\,.\,P) = \mathbf{1}$.

The cases that require the use of AC-equivalence are the following.

SCASE. In this case the transition is derived by

$$\textsc{sCase}\ \frac{P_i \xrightarrow[C]{\alpha} P'}{\mathbf{case}\ \widetilde{\varphi} : \widetilde{P} \xrightarrow[C\wedge\{\!|\mathbf{1}\vdash\varphi_i|\!\}]{\alpha} P'}\ \mathrm{bn}(\alpha)\#\varphi_i$$

By induction we get $M, D', \Psi, \widetilde{c}$ such that $C = (\nu\widetilde{c})\{\!|\Psi \vdash M \leftrightarrow y|\!\} \wedge D'$ with $y\#D'$ and $\mathcal{F}(P_i) \equiv_{\mathrm{AC}} (\nu\widetilde{c})\Psi$. Let $D = D' \wedge \{\!|\mathbf{1} \vdash \varphi_i|\!\}$; since $y\#\mathbf{case}\ \widetilde{\varphi} : \widetilde{P}$ we also have that $y\#D$. By well-formedness, P_i is guarded, so by Lemma 7.9 $\mathcal{F}(P_i) \equiv_{\mathrm{AC}} \mathbf{1}$. By transitivity $\mathcal{F}(P) = \mathbf{1} \equiv_{\mathrm{AC}} (\nu\widetilde{c})\Psi$.

SPAR. In this case the transition is derived by

$$\textsc{sPar}\ \frac{P \xrightarrow[C]{\alpha} P'}{P \mid Q \xrightarrow[\mathcal{F}(Q)\otimes C]{\alpha} P' \mid Q}\ \begin{array}{l}\mathrm{bn}(\alpha)\#Q \\ \alpha = \tau \vee \mathrm{subj}(\alpha)\#Q\end{array}$$

By induction there are $M, D', \Psi, \widetilde{c}$ such that $C = (\nu\widetilde{c})\{\!|\Psi \vdash M \leftrightarrow y|\!\} \wedge D'$ with $y\#D'$ and $\mathcal{F}(P) \equiv_{\mathrm{AC}} (\nu\widetilde{c})\Psi$. Let $D = \mathcal{F}(Q) \otimes D'$; since $y\#P|Q$ we also have that $y\#D$. By Lemma 7.8 $\mathcal{F}(P \mid Q) \equiv_{\mathrm{AC}} ((\nu\widetilde{c})\Psi) \otimes \mathcal{F}(Q) \equiv_{\mathrm{AC}} \mathcal{F}(Q) \otimes (\nu\widetilde{c})\Psi$.

SSCOPE. In this case the transition is derived by

$$\textsc{sScope}\ \frac{P \xrightarrow[C]{\alpha} P'}{(\nu b)P \xrightarrow[(\nu b)C]{\alpha} (\nu b)P'}\ b\#\alpha$$

By induction there exist $\widetilde{c}, \Psi, M$ and D' such that $C = (\nu\widetilde{c})(\Psi \vdash M \leftrightarrows y) \wedge D'$ with $y\#M, D$ and $\mathcal{F}(P) \equiv_{\text{AC}} (\nu\widetilde{c})\Psi$. Let $D = (\nu b)D'$; a fortiori $y\#(\nu b)D$. By Lemma 7.8 $\mathcal{F}((\nu b)P) \equiv_{\text{AC}} (\nu b)(\nu\widetilde{c})\Psi$.
sOPEN. As sSCOPE.
sREP. In this case the transition is derived by

$$\textsc{sRep}\ \frac{P \mid !P \xrightarrow[C]{\alpha} P'}{!P \xrightarrow[C]{\alpha} P'}$$

By induction there exist $\widetilde{c}, \Psi, M$ and D such that $C = (\nu\widetilde{c})(\Psi \vdash M \leftrightarrows y) \wedge D$ with $y\#M, D$ and $\mathcal{F}(P|!P) \equiv_{\text{AC}} (\nu\widetilde{c})\Psi$. By well-formedness, P is guarded, so by Lemma 7.9 $\mathcal{F}(P|!P) \equiv_{\text{AC}} \mathbf{1}$. By transitivity $\mathcal{F}(!P) = \mathbf{1} \equiv_{\text{AC}} (\nu\widetilde{c})\Psi$. □

We prove soundness and completeness of the symbolic semantics of this paper with respect to a polyadic version of the concrete semantics of broadcast psi-calculi [Borgström et al. 2011], which we show in Table VI.

The soundness theorem and its proof follow [Johansson et al. 2012], apart from the weaker preconditions of Rule sCOM (compared to OLD-sCOM), and the new cases for broadcast actions.

THEOREM 7.11 (SOUNDNESS OF SYMBOLIC TRANSITIONS).
If $P \xrightarrow[C]{\alpha} P'$ *and* $(\sigma, \Psi) \models C$ *and* $\text{bn}(\alpha)\#\sigma$ *then* $\Psi \rhd P\sigma \xrightarrow{\alpha\sigma} P'\sigma$.

PROOF. By induction on the inference of $P \xrightarrow[C]{\alpha} P'$. □

The proof of the completeness theorem follows [Johansson et al. 2012], apart from new cases for the broadcast rules. In the cCOM case of the proof, Lemma 7.10 is used to show that the symbolic transitions obtained from the induction hypothesis are of the right form to apply Rule sCOM.

THEOREM 7.12 (COMPLETENESS OF SYMBOLIC TRANSITIONS).

—*If* $\Psi \rhd P\sigma \xrightarrow{\tau} P'$ *then* $\exists C, Q\,.\ P \xrightarrow[C]{\tau} Q$, $Q\sigma = P'$ *and* $(\sigma, \Psi) \models C$.

—*If* $\Psi \rhd P\sigma \xrightarrow{\alpha} P'$, $\alpha \neq \tau$, $y\#P, \text{bn}(\alpha), \sigma$, *and* $\text{bn}(\alpha)\#\sigma, P$ *then* $\exists C, \alpha', Q.\ P \xrightarrow[C]{\alpha'} Q$, $Q\sigma = P'$, $\text{subj}(\alpha') = y$, $\alpha'\sigma' = \alpha$, *and* $(\sigma', \Psi) \models C$ *where* $\sigma' = \sigma[y := \text{subj}(\alpha)]$.

PROOF. By induction on the inference of $\Psi \rhd P\sigma \xrightarrow{\alpha} P'\sigma$. □

8. RELATED WORK

Our previous work [Borgström et al. 2011] presented the broadcast extension of psi-calculi, and a model of a routing protocol for ad-hoc networks. In the present paper we have given a corresponding symbolic semantics, and several new example models.

The precursors of the PWB are the Concurrency Workbench [Cleaveland et al. 1993] for CCS, and the Mobility Workbench [Victor and Moller 1994] for pi-calculus. The tool mCRL2 [Cranen et al. 2013] for ACP admits higher order sorted free algebras and equational logics. PAT3 [Liu et al. 2011] includes a CSP♯ [Sun et al. 2009] module where actions built over types like booleans, integers are extended with C♯ like programs. ProVerif [Blanchet 2011] is a verification tool for the applied pi-calculus [Abadi and Fournet 2001], an extension of the pi-calculus that is specialised for security protocol verification. The tool is parametric in a term language equipped with equations and

Table VI: Concrete semantics. Symmetric versions of cBrCom, cCom and cPar are elided. In Rules cBrCom and cBrMerge and cCom we assume that $\mathcal{F}(P) = (\nu\widetilde{b}_P)\Psi_P$ and $\mathcal{F}(Q) = (\nu\widetilde{b}_Q)\Psi_Q$ where $\widetilde{b}_P$ is fresh for $P, \widetilde{b}_Q, Q$ and Ψ, and that $\widetilde{b}_Q$ is fresh for $Q, \widetilde{b}_P, P$ and Ψ. In Rule cPar we assume that $\mathcal{F}(Q) = (\nu\widetilde{b}_Q)\Psi_Q$ where $\widetilde{b}_Q$ is fresh for Ψ, P and α. In Rules cOpen and cBrOpen the expression $\tilde{a} \cup \{b\}$ means the sequence $\tilde{a}$ with b inserted anywhere.

$$\textsc{cBrOut}\ \frac{\Psi \vdash M \dot{\prec} K}{\Psi \rhd \overline{M}!\widetilde{N}\,.\,P \xrightarrow{\overline{K}!\,\widetilde{N}} P} \qquad \textsc{cBrIn}\ \frac{\Psi \vdash K \dot{\succ} M \quad |\widetilde{x}| = |\widetilde{N}|}{\Psi \rhd \underline{M}?\,(\widetilde{x})\,.\,P \xrightarrow{\underline{K}?\,\widetilde{N}} P[\widetilde{x} := \widetilde{N}]}$$

$$\textsc{cBrMerge}\ \frac{\Psi_Q \otimes \Psi \rhd P \xrightarrow{\underline{K}?\,\widetilde{N}} P' \qquad \Psi_P \otimes \Psi \rhd Q \xrightarrow{\underline{K}?\,\widetilde{N}} Q'}{\Psi \rhd P \mid Q \xrightarrow{\underline{K}?\,\widetilde{N}} P' \mid Q'}$$

$$\textsc{cBrCom}\ \frac{\Psi_Q \otimes \Psi \rhd P \xrightarrow{\overline{K}!\,(\nu\widetilde{a})\widetilde{N}} P' \qquad \Psi_P \otimes \Psi \rhd Q \xrightarrow{\underline{K}?\,\widetilde{N}} Q'}{\Psi \rhd P \mid Q \xrightarrow{\overline{K}!\,(\nu\widetilde{a})\widetilde{N}} P' \mid Q'}\ \begin{array}{l}\widetilde{b}_P\widetilde{b}_Q\#K \\ \widetilde{a}\#Q\end{array}$$

$$\textsc{cBrOpen}\ \frac{\Psi \rhd P \xrightarrow{\overline{K}!\,(\nu\widetilde{a})\widetilde{N}} P'}{\Psi \rhd (\nu b)P \xrightarrow{\overline{K}!\,(\nu\widetilde{a}\cup\{b\})\widetilde{N}} P'}\ \begin{array}{l}b\#\widetilde{a}, \Psi, K \\ b \in \mathrm{n}(N)\end{array} \qquad \textsc{cBrClose}\ \frac{\Psi \rhd P \xrightarrow{\overline{K}!\,(\nu\widetilde{a})\widetilde{N}} P'}{\Psi \rhd (\nu b)P \xrightarrow{\tau} (\nu b)(\nu\widetilde{a})P'}\ \begin{array}{l}b \in \mathsf{n}(K) \\ b\#\Psi\end{array}$$

$$\textsc{cOut}\ \frac{\Psi \vdash M \leftrightarrow K}{\Psi \rhd \overline{M}\,\widetilde{N}\,.\,P \xrightarrow{\overline{K}\widetilde{N}} P} \qquad \textsc{cIn}\ \frac{\Psi \vdash M \leftrightarrow K \quad |\widetilde{x}| = |\widetilde{N}|}{\Psi \rhd \underline{M}(\widetilde{x})\,.\,P \xrightarrow{\underline{K}\,\widetilde{N}} P[\widetilde{x} := \widetilde{N}]}$$

$$\textsc{cCom}\ \frac{\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K \qquad \Psi_Q \otimes \Psi \rhd P \xrightarrow{\overline{M}\,(\nu\widetilde{a})\widetilde{N}} P' \qquad \Psi_P \otimes \Psi \rhd Q \xrightarrow{\underline{K}\,\widetilde{N}} Q' x}{\Psi \rhd P \mid Q \xrightarrow{\tau} (\nu\widetilde{a})(P' \mid Q')}\ \begin{array}{l}\widetilde{b}_P\#M \\ \widetilde{b}_Q\#K \\ \widetilde{a}\#Q\end{array}$$

$$\textsc{cOpen}\ \frac{\Psi \rhd P \xrightarrow{\overline{M}\,(\nu\widetilde{a})N} P'}{\Psi \rhd (\nu b)P \xrightarrow{\overline{M}\,(\nu\widetilde{a}\cup\{b\})N} P'}\ \begin{array}{l}b\#\widetilde{a}, \Psi, M \\ b \in \mathrm{n}(N)\end{array} \qquad \textsc{cCase}\ \frac{\Psi \rhd P_i \xrightarrow{\alpha} P' \qquad \Psi \vdash \varphi_i}{\Psi \rhd \mathbf{case}\ \widetilde{\varphi} : \widetilde{P} \xrightarrow{\alpha} P'}$$

$$\textsc{cRep}\ \frac{\Psi \rhd P \mid !P \xrightarrow{\alpha} P'}{\Psi \rhd !P \xrightarrow{\alpha} P'} \qquad \textsc{cPar}\ \frac{\Psi_Q \otimes \Psi \rhd P \xrightarrow{\alpha} P'}{\Psi \rhd P \mid Q \xrightarrow{\alpha} P' \mid Q}\ \mathrm{bn}(\alpha)\#Q$$

$$\textsc{cScope}\ \frac{\Psi \rhd P \xrightarrow{\alpha} P'}{\Psi \rhd (\nu b)P \xrightarrow{\alpha} (\nu b)P'}\ b\#\alpha, \Psi \qquad \textsc{cInv}\ \frac{\Psi \rhd P[\widetilde{x} := \widetilde{M}] \xrightarrow{\alpha} P'}{\Psi \rhd \mathrm{A}\langle\widetilde{M}\rangle \xrightarrow{\alpha} P'}\ \begin{array}{l}\mathrm{A}\langle\tilde{x}\rangle \Leftarrow P \\ |\widetilde{x}| = |\widetilde{M}|\end{array}$$

unidirectional rewrite rules, but works in a fixed logic (predicate logic with equality). ProVerif does not include a symbolic simulator or a general bisimulation checker.

Our symbolic semantics and bisimulation generation algorithm (slight variations of our previous work [Johansson et al. 2012]) are to a large extent based on the pioneering work by Hennessy and Lin [Hennessy and Lin 1995] for value-passing CCS, later specialised for the pi-calculus by Boreale and De Nicola [Boreale and De Nicola 1996] and independently by Lin [Lin 1996; Lin 2000].

9. FUTURE WORK

It would be interesting to investigate other notions of bisimulation for wireless communication [Merro 2007], including machine-checked proofs of their meta-theoretical properties. We have performed initial work [Åman Pohjola et al. 2013] on modelling discrete time, and are considering extensions to other quantitative aspects of wireless networks, including probabilities, distance, and energy.

ACKNOWLEDGMENTS

We thank the anonymous referees of ACSD 2013 for their comments on an earlier version of this paper.

REFERENCES

Martín Abadi and Cédric Fournet. 2001. Mobile Values, New Names, and Secure Communication. In *Proc. of POPL '01*. ACM Press, New York, NY, USA, 104–115. DOI:http://dx.doi.org/10.1145/373243.360213

Martín Abadi and Andrew D. Gordon. 1997. A Calculus for Cryptographic Protocols: The Spi Calculus. In *Fourth ACM Conference on Computer and Communications Security*. ACM Press, 36–47. DOI:http://dx.doi.org/10.1145/266420.266432

Johannes Åman Pohjola, Johannes Borgström, Joachim Parrow, Palle Raabjerg, and Ioana Rodhe. 2013. *Negative Premises in Applied Process Calculi*. Technical Report 2013-014. Dept of Information Technology, Uppsala University.

K. A. Bartlett, R. A. Scantlebury, and P. T. Wilkinson. 1969. A Note on Reliable Full-Duplex Transmission over Half-Duplex links. *Commun. ACM* 12, 5 (May 1969), 260–261. DOI:http://dx.doi.org/10.1145/362946.362970

Jesper Bengtson, Magnus Johansson, Joachim Parrow, and Björn Victor. 2011. Psi-calculi: A framework for mobile processes with nominal data and logic. *Logical Methods in Computer Science* 7, 1, Article 11 (2011), 44 pages. DOI:http://dx.doi.org/10.2168/LMCS-7(1:11)2011

Jesper Bengtson and Joachim Parrow. 2009. Psi-calculi in Isabelle. In *Proc. of TPHOLs 2009 (LNCS)*. Springer, 99–114. DOI:http://dx.doi.org/10.1007/978-3-642-03359-9_9

Bruno Blanchet. 2011. Using Horn Clauses for Analyzing Security Protocols. In *Formal Models and Techniques for Analyzing Security Protocols*, Véronique Cortier and Steve Kremer (Eds.). Vol. 5. IOS Press, 86–111. DOI:http://dx.doi.org/10.3233/978-1-60750-714-7-86

Michele Boreale and Rocco De Nicola. 1996. A Symbolic Semantics for the π-Calculus. *Information and Computation* 126, 1 (1996), 34–52. DOI:http://dx.doi.org/10.1006/inco.1996.0032

Johannes Borgström, Ramūnas Gutkovas, Ioana Rodhe, and Björn Victor. 2013. A Parametric Tool for Applied Process Calculi. In *Proc. of ACSD'13*. IEEE, Los Alamitos, CA, USA, 187–192. DOI:http://dx.doi.org/10.1109/ACSD.2013.22

Johannes Borgström, Shuqin Huang, Magnus Johansson, Palle Raabjerg, Björn Victor, Johannes Åman Pohjola, and Joachim Parrow. 2011. Broadcast Psi-calculi with an Application to Wireless Protocols. In *Proc. of SEFM '11 (LNCS)*. Springer, 74–89. DOI:http://dx.doi.org/10.1007/978-3-642-24690-6_7

Johannes Borgström, Shuqin Huang, Magnus Johansson, Palle Raabjerg, Björn Victor, Johannes Åman Pohjola, and Joachim Parrow. 2013. Broadcast Psi-calculi with an Application to Wireless Protocols. *Software and Systems Modeling* (2013). In press.

Maria Grazia Buscemi and Ugo Montanari. 2007. CC-Pi: A Constraint-Based Language for Specifying Service Level Agreements. In *Proceedings of ESOP 2007 (LNCS)*, Rocco De Nicola (Ed.), Vol. 4421. Springer, 18–32.

Marco Carbone and Sergio Maffeis. 2003. On the Expressive Power of Polyadic Synchronisation in π-calculus. *Nordic Journal of Computing* 10, 2 (2003), 70–98.

Rance Cleaveland, Joachim Parrow, and Bernhard Steffen. 1993. The Concurrency Workbench: a Semantics-Based Tool for the Verification of Concurrent Systems. *ACM Trans. Program. Lang. Syst.* 15, 1 (1993), 36–72. DOI:http://dx.doi.org/10.1145/151646.151648

Sjoerd Cranen, Jan Friso Groote, Jeroen J A Keiren, Frank P M Stappers, Erik P Vink, Wieger Wesselink, and Tim A C Willemse. 2013. An Overview of the mCRL2 Toolset and Its Recent Advances. In *Proc. of TACAS '13 (LNCS)*, Vol. 7795. Springer, 199–213. DOI:http://dx.doi.org/10.1007/978-3-642-36742-7_15

F. Ghassemi, W. Fokkink, and A. Movaghar. 2008. Restricted Broadcast Process Theory. In *Software Engineering and Formal Methods, 2008. SEFM '08. Sixth IEEE International Conference on*. 345–354. DOI:http://dx.doi.org/10.1109/SEFM.2008.25

Jens Chr. Godskesen. 2010. Observables for Mobile and Wireless Broadcasting Systems. In *Coordination Models and Languages*, Dave Clarke and Gul Agha (Eds.). LNCS, Vol. 6116. Springer, 1–15. DOI:http://dx.doi.org/10.1007/978-3-642-13414-2_1

Ramūnas Gutkovas and Johannes Borgström. 2013. The Psi-Calculi Workbench web page. (2013). http://www.it.uu.se/research/group/mobility/applied/psiworkbench

Matthew Hennessy and Huimin Lin. 1995. Symbolic Bisimulations. *Theoretical Computer Science* 138, 2 (1995), 353–389. DOI:http://dx.doi.org/10.1016/0304-3975(94)00172-F

Brian Huffman and Christian Urban. 2010. A New Foundation for Nominal Isabelle. In *Proc. of ITP'10*. Springer, 35–50. DOI:http://dx.doi.org/10.1007/978-3-642-14052-5_5

Magnus Johansson, Jesper Bengtson, Joachim Parrow, and Björn Victor. 2010. Weak Equivalences in Psi-calculi. In *Proc. of LICS 2010*. IEEE, 322–331. DOI:http://dx.doi.org/10.1109/LICS.2010.30

Magnus Johansson, Björn Victor, and Joachim Parrow. 2012. Computing strong and weak bisimulations for psi-calculi. *Journal of Logic and Algebraic Programming* 81, 3 (2012), 162–180. DOI:http://dx.doi.org/10.1016/j.jlap.2012.01.001

Huimin Lin. 1996. Symbolic Transition Graph with Assignment. In *Proc. of CONCUR '96 (LNCS)*, Vol. 1119. Springer, 50–65. DOI:http://dx.doi.org/10.1007/3-540-61604-7_47

Huimin Lin. 2000. Computing Bisimulations for Finite-Control pi-Calculus. *Journal of Computer Science and Technology* 15, 1 (2000), 1–9. DOI:http://dx.doi.org/10.1007/BF02951922

Yang Liu, Jun Sun, and Jin Song Dong. 2011. PAT 3: An Extensible Architecture for Building Multi-domain Model Checkers. In *Proc. of ISSRE '11*. IEEE, Los Alamitos, CA, USA, 190–199. DOI:http://dx.doi.org/10.1109/ISSRE.2011.19

Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. 2002. TAG: a Tiny AGgregation Service for Ad-hoc Sensor Networks. *SIGOPS Oper. Syst. Rev.* 36, SI (Dec. 2002), 131–146. DOI:http://dx.doi.org/10.1145/844128.844142

Massimo Merro. 2007. An Observational Theory for Mobile Ad Hoc Networks. *Electronical Notes in Theoretical Computer Science* 173 (April 2007), 275–293. DOI:http://dx.doi.org/10.1016/j.entcs.2007.02.039

Robin Milner, Joachim Parrow, and David Walker. 1992a. A Calculus of Mobile Processes, I. *Inf. Comput.* 100, 1 (1992), 1–40. DOI:http://dx.doi.org/10.1016/0890-5401(92)90008-4

Robin Milner, Joachim Parrow, and David Walker. 1992b. A Calculus of Mobile Processes, II. *Inf. Comput.* 100, 1 (1992), 41–77. DOI:http://dx.doi.org/10.1016/0890-5401(92)90009-5

Joachim Parrow, Johannes Borgström, Palle Raabjerg, and Johannes Åman Pohjola. 2013. Higher-order psi-calculi. *Mathematical Structures in Computer Science* FirstView (June 2013), 1–37. DOI:http://dx.doi.org/10.1017/S0960129513000170

Andrew M. Pitts. 2003. Nominal logic, a first order theory of names and binding. *Inf. Comput.* 186, 2 (2003), 165–193. DOI:http://dx.doi.org/10.1016/S0890-5401(03)00138-X

PolyML 2013. Poly/ML. (2013). http://www.polyml.org A full implementation of Standard ML.

Jun Sun, Yang Liu, Jin Song Dong, and Chunqing Chen. 2009. Integrating Specification and Programs for System Modeling and Verification. In *Proc. TASE '09*. IEEE, 127–135. DOI:http://dx.doi.org/10.1109/TASE.2009.32

Christian Urban and Christine Tasson. 2005. Nominal Techniques in Isabelle/HOL.. In *CADE (LNCS)*, Robert Nieuwenhuis (Ed.), Vol. 3632. Springer, 38–53. DOI:http://dx.doi.org/10.1007/11532231_4

Björn Victor and Faron Moller. 1994. The Mobility Workbench — A Tool for the π-Calculus. In *Proc. of CAV '94 (LNCS)*, David Dill (Ed.), Vol. 818. Springer, 428–440. DOI:http://dx.doi.org/10.1007/3-540-58179-0_73

Lucian Wischik and Philippa Gardner. 2005. Explicit fusions. *Theoretical Computer Science* 304, 3 (2005), 606–630. DOI:http://dx.doi.org/10.1016/j.tcs.2005.03.017

Received November 2013; revised March 2014; accepted September 2014

Online Appendix to: The Psi-Calculi Workbench: a Generic Tool for Applied Process Calculi

Johannes Borgström, Ramūnas Gutkovas, Ioana Rodhe and Björn Victor, Uppsala University

A. CORRECTNESS PROOFS FOR THE SYMBOLIC SEMANTICS

A.1. Correctness Proofs for the Symbolic Operational Semantics

The proofs for the soundness and completeness of the symbolic semantics with respect to the concrete broadcast semantics [Borgström et al. 2011] mainly follow [Johansson et al. 2012].

We begin by enumerating the additional axioms for substitution.

AXIOM 1. $\mathrm{n}(M\sigma) \supseteq \mathrm{n}(M) \setminus \mathrm{n}(\sigma)$.

AXIOM 2. $\mathrm{n}((\Psi \otimes \Psi')\sigma) = \mathrm{n}(\Psi\sigma \otimes \Psi'\sigma)$.

AXIOM 3. $(\Psi \otimes \Psi')\sigma \simeq \Psi\sigma \otimes \Psi'\sigma$.

AXIOM 4. $\mathbf{1}\sigma = \mathbf{1}$.

We then present a number of lemmas used in the proofs.

LEMMA A.1 (WEAKENING). $(\sigma, \Psi) \models C \Longrightarrow \forall \Psi' : (\sigma, \Psi \otimes \Psi') \models C$

PROOF. By induction over the structure of C.

$\mathbf{true}$ Trivial since all solutions satisfies $\mathbf{true}$.

$\mathbf{false}$ Trivial since $\mathbf{false}$ has no solutions.

$\{\!|\Psi'' \vdash \varphi|\!\}$ We have that $(\sigma, \Psi) \models \{\!|\Psi'' \vdash \varphi|\!\}$, so $\Psi''\sigma \otimes \Psi \vdash \varphi\sigma$. Let Ψ' be any assertion. By weakening $\Psi''\sigma \otimes \Psi \otimes \Psi' \vdash \varphi\sigma$, or in other words $(\sigma, \Psi \times \Psi') \models \{\!|\Psi'' \vdash \varphi|\!\}$.

$M = N$ Trivial.

$a\#X$ Trivial.

$a \in \mathsf{n}(M)$ Trivial.

$\exists x.C$ Here there exists $y\#\sigma, \Psi$ such that $(\sigma[y := M], \Psi) \models (x\ y).C$. By induction $(\sigma[y := M], \Psi \otimes \Psi') \models (x\ y).C$. By equivariance of $\vdash$ we may assume that $b\#\Psi'$, so by definition $(\sigma, \Psi \otimes \Psi') \models \exists x.C$.

$(\nu a)C$ We have that $(\sigma, \Psi) \models (\nu a)C$. By Definition 7.1 this means that $\exists b.b\#\sigma, \Psi, C$ such that $(\sigma, \Psi) \models (a\ b)C$. By equivariance of $\vdash$ we may assume that $b\#\Psi'$. By induction $(\sigma, \Psi \otimes \Psi') \models (a\ b)C$, so by definition $(\sigma, \Psi \times \Psi') \models (\nu a)C$.

$C \wedge C'$ By induction $(\sigma, \Psi \otimes \Psi') \models C$ and $(\sigma, \Psi \otimes \Psi') \models C'$, thus $(\sigma, \Psi \otimes \Psi') \models C \wedge C'$.

$C \vee C'$ By induction $(\sigma, \Psi \otimes \Psi') \models C$ or $(\sigma, \Psi \otimes \Psi') \models C'$, thus $(\sigma, \Psi \otimes \Psi') \models C \vee C'$.

$C \Rightarrow C'$ We have that $(\sigma, \Psi) \models C \Rightarrow C'$, i.e. $\forall \Psi''.(\sigma, \Psi \otimes \Psi'') \models C$ implies $(\sigma, \Psi \otimes \Psi'') \models C'$. We must check that $(\sigma, \Psi \otimes \Psi') \models C \Rightarrow C'$, i.e. $\forall \Psi'''.(\sigma, \Psi \otimes \Psi' \otimes \Psi''') \models C$ implies $(\sigma, \Psi \otimes \Psi' \otimes \Psi''') \models C'$, which holds since $\forall \Psi''.(\sigma, \Psi \otimes \Psi'') \models C$ implies $(\sigma, \Psi \otimes \Psi'') \models C'$, and in particular it holds for any $\Psi'' = \Psi' \otimes \Psi'''$ □

LEMMA A.2 (OPENING). *If* $a\#\sigma, \Psi$ *then* $(\sigma, \Psi) \models (\nu a)C$ *iff* $(\sigma, \Psi) \models C$.

PROOF. Immediate from the definition of solutions for $(\nu a)C$. □

DOI:10.1145/10.1145/2682570

LEMMA A.3 (CHANNEL SUBSTITUTION). *When* $(\sigma, \Psi) \models (\nu\widetilde{b})\{\!|\Psi' \vdash M \leftrightarrows N|\!\} \wedge C$ *and* $y\#\sigma, \Psi, \widetilde{b}, \Psi', M, N, C$ *and* $\widetilde{b}\#\sigma, \Psi$ *then* $(\sigma \cdot [y := M\sigma], \Psi) \models (\nu\widetilde{b})\{\!|\Psi' \vdash N \leftrightarrows y|\!\} \wedge C$.

PROOF. By expanding the definitions involved, using the freshness assumptions and the partial invertibility of $\leftrightarrows$. □

AXIOM 5. $\Psi \equiv_{\mathrm{AC}} \Psi' \implies \mathrm{n}(\Psi) = \mathrm{n}(\Psi')$.

LEMMA A.4. $\Psi \equiv_{\mathrm{AC}} \Psi' \implies \Psi\sigma \simeq \Psi'\sigma$ *and* $\mathrm{n}(\Psi\sigma) = \mathrm{n}(\Psi'\sigma)$.

PROOF. By induction on the derivation of $\Psi \equiv_{\mathrm{AC}} \Psi'$, using the symmetry, transitivity and reflexivity of $\simeq$ at the symmetry, transitivity and reflexivity cases, Axiom 3 at the base cases and the induction case, and Axiom 4 at the unit case. □

LEMMA A.5. *If* $\Psi\sigma \simeq \Psi'\sigma$ *and* $\mathrm{n}(\Psi\sigma) = \mathrm{n}(\Psi'\sigma)$*, then* $(\sigma, \Psi) \models C$ *iff* $(\sigma, \Psi') \models C$.

PROOF. Straightforward from the definition of $\models$. □

LEMMA A.6. *If* $\widetilde{c}\#\sigma$ *then*

(1) $\mathcal{F}(P) = (\nu\widetilde{c})\Psi \implies \exists\Psi' \,.\, \mathcal{F}(P\sigma) = (\nu\widetilde{c})\Psi'$ *and* $\Psi' \simeq \Psi\sigma$ *and* $\mathrm{n}(\Psi') = \mathrm{n}(\Psi\sigma)$.
(2) $\mathcal{F}(P\sigma) = (\nu\widetilde{c})\Psi' \implies \exists\Psi \,.\, \mathcal{F}(P) = (\nu\widetilde{c})\Psi$ *and* $\Psi' \simeq \Psi\sigma$ *and* $\mathrm{n}(\Psi') = \mathrm{n}(\Psi\sigma)$.

PROOF. By induction on the derivation of $\mathcal{F}(P)$ (resp $\mathcal{F}(P\sigma)$), using Axiom 3 at the parallel induction case and Axiom 4 at the trivial base cases. □

LEMMA A.7. *If* $\widetilde{a}\#C, \sigma, \Psi'$ *then* $(\sigma, \Psi') \models ((\nu\widetilde{a})\Psi) \otimes C$ *iff* $(\sigma, \Psi' \otimes \Psi\sigma) \models C$.

PROOF. By induction on C. The interesting case is $C = (\nu\widetilde{c})\{\!|\Psi'' \vdash \varphi|\!\}$ where $(\sigma, \Psi') \models ((\nu\widetilde{a})\Psi) \otimes C \iff \Psi' \otimes \Psi\sigma \otimes \Psi''\sigma \vdash \varphi\sigma \iff (\sigma, \Psi' \otimes \Psi\sigma) \models C$ using Axiom 3. □

LEMMA A.8. *If* $F \equiv_{\mathrm{AC}} G$ *and* $(\sigma, \Psi) \models F \otimes C$ *then* $(\sigma, \Psi) \models G \otimes C$.

PROOF. By Lemma A.4 and Lemma A.7. □

The following key lemma characterises the shape of the constraints of point-to-point input and output transitions. The first conjunct in the constraint is always a channel equivalence constraint (between the object M of the original prefix and the transition object variable y) that must hold under a frame $(\nu\widetilde{c})\Psi$ that is AC-equivalent to that of the original process P. The lemma is used in the proof of Theorem 7.11, to show that the frames of the transition constrints correspond to the frames of the originating processes, in the SCOM case. It is also used in the proof of Theorem 7.12, to show that the precondition on the shape of the transitions in Rule SCOM always holds.

LEMMA A.9 (LEMMA 7.10). *Let* $\alpha = \overline{y}\,(\nu\widetilde{a})\widetilde{N}$ *or* $\alpha = \underline{y}(\widetilde{x})$*. If* $P \xrightarrow[C]{\alpha} P'$ *and* $y\#P$ *then there exist* $\widetilde{c}, \Psi, M$ *and* D *such that* $\mathcal{F}(P) \equiv_{\mathrm{AC}} (\nu\widetilde{c})\Psi$ *and* $y\#\widetilde{c}, \Psi, M, D$ *and* $C = (\nu\widetilde{c})\{\!|\Psi \vdash M \leftrightarrows y|\!\} \wedge D$.

PROOF. By induction on the derivation of $P \xrightarrow[C]{\alpha} P'$.

Case SIN. In this case the transition is derived like

$$\textsc{sIn}\ \frac{}{\underline{K}(\widetilde{x}) \,.\, P \xrightarrow[\{\!|\mathbf{1} \vdash K \leftrightarrows y|\!\}]{\underline{y}(\widetilde{x})} P}\ y\#K, P, \widetilde{x}$$

Here $\widetilde{c} = \epsilon$, $\Psi = \mathbf{1}$, $M = K$, and $D = \mathbf{true}$, where $\mathcal{F}(\underline{K}(\widetilde{x}) \,.\, P) = \mathbf{1}$.

Case sOut. In this case the transition is derived like

$$\textsc{sOut}\ \frac{}{\overline{M}\widetilde{N}\,.\,P \xrightarrow[\{\!|\mathbf{1}\vdash M\leftrightarrow y|\!\}]{\overline{y}\widetilde{N}} P}\ y\#M,\widetilde{N},P$$

Here $\widetilde{c}=\epsilon$, $\Psi=\mathbf{1}$, $M=M$, and $D=\mathbf{true}$, where $\mathcal{F}(\overline{M}\,\widetilde{N}\,.\,P)=\mathbf{1}$.

Case sCase. In this case the transition is derived like

$$\textsc{sCase}\ \frac{P_i \xrightarrow[C]{\alpha} P'}{\mathbf{case}\ \widetilde{\varphi}:\widetilde{P} \xrightarrow[C\wedge\{\!|\mathbf{1}\vdash\varphi_i|\!\}]{\alpha} P'}\ \mathrm{bn}(\alpha)\#\varphi_i$$

By induction we get $M,D',\Psi,\widetilde{c}$ such that $C=(\nu\widetilde{c})\{\!|\Psi\vdash M\leftrightarrow y|\!\}\wedge D'$ with $y\#D'$ and $\mathcal{F}(P_i)\equiv_{\mathrm{AC}}(\nu\widetilde{c})\Psi$. Let $D=D'\wedge\{\!|\mathbf{1}\vdash\varphi_i|\!\}$; since $y\#\mathbf{case}\ \widetilde{\varphi}:\widetilde{P}$ we also have that $y\#D$. By well-formedness, P_i is guarded, so by Lemma 7.9 $\mathcal{F}(P_i)\equiv_{\mathrm{AC}}\mathbf{1}$. By transitivity $\mathcal{F}(P)=\mathbf{1}\equiv_{\mathrm{AC}}(\nu\widetilde{c})\Psi$.

Case sPar. In this case the transition is derived like

$$\textsc{sPar}\ \frac{P \xrightarrow[C]{\alpha} P'}{P\mid Q \xrightarrow[\mathcal{F}(Q)\otimes C]{\alpha} P'\mid Q}\ \begin{array}{c}\mathrm{bn}(\alpha)\#Q\\ \alpha=\tau\vee\mathrm{subj}(\alpha)\#Q\end{array}$$

By induction there are $M,D',\Psi,\widetilde{c}$ such that $C=(\nu\widetilde{c})\{\!|\Psi\vdash M\leftrightarrow y|\!\}\wedge D'$ with $y\#D'$ and $\mathcal{F}(P)\equiv_{\mathrm{AC}}(\nu\widetilde{c})\Psi$. Let $D=\mathcal{F}(Q)\otimes D'$; since $y\#P|Q$ we also have that $y\#D$. By Lemma 7.8 $\mathcal{F}(P\mid Q)\equiv_{\mathrm{AC}}((\nu\widetilde{c})\Psi)\otimes\mathcal{F}(Q)\equiv_{\mathrm{AC}}\mathcal{F}(Q)\otimes(\nu\widetilde{c})\Psi$.

Case sScope. In this case the transition is derived like

$$\textsc{sScope}\ \frac{P \xrightarrow[C]{\alpha} P'}{(\nu b)P \xrightarrow[(\nu b)C]{\alpha} (\nu b)P'}\ b\#\alpha$$

By induction there exist $\widetilde{c},\Psi,M$ and D' such that $C=(\nu\widetilde{c})(\Psi\vdash M\leftrightarrow y)\wedge D'$ with $y\#M,D$ and $\mathcal{F}(P)\equiv_{\mathrm{AC}}(\nu\widetilde{c})\Psi$. Let $D=(\nu b)D'$; a fortiori $y\#(\nu b)D$. By Lemma 7.8 $\mathcal{F}((\nu b)P)\equiv_{\mathrm{AC}}(\nu b)(\nu\widetilde{c})\Psi$.

Case sOpen. As sScope.

Case sRep. In this case the transition is derived like

$$\textsc{sRep}\ \frac{P\mid !P \xrightarrow[C]{\alpha} P'}{!P \xrightarrow[C]{\alpha} P'}$$

By induction there exist $\widetilde{c},\Psi,M$ and D such that $C=(\nu\widetilde{c})(\Psi\vdash M\leftrightarrow y)\wedge D$ with $y\#M,D$ and $\mathcal{F}(P|!P)\equiv_{\mathrm{AC}}(\nu\widetilde{c})\Psi$. By well-formedness, P is guarded, so by Lemma 7.9 $\mathcal{F}(P|!P)\equiv_{\mathrm{AC}}\mathbf{1}$. By transitivity $\mathcal{F}(!P)=\mathbf{1}\equiv_{\mathrm{AC}}(\nu\widetilde{c})\Psi$. □

Lemma A.10 (Change subject). *Let B be any finite set of names. If $\alpha=\overline{y}\,(\nu\widetilde{a})\widetilde{N}$ (resp. $\alpha=y(\widetilde{x})$)*

and $P \xrightarrow[(\nu\widetilde{b})\{\!|\Psi\vdash M\leftrightarrow y|\!\}\wedge C]{\alpha} P'$ then $\exists z$ such that $z\#\Psi,\widetilde{b},P,B,C$ and $P \xrightarrow[(\nu\widetilde{b})\{\!|\Psi\vdash M\leftrightarrow z|\!\}\wedge C]{\alpha'} P'$

with $\alpha'=\overline{z}\,(\nu\widetilde{a})\widetilde{N}$ (resp. $\alpha'=z(\widetilde{x})$).

PROOF. By induction on the derivation of the transition. The set of names B is necessary to be able to use the induction hypothesis in some of the induction cases. □

LEMMA A.11. *If* $P \xrightarrow[C]{\alpha} P'$ *and* $a\#P, \mathrm{bn}(\alpha)$ *then* $a\#P'$.

PROOF. The proof is by induction on the derivation of the transition.

Case SIN. In this case the transition is derived like

$$\textsc{sIn}\ \frac{}{\underline{M}(\widetilde{x}) \,.\, P \xrightarrow[\{\!|\mathbf{1} \vdash M \leftrightarrow y|\!\}]{\underline{y}(\widetilde{x})} P}\ y\#\Psi, M, P, \widetilde{x}$$

We know that $a\#\underline{M}(\widetilde{x}) \,.\, P, \widetilde{x}$. Then also $a\#P$.
Case SOUT. In this case the transition is derived like

$$\textsc{sOut}\ \frac{}{\overline{M}\widetilde{N} \,.\, P \xrightarrow[\{\!|\mathbf{1} \vdash M \leftrightarrow y|\!\}]{\overline{y}\widetilde{N}} P}\ y\#\Psi, M, \widetilde{N}, P$$

We know that $a\#\overline{M}\widetilde{N} \,.\, P$. Then also $a\#P$.
Case SCASE. In this case the transition is derived like

$$\textsc{sCase}\ \frac{P_i \xrightarrow[C]{\alpha} P'}{\mathbf{case}\ \widetilde{\varphi} : \widetilde{P} \xrightarrow[C \wedge \{\!|\mathbf{1} \vdash \varphi_i|\!\}]{\alpha} P'}\ \mathrm{bn}(\alpha)\#\varphi_i$$

We know that $a\#\mathbf{case}\ \widetilde{\varphi} : \widetilde{P}, \mathrm{bn}(\alpha)$. Then also $a\#P_i$. By induction we get that $a\#P'$.
Case SCOM. In this case the transition is derived like

$$\textsc{sCom}\ \frac{P \xrightarrow[C_P]{\overline{y}\ (\nu\widetilde{a})\widetilde{N}} P' \qquad Q \xrightarrow[C_Q]{\underline{z}(\widetilde{x})} Q'}{P \mid Q \xrightarrow[C]{\tau} (\nu\widetilde{a})(P' \mid Q'[\widetilde{x} := \widetilde{N}])}\ \begin{array}{l}\widetilde{a}\#Q,\\ y\#z\end{array}$$

We know that $a\#P \mid Q$. Let $p \subseteq \widetilde{a} \times (p \cdot \widetilde{a})$ be a permutation such that $a\#p \cdot \widetilde{a}$. By α-conversion we write the transition from P as $P \xrightarrow[p \cdot C_P]{\overline{y}\ (\nu p \cdot \widetilde{a})\widetilde{p \cdot N}} p \cdot P'$. By induction we get that $a\#p \cdot P'$. Let $q \subseteq \{\widetilde{x}\} \times (q \cdot \{\widetilde{x}\})$ be a permutation such that $a, p \cdot \widetilde{a}\#q \cdot \widetilde{x}$. By α-conversion we write the transition from Q as $Q \xrightarrow[q \cdot C_Q]{\underline{z(q \cdot x)}} q \cdot Q'$. By induction we get that $a\#q \cdot Q'$ and that $p \cdot \widetilde{a}\#q \cdot Q'$. Since $a\#P, p \cdot \widetilde{a}$ we also have that $a\#p \cdot \widetilde{N}$. This means that $a\#(q \cdot Q')[q \cdot \widetilde{x} := p \cdot \widetilde{N}]$ by one of the requirements on substitution. All together we get that $a\#(\nu p \cdot \widetilde{a})(p \cdot P' \mid (q \cdot Q')[q \cdot \widetilde{x} := p \cdot \widetilde{N}])$. By the substitution law for α-conversion we get that $a\#(\nu p \cdot \widetilde{a})(p \cdot P' \mid Q'[\widetilde{x} := p \cdot \widetilde{N}])$. Finally, by α-converting we get that $a\#(\nu\widetilde{a})(P' \mid Q'[\widetilde{x} := \widetilde{N}])$.
Case SPAR. In this case the transition is derived like

$$\textsc{sPar}\ \frac{P \xrightarrow[C]{\alpha} P'}{P \mid Q \xrightarrow[\mathcal{F}(Q) \otimes C]{\alpha} P' \mid Q}\ \begin{array}{c}\mathrm{bn}(\alpha)\#Q\\ \alpha = \tau \vee \mathrm{subj}(\alpha)\#Q\end{array}$$

We know that $a\#P \mid Q, \mathrm{bn}(\alpha)$. By induction we get that $a\#P'$. Then also $a\#P' \mid Q$.

Case sSCOPE. In this case the transition is derived like

$$\textsc{sScope}\ \frac{P\ \xrightarrow[C]{\alpha}\ P'}{(\nu b)P\ \xrightarrow[(\nu b)C]{\alpha}\ (\nu b)P'}\ b\#\alpha,\Psi$$

We know that $a\#(\nu b)P, \mathrm{bn}(\alpha)$. Let $p = (b\ c)$ such that $a\#c, p\cdot P, p\cdot \mathrm{bn}(\alpha)$. By equivariance the premise is rewritten to $p\cdot\left(p\cdot P\ \xrightarrow[p\cdot C]{p\cdot\alpha}\ p\cdot P'\right)$. By induction we get that $a\#p\cdot P'$. Then also $a\#(\nu p\cdot b)(p\cdot P')$. By α-equivalence we get that $a\#(\nu b)P'$.
Case sOPEN. In this case the transition is derived like

$$\textsc{sOpen}\ \frac{P\ \xrightarrow[C]{\overline{y}\ (\nu\widetilde{a})\widetilde{N}}\ P'}{(\nu b)P\ \xrightarrow[(\nu b)C]{\overline{y}\ (\nu\widetilde{a}\cup\{b\})\widetilde{N}}\ P'}\ \begin{array}{l} b\in \mathrm{n}(\widetilde{N}) \\ b\#\widetilde{a},\Psi,y\end{array}$$

We know that $a\#(\nu b)P, \widetilde{a}, b$. This gives us that $a\#P$. By induction we get that $a\#P'$.
Case sREP. In this case the transition is derived like

$$\textsc{sRep}\ \frac{P \mid !P\ \xrightarrow[C]{\alpha}\ P'}{!P\ \xrightarrow[C]{\alpha}\ P'}$$

We know that $a\#!P, \mathrm{bn}(\alpha)$. This gives us that $a\#P \mid !P$. By induction we get that $a\#P'$.
Case sBROUT. Here the transition is derived by

$$\textsc{sBrOut}\ \frac{x\#M,\widetilde{N},P}{\overline{M}\,\widetilde{N}\,.\,P\ \xrightarrow[\mathbf{1}\vdash M\dot{\prec}x]{\overline{x}!\ \widetilde{N}}\ P}$$

We know that $a\#P$.
Case sBRIN. Here the transition is derived by

$$\textsc{sBrIn}\ \frac{x,\widetilde{y}\#\Psi,M,P \quad x\neq\widetilde{y}}{\underline{M}(\widetilde{y})\,.\,P\ \xrightarrow[\mathbf{1}\vdash x\dot{\succ}M]{\underline{x}?(\widetilde{y})}\ P}$$

We know that $a\#\underline{M}(\widetilde{y})\,.\,P, \widetilde{y}$. This gives us that $a\#P$.
Case sBRMERGE. Here the transition is derived by

$$\textsc{sBrMerge}\ \frac{P\ \xrightarrow[C_P]{\underline{x}?(\widetilde{y})}\ P' \qquad Q\ \xrightarrow[C_Q]{\underline{x}?(\widetilde{y})}\ Q'}{P\mid Q\ \xrightarrow[(\mathcal{F}(Q)\otimes C_P)\wedge(\mathcal{F}(P)\otimes C_Q)]{\underline{x}?(\widetilde{y})}\ P'\mid Q'}$$

By induction $a\#P', Q'$, so $a\#P'\mid Q'$.
Case sBRCOM. Here the transition is derived by

$$\textsc{sBrCom}\ \frac{P\ \xrightarrow[C_P]{\overline{x}\ (\nu\widetilde{a})\widetilde{N}}\ P' \qquad Q\ \xrightarrow[C_Q]{\underline{x}?(\widetilde{y})}\ Q'}{P\mid Q\ \xrightarrow[(\mathcal{F}(Q)\otimes C_P)\wedge(\mathcal{F}(P)\otimes C_Q)]{\overline{x}\ (\nu\widetilde{a})\widetilde{N}}\ P'\mid Q'[\widetilde{y}:=\widetilde{N}]}\ \widetilde{a}\#Q$$

By induction $a\#P', Q'$. Here $\mathrm{n}(\widetilde{N})\subseteq \mathrm{n}(P)\cup\{\widetilde{a}\}$, so since $a\#P,\widetilde{a}$ we have $a\#\widetilde{N}$. Thus $a\#P'\mid Q'[\widetilde{y}:=\widetilde{N}]$.

Case sBrClose. Here the transition is derived by

$$\textsc{sBrClose}\ \frac{P\ \xrightarrow[C]{\overline{x}!\,(\nu\widetilde{a})\widetilde{N}}\ P'}{(\nu b)P\ \xrightarrow[\exists^b x.C]{\tau}\ (\nu b)(\nu\widetilde{a})P'}$$

Assume that $a\#b$. By induction $a\#P'$, so $a\#(\nu b)(\nu\widetilde{a})P'$. □

Lemma A.12.

$$\begin{aligned}\mathcal{F}((\nu a)P) = (\nu\widetilde{b}_{(\nu a)P})\Psi_{(\nu a)P} \implies\ & \exists\widetilde{b}_P, \Psi_P \textit{ such that}\\ & \mathcal{F}(P) = (\nu\widetilde{b}_P)\Psi_P\\ \wedge\ & \widetilde{b}_{(\nu a)P} = a\widetilde{b}_P\\ \wedge\ & \Psi_{(\nu a)P} = \Psi_P\end{aligned}$$

Proof. Just use the definitions involved. □

Lemma A.13.

$$\begin{aligned}\mathcal{F}(P \mid Q) = (\nu\widetilde{b}_{P\mid Q})\Psi_{P\mid Q} \implies\ & \exists\widetilde{b}_P, \widetilde{b}_Q, \Psi_P, \Psi_Q \textit{ such that}\\ & \mathcal{F}(P) = (\nu\widetilde{b}_P)\Psi_P\\ \wedge\ & \mathcal{F}(Q) = (\nu\widetilde{b}_Q)\Psi_Q\\ \wedge\ & \widetilde{b}_{P\mid Q} = \widetilde{b}_P\widetilde{b}_Q\\ \wedge\ & \Psi_{P\mid Q} = \Psi_P \otimes \Psi_Q\end{aligned}$$

Proof. Just use the definitions involved. □

Lemma A.14 (Change frame).
If $\Psi \rhd P\ \xrightarrow{\alpha}\ P'$, $\Psi \simeq \Psi'$, *and* $\mathrm{n}(\Psi) = \mathrm{n}(\Psi')$, *then* $\Psi' \rhd P\ \xrightarrow{\alpha}\ P'$.

Proof. By induction on the derivation of the transition. □

Lemma A.15 (Names are fresh in the constraint).
If $P\ \xrightarrow[C]{\alpha}\ P'$ *with* $\alpha = \underline{y}(\widetilde{x})$ *or* $\alpha = \underline{y}?(\widetilde{x})$, *and* $\widetilde{x}, z\#P, y$ *then* $\widetilde{x}, z\#C$.

Proof. By induction on the derivation of the transition.

Case sIn. In this case the transition is derived like

$$\textsc{sIn}\ \frac{}{\underline{M}(\widetilde{x})\,.\,P\ \xrightarrow[\{\!|\mathbf{1}\vdash M\leftrightarrow y|\!\}]{\alpha}\ P}\ y\#M, P, \widetilde{x}$$

We know that $\widetilde{x}, z\#y, \underline{M}(\widetilde{x})\,.\,P$, so $\widetilde{x}, z\#\{\!|\mathbf{1}\vdash M \leftrightarrow y|\!\}$.
Case sCase. In this case the transition is derived like

$$\textsc{sCase}\ \frac{P_i\ \xrightarrow[C]{\alpha}\ P'}{\mathbf{case}\ \widetilde{\varphi} : \widetilde{P}\ \xrightarrow[C\wedge\{\!|\mathbf{1}\vdash\varphi_i|\!\}]{\alpha}\ P'}$$

By induction we get that $\widetilde{x}, z\#C$. From $\widetilde{x}, z\#\varphi_i$ we get that $\widetilde{x}, z\#\{\!|\mathbf{1}\vdash\varphi_i|\!\}$.
Case sPar. In this case the transition is derived like

$$\textsc{sPar}\ \frac{P\ \xrightarrow[C]{\alpha}\ P'}{P\mid Q\ \xrightarrow[\mathcal{F}(Q)\otimes C]{\alpha}\ P'\mid Q}\ \begin{matrix}\widetilde{x}\#Q\\ y\#Q\end{matrix}$$

By induction $\widetilde{x}, z\#C$, and since $\widetilde{x}, z\#Q$ we also have $\widetilde{x}, z\#\mathcal{F}(Q)$. By equivariance of $\otimes$ we get $\widetilde{x}, z\#\mathcal{F}(Q)\otimes C$.

Case SSCOPE. In this case the transition is derived like

$$\textsc{sScope}\ \frac{P\ \xrightarrow[C]{\alpha}\ P'}{(\nu b)P\ \xrightarrow[(\nu b)C]{\alpha}\ (\nu b)P'}\ b\#\alpha,\Psi$$

We may assume that $b\#z$. By induction we get that $\widetilde{x}, z\#C$, so a fortiori $\widetilde{x}, z\#(\nu b)C$.
Case SREP. In this case the transition is derived like

$$\textsc{sRep}\ \frac{P \mid !P\ \xrightarrow[C]{\alpha}\ P'}{!P\ \xrightarrow[C]{\alpha}\ P'}$$

The desired result follows directly from induction.
Case SBRIN. Here the transition is derived by

$$\textsc{sBrIn}\ \frac{\widetilde{x}, y\#\Psi, M, P \quad y\#\widetilde{x}}{\underline{M}(\widetilde{x})\,.\,P\ \xrightarrow[\mathbf{1}\vdash y\dot{\succ}M]{\underline{y}?(\widetilde{x})}\ P}$$

We know that $\widetilde{x}, z\#y, \underline{M}(\widetilde{x})\,.\,P$, so $\widetilde{x}, z\#\{\!|\mathbf{1}\vdash y\dot{\succ}M|\!\}$.
Case SBRMERGE. Here the transition is derived by

$$\textsc{sBrMerge}\ \frac{P\ \xrightarrow[C_P]{\underline{y}?(\widetilde{x})}\ P' \qquad Q\ \xrightarrow[C_Q]{\underline{y}?(\widetilde{x})}\ Q'}{P \mid Q\ \xrightarrow[(\mathcal{F}(Q)\otimes C_P)\wedge(\mathcal{F}(P)\otimes C_Q)]{\underline{y}?(\widetilde{x})}\ P' \mid Q'}$$

By induction $\widetilde{x}, z\#C_P, C_Q$. By assumption $\widetilde{x}, z\#P, Q$, so $\widetilde{x}, z\#\mathcal{F}(P), \mathcal{F}(Q)$. By equivariance of $\otimes$ and $\wedge$ we then get $\widetilde{x}, z\#\mathcal{F}(Q)\otimes C_P \wedge \mathcal{F}(P)\otimes C_Q$. □

LEMMA A.16 (CONGRUENCE OF CONSTRAINT EQUIVALENCE).
If $\forall\sigma,\Psi.(\sigma,\Psi)\models C \Leftrightarrow (\sigma,\Psi)\models D$ *then* $\forall\sigma,\Psi.(\sigma,\Psi)\models(\nu a)C \Leftrightarrow (\sigma,\Psi)\models(\nu a)D$.

PROOF. Adding a restriction of a to a constraint amounts to removing the solutions involving a from the set of all solutions. In this case we remove the same solutions from both C and D, so the resulting sets of all substitutions will still be equal. □

LEMMA A.17. $\forall\sigma,\Psi.(\sigma,\Psi)\models(\nu a)(\nu b)C \Leftrightarrow (\sigma,\Psi)\models(\nu b)(\nu a)C$

PROOF. Both $(\nu a)(\nu b)$ and $(\nu b)(\nu a)$ remove the same set of solutions from C. □

A.2. Proof of Soundness Theorem

We prove soundness and completeness of the symbolic semantics of this paper with respect to the concrete semantics of broadcast psi-calculi [Borgström et al. 2011] (Table VI). The soundness theorem and its proof follow [Johansson et al. 2012], apart from the weaker preconditions of Rule SCOM (compared to OLD-COM), and the new cases for broadcast actions.

THEOREM A.18 (THEOREM 7.11).
If $P\ \xrightarrow[C]{\alpha}\ P'$ *and* $(\sigma,\Psi)\models C$ *and* $\mathrm{bn}(\alpha)\#\sigma$ *then* $\Psi\ \rhd\ P\sigma\ \xrightarrow{\alpha\sigma}\ P'\sigma$.

PROOF. By induction on the inference of $P\ \xrightarrow[C]{\alpha}\ P'$.

Case SIN. In this case the inference looks like

$$\textsc{sIn}\ \frac{}{\underline{M}(\widetilde{x}).P\ \xrightarrow[\{\!|\mathbf{1}\vdash M\leftrightarrows y|\!\}]{\underline{y}(\widetilde{x})}\ P}\ y\#M,P,\widetilde{x}$$

Since $\widetilde{x}\#\sigma$ we have that $(\underline{M}(\widetilde{x}).P)\sigma = \underline{M\sigma}(\widetilde{x}).P\sigma$ and that $(\underline{y}(\widetilde{x}))\sigma = \underline{y\sigma}(\widetilde{x})$. We then do the following derivation:

$$\textsc{cIn}\ \frac{\Psi\vdash M\sigma\leftrightarrows y\sigma}{\Psi\ \rhd\ \underline{M\sigma}(\widetilde{x}).P\sigma\ \xrightarrow{y\sigma(\widetilde{x})}\ P\sigma}$$

Case SOUT. In this case the inference looks like

$$\textsc{sOut}\ \frac{}{\overline{M}\widetilde{N}.P\ \xrightarrow[\{\!|\mathbf{1}\vdash M\leftrightarrows y|\!\}]{\overline{y}\widetilde{N}}\ P}\ y\#M,\widetilde{N},P$$

We then have a concrete transition

$$\textsc{cOut}\ \frac{\Psi\vdash M\sigma\leftrightarrows y\sigma}{\Psi\ \rhd\ (\overline{M}\widetilde{N}.P)\sigma\ \xrightarrow{(\overline{y}\widetilde{N})\sigma}\ P\sigma}$$

Case SCASE. In this case the inference looks like

$$\textsc{sCase}\ \frac{P_i\ \xrightarrow[C]{\alpha}\ P'}{\mathbf{case}\ \widetilde{\varphi}:\widetilde{P}\ \xrightarrow[C\wedge\{\!|\mathbf{1}\vdash\varphi_i|\!\}]{\alpha}\ P'}$$

Take (σ,Ψ) such that $(\sigma,\Psi)\models C\wedge\{\!|\mathbf{1}\vdash\varphi_i|\!\}$ and $\mathrm{bn}(\alpha)\#\sigma$. We must find a transition $\Psi\ \rhd\ (\mathbf{case}\ \widetilde{\varphi}:\widetilde{P})\sigma\ \xrightarrow{\alpha\sigma}\ P'\sigma$.
We then have that $\Psi\vdash\varphi_i\sigma$ and that (σ,Ψ) is also a solution to C. By induction we get that $\Psi\ \rhd\ P_i\sigma\ \xrightarrow{\alpha\sigma}\ P'\sigma$. We can now do the following derivation:

$$\textsc{cCase}\ \frac{\Psi\ \rhd\ P_i\sigma\ \xrightarrow{\alpha\sigma}\ P'\sigma\qquad\Psi\vdash\varphi_i\sigma}{\Psi\ \rhd\ (\mathbf{case}\ \widetilde{\varphi}:\widetilde{P})\sigma\ \xrightarrow{\alpha\sigma}\ P'\sigma}$$

Case SCOM. In this case the inference looks like

$$\textsc{sCom}\ \frac{P\ \xrightarrow[C'_P]{\overline{y}\,(\nu\widetilde{a})\widetilde{N}}\ P'\qquad Q\ \xrightarrow[C'_Q]{\underline{z}(\widetilde{x})}\ Q'}{P\mid Q\ \xrightarrow[C]{\tau}\ (\nu\widetilde{a})(P'\mid Q'[\widetilde{x}:=\widetilde{N}])}\ \begin{array}{c}\widetilde{a}\#Q\\ y\#z\\ y,z\#\widetilde{b}_P,P,\widetilde{b}_Q,Q,\widetilde{N},\widetilde{a}\end{array}$$

where $C'_P=(\nu\widetilde{c}_P)\{\!|\Psi'_P\vdash M_P\leftrightarrows y|\!\}\wedge C_P$, $C'_Q=(\nu\widetilde{c}_Q)\{\!|\Psi'_Q\vdash M_Q\leftrightarrows z|\!\}\wedge C_Q$ and $C=(\nu\widetilde{c}_P\widetilde{c}_Q)\{\!|\Psi_P\otimes\Psi_Q\vdash M_P\leftrightarrows M_Q|\!\}\wedge((\nu\widetilde{c}_Q)\Psi'_Q\otimes C_P)\wedge((\nu\widetilde{c}_Q)\Psi'_P\otimes C_Q)$.
We assume that $y,z\#\sigma,\Psi',C_P,C_Q$. If that is not the case we can use Lemma A.10 to find subjects for which it is true. We further assume that $\widetilde{a},\widetilde{x}\#\sigma$ (bound names are fresh). Let $\mathcal{F}(P)=(\nu\widetilde{b}_P)\Psi_P$ and $\mathcal{F}(Q)=(\nu\widetilde{b}_Q)\Psi_Q$. We assume that that $\widetilde{b}_P,\widetilde{b}_Q\#(\sigma,\Psi),P,Q,\widetilde{a}$.
By Lemma 7.10 $\mathcal{F}(P)\equiv_{\mathrm{AC}}(\nu\widetilde{c}_P)\Psi'_P$ and $\mathcal{F}(Q)\equiv_{\mathrm{AC}}(\nu\widetilde{c}_Q)\Psi'_Q$.
We know that $(\sigma,\Psi)\models(\nu\widetilde{c}_P)(\nu\widetilde{c}_Q)\{\!|\Psi'_P\otimes\Psi'_Q\vdash M_P\leftrightarrows M_Q|\!\}\wedge(\nu\widetilde{c}_Q)\Psi'_Q\otimes C_P$, so by Lemma A.17 and Lemma A.2 $(\sigma,\Psi)\models(\nu\widetilde{c}_P)\{\!|\Psi'_P\otimes\Psi'_Q\vdash M_P\leftrightarrows M_Q|\!\}\wedge\Psi'_Q\otimes C_P$.

By Lemma A.8 $(\sigma, \Psi) \models \Psi_Q \otimes ((\nu\widetilde{c}_P)\{\!\{\Psi'_P \vdash M_P \leftrightarrow M_Q\}\!\} \wedge C_P)$. so by Lemma A.7 $(\sigma, \Psi \otimes \Psi_Q\sigma) \models (\nu\widetilde{c}_P)\{\!\{\Psi'_P \vdash M_P \leftrightarrow M_Q\}\!\} \wedge C_P$. By substitutivity $(\sigma \cdot [y := M_Q\sigma], \Psi \otimes \Psi_Q\sigma) \models C'_P$, so by induction $\Psi \otimes \Psi_Q\sigma \rhd P\sigma \xrightarrow{(\overline{y}\ (\nu\widetilde{a})\widetilde{N})\sigma'} P'\sigma'$. By Lemma A.6 we get that $\mathcal{F}(Q\sigma) = (\nu\widetilde{b}_Q)\Psi_{Q\sigma}$ such that $\Psi_{Q\sigma} \simeq \Psi_Q\sigma$ and $\mathrm{n}(\Psi_{Q\sigma}) = \mathrm{n}(\Psi_Q\sigma)$. By Lemma A.14 $\Psi \otimes \Psi_{Q\sigma} \rhd P\sigma \xrightarrow{(\overline{y}\ (\nu\widetilde{a})\widetilde{N})\sigma'} Q'\sigma'$.

Similarly $(\sigma \cdot [z := M_P\sigma], \Psi \otimes \Psi_P\sigma) \models C'_Q$, so by induction $\Psi \otimes \Psi_P\sigma \rhd Q\sigma \xrightarrow{(\underline{z}(\widetilde{x}))\sigma'} Q'\sigma'$. By Lemma A.6 we get that $\mathcal{F}(P\sigma) = (\nu\widetilde{b}_P)\Psi_{P\sigma}$ such that $\Psi_{P\sigma} \simeq \Psi_P\sigma$ and $\mathrm{n}(\Psi_{P\sigma}) = \mathrm{n}(\Psi_P\sigma)$. By Lemma A.14 $\Psi \otimes \Psi_{P\sigma} \rhd Q\sigma \xrightarrow{(\underline{z}(\widetilde{x}))\sigma'} Q'\sigma'$.

Applying Lemma A.2 to $(\sigma, \Psi) \models \Psi_Q \otimes ((\nu\widetilde{c}_P)\{\!\{\Psi'_P \vdash M_P \leftrightarrow M_Q\}\!\})$ we get that $(\sigma, \Psi) \models \Psi_Q \otimes \{\!\{\Psi'_P \vdash M_P \leftrightarrow M_Q\}\!\}$, By Lemma A.7 we have $(\sigma, \Psi \otimes \Psi_Q\sigma) \models \{\!\{\Psi'_P \vdash M_P \leftrightarrow M_Q\}\!\}$, so by Lemma A.8 we get $(\sigma, \Psi \otimes \Psi_Q\sigma) \models \{\!\{\Psi_P \vdash M_P \leftrightarrow M_Q\}\!\}$. Thus $\Psi \otimes \Psi_P\sigma \otimes \Psi_Q\sigma \vdash M_P\sigma \leftrightarrow M_Q\sigma$, so $\Psi \otimes \Psi_{P\sigma} \otimes \Psi_{Q\sigma} \vdash M_P\sigma \leftrightarrow M_Q\sigma$.

We then have the following derivation (remember that y and z are fresh for basically everything but themselves):

$$\textsc{cCom}\ \frac{\Psi \otimes \Psi_{P\sigma} \rhd Q\sigma \xrightarrow{M_P\sigma(\widetilde{x})} Q'\sigma \qquad \Psi \otimes \Psi_{P\sigma} \otimes \Psi_{Q\sigma} \vdash M_P\sigma \leftrightarrow M_Q\sigma}{\Psi \rhd P\sigma \mid Q\sigma \xrightarrow{\tau} (\nu\widetilde{a})(P'\sigma \mid Q'\sigma[\widetilde{x} := \widetilde{N}\sigma])}\ \Psi \otimes \Psi_{P\sigma} \otimes \Psi_{Q\sigma} \rhd P\sigma \xrightarrow{\overline{M_Q\sigma}\ (\nu\widetilde{a})\widetilde{N\sigma}} P\sigma\ \widetilde{a}\#Q\sigma$$

Since $\widetilde{a}, \widetilde{x}\#\sigma$ we have that $(\nu\widetilde{a})(P'\sigma \mid Q'\sigma[\widetilde{x} := \widetilde{N}\sigma]) = (\nu\widetilde{a})(P' \mid Q'[\widetilde{x} := \widetilde{N}])\sigma$.

Case sPar. In this case the inference looks like

$$\textsc{sPar}\ \frac{\Psi \otimes \Psi_Q \rhd P \xrightarrow[C]{\alpha} P'}{P \mid Q \xrightarrow[\mathcal{F}(Q)\otimes C]{\alpha} P' \mid Q}\ \mathrm{bn}(\alpha)\#Q\ \alpha = \tau \vee \mathrm{subj}(\alpha)\#Q$$

We can assume that $\mathrm{subj}(\alpha)\#P$ (if not, use Lemma A.10 to find another subject). Assume that $\widetilde{x}\#\sigma, P \mid Q$.

Let $\mathcal{F}(Q) = (\nu\widetilde{b}_Q)\Psi_Q$ with $\widetilde{b}_Q\#\alpha, C, \Psi, \sigma$. By Lemma A.7 $(\sigma, \Psi \otimes \Psi_Q\sigma) \models C$. By induction we then get that $P \xrightarrow[C]{\alpha} P'$ has a matching transition $\Psi \otimes \Psi_Q\sigma \rhd P\sigma \xrightarrow{\alpha\sigma} P'\sigma$.

By Lemma A.6 we get that $\mathcal{F}(Q\sigma) = (\nu\widetilde{b}_Q)\Psi_{Q\sigma}$ such that $\Psi_{Q\sigma} \simeq \Psi_Q\sigma$ and $\mathrm{n}(\Psi_{Q\sigma}) = \mathrm{n}(\Psi_Q\sigma)$. By Lemma A.14 $\Psi \otimes \Psi_{Q\sigma} \rhd P\sigma \xrightarrow{\alpha\sigma} P'\sigma$, so we can do the following concrete inference:

$$\textsc{cPar}\ \frac{\Psi \otimes \Psi_{Q\sigma} \rhd P\sigma \xrightarrow{\alpha\sigma} P'\sigma}{\Psi \rhd P\sigma \mid Q\sigma \xrightarrow{\alpha\sigma} P'\sigma \mid Q\sigma}\ \mathrm{bn}(\alpha\sigma)\#Q\sigma$$

Case sScope. In this case the inference looks like

$$\textsc{sScope}\ \frac{P \xrightarrow[C]{\alpha} P'}{(\nu a)P \xrightarrow[(\nu a)C]{\alpha} (\nu a)P'}\ a\#\alpha, \Psi$$

We assume that $\mathrm{subj}(\alpha)\#(\nu a)P, \widetilde{x}$ (if not, use Lemma A.10 to find a new subject). We also assume $a\#\sigma, \Psi$ (bound names are fresh). By Lemma A.2 we then have that $(\sigma, \Psi) \models C$.

By induction we get that $P \xrightarrow[C]{\alpha} P'$ has a corresponding transition $\Psi \rhd P\sigma \xrightarrow{\alpha\sigma} P'\sigma$.

We can then do the following concrete inference:

$$\textsc{cScope}\ \frac{\Psi \rhd P\sigma \xrightarrow{\alpha\sigma} P'\sigma}{\Psi \rhd (\nu a)(P\sigma) \xrightarrow{\alpha\sigma} (\nu a)(P'\sigma)}\ a\#\alpha\sigma, \Psi$$

Since $a\#\sigma$ we have that $(\nu a)(P\sigma) = ((\nu a)P)\sigma$ and $(\nu a)(P'\sigma) = ((\nu a)P')\sigma$.
Case sOpen. In this case the inference looks like

$$\textsc{sOpen}\ \frac{P \xrightarrow[C]{\overline{y}\ (\nu\widetilde{a})\widetilde{N}} P'}{(\nu a)P \xrightarrow[(\nu a)C]{\overline{y}\ (\nu\widetilde{a}\cup\{a\})\widetilde{N}} P'}\ \begin{array}{l} a \in \mathrm{n}(\widetilde{N}) \\ a\#\widetilde{a}, y \end{array}$$

We can assume that $y\#P, \widetilde{a}, a$ (if not, use Lemma A.10 to find another subject). Since $(\sigma, \Psi) \models (\nu a)C$ we also have that $(\sigma, \Psi) \models C$.

By induction we get that $P \xrightarrow[C]{\overline{y}\ (\nu\widetilde{a})\widetilde{N}} P'$ has a corresponding transition $\Psi \rhd P\sigma \xrightarrow{(\overline{y}\ (\nu\widetilde{a})\widetilde{N})\sigma} P'\sigma$.
We assume that $a\#\sigma$ (bound names are fresh). By Axiom 1 $a \in \mathrm{n}(\widetilde{N}\sigma)$, so we have the following concrete inference.

$$\textsc{cOpen}\ \frac{\Psi \rhd P\sigma \xrightarrow{(\overline{y}\ (\nu\widetilde{a})\widetilde{N})\sigma} P'\sigma}{\Psi \rhd (\nu a)P\sigma \xrightarrow{(\overline{y}\ (\nu\widetilde{a}\cup\{a\})\widetilde{N})\sigma} P'\sigma}\ \begin{array}{l} a \in \mathrm{n}(\widetilde{N}\sigma) \\ a\#\widetilde{a}, \Psi\sigma, y\sigma \end{array}$$

Case sRep. In this case the inference looks like

$$\textsc{sRep}\ \frac{P \mid !P \xrightarrow[C]{\alpha} P'}{!P \xrightarrow[C]{\alpha} P'}$$

By induction $\Psi \rhd (P \mid !P)\sigma \xrightarrow{\alpha\sigma} P'\sigma$, so we can do the following derivation.

$$\textsc{cRep}\ \frac{\Psi \rhd P\sigma \mid !P\sigma \xrightarrow{\alpha\sigma} P'\sigma}{\Psi \rhd !P\sigma \xrightarrow{\alpha\sigma} P'\sigma}$$

Case sBrOut. Here the transition is derived by

$$\textsc{sBrOut}\ \frac{x\#, M, \widetilde{N}, P}{\overline{M}\,\widetilde{N}\,.\,P \xrightarrow[\mathbf{1}\vdash M \dot{\prec} x]{\overline{x}!\ \widetilde{N}} P}$$

We then have the corresponding concrete transition

$$\textsc{cBrOut}\ \frac{\Psi \vdash M\sigma \mathrel{\dot{\prec}} x\sigma}{\Psi \rhd (\overline{M}\widetilde{N}.P)\sigma \xrightarrow{(\overline{x}\widetilde{N})\sigma} P\sigma}$$

Case sBrIn. Here the transition is derived by

$$\textsc{sBrIn}\ \frac{x, \widetilde{y}\#\Psi, M, P \quad x\#\widetilde{y}}{\underline{M}(\widetilde{y})\,.\,P \xrightarrow[\mathbf{1}\vdash x \mathrel{\dot{\succ}} M]{\underline{x}?(\widetilde{y})} P}$$

Since $y\#\sigma$ we have that $(\underline{M}(\widetilde{y})\,.\,P)\sigma = \underline{M\sigma}(\widetilde{y})\,.\,P\sigma$ and that $(\underline{x}?(\widetilde{y}))\sigma = \underline{x\sigma}?(\widetilde{y})$. We then do the following derivation:

$$\textsc{cBrIn}\ \frac{\Psi \vdash x\sigma \dot{\succ} M\sigma}{\Psi \rhd \underline{M\sigma}(\widetilde{y}).P\sigma \xrightarrow{\underline{x\sigma}?(\widetilde{y})} P\sigma}$$

Case sBrMerge. Here the transition is derived by

$$\textsc{sBrMerge}\ \frac{P \xrightarrow[C_P]{\underline{x}?(\widetilde{y})} P' \qquad Q \xrightarrow[C_Q]{\underline{x}?(\widetilde{y})} Q'}{P \mid Q \xrightarrow[(\mathcal{F}(Q)\otimes C_P)\wedge(\mathcal{F}(P)\otimes C_Q)]{\underline{x}?(\widetilde{y})} P' \mid Q'}$$

By induction $\Psi \rhd P\sigma \xrightarrow{\underline{x\sigma}?(\widetilde{y})} P'\sigma$ and $\Psi \rhd Q\sigma \xrightarrow{\underline{x\sigma}?(\widetilde{y})} Q'\sigma$. By cBrMerge $\Psi \rhd P\sigma \mid Q\sigma \xrightarrow{\underline{x\sigma}?(\widetilde{y})} P'\sigma \mid Q'\sigma$.
Case sBrCom. Here the transition is derived by

$$\textsc{sBrCom}\ \frac{P \xrightarrow[C_P]{\overline{x}\ (\nu\widetilde{a})\widetilde{N}} P' \qquad Q \xrightarrow[C_Q]{\underline{x}?(\widetilde{y})} Q'}{P \mid Q \xrightarrow[(\mathcal{F}(Q)\otimes C_P)\wedge(\mathcal{F}(P)\otimes C_Q)]{\overline{x}\ (\nu\widetilde{a})\widetilde{N}} P' \mid Q'[\widetilde{y} := \widetilde{N}]}\ \widetilde{a}\#Q$$

By induction $\Psi \rhd P\sigma \xrightarrow{\overline{x\sigma}\ (\nu\widetilde{a})\widetilde{N\sigma}} P'\sigma$ and $\Psi \rhd Q\sigma \xrightarrow{\underline{x\sigma}?(\widetilde{y})} Q'\sigma$. By cBrCom $\Psi \rhd P\sigma \mid Q\sigma \xrightarrow{\overline{x\sigma}\ (\nu\widetilde{a})\widetilde{N\sigma}} P'\sigma \mid Q'[\widetilde{y} := \widetilde{N}]\sigma$.
Case sBrClose. Here the transition is derived by

$$\textsc{sBrClose}\ \frac{P \xrightarrow[C]{\overline{x}!\ (\nu\widetilde{a})\widetilde{N}} P'}{(\nu b)P \xrightarrow[\exists^b x.C]{\tau} (\nu b)(\nu\widetilde{a})P'}$$

By assumption there is K such that $b \in \mathrm{n}(K)$ and $(\sigma[x := K], \Psi) \models C$. We assume that $b\#\sigma$. Since $x\#P, \widetilde{a}$ we have $x\#P', \widetilde{N}$, so by induction $\Psi \rhd P\sigma \xrightarrow{\overline{K}!\ (\nu\widetilde{a})\widetilde{N\sigma}} P'\sigma$. We then have the following derivation.

$$\textsc{cBrClose}\ \frac{\Psi \rhd P\sigma \xrightarrow{\overline{K}!\ (\nu\widetilde{a})\widetilde{N\sigma}} P'\sigma}{\Psi \rhd (\nu b)P \xrightarrow{\tau} (\nu b)(\nu\widetilde{a})P'\sigma}\ b \in \mathrm{n}(K) \qquad \square$$

A.3. Proof of Completeness Theorem

The proof of the completeness theorem follows [Johansson et al. 2012], apart from the new cases for the broadcast rules, and the updated sCom rule.

Theorem A.19 (Theorem 7.12).

— *If* $\Psi \rhd P\sigma \xrightarrow{\tau} P'$ *then* $\exists C, Q\,.\, P \xrightarrow[C]{\tau} Q$, $Q\sigma = P'$ *and* $(\sigma, \Psi) \models C$.

— *If* $\Psi \rhd P\sigma \xrightarrow{\alpha} P'$, $\alpha \neq \tau$, $y\#P, \mathrm{bn}(\alpha), \sigma$, *and* $\mathrm{bn}(\alpha)\#\sigma, P$ *then* $\exists C, \alpha', Q.\ P \xrightarrow[C]{\alpha'} Q$, $Q\sigma = P'$, $\mathrm{subj}(\alpha') = y$, $\alpha'\sigma' = \alpha$, *and* $(\sigma', \Psi) \models C$ *where* $\sigma' = \sigma[y := \mathrm{subj}(\alpha)]$.

Proof. By induction on the inference of $\Psi \rhd P\sigma \xrightarrow{\alpha} P'\sigma$.

Case cIn. In this case the inference looks like

$$\textsc{cIn}\ \frac{\Psi \vdash M'\sigma \dot{\leftrightarrow} M}{\Psi \rhd (\underline{M'}(\widetilde{x}).P)\sigma \xrightarrow{\underline{M}(\widetilde{x})} P\sigma}$$

We know that $y\#\underline{M'}(\widetilde{x}) \,.\, P, \widetilde{x}, \sigma$ and that $\widetilde{x}\#\sigma, \underline{M'}(\widetilde{x}) \,.\, P$.
We let $Q = P$, and do the following derivation:

$$\textsc{sIn}\ \frac{}{\underline{M'}(\widetilde{x}).P \xrightarrow[\{\!|\mathbf{1} \vdash M' \leftrightarrows y|\!\}]{\underline{y}(\widetilde{x})} P}\ y\#M', P, \widetilde{x}$$

Since $\Psi \vdash M'\sigma \leftrightarrows M$ we have that $(\sigma[y := M], \Psi) \models \{\!|\mathbf{1} \vdash M' \leftrightarrows y|\!\}$.
Case cOut. In this case the inference looks like

$$\textsc{cOut}\ \frac{\Psi \vdash M'\sigma \leftrightarrows M}{\Psi \rhd (\overline{M'}\widetilde{N} \,.\, P)\sigma \xrightarrow{\overline{M}\widetilde{N}\sigma} P\sigma}$$

We know that $y\#M', \widetilde{N}, P$. We must find a constraint C such that $\overline{M'}\widetilde{N} \,.\, P \xrightarrow[C]{\overline{y}\widetilde{N}} P$ and $(\sigma[y := M], \Psi) \models C$. We let $Q = P$, $\widetilde{K} = \widetilde{N}$, and derive such a transition with

$$\textsc{sOut}\ \frac{}{\overline{M'}\widetilde{N} \,.\, P \xrightarrow[\{\!|\mathbf{1} \vdash M' \leftrightarrows y|\!\}]{\overline{y}\widetilde{N}} P}\ y\#M', \widetilde{N}, P$$

Since $\Psi \vdash M'\sigma \leftrightarrows M$ we have that $(\sigma[y := M], \Psi) \models \{\!|\mathbf{1} \vdash M' \leftrightarrows y|\!\}$.
Case cCase. In this case the inference looks like

$$\textsc{cCase}\ \frac{\Psi \rhd P_i\sigma \xrightarrow{\alpha} P' \qquad \Psi \vdash \varphi_i\sigma}{\Psi \rhd (\mathbf{case}\ \widetilde{\varphi} : \widetilde{P})\sigma \xrightarrow{\alpha} P'}$$

$\alpha = \tau$. By induction we know that $\Psi \rhd P_i\sigma \xrightarrow{\tau} P'$ has a matching transition $P_i \xrightarrow[C]{\tau} Q$ such that $(\sigma, \Psi) \models C$ and $Q\sigma = P'$. We also have that $(\sigma, \Psi) \models \{\!|\mathbf{1} \vdash \varphi_i|\!\}$. Together this gives us that $(\sigma, \Psi) \models C \wedge \{\!|\mathbf{1} \vdash \varphi_i|\!\}$.
$\alpha \neq \tau$. Since $y\#\mathbf{case}\ \widetilde{\varphi} : \widetilde{P}$ we have in particular that $y\#\varphi_i, P_i$. By induction we know that $\Psi \rhd P_i\sigma \xrightarrow{\alpha} P'$ has a matching transition $P_i \xrightarrow[C]{\alpha'} Q$ such that $(\sigma', \Psi) \models C$ and $Q\sigma = P'$. Since $\Psi \vdash \varphi_i\sigma$ we have that $(\sigma, \Psi) \models \{\!|\mathbf{1} \vdash \varphi_i|\!\}$, and since $y\#\varphi_i$ we also have that $(\sigma', \Psi) \models \{\!|\mathbf{1} \vdash \varphi_i|\!\}$. Together this gives us that $(\sigma', \Psi) \models C \wedge \{\!|\mathbf{1} \vdash \varphi_i|\!\}$.
We can then do the following derivation:

$$\textsc{sCase}\ \frac{P_i \xrightarrow[C]{\alpha_s} Q}{\mathbf{case}\ \widetilde{\varphi} : \widetilde{P} \xrightarrow[C \wedge \{\!|\mathbf{1} \vdash \varphi_i|\!\}]{\alpha_s} Q}$$

Case cCom. The interesting case is the cCom case, where the inference looks like

$$\textsc{cCom}\ \frac{\begin{array}{c}\Psi \otimes \Psi_{P\sigma} \otimes \Psi_{Q\sigma} \vdash M \leftrightarrows K \\ \Psi \otimes \Psi_{Q\sigma} \rhd P\sigma \xrightarrow{\overline{M}\,(\nu\widetilde{a})\widetilde{N}\sigma} P'\sigma \qquad \Psi \otimes \Psi_{P\sigma} \rhd Q\sigma \xrightarrow{\underline{K}(\widetilde{x})} Q'\sigma\end{array}}{\Psi \rhd (P \mid Q)\sigma \xrightarrow{\tau} (\nu\widetilde{a})(P' \mid Q'[\widetilde{x} := \widetilde{N}])\sigma}\ \widetilde{a}\#Q\sigma$$

Here $\mathcal{F}(P\sigma) = (\nu\widetilde{b}_{P\sigma})\Psi_{P\sigma}$ and $\mathcal{F}(Q\sigma) = (\nu\widetilde{b}_{Q\sigma})\Psi_{Q\sigma}$. We know that $\widetilde{b}_{P\sigma}\#\Psi, P\sigma, Q\sigma, \widetilde{b}_{Q\sigma}$ and $\widetilde{b}_{Q\sigma}\#\Psi, P\sigma, Q\sigma, \widetilde{b}_{P\sigma}, P$. We assume that $\widetilde{a}\#P, \widetilde{b}_{Q\sigma}, \sigma$ and $\widetilde{x}\#\widetilde{b}_{P\sigma}$. Let $y, z\# \ldots$. By Lemma A.11 we also have that $\widetilde{b}_{Q\sigma}\#P'\sigma$. By Lemma A.6 we get $\mathcal{F}(P) = (\nu\widetilde{b}_{P\sigma})\Psi_P$ with $\Psi_P\sigma \simeq \Psi_{P\sigma}$ and $\mathcal{F}(Q) = (\nu\widetilde{b}_{Q\sigma})\Psi_Q$ with $\Psi_Q\sigma \simeq \Psi_{Q\sigma}$.

By induction, $P \xrightarrow[C_P]{\overline{y}\,(\nu\widetilde{a})\widetilde{N'}} P'$ such that $\widetilde{N} = \widetilde{N'}\sigma$ and $(\sigma[y := M], \Psi \otimes \Psi_{Q\sigma}) \models C_P$.
By Lemma 7.10 $C_P = ((\nu\widetilde{c}_P)(\Psi'_P \vdash M_P \leftrightarrow y) \wedge C'_P$ with $(\nu\widetilde{c}_P)\Psi'_P \equiv_{\mathrm{AC}} (\nu\widetilde{b}_P)\Psi_P$.
In the same way, by induction $Q \xrightarrow[C_Q]{\underline{z}(\widetilde{x})} Q'$ such that $(\sigma[z := K], \Psi \otimes \Psi_{P\sigma}) \models C_Q$.
By Lemma 7.10 $C_Q = ((\nu\widetilde{c}_Q)(\Psi'_Q \vdash M_Q \leftrightarrow z) \wedge C'_Q$ with $(\nu\widetilde{c}_Q)\Psi'_Q \equiv_{\mathrm{AC}} (\nu\widetilde{b}_Q)\Psi_Q$.
We can then do the following inference:

$$\textsc{sCom}\ \frac{P \xrightarrow[(\nu\widetilde{c}_P)\{\!|\Psi'_P \vdash M_P \leftrightarrow y|\!\} \wedge C'_P]{\overline{y}\,(\nu\widetilde{a})\widetilde{N'}} P' \qquad Q \xrightarrow[(\nu\widetilde{c}_Q)\{\!|\Psi'_Q \vdash M_Q \leftrightarrow z|\!\} \wedge C'_Q]{\underline{z}(\widetilde{x})} Q'}{P \mid Q \xrightarrow[C]{\tau} (\nu\widetilde{a})(P' \mid Q'[\widetilde{x} := \widetilde{N'}])}\ \widetilde{a}\#Q$$

where $C = (\nu\widetilde{c}_P\widetilde{c}_Q)\{\!|\Psi'_P \otimes \Psi'_Q \vdash M_P \leftrightarrow M_Q|\!\} \wedge ((\nu\widetilde{c}_Q)\Psi'_Q \otimes C'_P) \wedge ((\nu\widetilde{c}_P)\Psi'_P \otimes C'_Q)$. It remains to show that $(\sigma, \Psi) \models C$. We consider each conjunct in turn.
By Lemma A.8 $(\sigma[z := K], \Psi \otimes \Psi_{P\sigma} \models (\nu\widetilde{b}_Q)(\Psi_Q \vdash M_Q \leftrightarrow z)$ and $(\sigma[y := M], \Psi \otimes \Psi_{Q\sigma}) \models (\nu\widetilde{b}_P)(\Psi_P \vdash M_P \leftrightarrow y)$. Thus $\Psi \otimes \Psi_{P\sigma} \otimes \Psi_Q\sigma \vdash M_Q\sigma \leftrightarrow K$ and $\Psi \otimes \Psi_{Q\sigma} \otimes \Psi_P\sigma \vdash M_P\sigma \leftrightarrow M$. By AC of entailment of $\leftrightarrow$ and $\otimes$ modulo $\simeq$ $\Psi \otimes \Psi_P\sigma \otimes \Psi_Q\sigma \vdash M_P\sigma \leftrightarrow M_Q\sigma$. By Lemma A.4 $\Psi \otimes \Psi'_P\sigma \otimes \Psi'_Q\sigma \vdash M_P\sigma \leftrightarrow M_Q\sigma$, so $(\sigma, \Psi) \models \Psi'_P\sigma \otimes \Psi'_Q \vdash M_P\sigma \leftrightarrow M_Q\sigma$. By Lemma A.2 $(\sigma, \Psi) \models ((\nu\widetilde{c}_P\widetilde{c}_Q)\Psi'_P \otimes \Psi'_Q \vdash M_P \leftrightarrow M_Q)$, which is the first conjunct.
By Lemma A.6 $(\sigma[y := M], \Psi \otimes \Psi_Q\sigma) \models C_P$ so by Lemma A.7 $(\sigma[y := M], \Psi) \models \mathcal{F}(Q) \otimes C'_P$. Since $y\#Q, C'_P$ we have $(\sigma, \Psi) \models \mathcal{F}(Q) \otimes C'_P$. By Lemma A.8 $(\sigma, \Psi) \models ((\nu\widetilde{c}_Q)\Psi'_Q) \otimes C'_P$.
In the same way, by Lemma A.6 $(\sigma[z := K], \Psi \otimes \Psi_P\sigma) \models C_Q$ so by Lemma A.7 $(\sigma[z := K], \Psi) \models \mathcal{F}(P) \otimes C'_Q$, Since $z\#P, C'_Q$ we have $(\sigma, \Psi) \models \mathcal{F}(P) \otimes C'_Q$. By Lemma A.8 $(\sigma, \Psi) \models ((\nu\widetilde{c}_P)\Psi'_P) \otimes C'_Q$.

Case cPar. In this case the inference looks like

$$\textsc{cPar}\ \frac{\Psi \otimes \Psi_{Q\sigma} \rhd P\sigma \xrightarrow{\alpha} P'\sigma}{\Psi \rhd (P \mid Q)\sigma \xrightarrow{\alpha} (P' \mid Q)\sigma}\ \mathrm{bn}(\alpha)\#Q\sigma$$

where $\mathcal{F}(Q\sigma) = (\nu\widetilde{b}_{Q\sigma})\Psi_{Q\sigma}$ with $\widetilde{b}_{Q\sigma}\#\Psi, \Psi\sigma, P\sigma, \alpha, y$. By Lemma A.6 $\mathcal{F}(Q) = (\nu\widetilde{b}_{Q\sigma})\Psi_Q$ such that $\Psi_{Q\sigma} \simeq \Psi_Q\sigma$ and $\mathrm{n}(\Psi_{Q\sigma}) = \mathrm{n}(\Psi_Q\sigma)$.
By Lemma A.11 we also have that $\widetilde{b}_{Q\sigma}\#P'\sigma$. Since $y\#\widetilde{b}_{Q\sigma}, Q, \sigma$ we get that $y\#\Psi_Q\sigma$. Together with $y\#\sigma$ this gives us that $y\#\Psi_Q$.
By induction we know that $\Psi \otimes \Psi_{Q\sigma} \rhd P\sigma \xrightarrow{\alpha} P'\sigma$ has a matching transition $P \xrightarrow[C]{\alpha'} P'$ such that $(\sigma', \Psi \otimes \Psi_{Q\sigma}) \models C$.
We can then do the following symbolic inference:

$$\textsc{sPar}\ \frac{P \xrightarrow[C]{\alpha'} P'}{P \mid Q \xrightarrow[\mathcal{F}(Q)\otimes C]{\alpha'} P' \mid Q}\ \begin{matrix}\widetilde{x}\#Q \\ y\#Q\end{matrix}$$

Lemma A.6 yields that $(\sigma', \Psi \otimes \Psi_Q\sigma) \models C$, so by Lemma A.7 $(\sigma', \Psi) \models \mathcal{F}(Q) \otimes C$.
Case cScope. In this case the transition is

$$\Psi \rhd ((\nu a)P)\sigma \xrightarrow{\alpha} ((\nu a)P')\sigma$$

Let b be a sufficiently fresh name, and let $p = (a\ b)$. By applying the substitution and using α-conversion to avoid capture, this transition is equivalent to

$$\Psi \rhd (\nu b)((p \cdot P)\sigma) \xrightarrow{\alpha} (\nu b)((p \cdot P')\sigma)$$

This transition is inferred like

$$\textsc{cScope}\ \frac{\Psi \rhd (p \cdot P)\sigma \xrightarrow{\alpha} (p \cdot P')\sigma}{\Psi \rhd (\nu b)((p \cdot P)\sigma) \xrightarrow{\alpha} (\nu b)((p \cdot P')\sigma)}\ b\#\alpha, \Psi$$

We know that $y\#(\nu a)P$. Since $y\#(\nu b)(p \cdot P)$ and $b\#y$ we have that $y\#p \cdot P$.
By induction we have that $\Psi \rhd (p \cdot P)\sigma \xrightarrow{\alpha} (p \cdot P')\sigma$ has a matching transition $p \cdot P \xrightarrow[p \cdot C]{\alpha'} p \cdot P'$ such that $(\sigma', \Psi) \models C$.
We pick $b\#\sigma, \alpha$, so $b\#\alpha'$ by Axiom 1.
We then do the following symbolic inference:

$$\textsc{sScope}\ \frac{p \cdot P \xrightarrow[p \cdot C]{\alpha'} p \cdot P'}{(\nu b)(p \cdot P) \xrightarrow[(\nu b)p \cdot C]{\alpha'} (\nu b)(p \cdot P')}\ b\#\alpha$$

Since $(\sigma', \Psi) \models C$ and $b\#\sigma, \Psi, y, \mathrm{subj}(\alpha)$ we also have that $(\sigma', \Psi) \models (\nu b)p \cdot C$.
By α-converting the final transition we get that

$$(\nu a)(P) \xrightarrow[(\nu a)C]{\alpha'} (\nu a)(P')$$

Case cOpen. In this case the transition looks like

$$(\nu a)P\sigma \xrightarrow{\overline{M}\ (\nu \widetilde{a} \cup \{a\})\widetilde{N\sigma}} P'\sigma$$

Let b be a sufficiently fresh name, and let $p = (a\ b)$. By applying the substitution and using α-conversion to avoid capture, this transition is equivalent to

$$(\nu b)((p \cdot P)\sigma) \xrightarrow{\overline{M}\ (\nu \widetilde{a} \cup \{b\})\widetilde{(p \cdot \widetilde{N})\sigma}} (p \cdot P')\sigma$$

This transition is inferred like

$$\textsc{cOpen}\ \frac{\Psi \rhd (p \cdot P)\sigma \xrightarrow{\overline{M}\ (\nu \widetilde{a})\widetilde{(p \cdot \widetilde{N})\sigma}} (p \cdot P')\sigma}{\Psi \rhd (\nu b)((p \cdot P)\sigma) \xrightarrow{\overline{M}\ (\nu \widetilde{a} \cup \{b\})\widetilde{(p \cdot \widetilde{N})\sigma}} (p \cdot P')\sigma}\ \begin{array}{l} b \in \mathrm{n}((p \cdot \widetilde{N})\sigma) \\ b\#\widetilde{a}, \Psi\sigma, M \end{array}$$

We know that $y\#(\nu a)P, x$, $x\#\sigma, (\nu a)P$. Since $y\#(\nu b)(p \cdot P)$ and $b\#y$ we have that $y\#p \cdot P$, and similarly we get that $x\#p \cdot P$
By induction we have that $\Psi \rhd (p \cdot P)\sigma \xrightarrow{\overline{M}\ (\nu \widetilde{a})\widetilde{(p \cdot \widetilde{N})\sigma}} (p \cdot P')\sigma$ has a matching transition $p \cdot P \xrightarrow[p \cdot C]{\overline{y}\ (\nu \widetilde{a})\widetilde{(p \cdot \widetilde{N})}} p \cdot P'$ such that $(\sigma[y := M], \Psi) \models p \cdot C$.
We then infer:

$$\textsc{sOpen}\ \frac{p \cdot P \xrightarrow[p \cdot C]{\overline{y}\ (\nu \widetilde{a})\widetilde{p \cdot \widetilde{N}}} p \cdot P'}{(\nu b)(p \cdot P) \xrightarrow[(\nu b)p \cdot C]{\overline{y}\ (\nu \widetilde{a} \cup \{b\})\widetilde{p \cdot \widetilde{N}}} p \cdot P'}\ \begin{array}{l} b \in \mathrm{n}(p \cdot \widetilde{N}) \\ b\#\widetilde{a}, y \end{array}$$

Since $b\#\sigma, \Psi, M, y$ and we have that $(\sigma', \Psi) \models p \cdot C$ we also have that $(\sigma', \Psi) \models (\nu b)p \cdot C$.

By α-converting the final transition we get:

$$(\nu a)P \xrightarrow[(\nu a)C]{\overline{y}\,(\nu\widetilde{a}\cup\{a\})\widetilde{N}} P'$$

Case cRep. In this case the inference looks like

$$\textsc{cRep}\ \frac{\Psi \rhd P\sigma \mid !P\sigma \xrightarrow{\alpha} P'\sigma}{\Psi \rhd !P\sigma \xrightarrow{\alpha} P'\sigma}$$

If $\alpha \neq \tau$ and $y\#!P, \mathrm{bn}(\alpha)$, we get that $y\#P, !P, \mathrm{bn}(\alpha)$.
By induction we then get that $P \mid !P \xrightarrow[C]{\alpha'} P'$ and that $(\sigma', \Psi) \models C$.
We do the following derivation

$$\textsc{sRep}\ \frac{P \mid !P \xrightarrow[C]{\alpha'} P'}{!P \xrightarrow[C]{\alpha'} P'}$$

Case cBrIn. Here the transition is derived by

$$\textsc{cBrIn}\ \frac{\Psi \vdash K \mathrel{\dot{\succ}} M\sigma \qquad \widetilde{x}\#\Psi, M\sigma, P\sigma}{\Psi \rhd \underline{M\sigma}(\widetilde{x})\,.\,P\sigma \xrightarrow{\underline{K?(\widetilde{x})}} P\sigma}$$

We know that $y\#\underline{M}(\widetilde{x})\,.\,P, \widetilde{x}, \sigma$ and assume that $\widetilde{x}\#\sigma, M$.
We let $Q = P$, and do the following derivation:

$$\textsc{sBrIn}\ \frac{\widetilde{x}, y\#M, P \quad y\#\widetilde{x}}{\underline{M}(\widetilde{x})\,.\,P \xrightarrow[\mathbf{1}\vdash y\mathrel{\dot{\succ}}M]{\underline{y}?(\widetilde{x})} P}$$

Since $\Psi \vdash K \mathrel{\dot{\succ}} M\sigma$ we get $(\sigma[y := K], \Psi) \models \mathbf{1} \vdash y \mathrel{\dot{\succ}} M$.
Case cBrOut. Here the transition is derived by

$$\textsc{cBrOut}\ \frac{\Psi \vdash M\sigma \mathrel{\dot{\prec}} K}{\Psi \rhd \overline{M\sigma}\,\widetilde{N\sigma}\,.\,P\sigma \xrightarrow{\overline{K}!\,\widetilde{N\sigma}} P\sigma}$$

We know that $y\#\overline{M}\,\widetilde{N}\,.\,P, \sigma$. We let $Q = P$, and do the following derivation:

$$\textsc{sBrOut}\ \frac{x\#M, N, P}{\overline{M}\,\widetilde{N}\,.\,P \xrightarrow[\mathbf{1}\vdash M\mathrel{\dot{\prec}}x]{\overline{x}!\,\widetilde{N}} P}$$

Case cBrMerge. Here the transition is derived by

$$\textsc{cBrMerge}\ \frac{\Psi \otimes \Psi_{Q\sigma} \rhd P\sigma \xrightarrow{\underline{K?(\widetilde{y})}} P' \qquad \Psi \otimes \Psi_{P\sigma} \rhd Q\sigma \xrightarrow{\underline{K?(\widetilde{y})}} Q'}{\Psi \rhd P\sigma \mid Q\sigma \xrightarrow{\underline{K?(\widetilde{y})}} P' \mid Q'}$$

Here $\mathcal{F}(P\sigma) = (\nu\widetilde{b}_{P\sigma})\Psi_{P\sigma}$ and $\mathcal{F}(Q\sigma) = (\nu\widetilde{b}_{Q\sigma})\Psi_{Q\sigma}$. We know that $\widetilde{b}_{P\sigma}\#\Psi, P\sigma, Q\sigma, \widetilde{b}_{Q\sigma}$ and $\widetilde{b}_{Q\sigma}\#\Psi, P\sigma, Q\sigma, \widetilde{b}_{P\sigma}, P$. We assume that $x\#\widetilde{b}_{P\sigma}$.
By induction $P \xrightarrow[C_P]{\underline{x}(\widetilde{y})} P''$ and $Q \xrightarrow[C_Q]{\underline{x}(\widetilde{y})} Q''$ such that $(\sigma[x := K], \Psi \otimes \Psi_{Q\sigma}) \models C_P$ and $(\sigma[x := K], \Psi \otimes \Psi_{P\sigma}) \models C_Q$ and $P' = P''\sigma$ and $Q' = Q''\sigma$.

We then have the following derivation.

$$\textsc{sBrMerge}\ \frac{P\ \xrightarrow[C_P]{\underline{x}?(\widetilde{y})}\ P'' \qquad Q\ \xrightarrow[C_Q]{\underline{x}?(\widetilde{y})}\ Q''}{P \mid Q\ \xrightarrow[(\mathcal{F}(Q)\otimes C_P)\wedge(\mathcal{F}(P)\otimes C_Q)]{\underline{x}?(\widetilde{y})}\ P'' \mid Q''}$$

By Lemma A.6 we get $\mathcal{F}(P) = (\nu\widetilde{b}_{P\sigma})\Psi_P$ with $\Psi_P\sigma \simeq \Psi_{P\sigma}$ and $\mathcal{F}(Q) = (\nu\widetilde{b}_{Q\sigma})\Psi_Q$ with $\Psi_Q\sigma \simeq \Psi_{Q\sigma}$. By Lemma A.7 $(\sigma[x := K], \Psi) \models \mathcal{F}(Q) \otimes C_P$ and $(\sigma[x := K], \Psi) \models \mathcal{F}(P) \otimes C_Q$.
Case cBrCom. Here the transition is derived by

$$\textsc{cBrCom}\ \frac{\begin{array}{c}\Psi \otimes \Psi_{Q\sigma} \rhd P\sigma\ \xrightarrow{\overline{K}\ (\nu\widetilde{a})\widetilde{N}}\ P' \\ \Psi \otimes \Psi_{P\sigma} \rhd Q\sigma\ \xrightarrow{\underline{K}?(\widetilde{x})}\ Q'\end{array}}{\Psi \rhd P\sigma \mid Q\sigma\ \xrightarrow{\overline{y}\ (\nu\widetilde{a})\widetilde{N}}\ P' \mid Q'[y := N]}\ \widetilde{a}\#Q\sigma$$

Here $\mathcal{F}(P\sigma) = (\nu\widetilde{b}_{P\sigma})\Psi_{P\sigma}$ and $\mathcal{F}(Q\sigma) = (\nu\widetilde{b}_{Q\sigma})\Psi_{Q\sigma}$. We know that $\widetilde{b}_{P\sigma}\#\Psi, P\sigma, Q\sigma, \widetilde{b}_{Q\sigma}$ and $\widetilde{b}_{Q\sigma}\#\Psi, P\sigma, Q\sigma, \widetilde{b}_{P\sigma}, P$. We assume that $\widetilde{x}\#P, \widetilde{b}_{P\sigma}$.
By induction $P\ \xrightarrow[C_P]{\overline{y}\ (\nu\widetilde{a})\widetilde{M}}\ P''$ and $Q\ \xrightarrow[C_Q]{\underline{y}(\widetilde{x})}\ Q''$ such that $(\sigma[y := K], \Psi \otimes \Psi_{Q\sigma}) \models C_P$ and $(\sigma[y := K], \Psi \otimes \Psi_{P\sigma}) \models C_Q$ and $P' = P''\sigma$ and $\widetilde{M}\sigma = \widetilde{N}$ and $Q' = Q''\sigma$.
We then have the following derivation.

$$\textsc{sBrCom}\ \frac{P\ \xrightarrow[C_P]{\overline{y}\ (\nu\widetilde{a})\widetilde{M}}\ P'' \qquad Q\ \xrightarrow[C_Q]{\underline{y}?(\widetilde{x})}\ Q''}{P \mid Q\ \xrightarrow[(\mathcal{F}(Q)\otimes C_P)\wedge(\mathcal{F}(P)\otimes C_Q)]{\overline{y}\ (\nu\widetilde{a})\widetilde{M}}\ P'' \mid Q''[\widetilde{y} := \widetilde{N}]}\ \widetilde{a}\#Q$$

By Lemma A.6 we get $\mathcal{F}(P) = (\nu\widetilde{b}_{P\sigma})\Psi_P$ with $\Psi_P\sigma \simeq \Psi_{P\sigma}$ and $\mathcal{F}(Q) = (\nu\widetilde{b}_{Q\sigma})\Psi_Q$ with $\Psi_Q\sigma \simeq \Psi_{Q\sigma}$. By Lemma A.7 $(\sigma[y := K], \Psi) \models \mathcal{F}(Q) \otimes C_P$ and $(\sigma[y := K], \Psi) \models \mathcal{F}(P) \otimes C_Q$.
Case cBrClose. Here the transition is derived by

$$\textsc{cBrClose}\ \frac{\Psi \rhd P\sigma\ \xrightarrow{\overline{K}!\ (\nu\widetilde{a})\widetilde{N}}\ P'}{\Psi \rhd (\nu b)P\sigma\ \xrightarrow{\tau}\ (\nu b)(\nu\widetilde{a})P'}\ b \in \mathrm{n}(K)$$

By induction $P\ \xrightarrow[C]{\overline{y}!\ (\nu\widetilde{a})\widetilde{M}}\ P''$ such that $\widetilde{M}\sigma = \widetilde{N}$ and $P''\sigma = P'$ and $(\sigma[y := K], \Psi) \models C$. We then do

$$\textsc{sBrClose}\ \frac{P\ \xrightarrow[C]{\overline{x}!\ (\nu\widetilde{a})\widetilde{M}}\ P'}{(\nu b)P\ \xrightarrow[\exists^b x.C]{\tau}\ (\nu b)(\nu\widetilde{a})P'}$$

where $(\sigma, \Psi) \models \exists^b x.C$. $\square$