Compositional Equivalences Based on Open pNets

Rabéa Ameur-Boulifa, Ludovic Henrio, Eric Madelaine

Compositional Equivalences Based on Open pNets

Rabéa Ameur-Boulifa

LTCI, Télécom Paris, Institut Polytechnique de Paris, France

Ludovic Henrio*

Univ Lyon, EnsL, UCBL, CNRS, Inria, LIP, F-69342, LYON Cedex 07, France.

Eric Madelaine

INRIA Sophia Antipolis Méditérannée, UCA, BP 93, 06902 Sophia Antipolis, France

Abstract

Establishing equivalences between programs is crucial both for verifying correctness of programs and for justifying optimisations and program transformations. There exist several equivalence relations for programs, and bisimulations are among the most versatile of these equivalences. Among bisimulations one distinguishes strong bisimulation that requires that each action of a program is simulated by a single action of the equivalent program, and weak bisimulation that allows some of the actions to be invisible, and thus not simulated.

pNet is a generalisation of automata that model open systems. They feature variables and hierarchical composition. Open pNets are pNets with holes, i.e. placeholders that can be filled later by sub-systems. However, there is no standard tool for defining the semantics of an open system in this context. This article first defines *open automata* that are labelled transition systems with parameters and holes. Relying on open automata, it then defines bisimilarity relations for the comparison of systems specified as pNets. We first present a strong bisimilarity for open pNets called FH-bisimilarity. Next we offer an equivalence relation similar to the classical *weak bisimulation* equivalence, and study its properties. Among these properties we are interested in compositionality: if two systems are proven equivalent they will be indistinguishable by their context, and they will also be indistinguishable when their holes are filled with equivalent systems. We identify sufficient conditions to ensure compositionality of strong and weak bisimulation. The contributions of this article are illustrated using a transport protocol as running example.

Keywords: Bisimulation, compositionality, automata, semantics.

*Corresponding author

Email addresses: rabea.ameur-boulifa@telecom-paris.fr (Rabéa Ameur-Boulifa), ludovic.henrio@cnrs.fr (Ludovic Henrio), eric.madelaine@inria.fr (Eric Madelaine)

1. Introduction

In the nineties, several works extended the basic behavioural models based on labelled transition systems to address value-passing or parameterised systems, using various symbolic encodings of the transitions [15, 35]. These works use the term *parameter* to designate variables whose value have a strong influence the system structure and behaviour. In parameterised systems, parameters can typically be the number of processes in the system or the way they interact. In [32, 25], Lin, Ingolfsdottir and Hennessy developed a full hierarchy of bisimulation equivalences, together with a proof system, for value passing CCS, including notions of symbolic behavioural semantics and various symbolic bisimulations (early and late, strong and weak, and their congruent versions). They also extended this work to models with explicit assignments [38]. Separately Rathke [26] defined another symbolic semantics for a parameterised broadcast calculus, together with strong and weak bisimulation equivalences, and developed a symbolic model-checker based on a tableau method for these processes. Thirty years later, no verification platform uses this kind of approaches to provide proof methods for value-passing processes or open process expressions, perhaps because of the difficulty to apply these methods on industrial systems.

This article provides a theoretical background that allows us to implement such a verification platform. We build upon the concept of pNets that we have employed to give a behavioural semantics of distributed components and verify the correctness of distributed applications in the past 15 years. pNets is a low level semantic framework for expressing the behaviour of various classes of distributed languages, and as a common internal format for our tools. pNets support the specification of parameterised hierarchical labelled transition systems: labelled transition systems with parameters can be combined hierarchically.

We develop here a semantics for a model of interacting processes with parameters and holes. Our approach is originally inspired from Structured Operational Semantics with conditional premises as in [20, 45]. But we aim at a more constructive and implementable approach to compute the semantics (intuitively transitions including first order predicates) and to check equivalences for these open systems. The main interest of our symbolic approach is to define a method to prove properties directly on open structures; these properties will then be preserved by any correct instantiation of the holes. As a consequence, our model allows us to reason about composition operators as well as about realistic distributed systems. The parametric nature of the model and the properties of compositionality of the equivalence relations are thus the main strengths of our approach.

pNets. pNet is a convenient model to model concurrent systems in a hierarchical and parameterised way. The coordination between processes is expressed as synchronisation vectors that allow for the definition of complex and expressive synchronisation patterns. Open pNets are pNets for which some elements in the

hierarchy are still undefined, such undefined elements are called *holes*. A hole can be *filled* later by providing another pNet. This article first defines pNets and illustrates with an example how they can be used to provide the model of a communicating system.

A semantics for open pNets based on open automata. The semantics of pNets can be expressed as a translation to a labelled transition system (LTS), but only if the pNet has no parameter and no hole. Adding parameters to a LTS is quite standard [38] but enabling holes inside LTSs is not a standard notion.

To define a semantics for open pNets we thus need LTSs that have both standard variable parameters, and process parameters, i.e. holes that can be filled by processes. We call such LTSs with parameters and holes *open automata*. The main goal of this article is to define the theory behind open automata and to use them to provide a semantics and prove compositionality properties for open pNets. The transitions of open automata are much more complex than transitions of an LTS as the firing of a transition depends on parameters and actions that are symbolic. This article defines the notion of *open transition*, namely a transition that is symbolic in terms of parameters and coordinated actions.

Note that even if open transitions look similar to the notion of Transition System Specification [23, 22] and other forms of SOS rules, they are *not* structural rules, but rules defining the behaviour of the global states of the system.

Unlike pNets, open automata are not hierarchical structures, we consider them here as a mathematical structure that is convenient for formal reasoning but not adapted to the definition of a complex and structured system. Open transitions are expressed in terms of logics while the communication in pNets is specified as synchronisation vectors that specify synchronised actions. Open automata could alternatively be seen as an algebra that can be studied independently from its application to pNets but their compositionality properties make more sense in a hierarchical model like pNets.

Previous works and contribution

While most of our previous works relied on closed, fully-instantiated semantics [6, 2, 27], it is only recently that we could design a first version of a parameterised semantics for pNets with a strong bisimulation equivalence [28]. This article builds upon this previous parameterised semantics and provides a clean and complete version of the semantics with a slightly simplified formalism that makes proofs easier. It also adds a notion of global state to automata. Moreover, in [28] the study of compositionality was only partial, and in particular the proof that bisimilarity is an equivalence is one new contribution of this article and provides a particularly interesting insight on the semantic model we use. The new formalism allowed us to extend the work and define weak bisimulation for open automata, which is entirely new. This allows us to define a weak bisimulation equivalence for open pNets with valuable compositionality properties. To summarise, the contribution of this paper are the following:

- The definition of open automata: an algebra of parameterised automata with holes, and a strong bisimulation relation. This is an adaptation of [28] with an additional result stating that strong FH-bisimilarity is indeed an equivalence relation.
- A semantics for open pNets expressed as translation to open automata. This is an adaptation of [28] with a complete proof that strong FH-bisimilarity is compositional.
- A theory of weak bisimulation for open automata, and a study of its properties. It relies on the definition of weak open transitions that are derived from transitions of the open automaton by concatenating invisible action transitions with one (visible or not) action transition. The precise and sound definition of the concatenation is also a major contribution of this article.
- A resulting weak FH-bisimilarity equivalence for open pNets and a simple static condition on synchronisation vectors inside pNets that is sufficient to ensure that weak FH-bisimilarity is compositional.
- An illustrative example based on a transport protocol, showing the construction of the weak open transitions, and the proof of weak FH-bisimulation.

What is new about open automata bisimulation?

Bisimulation over a symbolic and open model like open pNets or open automata is different from the classical notion of bisimulation because it cannot rely on the equality over a finite set of action labels. Classical bisimulations require to exhibit, for each transition of one system, a transition of the other system that simulates it. Instead, bisimulation for open automata relies on the simulation of each open transition of one automaton by *a set of* open transitions of the other one, that should cover all the cases where the original transition can be triggered. This is similar to the early and late symbolic bisimulation equivalences for value-passing CCS [25], though we use more general definitions in our setting.

Compositionality of bisimilarity in our model comes from the specification of the interactions, including actions of the holes. This is quite different from the works on contextual equivalences, e.g. [35, 36]; we will provide a detailed comparison in Section 6. In pNets, synchronisation vectors define the possible interactions between the pNet that fills the hole and the surrounding pNets. In open automata, this is reflected by symbolic hypotheses that depend on the actions of the holes. This additional specification is the price to pay to obtain the compositionality of bisimilarity that cannot be guaranteed in traditional process algebras.

This approach also allows us to specify a sufficient condition on transitions to make weak bisimilarity compositional; namely it is not possible to synchronise on invisible actions from the holes or prevent them to occur. This is loosely related to works on the syntactic conditions on SOS rules to check whether

weak bisimulation is a congruence for some process algebra operators [23]. Our approach is semantical and more global: our sufficient condition applies to all the synchronisations at a given composition level of an (open) system and not on individual rules. It is expressed on the open automaton (see Definition 15).

Structure

This article is organised as follows. Section 2 provides the definition of pNets and introduces the notations used in this paper, including the definition of open pNets. Section 3 defines open automata, i.e. automata with parameters and transitions conditioned by the behaviour of "holes"; a strong bisimulation equivalence for open automata is also presented in this section. Section 4 gives the semantics of open pNets expressed as open automata, and states compositionality properties of strong bisimilarity for open pNets. Section 5 defines a weak bisimulation equivalence on open automata and derives weak bisimilarity for pNets, together with compositionality properties of weak bisimilarity. Finally, Section 6 discusses related works and Section 7 concludes the paper.

2. Background and Notations

This section introduces the notations we will use in this article, and recalls the definition of pNets [28] with an informal semantics of the pNet constructs. The only significant difference compared to our previous definitions (from [28]) is that we remove here the restriction that was stating that variables should be local to a state of a labelled transition system.

2.1. Notations

Term algebra. Our models rely on a notion of parameterised actions, which are symbolic expressions using data types and variables. As our model aims at encoding the low-level behaviour of possibly very different programming languages, we do not want to impose one specific algebra for denoting actions, nor any specific communication mechanism. So we leave the constructors of the algebra that will be used to build expressions and actions unspecified. Moreover, we use a generic *action interaction* mechanism, based on (some sort of) unification between two or more action expressions, to express various kinds of communication or synchronisation mechanisms.

Formally, we assume the existence of a term algebra $\mathbb{T}$, and denote as Σ the signature of the data and action constructors. Within $\mathbb{T}$, we distinguish a set of data expressions $\mathbb{E}$, including a set of boolean expressions $\mathbb{B}$ ($\mathbb{B} \subseteq \mathbb{E}$), and a set of action expressions called the action algebra $\mathbb{A}$, with $\mathbb{A} \subseteq \mathbb{T}, \mathbb{E} \cap \mathbb{A} = \emptyset$; naturally action terms will use data expressions as sub-terms[1]. The function $vars(t)$ identifies the set of variables in a term $t \in \mathbb{T}$.

[1] In our tools, we use datatypes for the different kinds of terms. In this article, we use different sets of variables for terms of different kinds.

We let e_i range over expressions ($e_i \in \mathbb{E}$), op be operators, and x_i and y_i range over variable names. We additionally rely on a set of action names, ranged over by a, b, ... We define two kinds of parameterised actions. The first kind supports two kinds of parameters: input parameters that are variables and output parameters that can be any expression. The second kind makes no distinction between input and output parameters. The actions that distinguish input variables will be used in the definition of *pLTS* and are defined as follows:

$$\begin{array}{rcll} \alpha \in \mathbb{A} & ::= & a(p_1, \ldots, p_n) & \text{action terms} \\ p_i & ::= & ?x \mid e_i & \text{parameters (input var or expression)} \\ e_i & ::= & \mathit{Value} \mid x \mid op(e_1, .., e_n) & \text{Expressions} \end{array}$$

The *input variables* in an action term are those marked with a *?*. We additionally impose that each input variable does not appear anywhere else in the same action term: $p_i = ?x \Rightarrow \forall j \neq i.\, x \notin \mathit{vars}(p_j)$. We define $iv(t)$ as the set of input variables of a term t (without the '?' marker). Input variables are used in guards and to update the local state, they can only appear in well-identified expressions. Action algebras can encode naturally usual point-to-point message passing calculi (using $a(?x_1, ..., ?x_n)$ for inputs, $a(v_1, .., v_n)$ for outputs), but they also allow for more general synchronisation mechanisms, like gate negotiation in Lotos, or broadcast communications.

The set of actions that do not distinguish input variables is denoted $\mathbb{A}_S$, it will be used in synchronisation vectors of pNets:

$$\alpha \in \mathbb{A}_S \quad ::= \quad a(e_1, \ldots, e_n)$$

Indexed sets. This article extensively uses indexed structures (maps) over some countable indexed sets. The indices can typically be integers, bounded or not. We use indexed sets in pNets because we want to consider a set of processes, and specify separately how to synchronise them. Roughly this could also be realised using tuples, however indexed sets are more general, can be infinite, and give a more compact representation than using the position in a possibly long tuple.

An indexed family is denoted as follows: $t_i^{i \in I}$ is a family of elements t_i indexed over the set I. Such a family is equivalent to the mapping $(i \mapsto t_i)^{i \in I}$, and we will also use mapping notations to manipulate indexed sets. To specify the set over which the structure is indexed, indexed structures are always denoted with an exponent of the form $i \in I$. Consequently, $t_i^{i \in I}$ defines first I the set over which the family is indexed, and then t_i the elements of the family. For example $t_i^{i \in \{3\}}$ is the mapping with a single entry t_3 at index 3; exceptionally, for mappings with only a few entries we use the notation $(3 \mapsto t_3)$ instead. In this article, sentences of the form "there exists $t_i^{i \in I}$" means there exist I and a function that maps each element of I to a term t_i.

When this is not ambiguous, we shall abuse notations for sets, and typically write "indexed set over I" when formally we should speak of multisets, and "$x \in A_i^{i \in I}$" to mean $\exists i \in I.\, x = A_i$. To simplify equations, an indexed set can be denoted $\bar{t}$ instead of $t_i^{i \in I}$ when I is irrelevant or clear from the context.

The disjoint union on sets is $\uplus$ and we only use $A \uplus B$ when A and B are disjoint. We extend it to union of indexed sets provided they are indexed over disjoint families; $\uplus$ is then defined by the merge of the two sets. The elements of the union of two indexed sets are then accessed by using an index of one of the two joined families. The subtraction operation on indexed sets is $\setminus$, it reduces the set of indices such that $\mathrm{dom}(A \setminus B) = \mathrm{dom}(A) \setminus B$.

Substitutions. This article also uses substitutions. Applying a substitution inside a term t is denoted $t\{\!\{y_i \leftarrow e_i\}\!\}^{i \in I}$ and consists in replacing in parallel all the occurrences of variables y_i in the term t by the terms e_i. Note that a substitution is defined by a partial function that is applied on the variables inside a term. We let *Post* range over partial functions that are used as substitution and use the notation $\{y_i \leftarrow e_i\}^{i \in I}$ to define such a partial function[2]. These partial functions are sometimes called *substitution functions* in the following. Thus, $\{\!\{Post\}\!\}$ is the operation that applies, in a parallel manner, the substitution defined by the partial function *Post*. $\odot$ is a composition operator on these partial functions, such that for any term t we have: $t\{\!\{Post \odot Post'\}\!\} = (t\{\!\{Post'\}\!\})\{\!\{Post\}\!\}$. This property must also be valid when the substitution does not operate on all variables. We thus define a composition operation as follows:

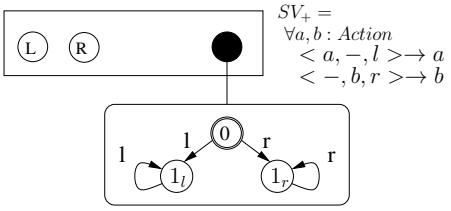
$$(x_k \leftarrow e_k)^{k \in K} \odot (x'_{k'} \leftarrow e'_{k'})^{k' \in K'} = (x_k \leftarrow e_k\{\!\{(x'_{k'} \leftarrow e'_{k'})^{k' \in K'}\}\!\})^{k \in K} \cup (x'_{k'} \leftarrow e'_{k'})^{k' \in K''}$$

where $K'' = \{k' \in K' | x'_{k'} \notin \{x_k\}^{k \in K}\}$

2.2. The principles of Parameterised Networks (pNets)

pNets are tree-like structures, where the leaves are either *parameterised labelled transition systems (pLTSs)*, expressing the behaviour of basic processes, or *holes*, used as placeholders for unknown processes. Every node of the tree is a pNet, it acts as a synchronising artefacts, using a set of *synchronisation vectors* that express the possible synchronisation between the actions of a subset of the sub-trees. The pNets model is hierarchical in the structure of the processes, this contrasts with Statecharts [24] that model high-level behaviour by organising the states (but not processes) in a hierarchy. We introduce pNets through a simple example below, and define formally pLTSs and pNets afterwards:

Example 1 (CCS choice). Here is the encoding of a choice operator.



It consists of one pNet (Definition 2 below) with two holes and a subnet. The pNet is represented by the top box with three circles and two synchronisation vectors on the right. The sub-net is a pLTS that is represented by the bottom box.

[2]When using this notation, we suppose, without loss of generality that each y_i is different.

Each hole is represented by an empty disc, when the hole is filled it becomes a black disc. The left hole is indexed by L and the right hole by R. The sub-net is a LTS with three states and emitting actions l and r. In the synchronisation vector a and b range over arbitrary action terms , e.g. including action parameters; l and r on the contrary are specific actions. The set of synchronisation vectors is infinite but admits a simple finite representation.

The behaviour of the pNet is defined with synchronisation vectors also shown on the figure. In the examples, we write them on the form $< a, -, l > \rightarrow a$. This states that if the first hole L performs the action a and the third sub-net, i.e. the LTS, performs the action l, both of them progress synchronously, and an action a is emitted by the pNet. The symbol $-$ at the second position denotes that the second hole does nothing. On the formal side, numbering and ordering the vectors is cumbersome, this is why we adopt indexed families of actions. The LTS is sometimes called the "control part", it controls the evolution of the rest of the pNet. The first action of one of the holes decides which branch of the LTS is activated; all subsequent actions will be performed by the same side.

2.3. Parameterised Labelled Transition systems (pLTS)

A pLTS is a labelled transition system with variables; variables can be used inside states, actions, guards, and assignments. Note that we make no assumption on finiteness of the set of states nor on finite branching of the transition relation. Compared to our previous works [28, 2] we only global variables, which makes the model easier to use.

Definition 1 (pLTS). A pLTS is a tuple $pLTS \triangleq \langle\!\langle S, s_0, V, \rightarrow \rangle\!\rangle$ where:

- S is a set of states.
- $s_0 \in S$ is the initial state.
- V is a set of global variables for the pLTS.
- $\rightarrow \subseteq S \times L \times S$ is the transition relation and L is the set of labels. Labels have the form:
  $\langle \alpha,\ e_b,\ (x_j := e_j)^{j \in J} \rangle$, where $\alpha \in \mathbb{A}$ is a parameterised action, $e_b \in \mathbb{B}$ is a guard, and the variables x_j (that are pairwise distinct) are assigned the expressions $e_j \in \mathbb{E}$. If $s \xrightarrow{\langle \alpha,\ e_b,\ (x_j := e_j)^{j \in J} \rangle} s' \in \rightarrow$ then $vars(\alpha) \backslash iv(\alpha) \subseteq V$, $vars(e_b) \subseteq V \cup vars(\alpha)$, and $\forall j \in J.\ (vars(e_j) \subseteq V \cup iv(\alpha) \wedge x_j \in V)$.

A set of assignments between two states is performed in parallel so that their order do not matter and they all use the values of variables before the transition or the values received as action parameters.

2.4. Parameterised Networks (pNets)

Now we define pNet nodes as constructors for hierarchical behavioural structures. A pNet has a set of sub-pNets that can be either pNets or pLTSs, and

a set of holes, playing the role of process parameters. A pNet is thus a composition operator that can receive processes as parameters; it expresses how the actions of the sub-processes synchronise.

Each sub-pNet exposes a set of actions, called *internal actions*. *Synchronisation vectors* define the synchronisation between global actions exposed by the pNet and internal actions of its sub-pNets: it synchronises one or several internal actions, and exposes a single resulting global action.

We now define the structure of pNets, the following definition relies on the definition of holes, leaves and sorts formalised below in Definition 3. Informally, holes are process parameters, leaves provide the set of pLTSs at the leaves of the hierarchical structure of a pNet, and sorts give the signature of a pNet, i.e. the actions it exposes.

Definition 2 (pNets). A pNet P is a hierarchical structure where leaves are pLTSs and holes

$$P \triangleq pLTS \mid \langle\!\langle P_i^{i\in I}, Sort_j^{j\in J}, SV_k^{k\in K} \rangle\!\rangle$$

We denote $vars(P)$ the set of variables used by the pLTSs inside P and $Sort(P)$ the signature of the actions emitted by P; both are defined below, in Definition 3. A pNet is composed of the following:

- I is a set of indices and $P_i^{i\in I}$ is the family of sub-pNets indexed over I. $vars(P_i)$ and $vars(P_j)$ must be disjoint for $i \neq j$.
- J is a set of indices, called *holes*. I and J are *disjoint*: $I \cap J = \emptyset$, $I \cup J \neq \emptyset$.
- $Sort_j \subseteq \mathbb{A}_S$ is a set of action terms, denoting the *sort* of hole j.
- $SV_k^{k\in K}$ is a set of synchronisation vectors.
  $\forall k \in K.\, SV_k = \alpha_l^{l\in I_k \uplus J_k} \rightarrow \alpha'_k[e_k]$ where $\alpha'_k \in \mathbb{A}_S$, $I_k \subseteq I$, $J_k \subseteq J$, $\forall i \in I_k.\, \alpha_i \in Sort(P_i)$, $\forall j \in J_k.\, \alpha_j \in Sort_j$, and $vars(\alpha'_k) \subseteq \bigcup_{l\in I_k \uplus J_k} vars(\alpha_l)$. The global action of a vector SV_k is α'_k. $e_k \in \mathbb{B}$ is a guard associated to the vector such that $vars(e_k) \subseteq \bigcup_{l\in I_k \uplus J_k} vars(\alpha_l)$.

Synchronisation vectors are identified modulo renaming of variables that appear in their action terms, e.g. the vectors $< a(x), b(x) > \rightarrow \tau$ and $< a(y), b(y) > \rightarrow \tau$ are equivalent.

The preceding definition relies on the auxiliary functions defined below:

Definition 3 (Sorts, holes, leaves, variables of pNets).

- The sort of a pNet is its signature, i.e. the set of actions in $\mathbb{A}_S$ it can perform, where each action signature is an action label plus the arity of the action.

$$\begin{aligned}
&\mathrm{Sort}(\langle\!\langle S, s_0, V, \rightarrow \rangle\!\rangle) = \{\mathrm{Sort}(\alpha) | s \xrightarrow{\langle \alpha,\ e_b,\ (x_j := e_j)^{j\in J} \rangle} s' \in \rightarrow\} \\
&\mathrm{Sort}(\langle\!\langle \overline{P}, \overline{Sort}, \overline{SV} \rangle\!\rangle) = \{\mathrm{Sort}(\alpha') | \, \overline{\alpha} \rightarrow \alpha'[e_b] \in \overline{SV}\} \\
&\mathrm{Sort}(\alpha(p_1, .., p_n)) = (\alpha, n)
\end{aligned}$$

- The set of variables of a pNet P, denoted $vars(P)$ is disjoint union the set of variables of all pLTSs that compose P.
- The set of holes $\text{Holes}(P)$ of a pNet is the set of indices of the holes of the pNet itself plus the indices of all the holes of its sub-pNets. It is defined inductively (we suppose that those index sets are disjoint):

$$\begin{array}{l}\text{Holes}(\langle\!\langle S, s_0, V, \rightarrow \rangle\!\rangle) = \emptyset \\ \text{Holes}(\langle\!\langle P_i^{i\in I}, Sort_j^{j\in J}, \overline{SV} \rangle\!\rangle) = J \uplus \bigcup_{i\in I} \text{Holes}(P_i) \\ \forall i \in I.\ \text{Holes}(P_i) \cap J = \emptyset \\ \forall i_1, i_2 \in I.\, i_1 \neq i_2 \Rightarrow \text{Holes}(P_{i_1}) \cap \text{Holes}(P_{i_2}) = \emptyset\end{array}$$

- The set of leaves of a pNet is the set of all pLTSs occurring in the structure, as an indexed family of the form $\text{Leaves}(P) = \langle\!\langle P_i \rangle\!\rangle^{i\in L}$.

$$\begin{array}{l}\text{Leaves}(\langle\!\langle S, s_0, V, \rightarrow \rangle\!\rangle) = \emptyset \\ \text{Leaves}(\langle\!\langle P_i^{i\in I}, \overline{Sort}, \overline{SV} \rangle\!\rangle) = \biguplus_{i\in I} \text{Leaves}(P_i) \uplus \{i \mapsto P_i | P_i \text{ is a } pLTS\}\end{array}$$

For example, the controller of Example 1 has the sort $\{l, r\}$ and holes $\{L, R\}$. Note that $\text{Holes}(P)$ is a set of indices because holes are characterized only by their indices, while entities at the leaves are pLTSs and thus $\text{Leaves}(P)$ is a set of pLTSs. A pNet Q is *closed* if it has no hole: $\text{Holes}(Q) = \emptyset$; else it is said to be *open*. Sort comes naturally with a compatibility relation that is similar to a type-compatibility check. We simply say that two sorts are compatible if they consist of the same actions with the same arity. In practice, it is sufficient to check the equality of the two sets of action signatures of the two sorts[3].

The informal semantics of pNets is as follows. pLTSs behave more or less like classical automata with conditional branching and variables. The actions on the *pLTS*s can send or receive values, potentially modifying the value of variables. pNets are synchronisation entities: a pNet node composes several sub-pNets and defines how the sub-pNets interact, where a sub-pNet is either a pNet or a pLTS. The synchronisation between sub-pNets is defined by synchronisation vectors (originally introduced in [3]) that express how an action of a sub-pNet can be synchronised with actions of other sub-pNet, and how the resulting synchronised action is visible from outside of the pNet. The synchronisation mechanism is very expressive, including pattern-matching/unification between the parameterized actions within the vector, and an additional predicate over their variables. Consider a pNet node that assembles several pLTSs, the synchronisation vectors specify the way that transitions of the composed pNet are built from the transitions of the sub-nets. This can be seen as "conditional transitions" of the pNet, or alternatively, as a syntax to encode structural

[3] A more complex compatibility relation could be defined, but this is out of the scope of this article.

operational semantics (SOS rules) [42] of the system: each vector expresses not only the actions emitted by the pNet but also what transitions of the composed pLTSs must occur to trigger this global transition. Synchronisation vectors can also express the exportation of an action of a sub-pNet to the next level, or to hide an interaction and make it non-observable. Finally, a pNet can leave sub-pNets undefined and instead declare holes with a well-defined signature. Holes can then be filled with a sub-pNet. This is defined as follows.

Definition 4 (pNet composition). An open pNet: $P = \langle\!\langle P_i^{i\in I}, Sort_j^{j\in J}, \overline{SV} \rangle\!\rangle$ can be (partially) filled by providing a pNet Q to fill one of its holes. Suppose $j_0 \in J$ and $\mathrm{Sort}(Q) \subseteq Sort_{j_0}$, then:

$$P\,[Q]_{j_0} = \langle\!\langle P_i^{i\in I} \uplus \{j_0 \mapsto Q\}, Sort_j^{j\in J\setminus\{j_0\}}, \overline{SV} \rangle\!\rangle$$

pNets are composition entities equipped with a rich synchronisation mechanism: synchronisation vectors allow the expression of synchronisation between any number of entities and at the same time the passing of data between processes. Their strongest feature is that the data emitted by processes can be used inside the synchronisation vector to do addressing: it is easy to synchronise a process indexed by n with the action $a(v, n)$ of another process. This is very convenient to model systems and encode futures or message routing.

pNets have been used to model distributed components using the Grid Component Model, illustrating the expressiveness of the model [2]. These works show that pNets are convenient to express the behaviour of a system in a compositional way. Unfortunately, the semantics of pNets and the existing tools at that point were only able to deal with a closed and completely instantiated system: pNets could be used as composition operators in the definition of the semantics, which was sufficient to perform finite-state model checking on a closed system, but there was no theory for the use of pNets as operators and no tool for proving properties on open system. Consequently, much of the formalisation efforts did not use holes and the interplay between holes, sorts, and synchronisation vector was not formalised. In previous works [2], only closed pNets were equipped with a semantics, which was defined as labelled transition systems. The theory of pNets as operators for open systems is given in the following sections. Comparing formally the existing direct operational semantics and the semantics derived from open automata in the current article would be an interesting partial proof of soundness for our semantics. The proof could only be partial as the formal semantics that exists only consider closed and fully instantiated pNets. Proving an equivalence between the semantics presented in this article and the operational one shown in [2] is outside the scope of this article because we focus here on the modelling of holes that were not considered in the previous semantics. It is however easy to see that, in case there is no hole the structure of the open automaton that defines the semantics here is very close to the pLTS that is used to define the semantics, even though the formalisms are slightly different.

2.5. Running example

To illustrate this work, we use a simple communication protocol, that provides safe transport of data between two processes, over unsafe media.

Figure 1 (left) shows the example principle, which corresponds to the hierarchical structure of a pNet: two unspecified processes P and Q (holes) communicate messages, with a data value argument, through the two protocol entities. Process P sends a `p-send(m)` message to the *Sender*; this communication is denoted as `in(m)`. At the other end, process Q receives the message from the *Receiver*. The holes P and Q can also have other interactions with their environment, represented here by actions `p-a` and `q-b`. The underlying network is modelled by a medium entity transporting messages from the sender to the receiver, and that is able to detect transport errors and signal them to the sender. The return *ack* message from *Receiver* to *Sender* is supposed to be safe. The final transmission of the message to the recipient (the hole Q) includes the value of the "error counter" *ec*.

Figure 1 (right) shows a graphical view of the pNet *SimpleProtocolSpec* that specifies the system. The pNet is made of the composition of two pNets: a *SimpleSystem* node, and a *PerfectBuffer* sub-pNet. The full system implementation should be equivalent (e.g. weakly bisimilar) to this *SimpleProtocolSpec*. The pNet has a tree-like structure. The root node of the tree *SimpleSystem* is the top level of the pNet structure. It acts as the parallel operator. It consists of three nodes: two holes P and Q and one sub-pNet, denoted *PerfectBuffer*. Nodes of the tree are synchronised using four synchronisation vectors, that express the possible synchronisations between the parameterised actions of a subset of the nodes. For instance, in the vector $<$ `p-send(m)`, `in(m)`, _ $>\rightarrow$ `in(m)` only P and *PerfectBuffer* nodes are involved in the synchronisation. The synchronisation between these processes occurs when process P performs `p-send(m)` action sending a message, and the *PerfectBuffer* accepts the message through an `in(m)` action at the same time; the result that will be returned at upper level is the action `in(m)`.

Figure 2 shows the pNet model of the protocol implementation, called *SimpleProtocolImpl*. When the *Medium* detects an error (modelled by a local τ action), it sends back a `m-error` message to the *Sender*. The *Sender* increments its local error counter *ec*, and resends the message (including *ec*) to the *Medium*, that will, eventually, transmit m, ec to the *Receiver*.

3. A model of process composition

The semantics of open pNets will be defined as an open automaton. An open automaton is an automaton where each transition composes transitions of several LTSs with action of some holes, the transition occurs if some predicates hold, and can involve a set of state modifications. This section defines open automata and a bisimulation theory for them. This section is an improved version of the formalism described in [28], extending the automata with a notion of global variable, which makes the state of the automaton more explicit. We

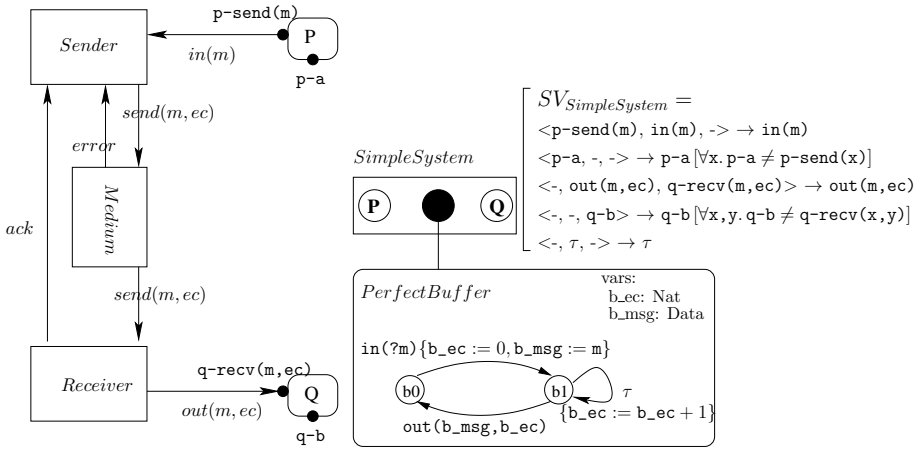


Figure 1: pNet structure of the example and its specification expressed as a pNet called *SimpleProtocolSpec*

also adopt a semantics and logical interpretation of the automata that intuitively can be stated as follows: "if a transition belongs to an open automaton, any refinement of this transition also belongs to the automaton". Our open automata are clearly inspired by the work of De Simone on SOS rule [15] formats. A precise comparison with related works can be found in Section 6.

3.1. Open automata

Open automata (OA) are not composition structures but they are made of transitions that are dependent of the actions of the holes, and they can use variables (potentially with only symbolic values).

Definition 5 (Open transitions). An *open transition* (OT) over a set J of holes with sorts $Sort_j^{j \in J}$, a set V of variables, and a set of states $\mathcal{S}$ is a structure of the form:

$$\frac{\beta_j^{j \in J'}, Pred, Post}{s \xrightarrow{\alpha} s'}$$

where $J' \subseteq J$ is the set of holes involved in the transition; $s, s' \in \mathcal{S}$ are states of the automaton; and β_j is a transition of the hole j, with $\mathrm{Sort}(\beta_j) \subseteq Sort_j$. α is the resulting action of this open transition. *Pred* is a predicate, *Post* is a set of assignments that are effective after the open transition, and are represented as a substitution function: $(x_k \leftarrow e_k)^{k \in K}$. Predicates and expressions of an open transition can refer to the variables inside V and the different terms β_j and α.

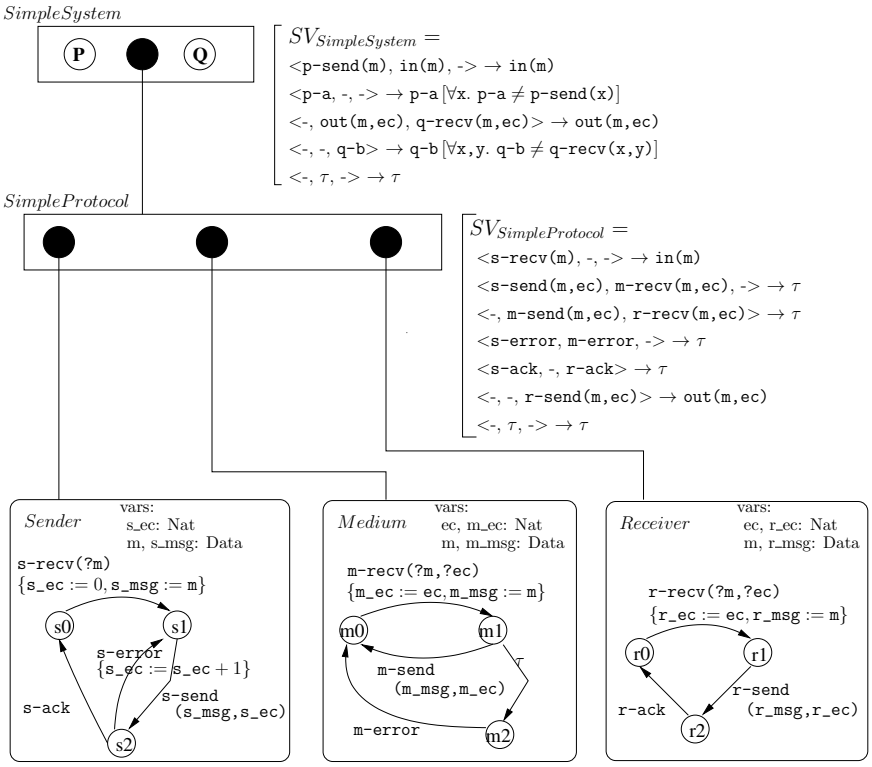


Figure 2: The *SimpleProtocolImpl* pNet resulting from the composition of the *SimpleSystem* and the *SimpleProtocol* pNets.

More precisely:

$$vars(Pred) \subseteq V \cup vars(\alpha) \cup \bigcup_{j \in J'} vars(\beta_j) \quad \wedge$$
$$\forall k.\, x_k \in V \quad \wedge \quad \forall k.\, vars(e_k) \subseteq V \cup vars(\alpha) \cup \bigcup_{j \in J'} vars(\beta_j)$$

The assignments are applied simultaneously because the variables in V can be in both sides (x_ks are distinct). Open transitions are identified modulo logical equivalence on their predicate.

It is important to understand the difference between the red dotted rule and a classical inference rule. They correspond to two different logical levels. On one side, classical (black) inference rules act at the mathematical level of the paper proofs (as e.g. the rules in Definition 13). They use an expressive logic (like any other computer science article). On the other side, open transition rules (with dotted lines) are logical implications that belong to the open automata algebra. Their logic has a specific syntax that can be mechanized; this logic includes the boolean expressions $\mathbb{B}$, boolean operators, and term equality.

An open automaton is an automaton where transitions are open transitions.

Definition 6 (Open automaton). An *open automaton* is a structure $A = \langle\!\langle J, \mathcal{S}, s_0, V, \mathcal{T} \rangle\!\rangle$ where:

- J is a set of indices.
- $\mathcal{S}$ is a set of states and s_0 is an initial state belonging to $\mathcal{S}$.
- V is a set of variables of the automaton and each $v \in V$ may have an initial value $init(v)$.
- $\mathcal{T}$ is a set of open transitions and for each $t \in \mathcal{T}$ there exists J' with $J' \subseteq J$, such that t is an open transition over J' and $\mathcal{S}$.

While the definition and usage of the open transition can be considered purely syntactically, we take in this article a semantic and logical understanding of open automata. We see open transitions as logical formulas with a constrained syntax and logics rather than purely syntactical terms. Consequently, the open transition sets in open automata are closed by a simple form of refinement that allows us to refine the predicate, or substitute any free variable by an expression. Formally, for each predicate $Pred$ for each partial function $Post$, if $V \cap \mathrm{dom}(Post) = \emptyset$, we have:

$$\frac{\overline{\beta}, Pred', Post'}{s \xrightarrow{\alpha} s'} \in \mathcal{T} \quad \Longrightarrow \quad \frac{\overline{\beta}\{\!\{Post\}\!\}, Pred'\{\!\{Post\}\!\} \wedge Pred, Post \odot Post'}{s \xrightarrow{\alpha\{\!\{Post\}\!\}} s'} \in \mathcal{T}$$

Because of the semantic interpretation of open automata, the set of open transition of an open automaton is infinite (for example because every free variable can be substituted by any term). This raises an issue when a finite representation is needed, which is the case both in our tools, and when writing examples. When needed, we can rely on a *canonical representation* of the open automaton, provided that a finite subset of the open transitions is sufficient to generate, by substitution, the other ones. Thus, we use this canonical representation in our examples. In the following, we will abusively write that we define an "open automaton" when we provide its canonical representation.

Another aspect of the semantic interpretation is that we consider terms up to semantic equivalence, i.e. equivalence of two predicates $Pred$ and $Pred'$ can be denoted $Pred = Pred'$, where the $=$ symbol is interpreted semantically.

Though the definition is simple, the fact that transitions are complex structures relating events must not be underestimated. The first element of theory for open automata, i.e. the definition of a strong bisimulation, is given below.

3.2. Bisimulation for open Automata

We define now a bisimulation relation tailored to open automata and their parametric nature. This relation relates states of the open automata and guarantees that the related states are observationally equivalent, i.e. equivalent

states can trigger transitions with identical action labels. Its key characteristics are 1) the introduction of predicates in the bisimulation relation: the relation between states may depend on the value of the variables; 2) bisimulation relates elements of the open transitions and takes into account predicates over variables, actions of the holes, and state modifications. We name it FH-bisimulation, as a short cut for the "Formal Hypotheses" over the holes behaviour manipulated in the transitions, but also as a reference to the work of De Simone [15], that pioneered this idea. Indeed, our definition uses both hypotheses on the behaviour of holes, as in [15], and symbolic manipulation of action expressions, as in symbolic bisimulations of [25].

One of the original aspects of FH-bisimulation is due to the symbolic nature of open automata. Indeed, a single state of the automaton represents a potentially infinite number of concrete states, depending on the value of the automaton variables, and a single open transition of the automaton may also be instantiated with an unbounded number of values for the transition parameters. Consequently it would be too restrictive to impose that each transition of one automaton is matched by exactly one transition of the bisimilar automaton. Thus the definition of bisimulation requires that, for each open transition of one automaton, there exists a matching set of open transitions covering the original one. Indeed depending on the value of action parameters or automaton variables, different open transitions might simulate the same one.

The parametric nature of the automata entails a second original aspect of FH-bisimulation: the nature of the bisimulation relation itself. A classical relation between states can be seen as a function mapping pairs of states to a boolean value (true if the states are related, false if they are not). An FH-bisimulation relation maps pairs of states to boolean expressions that use variables of the two systems. Formally, a relation over the states of two open automata $\langle\!\langle J, \mathcal{S}_1, s_0, V_1, \mathcal{T}_1 \rangle\!\rangle$ and $\langle\!\langle J, \mathcal{S}_2, t_0, V_2, \mathcal{T}_2 \rangle\!\rangle$ has the signature $\mathcal{S}_1 \times \mathcal{S}_2 \to \mathbb{B}$. We suppose without loss of generality that the variables of the two open automata are disjoint. We adopt a notation similar to standard relations and denote it $\mathcal{R} = \{(s, t | \mathit{Pred}_{s,t})\}$, where: 1) For any pair $(s, t) \in \mathcal{S}_1 \times \mathcal{S}_2$, there is a single $(s, t | \mathit{Pred}_{s,t}) \in \mathcal{R}$ stating that s and t are related if $\mathit{Pred}_{s,t}$ is True, i.e. the states are related when the value of the automata variables satisfy the predicate $\mathit{Pred}_{s,t}$. 2) The free variables of $\mathit{Pred}_{s,t}$ belong to V_1 and V_2, i.e. $\mathit{vars}(\mathit{Pred}_{s,t}) \subseteq V_1 \cup V_2$. FH-bisimulation is defined formally[4]:

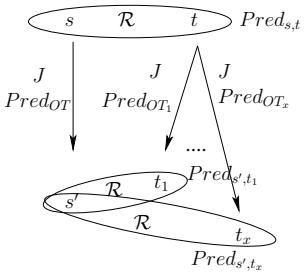
Definition 7 (Strong FH-bisimulation).
Suppose $A_1 = \langle\!\langle J, \mathcal{S}_1, s_0, V_1, \mathcal{T}_1 \rangle\!\rangle$ and $A_2 = \langle\!\langle J, \mathcal{S}_2, t_0, V_2, \mathcal{T}_2 \rangle\!\rangle$ are open automata with identical holes of the same sort, with disjoint sets of variables ($V_1 \cap V_2 = \emptyset$).

Then $\mathcal{R}$ is an FH-bisimulation if and only if for all states $s \in \mathcal{S}_1$ and $t \in \mathcal{S}_2$, $(s, t | \mathit{Pred}_{s,t}) \in \mathcal{R}$, we have the following:

- For any open transition OT in $\mathcal{T}_1$:

[4]In this article, we denote β_{jx} a double indexed set, instead of the classical $\beta_{j,\,x}$. Indeed the standard notation would be too heavy in our case.

$$\frac{\beta_j^{j\in J'}, Pred_{OT}, Post_{OT}}{s \xrightarrow{\alpha} s'}$$



there exists an indexed set of open transitions $OT_x^{x\in X} \subseteq \mathcal{T}_2$:

$$\frac{\beta_{jx}^{j\in J_x}, Pred_{OT_x}, Post_{OT_x}}{t \xrightarrow{\alpha_x} t_x}$$

such that $\forall x.\, J' = J_x$ and there exists some $Pred_{s',t_x}$ such that $(s', t_x | Pred_{s',t_x}) \in \mathcal{R}$ and

$$Pred_{s,t} \wedge Pred_{OT} \implies \bigvee_{x\in X} (\forall j.\beta_j = \beta_{jx} \wedge Pred_{OT_x} \wedge \alpha = \alpha_x \wedge Pred_{s',t_x}\{\!\{Post_{OT} \uplus Post_{OT_x}\}\!\})$$

- and symmetrically any open transition from t in $\mathcal{T}_2$ can be covered by a set of transitions from s in $\mathcal{T}_1$.

Two open automata are FH-bisimilar if there exists an FH-bisimulation that relates their initial states[5]. We call this relation *FH-bisimilarity*.

Classically, $Pred_{s',t_x}\{\!\{Post_{OT} \uplus Post_{OT_x}\}\!\}$ applies in parallel the substitution defined by the partial functions $Post_{OT}$ and $Post_{OT_x}$ (parallelism is crucial inside each *Post* set but not between $Post_{OT}$ and $Post_{OT_x}$ that are independent), applying the assignments of the involved rules. We can prove that bisimilarity is an equivalence relation.

Note that, if there is a FH-bisimulation $\mathcal{R}$ such that $(s, t|Pred) \in \mathcal{R}$, and additionally $Pred' \Rightarrow Pred$ and it does not mean that there is a FH-bisimulation $\mathcal{R}'$ such that $(s, t|Pred')$, indeed pair of states such that $Pred$ is true and $Pred'$ is false might be necessary to prove the FH-bisimulation.

Example 2. The simulation of one transition by many others is one non-standard aspect of this definition. This is made necessary by the parameterised nature of our model. Consider the following open transition.

$$\frac{\overline{\beta}, True, \{\!\{y \leftarrow x\}\!\}}{s_1 \xrightarrow{\alpha(x)} s_1'}$$

Bisimulation should allow it to be matched by the two following ones (depending on the value of x), to prove that the relation $\mathcal{R} = \{(s_1, s_2, True), (s_1', s_2', True)\}$ is a bisimulation.

$$\frac{\overline{\beta}, x \geq 0, \{\!\{y \leftarrow x\}\!\}}{s_2 \xrightarrow{\alpha(x)} s_2'} \qquad \frac{\overline{\beta}, x < 0, \{\!\{y \leftarrow x\}\!\}}{s_2 \xrightarrow{\alpha(x)} s_2'}$$

[5] In other words, the predicate relation associated to the initial states is *True*.

This example illustrates the necessity of multiple transitions in the definition of bisimulation in a naive and minimalistic way. It can easily be extended into a non-trivial example with more states and different usage of the variables.

Theorem 1 (FH-bisimilarity is an equivalence). *FH-bisimilarity is reflexive, symmetric and transitive.*

The proof of this theorem can be found in Appendix A.1. The only non-trivial part of the proof is about transitivity. It relies on the following elements. First, the transitive composition of two relations with predicate is defined; this is not exactly standard as it requires to define the right predicate for the transitive composition and producing a single predicate to relate any two states. Then the fact that one open transition is simulated by a family of open transitions leads to a doubly indexed family of simulating open transition; this needs particular care, also because of the use of renaming (*Post*) when proving that the predicates satisfy the definition (property on $Pred_{s,t} \wedge Pred_{OT}$ in the definition).

Finite versus infinite open automata, and decidability

As mentioned in page 6, we adopt here a semantic view on open automata. More precisely, in [29], we define *semantic open automata* (infinite as in Definition 6), and *structural open automata* (finite) that can be generated as the semantics of pNets (see Definition 9), and used in their implementation. Then we define an alternative version of our bisimulation, called structural FH-bisimulation, based on structural open automata, and prove that the *semantic* and *structural* FH-bisimulations coincide. In the sequel, all mentions of finite automata, and algorithms for bisimulations, implicitly refer to their *structural* versions.

If we assume that everything is finite (states and transitions in the open automata), then it is easy to prove that it is decidable whether a relation is a FH-bisimulation, provided the logic of the predicates is decidable (a proof of this claim can be found in [28]). Formally:

Theorem 2 (Decidability of FH-bisimulation). *Let A_1 and A_2 be finite open automata and $\mathcal{R}$ a relation over their states $\mathcal{S}_1$ and $\mathcal{S}_2$ constrained by a set of predicates. Assume that the predicate inclusion is decidable over the action algebra $\mathbb{A}$. Then it is decidable whether the relation $\mathcal{R}$ is an FH-bisimulation.*

4. Semantics of Open pNets

This section defines the semantics of an open pNet as a translation into an open automaton. In this translation, the states of the open automaton are obtained as products of the states of the pLTSs at the leaves of the composition. The predicates on the transitions result both from the predicates on the transitions of the pLTSs, and from the synchronisation vectors involved in the transition.

The definition of bisimulation for open automata allows us to derive a bisimilarity relation for open pNets. As pNets are composition structures, it then makes sense to prove compositionality lemmas: we prove that the composition of strongly bisimilar pNets are themselves bisimilar.

4.1. Deriving an open automaton from an open pNet

To derive an open automaton from a pNet, we first describe the set of states of the automaton. Then we show the construction rule for transitions of the automaton, which relies on the derivation of predicates unifying synchronisation vectors and the actions of the pNets involved in a given synchronisation.

States of open pNets are tuples of states. We denote them as $\triangleleft \dots \triangleright$ for distinguishing tuple states from other tuples.

Definition 8 (States of open pNets). A state of an open pNet is a (not necessarily finite) tuple of the states of its leaves.

For any pNet P, let Leaves$(P) = \langle\!\langle S_i, s_{i0}, V, \rightarrow_i \rangle\!\rangle^{i \in L}$ be the set of pLTS at its leaves, then $States(P) = \{\triangleleft s_i^{i \in L} \triangleright \,| \forall i \in L. s_i \in S_i\}$. A pLTS being its own single leave: $States(\langle\!\langle S, s_0, V, \rightarrow \rangle\!\rangle) = \{\triangleleft s \triangleright \,| s \in S\}$.

The initial state is defined as: $InitState(P) = \triangleleft {s_{i0}}^{i \in L} \triangleright$.

To be precise, the state of each pLTS is entirely characterized by both the state of the automaton, and the values of its variables V.

Predicates. We define a predicate $Pred_{sv}$ relating a synchronisation vector (of the form $(\alpha'_i)^{i \in I}, (\beta'_j)^{j \in J} \rightarrow \alpha'[e_b]$), the actions of the involved sub-pNets and the resulting actions. This predicate verifies:

$$Pred_{sv}\Big(\big((\alpha'_i)^{i \in I}, (\beta'_j)^{j \in J} \rightarrow \alpha'[e_b]\big), \alpha_i^{i \in I}, \beta_j^{j \in J}, \alpha\Big) \Leftrightarrow$$
$$\forall i \in I. \alpha_i = \alpha'_i \wedge \forall j \in J. \beta_j = \beta'_j \wedge \alpha = \alpha' \wedge e_b$$

Somehow, this predicate entails a verification of satisfiability in the sense that if the predicate $Pred_{sv}$ is not satisfiable, then the transition associated with the synchronisation will not occur in the considered state, or equivalently will occur with a *False* precondition. If the action families do not match or if there is no valuation of variables such that the above formula can be ensured then the predicate is undefined.

The definition of this predicate is not constructive. In our tool [44], we construct a logical formula encoding the matching and unification condition involved, and we let an SMT engine (in the current implementation Z3 [33]) decide its satisfiability.

Example 3 (An open-transition). At the upper level, the *SimpleSystem* pNet of Figure 2 has 2 holes and *SimpleProtocol* as a sub-pNet, itself containing 3 pLTSs. One of its possible open transitions (synchronizing the hole P with the *Sender* within the *SimpleProtocol*) is:

$$\frac{s \xrightarrow{\langle \alpha,\ e_b,\ (x_j := e_j)^{j\in J}\rangle} s' \in \rightarrow}{\langle\!\langle S, s_0, \rightarrow \rangle\!\rangle \models \dfrac{\emptyset, e_b, \{x_j \leftarrow e_j\}^{j\in J}}{\lhd s \rhd \xrightarrow{\alpha} \lhd s' \rhd}} \quad \textbf{Tr1}$$

and

$$\frac{\begin{array}{c}\text{Leaves}(\langle\!\langle P_m^{m\in I}, \overline{Sort}, SV_k^{k\in K}\rangle\!\rangle) = pLTS_l^{l\in L} \quad k \in K \\ SV_k = (\alpha'_m)^{m\in I_1 \uplus I_2 \uplus J} \rightarrow \alpha'[e_b] \\ \forall m \in I_1.\ P_m \models \dfrac{\beta_j^{j\in J_m}, Pred_m, Post_m}{\lhd s_i^{i\in L_m} \rhd \xrightarrow{\alpha_m} \lhd (s'_i)^{i\in L_m} \rhd} \\ \forall m \in I_2.\ P_m \models \dfrac{\emptyset, Pred_m, Post_m}{\lhd s_m \rhd \xrightarrow{\alpha_m} \lhd s'_m \rhd} \qquad J' = \biguplus_{m\in I_1} J_m \uplus J \\ Pred = \bigwedge_{m\in I_1 \uplus I_2} Pred_m \wedge Pred_{sv}(SV_k, \alpha_m^{m\in I_1\uplus I_2}, \beta_j^{j\in J}, \alpha) \\ \forall i \in L \setminus \left(\biguplus_{m\in I_1} L_m \uplus I_2\right).\ s'_i = s_i \qquad \text{fresh}(\alpha'_m, \alpha', \beta_j^{j\in J}, \alpha)\end{array}}{\langle\!\langle P_m^{m\in I}, \overline{Sort}, SV_k^{k\in K}\rangle\!\rangle \models \dfrac{\beta_j^{j\in J'}, Pred, \biguplus_{m\in I_1\uplus I_2} Post_m}{\lhd s_i^{i\in L} \rhd \xrightarrow{\alpha} \lhd (s'_i)^{i\in L} \rhd}} \quad \textbf{Tr2}$$

Figure 3: Rules **Tr1** and **Tr2** defining the semantics of open pNets

$$OT_1 = \frac{\{\texttt{P} \mapsto \texttt{p-send(m)}\}, [\texttt{m=m'}], (\texttt{s\_msg} \leftarrow \texttt{m})}{\lhd s_0, m_0, r_0 \rhd \xrightarrow{\texttt{in(m')}} \lhd s_1, m_0, r_0 \rhd}$$

The global states here are triples, the product of states of the 3 pLTSs (holes have no state). The assignment performed by the open transition uses the variable `m` from the action of hole `P` to set the value of the sender variable named `s_msg`.

We build the semantics of open pNets as an open automaton over the states given by Definition 8. The open transitions first project the global state into states of the leaves, then apply pLTS transitions on these states, and compose them with the sort of the holes. The semantics instantiates fresh variables using the predicate $\text{fresh}(x)$, additionally, for an action α, $\text{fresh}(\alpha)$ means all variables in α are fresh.

Definition 9 (Semantics of open pNets). The semantics of a pNet P is an open automaton $A = \langle\!\langle Holes(P), States(P), InitState(P), vars(P), \mathcal{T}\rangle\!\rangle$ where $\mathcal{T}$ is the smallest set of open transitions such that $\mathcal{T} = \{OT \mid P \models OT\}$ and $P \models OT$ is defined by the rules in Figure 3.

- The rule **Tr1** for a pLTS checks that the guard is verified and transforms assignments into post-conditions.
- The rule **Tr2** deals with pNet nodes: for each possible synchronisation vector (of index k) applicable to the rule subject, the premises include one *open transition* for each sub-pNet involved, one possible *action* for each hole involved, and the predicate relating these with the resulting action of the vector. The sub-pNets involved are split between two sets, I_2 for sub-pNets that are pLTSs (with open transitions obtained by rule **Tr1**), and I_1 for the sub-pNets that are not pLTSs (with open transitions obtained by rule **Tr2**), J is the set of holes involved in the transition[6].

A key to understand **Tr2** is that the open transitions are expressed in terms of the leaves and holes of the whole pNet structure, i.e. a flattened view of the pNet. For example, L is the index set of the Leaves, L_m the index set of the leaves of one sub-pNet indexed m, so all L_m are disjoint subsets of L. Thus the states in the open transitions, at each level, are tuples including states of all the leaves of the pNet, not only those involved in the chosen synchronisation vector.

Note that the construction is symbolic, and each open transition deduced expresses a whole family of behaviours, for any possible value of the variables.

In [28], we have shown a detailed example of the construction of a complex open transition, building a deduction tree using rules **Tr1** and **Tr2**. We have also shown in [28] that an open pNet with finite synchronisation sets, finitely many leaves and holes, and each pLTS at leaves having a finite number of states and (symbolic) transitions, induces a finite automaton. The algorithm for building such an automaton can be found in [43].

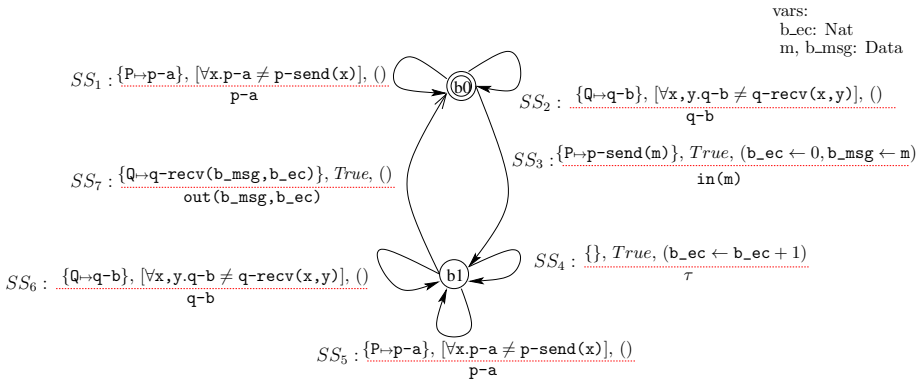


Figure 4: Open automaton for *SimpleProtocolSpec*

[6]Formally, if $SV_k = (\alpha')_m^{m \in M} \rightarrow \alpha'[e_b]$ is a synchronisation vector of P then $J = M \cap \text{Holes}(P)$, $I_2 = M \cap \text{Leaves}(P)$, $I_1 = M \setminus J \setminus I_2$. We could replace I_1 and I_2 by their formal definition in **Tr2** but the rule would be more difficult to read.

Example

Figure 4 shows the open automaton computed from the *SimpleProtocolSpec* pNet given in Figure 1. For later references, we name SS_i the transitions of this (strong) specification automaton while transitions of the *SimpleProtocolImpl* pNet are labelled SI_i. In the figures we annotate each open automaton with the set of its variables.

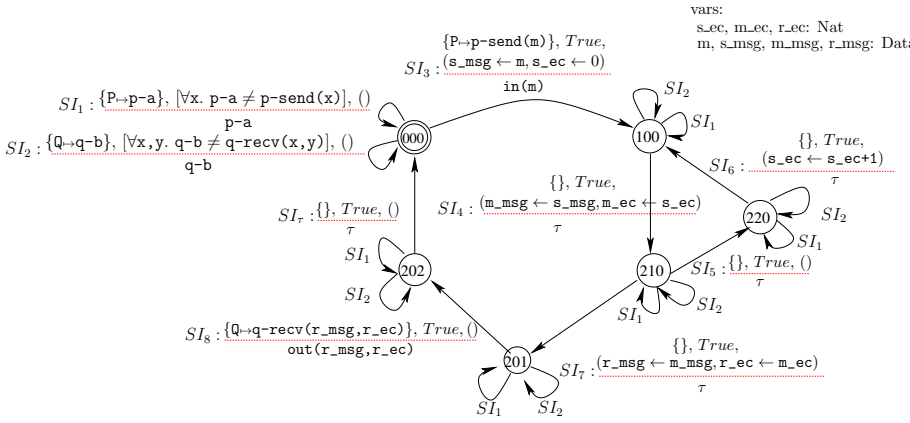


Figure 5: Open automaton for *SimpleProtocolImpl*

Figure 5 shows the open automaton of *SimpleProtocolImpl* from Figure 2. In this drawing, we have short labels for states, representing $\triangleleft s_0, m_0, r_0 \triangleright$ by `000`. Note that open transitions are denoted $\texttt{SI}_i$ and tau open transition by $\texttt{SI}_\tau$. The resulting behaviour is quite simple: we have a main loop including receiving a message from P and transmitting the same message to Q, with some intermediate τ actions from the internal communications between the protocol processes. In most of the transitions, you can observe that data is propagated between the successive pLTS variables (holding the message, and the error counter value). On the right of the figure, there is a loop of τ actions ($\texttt{SI}_4$, $\texttt{SI}_5$ and $\texttt{SI}_6$) showing the handling of errors and the incrementation of the error counter.

4.2. pNet Composition Properties: composition of open transitions

The semantics of open pNets allows us to prove two crucial properties relating pNet composition with pNet semantics: open transition of a composed pNet can be decomposed into open transitions of its composing sub-pNets, and conversely, from the open transitions of sub-pNets, an open transition of the composed pNet can be built.

We start with a decomposition property: from one open transition of $P[Q]_{j_0}$, we exhibit corresponding behaviours of P and Q, and determine the relation between their predicates.

Lemma 1 (Open transition decomposition). *Consider two pNets P and Q that are not pLTSs*[7]*. Let* $\text{Leaves}(Q) = p_l^{l \in L_Q}$ *and suppose:*

$$P[Q]_{j_0} \models \frac{\beta_j^{j \in J}, Pred, Post}{\lhd s_i^{i \in L} \rhd \xrightarrow{\alpha} \lhd {s'}_i^{i \in L} \rhd}$$

with $J \cap \text{Holes}(Q) \neq \emptyset$ *or* $\exists i \in L_Q.\, s_i \neq s'_i$*, i.e. Q takes part in the reduction. Then there exist α_Q, $Pred'$, $Pred''$, $Post'$, $Post''$ s.t.:*

$$P \models \frac{\beta_j^{j \in (J \setminus \text{Holes}(Q)) \cup \{j_0\}}, Pred', Post'}{\lhd s_i^{i \in L \setminus L_Q} \rhd \xrightarrow{\alpha} \lhd {s'}_i^{i \in L \setminus L_Q} \rhd}$$

$$\text{and} \qquad Q \models \frac{\beta_j^{j \in J \cap \text{Holes}(Q)}, Pred'', Post''}{\lhd s_i^{i \in L_Q} \rhd \xrightarrow{\alpha_Q} \lhd {s'}_i^{i \in L_Q} \rhd}$$

and $Pred \iff Pred' \wedge Pred'' \wedge \alpha_Q = \beta_{j_0}$*,* $Post = Post' \uplus Post''$ *where $Post''$ is the restriction of $Post$ over variables of Q.*

Lemma 2 is combining an open transition of P with an open transition of Q, and building a corresponding transition of $P[Q]_{j_0}$ by assembling their elements.

Lemma 2 (Open transition composition). *Suppose* $j_0 \in J$ *and:*

$$P \models \frac{\beta_j^{j \in J}, Pred, Post}{\lhd s_i^{i \in L} \rhd \xrightarrow{\alpha} \lhd {s'}_i^{i \in L} \rhd} \qquad \text{and} \qquad Q \models \frac{\beta_j^{j \in J_Q}, Pred', Post'}{\lhd s_i^{i \in L_Q} \rhd \xrightarrow{\alpha_Q} \lhd {s'}_i^{i \in L_Q} \rhd}$$

Then, we have:

$$P[Q]_{j_0} \models \frac{\beta_j^{(j \in J \setminus \{j_0\}) \uplus J_Q}, Pred \wedge Pred' \wedge \alpha_Q = \beta_{j_0}, Post \uplus Post'}{\lhd s_i^{i \in L \uplus L_Q} \rhd \xrightarrow{\alpha} \lhd {s'}_i^{i \in L \uplus L_Q} \rhd}$$

Note that this does not mean that any two pNets can be composed and produce an open transition. Indeed, the predicate $Pred \wedge Pred' \wedge \alpha_Q = \beta_{j_0}$ is often not satisfiable, in particular if the action α_Q cannot be matched with β_{j_0}. Note also that β_{j_0} is only used as an intermediate term inside formulas in the composed open transition: it does not appear as global action, and will not appear as an action of a hole.

4.3. Bisimulation for open pNets – a composable bisimulation theory

As our symbolic operational semantics provides an open automaton, we can apply the notion of strong (symbolic) bisimulation on automata to open pNets.

Definition 10 (FH-bisimulation for open pNets). Two pNets are FH-bisimilar if their associated open automata are bisimilar.

[7] A similar lemma can be proven for a pLTS Q.

We can now prove that pNet composition preserves FH-bisimilarity. More precisely, one can define two preservation properties, namely 1) when one hole of a pNet is filled by two bisimilar other (open) pNets; and 2) when the same hole in two bisimilar pNets are filled by the same pNet, in other words, composing a pNet with two bisimilar contexts. The general case will be obtained by transitivity of the bisimilarity relation (Theorem 1).

Theorem 3 (Congruence). *Consider an open pNet* $P = \langle\!\langle P_i^{i\in I}, Sort_j^{j\in J}, \overline{SV}\rangle\!\rangle$. *Let* $j_0 \in J$ *be a hole. Let* Q *and* Q' *be two FH-bisimilar pNets such that*[8] $\mathrm{Sort}(Q) = \mathrm{Sort}(Q') = Sort_{j_0}$. *Then* $P[Q]_{j_0}$ *and* $P[Q']_{j_0}$ *are FH-bisimilar.*

Theorem 4 (Context equivalence). *Consider two open pNets* $P = \langle\!\langle P_i^{i\in I}, Sort_j^{j\in J}, \overline{SV}\rangle\!\rangle$ *and* $P' = \langle\!\langle {P'}_i^{i\in I}, Sort_j^{j\in J}, \overline{SV'}\rangle\!\rangle$ *that are FH-bisimilar (they thus have the same holes). Let* $j_0 \in J$ *be a hole, and* Q *be a pNet such that* $\mathrm{Sort}(Q) = Sort_{j_0}$. *Then* $P[Q]_{j_0}$ *and* $P'[Q]_{j_0}$ *are FH-bisimilar.*

Finally, the previous theorems can be composed to state a general theorem about composability and FH-bisimilarity.

Theorem 5 (Composability). *Consider two FH-bisimilar pNets with an arbitrary number of holes, when replacing, inside those two original pNets, a subset of the holes by FH-bisimilar pNets, we obtain two FH-bisimilar pNets.*

This theorem is quite powerful, as it somehow implies that the theory of open pNets can be used to study properties of process composition. Open pNets can indeed be applied to study process operators and process algebras, as shown in [28] where compositional properties are extremely useful. In the case of interaction protocols [12], compositionality of bisimulation can justify abstractions used in some parts of the application.

5. Weak bisimulation

Weak symbolic bisimulation [25] was introduced to relate transition systems that have indistinguishable behaviour, with respect to some definition of *internal actions* that are considered local to some subsystem, and consequently cannot be observed, nor used for synchronisation with their context. The notion of non-observable actions varies in different contexts, e.g. *tau* in CCS [40, 41], and *i* in Lotos [10]. We could define classically a set of *internal/non-observable actions* depending on a specific action algebra. However in this paper, to simplify the notations, we will simply use τ as the single non-observable action; the generalisation of our results to a set of non-observable actions is trivial. Naturally, a non-observable action cannot be synchronised with actions of other

[8] Note that $\mathrm{Sort}(Q) = \mathrm{Sort}(Q')$ is ensured by strong bisimilarity.

systems in its environment. We show here that under such assumption of non-observability of τ actions, see Definition 11, we can define a weak bisimulation relation that is compositional, in the sense of open pNet composition. In this section we will first define a notion of weak open transition similar to open transition. In fact a weak open transition is made of several open transitions labelled as non-observable transitions, plus potentially one observable open transition. This allows us to define weak open automata, and a weak bisimulation relation based on these weak open automata. Finally, we apply this weak bisimulation to open pNets, obtain a weak bisimilarity relation for open pNets, and prove that this relation has compositional properties.

5.1. Preliminary definitions and notations

We first specify in terms of open transition, what it means for an action to be non-observable. We first define (in Definition 11) systems that cannot observe τ actions of sub-systems; namely pNets that cannot change their state, or emit an observable action when one of its holes emits a τ action.

More precisely, we state that τ is not observable if the automaton always allows any τ transition from holes, and additionally the global transition resulting from a τ action of a hole is a τ transition not changing the pNet's state. We define $\mathrm{Id}(V)$ as the identity function on the set of variables V.

Definition 11 (Non-observability of τ actions for open automata). An open automaton $A = \langle\!\langle J, \mathcal{S}, s_0, V, \mathcal{T} \rangle\!\rangle$ *cannot observe* τ *actions* if and only if for all j in J and s in $\mathcal{S}$ we have:

1. $$\frac{(j \mapsto \tau),\ \mathit{True},\ \mathrm{Id}(V)}{s \xrightarrow{\tau} s} \in \mathcal{T}$$

   and

2. for all β_j, J, α, s, s', *Pred*, *Post* such that

   $$\frac{\beta_j^{j \in J},\ \mathit{Pred},\ \mathit{Post}}{s \xrightarrow{\alpha} s'} \in \mathcal{T}$$

   If there exists j such that $\beta_j = \tau$ then we have:

   $$\alpha = \tau \wedge s = s' \wedge \mathit{Pred} = \mathit{True} \wedge \mathit{Post} = \mathrm{Id}(V) \wedge J = \{j\}$$

The first statement of the definition states that the open automaton must allow a hole to do a silent action at any time, and must not observe it, i.e. it cannot change its internal state because a hole did a τ transition. The second statement ensures that there cannot be in the open automaton other transitions that would be able to observe a τ action from a hole: statement (2) states that all the open transitions where a hole does a τ action must be of the shape given in statement (1). In this second statement, the condition $J = \{j\}$ is a bit restrictive, it could

safely be replaced by $\forall j \in J. \beta_j = \tau$, allowing the other holes to perform τ transitions too (because these τ actions cannot be observed). This possible synchronisation of τ actions would not be a problem as condition 1 still ensures that each process can do a τ separately.

By definition, one weak open transition contains several open transitions, where each open transition can require an observable action from a given hole, the same hole might have to emit several observable actions for a single weak open transition to occur. Consequently, for a weak open transition to trigger, a sequence of actions from a given hole may be required.

Thus, we let γ range over sequences of action terms and use $\oplus$ as the concatenation operator that appends sequences of action terms: given two sequences of action terms $\gamma \oplus \gamma'$ concatenates the two sequences. The operation is lifted to indexed sets of sequences: at each index i, $\overline{\gamma_1} \oplus \overline{\gamma_2}$ concatenates the sequences of actions at index i of $\overline{\gamma_1}$ and the one at index i of $\overline{\gamma_2}$[9]. $[a]$ denotes a sequence with a single element. These new actions are sequences of observable actions, we thus need an operator to build them from a set of actions that occur in open transitions, i.e. an operator that takes a set of actions performed by one hole and produces a sequence of observable actions. Thus we define $(\overline{\beta})^{\nabla}$ as the mapping $\overline{\beta}$ with only observable actions of the holes in I, but where each element is either empty or a list of length 1:

$$(\beta_i^{i \in I})^{\nabla} = [\beta_i]^{i \in I'} \text{ where } I' = \{i | i \in I \wedge \beta_i \neq \tau\}$$

As an example the $(\overline{\beta})^{\nabla}$ built from the transition OT_1 in Example 3, page 19 is $\mathtt{P}\mapsto[\mathtt{p\text{-}send(m)}]$. Remark that in our simple example no τ transition involves any visible action from a hole, so we have no β sequences of length longer than 1 in the weak automaton.

5.2. Weak open transition definition

Because of the non-observability property (Definition 11), it is possible to add any number of τ transitions of the holes before or after any open transition freely. This property justifies the fact that we can abstract away from τ transitions from holes in the definition of a weak open transition. We define weak open transitions similarly to open transitions except that holes can perform *sequences of observable actions* instead of single actions (observable or not). Compared to the definition of open transition, this small change has a significant impact as a single weak transition is the composition of several transitions of the holes.

Definition 12 (Weak open transition (WOT)). A weak open transition over a set J of holes with sorts $Sort_j^{j \in J}$ and a set of states $\mathcal{S}$ is a structure of the form:

$$\frac{\gamma_j^{j \in J'}, Pred, Post}{s \overset{\alpha}{\Rightarrow} s'}$$

[9] One of the two sequences is empty when $i \notin \mathrm{dom}(\overline{\gamma_1})$ or $i \notin \mathrm{dom}(\overline{\gamma_2})$.

$$\frac{}{\overset{\emptyset,\, True,\, \mathrm{Id}(V)}{\cdots\cdots\cdots\cdots}\; s \overset{\tau}{\Rightarrow} s} \in \mathcal{WT} \quad \textbf{WT1} \qquad \text{and} \qquad \frac{\overset{\overline{\beta},\, Pred,\, Post}{\cdots\cdots\cdots\cdots}\; s \overset{\alpha}{\rightarrow} s' \in \mathcal{T}}{\overset{(\overline{\beta})^{\nabla},\, Pred,\, Post}{\cdots\cdots\cdots\cdots}\; s \overset{\alpha}{\Rightarrow} s' \in \mathcal{WT}}\textbf{WT2}$$

and

$$\frac{\begin{array}{c}\overset{\overline{\gamma_1},\, Pred_1,\, Post_1}{\cdots\cdots\cdots\cdots}\; s \overset{\tau}{\Rightarrow} s_1 \in \mathcal{WT} \qquad \overset{\overline{\gamma_2},\, Pred_2,\, Post_2}{\cdots\cdots\cdots\cdots}\; s_1 \overset{\alpha}{\Rightarrow} s_2 \in \mathcal{WT} \\ \overset{\overline{\gamma_3},\, Pred_3,\, Post_3}{\cdots\cdots\cdots\cdots}\; s_2 \overset{\tau}{\Rightarrow} s' \in \mathcal{WT} \qquad \overline{\gamma} = \overline{\gamma_1} \oplus \overline{\gamma_2}\{\!\{Post_1\}\!\} \oplus \overline{\gamma_3}\{\!\{Post_2 \odot Post_1\}\!\} \\ \alpha' = \alpha\{\!\{Post_1\}\!\} \qquad Pred = Pred_1 \wedge Pred_2\{\!\{Post_1\}\!\} \wedge Pred_3\{\!\{Post_2 \odot Post_1\}\!\}\end{array}}{\overset{\overline{\gamma},\, Pred,\, Post_3 \odot Post_2 \odot Post_1}{\cdots\cdots\cdots\cdots\cdots\cdots\cdots}\; s \overset{\alpha'}{\Rightarrow} s' \in \mathcal{WT}}\textbf{WT3}$$

Figure 6: Weak transition definition

Where $J' \subseteq J$, $s, s' \in \mathcal{S}$ and γ_j is a list of transitions of the hole j, with each element of the list in $Sort_j$. α is an action label denoting the resulting action of this open transition. *Pred* and *Post* are defined similarly to Definition 5. We use $\mathcal{WT}$ to range over sets of weak open transitions.

A weak open automaton $\langle\!\langle J, \mathcal{S}, s_0, V, \mathcal{WT} \rangle\!\rangle$ is similar to an open automaton except that $\mathcal{WT}$ is a set of weak open transitions over J and $\mathcal{S}$.

A weak open transition labelled α can be seen as a sequence of open transitions that are all labelled τ except one that is labelled α; however conditions on predicates, effects, and states must be verified for this sequence to be fired.

We are now able to build a weak open automaton from an open automaton. This is done in a way that resembles the process of τ saturation: we add τ open transitions before or after another open transition, regardless of whether it is observable or not.

Definition 13 (Building a weak open automaton).
Let $A = \langle\!\langle J, \mathcal{S}, s_0, V, \mathcal{T} \rangle\!\rangle$ be an open automaton. The weak open automaton *derived* from A is an open automaton $\langle\!\langle J, \mathcal{S}, s_0, V, \mathcal{WT} \rangle\!\rangle$ where $\mathcal{WT}$ is derived from $\mathcal{T}$ by saturation, applying the rules of Figure 6.

Rule **WT1** states that it is always possible to perform a non-observable transition, where the state is unchanged and the holes perform no action. Rule **WT2** states that each open transition is a weak open transition. Finally, Rule **WT3** allows any number of τ transitions before or after any weak open transition. This rule carefully composes predicates, effects, and actions of the holes. Indeed, predicate $Pred_2$ manipulates variables of s_1 that result from the first weak

open transition. Their values thus depend on the initial state but also on the effect (as a substitution function $Post_1$) of the first weak open transition. In the same manner, $Pred_3$ must be applied the substitution defined by the composition $Post_2 \odot Post_1$. Similarly, effects on variables must be applied to obtain the global effect of the composed weak open transition, to observable actions of the holes, and to the global action of the weak open transition.

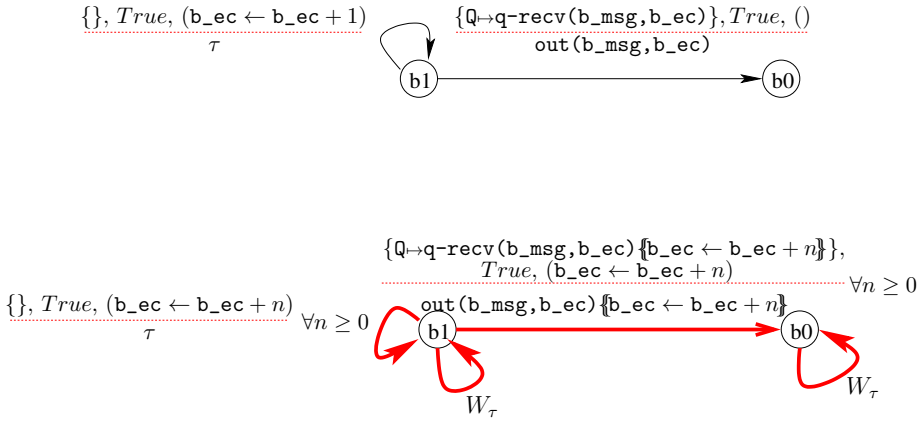


Figure 7: Construction of an example of weak open transition

Example 4 (A weak open-transition). Figure 7 shows the construction of one of the weak transitions of the open automaton of *SimpleProtocolSpec*. On the top we show the subset of the original open automaton (from Figure 4) considered here, and at the bottom the generated weak transition. For readability, we abbreviate the weak open transitions encoded by $\frac{\{\}, True, ()}{s \stackrel{\tau}{\Rightarrow} s'}$ as W_τ. The weak open transition shown here is the transition delivering the result of the algorithm to hole Q by applying rules: **WT1**,**WT2**, and **WT3**. First rule **WT1** adds a WT_τ loop on each state. Rule **WT2** transforms each 2 OTs into WOTs. Then consider application of Rule **WT3** on a sequence of 3 WOTs. $\frac{\{\}, True, (\texttt{b\_ec} \leftarrow \texttt{b\_ec} + 1)}{\texttt{b1} \stackrel{\tau}{\Rightarrow} \texttt{b1}}$; $\frac{\{\}, True, (\texttt{b\_ec} \leftarrow \texttt{b\_ec} + 1)}{\texttt{b1} \stackrel{\tau}{\Rightarrow} \texttt{b1}}$; $\frac{\{\}, True, ()}{\texttt{b1} \stackrel{\tau}{\Rightarrow} \texttt{b1}}$. The result will be: $\frac{\{\}, True, (\texttt{b\_ec} \leftarrow \texttt{b\_ec} + 2)}{\texttt{b1} \stackrel{\tau}{\Rightarrow} \texttt{b1}}$. We can iterate this construction an arbitrary number of times, getting for any natural number n a weak open transition: $\frac{\emptyset, True, (\texttt{ec} \leftarrow \texttt{ec} + n)}{\texttt{b1} \stackrel{\tau}{\Rightarrow} \texttt{b1}} \forall n \geq 0$. Finally, applying again **WT3**, and using the central open transition having `out(b_msg,b_ec)` as α, we get the resulting weak open transition between b1 and b0 (as shown in Figure 7). Applying the substitutions finally yields the weak transitions family WS_7 in Figure 8.

Example 5 (Weak open automata). Figures 8 and 9 respectively show the

weak automata of *SimpleProtocolSpec* and *SimpleProtocolImpl*. We encode weak open transitions by WS on the specification model and by WI on the implementation model.

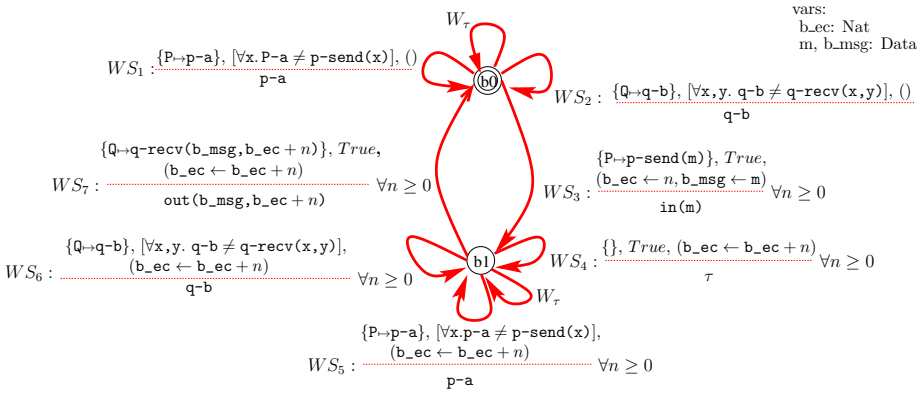


Figure 8: Weak Open Automaton of *SimpleProtocolSpec*

For readability, we only give names to the weak open transitions of *SimpleProtocolImpl* in Figure 9; we detail some of these transitions below and the full list is included in Appendix C. Let us point out that the weak OT loops (WI_1,WI_2 and W_τ) on state 000 are also present in all other states, we did not repeat them. Additionally, many WOTs are similar, and numbered accordingly as 3, 3a, 3b, 3c and 8, 8a, 8b, 8c respectively: they only differ by their respective source or target states; the "variant" WOTs appear in blue in Figure 9, note that composed WOTs also appear in blue.

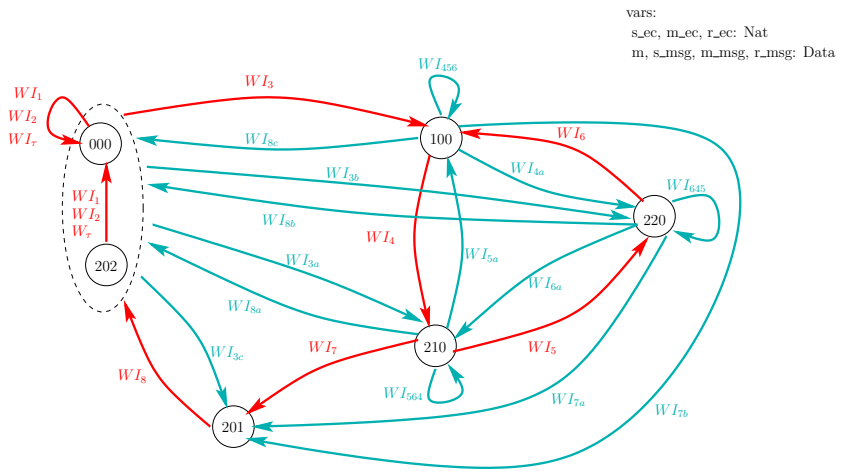


Figure 9: Weak Open Automaton of *SimpleProtocolImpl*

Now let us give some details about the construction of the weak automaton of the *SimpleProtocolImpl* pNet, obtained by application of the weak rules as explained above. We concentrate on weak open transitions WI_3 and WI_4. Let us denote as $post_n$ the effect (as a substitution function) of the strong open transitions SI_n from Figure 5:

$post_3 = (\texttt{s\_msg} \leftarrow \texttt{m}, \texttt{s\_ec} \leftarrow 0)$
$post_4 = (\texttt{m\_msg} \leftarrow \texttt{s\_msg}, \texttt{m\_ec} \leftarrow \texttt{s\_ec})$
$post_5 = ()$
$post_6 = (\texttt{s\_ec} \leftarrow \texttt{s\_ec+1})$

Then the effect of one single $100 \xrightarrow{OT_4} 210 \xrightarrow{OT_5} 220 \xrightarrow{OT_6} 100$ loop is[10]:

$$post_{456} = post_6 \odot post_5 \odot post_4 = (\texttt{s\_ec} \leftarrow \texttt{s\_ec} + 1)$$

So if we denote $post_{456*}$ any iteration of this loop, we get $post_{456*} = (\texttt{s\_ec} \leftarrow \texttt{s\_ec} + n)$ for any $n \geq 0$, and the *Post* of the weak OT WI_3 is:
$Post_3 = post_{456*} \odot post_3 = (\texttt{s\_msg} \leftarrow m, \texttt{s\_ec} \leftarrow n), \forall n \geq 0$ and *Post* of WI_{3a} is:
$post_4 \odot post_{456*} \odot post_3 = (\texttt{m\_msg} \leftarrow m, \texttt{m\_ec} \leftarrow n), \forall n \geq 0.$

We can now show some of the weak OTs of Figure 9 (the full table is included in Appendix C). As we have seen above, the effect of rule WT_3 when a silent action have an effect on the variable *ec* will generate an infinite family of WOTs, depending on the number of iterations through the loops. We denote these families using a "meta-variable" n, ranging over `Nat`.

$$WI_1 = \frac{\{\texttt{P} \mapsto \texttt{p-a}\}, [\forall \texttt{x}.\texttt{p-a} \neq \texttt{p-send(x)}], ()}{s \xRightarrow{\texttt{p-a}} s} \text{ (for any } s \in S)$$

$$\forall n \geq 0.\ WI_3(n) = \frac{\{\texttt{P} \mapsto \texttt{p-send(m)}\},\ True,\ (\texttt{s\_msg} \leftarrow \texttt{m}, \texttt{s\_ec} \leftarrow n)}{000 \xRightarrow{\texttt{in(m)}} 100}$$

$$\forall n \geq 0.\ WI_4(n) = \frac{\{\},\ True,\ (\texttt{m\_msg} \leftarrow \texttt{s\_msg}, \texttt{m\_ec} \leftarrow \texttt{s\_ec} + n, \texttt{s\_ec} \leftarrow \texttt{s\_ec} + n)}{100 \xRightarrow{\tau} 210}$$

$$\forall n \geq 0.\ WI_{456}(n) = \frac{\{\},\ True,\ (\texttt{s\_ec} \leftarrow \texttt{s\_ec} + n)}{100 \xRightarrow{\tau} 100}$$

The *Post* of the weak OT WI_{6a} is:

$$\begin{aligned} Post_{6a} &= post_4 \odot post_{456*} \odot post_6 \\ &= (\texttt{m\_msg} \leftarrow \texttt{s\_msg}, \texttt{m\_ec} \leftarrow \texttt{s\_ec}) \odot (\texttt{s\_ec} \leftarrow \texttt{s\_ec} + n) \odot (\texttt{s\_ec} \leftarrow \texttt{s\_ec} + 1) \\ &= (\texttt{m\_msg} \leftarrow \texttt{s\_msg}, \texttt{m\_ec} \leftarrow \texttt{s\_ec} + 1 + n, \texttt{s\_ec} \leftarrow \texttt{s\_ec} + 1 + n) \end{aligned}$$

So we get:

$$\forall n \geq 0.\ WI_{6a}(n) = \frac{\{\},\ True,\ (\texttt{m\_ec} \leftarrow \texttt{s\_ec} + 1 + n, \texttt{s\_ec} \leftarrow \texttt{s\_ec} + 1 + n)}{220 \xRightarrow{\tau} 210}$$

[10] when showing the result of *Posts* composition, we will omit the identity substitution functions introduced by the $\odot$ definition in page 7.

5.3. Composition properties: composition of weak open transitions

We now have two different semantics for open pNets: a strong semantics, defined as an open automaton, and as a weak semantics, defined as a weak open automaton. Like the open automaton, the weak open automaton features valuable composition properties. We can exhibit a composition property and a decomposition property that relate open pNet composition with their semantics, defined as weak open automata. These are however technically more complex than the ones for open automata because each hole performs a set of actions, and thus a composed transition is the composition of one transition of the top-level pNet and a sequence of transitions of the sub-pNet that fills its hole. Composition and decomposition properties can be found as Lemma 6, Lemma 7, and Lemma 8 in Appendix B.2.

5.4. Weak FH-bisimulation

For defining a bisimulation relation between weak open automata, two options are possible. One option is that we define a simulation similar to the strong simulation but based on weak open automata, this would look like the FH-simulation but would need to be adapted to weak open transitions. Alternatively, we could define directly and classically a weak FH-simulation as a relation between two open automata, relating the open transitions of the first one with the transitions of the weak open automaton derived from the second one.

The definition below specifies how a set of weak open transitions can simulate an open transition, and under which condition; this is used to relate, by weak FH-bisimulation, two open automata by reasoning on the weak open automata that can be derived from the strong ones.

Definition 14 (Weak FH-bisimulation).
Let $A_1 = \langle\!\langle J, \mathcal{S}_1, s_0, V_1, \mathcal{T}_1 \rangle\!\rangle$ and $A_2 = \langle\!\langle J, \mathcal{S}_2, t_0, V_2, \mathcal{T}_2 \rangle\!\rangle$ be open automata with disjoint sets of variables. Let $\langle\!\langle J, \mathcal{S}_1, s_0, V_1, \mathcal{WT}_1 \rangle\!\rangle$ and $\langle\!\langle J, \mathcal{S}_2, t_0, V_2, \mathcal{WT}_2 \rangle\!\rangle$ be the weak open automata derived from A_1 and A_2 respectively. Let $\mathcal{R}$ a relation over $\mathcal{S}_1$ and $\mathcal{S}_2$, as in Definition 7.

Then $\mathcal{R}$ is a weak FH-bisimulation iff for any states $s \in \mathcal{S}_1$ and $t \in \mathcal{S}_2$ such that $(s, t|Pred_{s,t}) \in \mathcal{R}$, we have the following:

- For any open transition OT in $\mathcal{T}_1$:

$$\frac{\beta_j^{j \in J'}, Pred_{OT}, Post_{OT}}{s \xrightarrow{\alpha} s'}$$

there exists an indexed set of weak open transitions $WOT_x^{x \in X} \subseteq \mathcal{WT}_2$:

$$\frac{\gamma_{jx}^{j \in J_x}, Pred_{OT_x}, Post_{OT_x}}{t \xRightarrow{\alpha_x} t_x}$$

such that $\forall x.\{j \in J' | \beta_j \neq \tau\} = J_x, (s', t_x | Pred_{s',t_x}) \in \mathcal{R}$; and

$$Pred_{s,t} \wedge Pred_{OT} \implies \bigvee_{x \in X} \left(\forall j \in J_x . (\beta_j)^{\nabla} = \gamma_{jx} \wedge Pred_{OT_x} \wedge \alpha = \alpha_x \wedge Pred_{s',t_x} \{\!\{ Post_{OT} \uplus Post_{OT_x} \}\!\} \right)$$

- and symmetrically any open transition from t in $\mathcal{T}_2$ can be covered by a set of weak transitions from s in $\mathcal{WT}_1$.

Two open automata are *weak FH-bisimilar* if there exists a weak FH-bisimulation relation that relates their initial states. This relation is called *weak FH-bisimilarity*. Two pNets are weak FH-bisimilar if their associated open automata are weakly bisimilar.

Compared to strong bisimulation, except the use of weak open transitions to simulate an open transition, the condition on predicate is slightly changed concerning actions of the holes. Indeed, only the visible actions of the holes must be compared and they form a list of actions, but of length at most one.

Our first important result is that weak FH-bisimilarity is an equivalence in the same way as strong FH-bisimilarity.

Theorem 6 (Weak FH-bisimilarity is an equivalence).
Weak FH-bisimilarity is reflexive, symmetric and transitive.

The proof is detailed in Appendix B.1, it follows a similar pattern as the proof that strong FH-bisimilarity is an equivalence, but technical details are different, and in practice we rely on a variant of the definition of weak FH-bisimilarity; this equivalent version simulates a *weak* open transition with a set of weak open transition. The careful use of the best definition of weak FH-bisimilarity makes the proof similar to the strong FH-bisimilarity case.

Proving bisimulation in practice

In practice, we are dealing with finite representations of the (infinite) open automata. In [29], we defined a slightly modified definition of the "coverage" proof obligation, in the case of strong FH-bisimulation. This modification is required to manage in a finite way all possible instantiations of an OT. In the case of weak FH-bisimulation, the proof obligation from Definition 14 becomes:

$$\forall fv_{OT} . \left\{ Pred_{s,t} \wedge Pred_{OT} \implies \bigvee_{x \in X} \left[\exists fv_{OT_x} . \left(\forall j \in J_x . (\beta_j)^{\nabla} = \gamma_{jx} \wedge Pred_{OT_x} \wedge \alpha = \alpha_x \wedge Pred_{s',t_x} \{\!\{ Post_{OT} \uplus Post_{OT_x} \}\!\} \right) \right] \right\}$$

where fv_{OT} denotes the set of free variables of all expressions in OT.

5.5. Weak FH-bisimulation for open pNets

Before defining a weak open automaton for the semantics of open pNets, it is necessary to state under which condition a pNet is unable to observe silent actions of its holes. In the setting of pNets this can simply be expressed as a condition on the synchronisation vectors. Precisely, the set of synchronisation vectors must contain vectors that let silent actions go through the pNet, i.e. synchronisation vectors where one hole does a τ transition, and the global visible action is a τ. Additionally, no other synchronisation vector must be able to react on a silent action from a hole, i.e. if a synchronisation vector observes a τ from a hole it cannot synchronise it with another action nor emit an action that is not τ. This is formalised as follows:

Definition 15 (Non-observability of silent actions for pNets). A pNet $\langle\!\langle P_i^{i\in I}, Sort_j^{j\in J}, \overline{SV}\rangle\!\rangle$ *cannot observe silent actions* if it verifies: $\forall i \in I \uplus J.\, (i{\mapsto}\tau) \to \tau[True] \in \overline{SV}$ and

$$\forall \left((\alpha_i)^{i\in I'} \to \alpha'[e_b] \in \overline{SV} \right), \forall i \in I' \cap J.\, \alpha_i = \tau \implies \alpha' = \tau \ \wedge \ I' = \{i\}$$

With this definition, it is easy to check that the open automaton that gives the semantics of such an open pNet cannot observe silent actions in the sense of Definition 11.

Property 1 (Non-observability of silent actions). *The semantics of a pNet, as provided in Definition 9, that cannot observe silent actions is an open automaton that cannot observe silent actions.*

Under this condition, it is safe to define the weak open automaton that provides a weak semantics to a given pNet. This is simply obtained by applying Definition 13 to generate a weak open automaton from the open automaton that is the strong semantics of the open pNet, as provided by Definition 9.

Definition 16 (Semantics of pNets as a weak open automaton). Let A be the open automaton expressing the semantics of an open pNet P; let $\langle\!\langle J, \mathcal{S}, s_0, V, \mathcal{WT}\rangle\!\rangle$ be the weak open automaton derived from A; we call this weak open automaton the weak semantics of the pNet P. Then, we denote $P \models WOT$ whenever $WOT \in \mathcal{WT}$.

From the definition of the weak open automata of pNets, we can now study the properties of weak bisimulation concerning open pNets.

5.6. Properties of weak bisimulation for open pNets

When silent actions cannot be observed, weak FH-bisimilarity is a congruence for open pNets: if P and Q are weakly bisimilar to P' and Q' then the composition of P and Q is weakly bisimilar to the composition of P' and Q', where composition is the hole replacement operator: $P[Q]_j$ and $P'[Q']_j$ are weak FH-bisimilar. This can be shown by proving the two following theorems.

The detailed proof of these theorem can be found in Appendix B.2. The proof strongly relies on the fact that weak FH-bisimulation is an equivalence, but also on the composition properties for open automata.

Theorem 7 (Congruence for weak FH-bisimilarity). *Consider an open pNet P that cannot observe silent actions, of the form $P = \langle\!\langle P_i^{i\in I}, Sort_j^{j\in J}, \overline{SV}\rangle\!\rangle$. Let $j_0 \in J$ be a hole. Let Q and Q' be two weak FH-bisimilar pNets such that*[11] $\mathrm{Sort}(Q) = \mathrm{Sort}(Q') \subseteq Sort_{j_0}$. *Then $P[Q]_{j_0}$ and $P[Q']_{j_0}$ are weak FH-bisimilar.*

Theorem 8 (Context equivalence for weak FH-bisimilarity). *Consider two open pNets $P = \langle\!\langle P_i^{i\in I}, Sort_j^{j\in J}, \overline{SV}\rangle\!\rangle$ and $P' = \langle\!\langle {P'}_i^{i\in I}, Sort_j^{j\in J}, \overline{SV'}\rangle\!\rangle$ that are weak FH-bisimilar (recall they must have the same holes to be FH-bisimilar) and that cannot observe silent actions. Let $j_0 \in J$ be a hole, and Q be a pNet such that* $\mathrm{Sort}(Q) \subseteq Sort_{j_0}$. *Then $P[Q]_{j_0}$ and $P'[Q]_{j_0}$ are weak FH-bisimilar.*

Finally, the previous theorems can be composed to state a general theorem about composability and weak FH-bisimilarity.

Theorem 9 (Composability of weak FH-bisimilarity). *Consider two weak FH-bisimilar pNets with an arbitrary number of holes, such that the two pNets cannot observe silent actions. When replacing, inside those two original pNets, a subset of the holes by weak FH-bisimilar pNets, we obtain two weak FH-bisimilar pNets.*

Example 6 (CCS Choice). Consider the + operator of CCS, shown in Example 1. The pNet does not satisfy Definition 15. Indeed, if a or b is τ then the + operator can observe the τ transition. It is well-known that weak bisimilarity is not a congruence in CCS, this corresponds to the fact that the + operator can observe the τ transitions. Thus, even if we can define a weak FH-bisimilarity for CCS with + it does not verify the necessary requirements for being a congruence. On the other side, the parallel operator defined similarly *satisfies* Definition 15, and indeed bisimilarity is a congruence for the parallel operator in CCS.

Running example

In Section 5 we have shown the full saturated weak automaton for both *SimpleProtocolSpec* and *SimpleProtocolImpl*. We will show here how we can check if some given relation between these two automata is a weak FH-bisimulation.

Preliminary remarks:

- Both pNets trivially verify the "non-observability" condition: the vectors having τ as an action of a sub-net are of the form "$< -, \tau, - > \rightarrow \tau$".

[11] Note that $\mathrm{Sort}(Q) = \mathrm{Sort}(Q')$ is ensured by weak FH-bisimilarity.

SimpleProtocolSpec states	*SimpleProtocolImpl* states	Predicate
b0	000	True
b0	202	True
b1	100	`b_msg = s_msg` $\wedge$ `b_ec = s_ec`
b1	210	`b_msg = m_msg` $\wedge$ `b_ec = m_ec`
b1	220	`b_msg = s_msg` $\wedge$ `b_ec = s_ec`
b1	201	`b_msg = r_msg` $\wedge$ `b_ec = r_ec`

Table 1: Bisimulation for the running example.

- We must take care of variable name conflicts: in our example, the variables of the 2 systems already have different names, but the action parameters occurring in the transitions (`m`, `msg`, `ec`) are the same, that is not correct. In the tools, this is managed by the static semantic layer; in the example, we rename the only conflicting variables m into $m1$ for *SimpleProtocolSpec*, and $m2$ for *SimpleProtocolImpl*.

Now consider the relation $\mathcal{R}$ defined by the triples in Table 1.

Checking that $\mathcal{R}$ is a weak FH-bisimulation means checking, for each of these triples, that each (strong) OT of one of the states corresponds to a set of WOTs of the other, using the conditions from Definition 14. We give here one example: consider the second triple from the table, and transition SS_3 from state `b0`. Its easy to guess that it will correspond to $WI_3(0)$ of state 202 (and equivalently state 000, see Figure 9):

$$SS_3 = \frac{\{\texttt{P}\mapsto\texttt{p-send(m1)}\}, True, (\texttt{b\_msg} \leftarrow \texttt{m1}, \texttt{b\_ec} \leftarrow 0)}{\text{b0} \xrightarrow{\texttt{in(m1)}} \text{b1}}$$

$$WI_3(0) = \frac{\{\texttt{P}\mapsto\texttt{p-send(m2)}\}, True, (\texttt{s\_msg} \leftarrow \texttt{m2}, \texttt{s\_ec} \leftarrow 0)}{000 \xrightarrow{\texttt{in(m2)}} 100}$$

Let us check formally the conditions:

- Their sets of active (non-silent) holes is the same: $J' = J_x = \{\texttt{P}\}$.
- Triple $(\text{b1}, 100, \texttt{b\_msg = s\_msg} \wedge \texttt{b\_ec = s\_ec})$ is in $\mathcal{R}$.
- The verification condition

$$\forall fv_{OT}.\{Pred \wedge Pred_{OT} \implies \bigvee_{x\in X} \Big[\exists fv_{OT_x}. \big(\forall j \in J_x.(\beta_j)^{\nabla} = \gamma_{jx} \wedge Pred_{OT_x} \wedge \alpha = \alpha_x \wedge Pred_{s',t_x}\{\!\{Post_{OT} \uplus Post_{OT_x}\}\!\}\big)\Big]\}$$

gives us:

$\forall \texttt{m1}.\{True \wedge True \implies \exists \texttt{m2}.$
$(\texttt{[p-send(m1)]} = \texttt{[p-send(m2)]} \wedge True \wedge \texttt{in(m1)} = \texttt{in(m2)} \wedge$

$(\texttt{b\_msg} = \texttt{s\_msg} \wedge \texttt{b\_ec} = \texttt{s\_ec})\{\!\{(\texttt{b\_msg} \leftarrow \texttt{m1}, \texttt{b\_ec} \leftarrow 0) \uplus (\texttt{s\_msg} \leftarrow \texttt{m2}, \texttt{s\_ec} \leftarrow 0)\}\!\})\}$

That is reduced to:

$\forall \texttt{m1}.\exists \texttt{m2}.\, (\texttt{p-send(m1)} = \texttt{p-send(m2)} \wedge \texttt{in(m1)} = \texttt{in(m2)} \wedge \texttt{m1} = \texttt{m2} \wedge 0 = 0)$

That is a tautology.

6. Related Works

To the best of our knowledge, there are not many research works on Weak Bisimulation Equivalences between such complicate system models (open, symbolic, data-aware, with loops and assignments). We give a brief overview of other related publications, focussing first on Open and Compositional approaches, then on Symbolic Bisimulation for data-sensitive systems.

Open and compositional systems

In [34, 33], the authors investigate several methodologies for the compositional verification of software systems, with the aim to verify reconfigurable component systems. To improve scalability and compositionality, the authors decompose the verification problem that is to be resolved by a SMT (satisfiability modulo theory) solver into independent sub-problems on independent sets of variables. These works clearly highlight the interest of incremental and compositional verification in a very general setting. In our own work on open pNets, adding more structure to the composition model, we show how to enforce a compositional proof system that is more versatile than independent sets of variables as the composition is structured and allows arbitrary synchronisations between sub-entities. Our theory has also been encoded into an SMT solver and it would be interesting to investigate how the examples of evolving systems studied by Johnson et al. could be encoded into pNet and verified by our framework. However, the models of Johnson et al. are quite different from ours, in particular they are much less structured, and translating them is clearly outside the scope of this article.

In previous work [19], we also have shown how (closed) pNet models could be used to encode and verify finite instances of reconfigurable component systems.

Methodologies for reasoning about abstract semantics of open systems can be found in [4, 5, 17], authors introduce behavioural equivalences for open systems from various symbolic approaches. Working in the setting of process calculi, some close relations exist with the work of the authors of [4, 5], where both approaches are based on some kinds of labelled transition systems. The distinguishing feature of their approach is that the transitions systems are labelled with logical formulae that provides an abstract characterization of the structure that a hole must possess and of the actions it can perform in order to allow a transition to fire. Logical formulae are suitable formalisms that capture the general class of components that can act as the placeholders of the

system during its evolution. In our approach we purposely leave the algebra of action terms undefined but the only operation we allow on action of holes is the comparison with other actions. Defining properly the interaction between a logical formulae in the action and the logics of the pNet composition seems very difficult. mCRL2 [21] is another effective model for specifying and proving properties of concurrent systems. mCRL2 has an established tool-suite and share similarities with pNets. However, pNets feature hierarchical composition with more structure than mCRL2 that composes processes with a parallel operator. Synchronisation of processes is expressed very differently; it is difficult to precisely compare multi-actions of mCRL2 with synchronisation vectors of pNets but synchronisation vector of pNets enforce a synchronisation based on the structure while in mCRL2 synchronisation is specified in a versatile, flexible, but less structured way.

In the same vein as context systems [36], pNets is a formalism for modular and possibly incomplete description of concurrent systems. The two formalisms are however different as the theory of contexts relies on a form of rewrite rules, while pNets rely on parametric automata to express the system behaviour. pNets have similar features as context systems [36] and static constructs [31]. Indeed all these approaches allow for modular and possibly incomplete description and structural composition of systems. The main originality of pNets compared to these other compositional approaches is the parameterised nature of the specification, which enables reasoning on value-passing systems but also on rich synchronisations that depend on the value of parameters.

Decomposition techniques

Quotienting of process algebras [36] and decomposition techniques for mCRL2 [37] share similarities with our approach; they propose to overcome the state-space-explosion problem by decomposing formulas to be verified according to the process composition. The decomposed problem must be equivalent to the original one. However these techniques are expressed in a very different setting from ours and it is difficult to precisely relate them to the more structural and parameterised point of view we adopt here. We could try to apply such automatic decomposition techniques to open pNets, but deriving a decomposition for systems synchronised in a very parameterised way like we do requires further investigations. Both parallel composition [36] and mCRL2 [37] feature a concrete verification setting where decomposition is useful, while open automata provide a more general setting that could be used to represent both frameworks and hopefully generalise process decomposition results of [36, 37].

Logical and semantics approaches

Among the approaches for modelling open systems, one can cite [7] that uses transition conditions depending on an external environment, and introduce bisimulation relations based on this approach. The approach of [7] is highly based on logics and their bisimulation theory is richer than ours in this aspect, while our theory is highly structural and focuses on relation between structure and equivalence. Also, we see composition as a structural operation putting

systems together, and do not focus on the modelling of an unknown outside world. Overall we believe that the two approaches are complementary but comparing precisely the two different bisimulation theories is not trivial.

There is also a clear relation with the seminal works on rule formats for Structured Operational Semantics, e.g. De Simone format, GSOS, and conditional rules with or without negative premises [15, 9, 23, 45]. The Open pNets model provides a way to define operators similar to these rules formats, but with quite different aim and approach. A formal comparison would be interesting, though not trivial. What we can say easily is that: the pNet format syntactically encompasses De Simone, GSOS, and conditional premises rules. Then our compositionality result is more powerful than their classical results, but this is not a surprise, as we rely on a (sufficient) syntactic hypothesis on a particular system, rather than the general rules defining an operator. Last, we intentionally do not accept negative premises, that would be more to put into practice in our implementation. This extension could be studied in future work.

Symbolic and data-sensitive systems

As mentioned in the *Introduction*, we were substantially inspired by the works of Lin et al. [32, 25, 38]. They developed the theory of symbolic transition graphs (STG), and the associated symbolic (early and late, strong and weak) bisimulations. Moreover, they studied STGs with assignments as a model for message-passing processes. Our work extends those contributions in several ways: first our models are compositional, and our bisimulations come with effective conditions for being preserved by pNet composition (i.e. congruent), even for the weak version. This result is more general than the bisimulation congruences for value-passing CCS in [32]. Then our settings for management of data types are much less restrictive, thanks to our use of satisfiability engines, while Lin's algorithms were limited to data-independent systems.

In a similar way, [1] presents a notion of "data-aware" bisimulations on data graphs, in which computation of such bisimulations is studied based on XPath logical language extended with tests for data equality.

Research related to the keyword "Symbolic Bisimulation" refer to two very different domains, namely BDD-like techniques for modelling and computing finite-state bisimulations, that are not related to our topic; and symbolic semantics for data-dependant or high-order systems, that are very close in spirit to our approach. In this last area, we can mention Calder's work [14], that defines a symbolic semantic for full Lotos, with a symbolic bisimulation over it; Borgstrom et al., Liu et al, Delaune et al. and Buscemi et al. providing symbolic semantics and equivalence for different variants of pi calculus respectively [11, 16, 39, 13]; and more recently Feng et al. provide a symbolic bisimulation for quantum processes [18]. All the above works are based on models definitely different from ours, and none of them allows system to be as much parameterised as open pNets; this additional expressiveness is due to the open and symbolic nature of our constructs.

7. Conclusion and discussion

pNets (Parameterised Networks of Automata) is a formalism adapted to the representation of the behaviour of parallel or distributed systems. One strength of pNets is their parameterised nature, making them suitable for to the representation of systems of arbitrary size, and making the modelling of parameterised systems possible. Parameters are also crucial to reason about interaction protocols that can address one entity inside an indexed set of processes. pNets have been successfully used to represent behavioural specification of parallel and distributed components and verify their correctness [2, 27]. VCE is the specification and verification platform that uses pNets as an intermediate representation. In this platform we have developed tool support for computing the symbolic semantics in term of open automata; this is presented in [43, 44], together with a case-study based on the on-board control software of satellites. In [8] we present how to encode reactive systems from the BIP specification language and check their temporal properties using VCE. In [29, 30] we describe our strong bisimulation algorithms, with illustration on the equivalence of different encodings of operators.

Open pNets are pNets with holes; they are adapted to represent processes parameterised by the behaviour of other processes, like composition operators or interaction protocols that synchronise the actions of processes that can be provided afterwards. Open pNets are hierarchical composition of automata with holes and parameters. We defined here a semantics for open pNets and a complete bisimulation theory for them. The semantics of open pNets relies on the definition of open automata that are automata with holes and parameters, but no hierarchy. Open automata are a flattened view of the pNet; their behaviour is expressed as open transitions that allow for a more semantic interpretation of process parameters (holes) than pNets. In the end, open automata are labelled transition systems with parameters and holes, a notion that is useful to define semantics, but makes less sense for the high level modelling of a system, compared to pNets. Open automata is the formalism that makes it possible to define FH-bisimilarity.

This article defines a strong and a weak bisimulation relation that are adapted to parameterised systems and hierarchical composition. FH-bisimulation handles pNet parameters in the sense that two states might be or not in relation depending on the value of parameters. Strong FH-bisimilarity is compositional in the sense that it is maintained when composing processes. We also identified a simple and realistic condition on the semantics of non-observable actions that allows weak FH-bisimilarity to be also compositional. Overall we believe that this article paved the way for a solid theoretical foundation for compositional verification of parallel and distributed systems.

The pNets formalism supports the refinement checking at the automaton level through a simulation, with symbolic evaluation of guards and transitions. The definition of simulation on open automata should be stronger than a classical simulation since it matches a transition with a family of transitions. Such a relation should be able to check the refinement by taking into account state

duplication, transition removal, guard strengthening, variable modification. Additionally, composition of pNets gives the possibility to either add new holes to a system or fill holes. A useful simulation relation should thus support the comparison of automata that do not have the same number of holes. Designing such a simulation relation is a non-trivial extension that we leave for future work.

We are currently looking at further properties of FH-bisimulation, but also the relations with existing equivalences on both closed and open systems. In particular, our model being significantly different from those considered in [32], it would be interesting to compare our "FH" family of bisimulations with the hierarchy of symbolic bisimulations from those authors. We also plan to apply open pNets to the study of complex composition operators in a symbolic way, for example in the area of parallel skeletons, or distributed algorithms.

Recently we published preliminary work on methods for checking weak FH-bisimulation [46]. The challenges here, in the context of our symbolic systems, are not so much algorithmic complexity, as was the case with classical weak bisimulation on finite models, but decidability and termination. The naive approach, using an explicit construction of the weak transition, may in itself introduce non-termination, so we prefer a direct implementation of the weak bisimulation definition, without constructing the weak automata beforehand, but searching *on demand* to construct the required weak transitions. We illustrate this approach on a simple error-correcting transport protocol case-study. Beside, we explore in [47] more pragmatic approaches using weak bisimulation preserving (pattern-based) reduction rules.

References

[1] Sergio Abriola, Pablo Barceló, Diego Figueira, and Santiago Figueira. Bisimulations on data graphs. *Journal of Artificial Intelligence Research*, 61:171–213, 2018.

[2] Rabéa Ameur-Boulifa, Ludovic Henrio, Oleksandra Kulankhina, Eric Madelaine, and Alexandra Savu. Behavioural semantics for asynchronous components. *Journal of Logical and Algebraic Methods in Programming*, 89:1–40, 2017.

[3] André Arnold. Synchronised behaviours of processes and rational relations. *Acta Informatica*, 17:21–29, 1982.

[4] Paolo Baldan, Andrea Bracciali, and Roberto Bruni. Bisimulation by unification. In *AMAST 2002 - Algebraic Methodology and Software Technology, 9th International Conference*, volume 2422 of *Lecture Notes in Computer Science*, pages 254–270. Springer, 2002.

[5] Paolo Baldan, Andrea Bracciali, and Roberto Bruni. A semantic framework for open processes. *Theoretical Computer Science*, 389(3):446–483, 2007.

[6] Tomás Barros, Rabéa Ameur-Boulifa, Antonio Cansado, Ludovic Henrio, and Eric Madelaine. Behavioural models for distributed fractal components. *Annales des Télécommunications*, 64(1-2):25–43, 2009.

[7] Harsh Beohar, Barbara König, Sebastian Küpper, and Alexandra Silva. Conditional transition systems with upgrades. *Science of Computer Programming*, 186:102320, 2020.

[8] Simon Bliudze, Ludovic Henrio, and Eric Madelaine. Verification of concurrent design patterns with data. In Hanne Riis Nielson and Emilio Tuosto, editors, *COORDINATION 2019 - 21st International Conference on Coordination Models and Languages*, volume LNCS-11533 of *Coordination Models and Languages*, pages 161–181. Springer International Publishing, 2019.

[9] Bard Bloom, Sorin Istrail, and Albert R. Meyer. Bisimulation can't be traced. *Journal of the ACM*, 42(1):232–268, 1995.

[10] Tommaso Bolognesi and Ed Brinksma. Introduction to the ISO specification language LOTOS. *Computer Networks*, 14(1):25–59, 1987.

[11] Johannes Borgström, Sébastien Briais, and Uwe Nestmann. Symbolic Bisimulation in the Spi Calculus. In *CONCUR 2004 - Concurrency Theory, 15th International Conference*, volume 3170 of *Lecture Notes in Computer Science*, pages 161–176. Springer, 2004.

[12] Rabéa Ameur Boulifa, Raluca Halalai, Ludovic Henrio, and Eric Madelaine. Verifying safety of fault-tolerant distributed components. In *FACS 2011 - 8th International Symposium on Formal Aspects of Component Software*, Lecture Notes in Computer Science. Springer, 2011.

[13] Maria Grazia Buscemi and Ugo Montanari. Open Bisimulation for the Concurrent Constraint Pi-Calculus. In *ESOP 2008 - Programming Languages and Systems, 17th European Symposium on Programming*, volume 4960 of *Lecture Notes in Computer Science*, pages 254–268. Springer, 2008.

[14] Muffy Calder and Carron Shankland. A symbolic semantics and bisimulation for full LOTOS. In *FORTE 2001 - 21st International Conference on Formal Techniques for Networked and Distributed Systems*, pages 185–200. Springer, 2001.

[15] Robert De Simone. Higher-level synchronising devices in MEIJE-SCCS. *Theoretical Computer Science*, 37:245–267, 1985.

[16] Stéphanie Delaune, Steve Kremer, and Mark Ryan. Symbolic Bisimulation for the Applied Pi Calculus. In *FSTTCS 2007 - Foundations of Software Technology and Theoretical Computer Science, 27th International Conference*, volume 4855 of *Lecture Notes in Computer Science*, pages 133–145. Springer, 2007.

[17] Jérémy Dubut. Bisimilarity of Diagrams. In *RAMiCS 2020 - Relational and Algebraic Methods in Computer Science, 18th International Conference*, volume 12062 of *Lecture Notes in Computer Science*, pages 65–81. Springer, 2020.

[18] Yuan Feng, Yuxin Deng, and Mingsheng Ying. Symbolic bisimulation for quantum processes. *ACM Transactions on Computational Logic (TOCL)*, 15(2):14, 2014.

[19] Nuno Gaspar, Ludovic Henrio, and Eric Madelaine. Formally reasoning on a reconfigurable component-based system - A case study for the industrial world. In *FACS 2013 - Formal Aspects of Component Software, 10th International Symposium,*, volume 8348 of *Lecture Notes in Computer Science*, pages 137–156. Springer, 2013.

[20] Jan Friso Groote. Transition system specifications with negative premises. *Theoretical Computer Science*, 118(2):263–299, 1993.

[21] Jan Friso Groote, Jeroen J. A. Keiren, Bas Luttik, Erik P. de Vink, and Tim A. C. Willemse. Modelling and analysing software in mcrl2. In Farhad Arbab and Sung-Shik Jongmans, editors, *Formal Aspects of Component Software*, pages 25–48, Cham, 2020. Springer International Publishing.

[22] Jan Friso Groote, Mohammad Reza Mousavi, and Michel A. Reniers. A hierarchy of SOS rule formats. *Electronic Notes in Theoretical Computer Science*, 156(1):3–25, 2006. Proceedings of the Second Workshop on Structural Operational Semantics.

[23] Jan Friso Groote and Frits Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, 1992.

[24] David Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming.*, 8(3):231–274, 1987.

[25] Matthew Hennessy and Huimin Lin. Symbolic bisimulations. *Theoretical Computer Science*, 138(2):353–389, 1995.

[26] Matthew Hennessy and Julian Rathke. Bisimulations for a calculus of broadcasting systems. *Theoretical Computer Science*, 200(1-2):225–260, 1998.

[27] Ludovic Henrio, Oleksandra Kulankhina, Siqi Li, and Eric Madelaine. Integrated environment for verifying and running distributed components. In *FASE 2016 - 19th International Conference on Fundamental Approaches to Software Engineering*, pages 66–83. Springer Berlin Heidelberg, 2016.

[28] Ludovic Henrio, Eric Madelaine, and Min Zhang. A Theory for the Composition of Concurrent Processes. In *FORTE 2016 - 36th International Conference on Formal Techniques for Distributed Objects, Components, and Systems*, volume 9688, pages 175 – 194. Springer, 2016.

[29] Zechen Hou and Eric Madelaine. Symbolic Bisimulation for Open and Parameterized Systems. In *PEPM 2020 - Workshop on Partial Evaluation and Program Manipulation*. ACM SIGPLAN, 2020.

[30] Zechen Hou, Eric Madelaine, Jing Liu, and Yuxin Deng. Symbolic Bisimulation for Open and Parameterized Systems - Extended version. Research Report RR-9304, Inria & Université Cote d'Azur, CNRS, I3S, Sophia Antipolis, France ; East China Normal University (Shanghai), November 2019.

[31] Hans Hüttel and Kim Guldstrand Larsen. The use of static constructs in A modal process logic. In *Logic at Botik '89, Symposium on Logical Foundations of Computer Science*, volume 363 of *Lecture Notes in Computer Science*, pages 163–180. Springer, 1989.

[32] Anna Ingólfsdóttir and Huimin Lin. A symbolic approach to value-passing processes. In *Handbook of Process Algebra*, pages 427–478. North-Holland/Elsevier, 2001.

[33] Kenneth Johnson and Radu Calinescu. Efficient Re-resolution of SMT Specifications for Evolving Software Architectures. In *QoSA 2014 - 10th International ACM Sigsoft Conference on Quality of Software Architectures*, pages 93–102. ACM, 2014.

[34] Kenneth Johnson, Radu Calinescu, and Shinji Kikuchi. An incremental verification framework for component-based software systems. In *CBSE 2013 - 16th International ACM Sigsoft Symposium on Component-based Software Engineering*, pages 33–42. ACM, 2013.

[35] Kim G. Larsen. A context dependent equivalence between processes. *Theoretical Computer Science*, 49:184–215, 1987.

[36] Kim Guldstrand Larsen and Liu Xinxin. Compositionality through an operational semantics of contexts. *Journal of Logic and Computation*, 1(6):761–795, 1991.

[37] Maurice Laveaux and Tim A. C. Willemse. Decomposing monolithic processes in a process algebra with multi-actions. In *ICE*, volume 347 of *EPTCS*, pages 57–76, 2021.

[38] Huimin Lin. Symbolic transition graph with assignment. In *CONCUR'96 - Concurrency Theory, 7th International Conference*, volume 1119, pages 50–65. Springer Berlin Heidelberg, 1996.

[39] Jia Liu and Huimin Lin. A Complete Symbolic Bisimulation for Full Applied Pi Calculus. In *SOFSEM 2010 - 36th Conference on Current Trends in Theory and Practice of Computer Science*, volume 5901, pages 552–563. Springer, 2010.

[40] Robin Milner. *A Calculus of Communicating Systems*. Springer-Verlag, Berlin, Heidelberg, 1982.

[41] Robin Milner. *Communication and Concurrency*. Int. Series in Computer Science. Prentice-Hall, Englewood Cliffs, New Jersey, 1989. SU Fisher Research 511/24.

[42] Gordon D. Plotkin. A structural approach to operational semantics. *Journal of Logic and Algebraic Programming*, 60-61:17–139, 2004.

[43] Xudong Qin, Simon Bliudze, Eric Madelaine, and Min Zhang. Using SMT engine to generate symbolic automata. In *AVOCS 2018 - 18th International Workshop on Automated Verification of Critical Systems*, volume 076. Electronic Communications of the EASST, 2018.

[44] Xudong Qin, Simon Bliudze, Eric Madelaine, and Min Zhang. Using SMT engine to generate Symbolic Automata -Extended version. Research Report RR-9177, Inria & Université Cote d'Azur, CNRS, I3S, Sophia Antipolis, France ; inria, June 2018.

[45] Robert Jan Van Glabbeek. The meaning of negative premises in transition system specifications II. *The Journal of Logic and Algebraic Programming*, 60-61:229–258, 2004.

[46] Biyang Wang, Eric Madelaine, and Min Zhang. New symbolic model and equivalences checking for open automata. In *SMC 2021 - IEEE International Conference on Systems, Man, and Cybernetics*, pages 2360–2367. IEEE, 2021.

[47] Biyang Wang, Eric Madelaine, and Min Zhang. Symbolic Weak Equivalences: Extension, Algorithms, and Minimization - Extended version. Research Report RR-9389, Inria, Université Cote d'Azur, CNRS, I3S, Sophia Antipolis, France; East China Normal University (Shanghai), 2021.

Appendix A. Proof on FH-bisimulation

Appendix A.1. Bisimilarity is an equivalence: Proof of Theorem 1

Suppose $\mathcal{R}$ is an FH-bisimilarity. Then $\mathcal{R}$ is an equivalence, that is, $\mathcal{R}$ is reflexive, symmetric and transitive.

Proof. It is trivial to check reflexivity and symmetry. Here we focus on the transitivity. To prove transitivity of strong FH-bisimilarity on pNets it is sufficient to prove transitivity of the strong FH-bisimilarity on states. Consider 3 open automata $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_3$ and states s, t, u in those automata[12]. Suppose we have $\mathcal{R}$ an FH-bisimulation relation between states of $\mathcal{T}_1$ and of $\mathcal{T}_2$; members of $\mathcal{R}$ are of the form $(s, t|Pred_{s,t})$. Suppose we also have $\mathcal{R}'$ an FH-bisimulation relation between states of $\mathcal{T}_2$ and of $\mathcal{T}_3$; members of $\mathcal{R}'$ are of the form $(t, u|Pred_{t,u})$.

Let $\mathcal{R}''$ be the relation:

$$\mathcal{R}'' = \{(s, u|Pred_{s,u}) \;\Big|\; Pred_{s,u} = \bigvee_{\substack{(s,t|Pred_{s,t}) \in \mathcal{R} \\ (t,u|Pred_{t,u}) \in \mathcal{R}'}} Pred_{s,t} \wedge Pred_{t,u}\}$$

This relation is the adaptation of the transitivity to the conditional relationship that defines a bisimulation. Indeed the global disjunction together with the conjunction of predicates plays exactly the role of the intermediate element in a transitivity rule: "there exists an intermediate state" corresponds to the global disjunction, and the conjunction of states expresses the intermediate predicate is used to ensure satisfiability of the predicate relating the first state to the last one.

The relation is built as follows: for each pair of states s, u, for each state t such that $\mathcal{R}$ relates s and t, and $\mathcal{R}'$ relates t and u, we take the conjunction of the two predicates. The predicates for different values of t are collected by a disjunction.

We will show that $\mathcal{R}''$ is an FH-bisimulation. Consider $(s, u|Pred_{s,u}) \in \mathcal{R}''$. Then there is a set of states of $\mathcal{T}_2$ relating s and u, let $(t_p)^{p\in P}$ be this family. We have $Pred_{s,u} = \bigvee_{p\in P} Pred_{s,p} \wedge Pred_{p,u}$.

For any $p \in P$ by definition of $\mathcal{R}''$, $(s, t_p|Pred_{s,p}) \in \mathcal{R}$, and $(t_p, u|Pred'_{p,u}) \in \mathcal{R}'$. We have the following by definition of bisimulation: For any open transition OT in $\mathcal{T}_1$ originating from s.

$$\frac{\beta_j^{j\in J_1}, Pred_{OT}, Post_{OT}}{s \xrightarrow{\alpha} s'}$$

[12] We omit the constraints stating that each s_x, t_x, u_x is in the states of $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_3$ for the sake of readability.

There exists an indexed set of open transitions $OT_{px}^{x\in X} \subseteq \mathcal{T}_2$:

$$\frac{\beta_{jpx}^{j\in J_{px}}, Pred_{OT_{px}}, Post_{OT_{px}}}{t_p \xrightarrow{\alpha_{px}} t_{px}} \qquad (*)$$

such that $\forall x, J_1 = J_{px}, (s', t_{px}|Pred_{px}) \in \mathcal{R}$; and

$$Pred_{s,p} \wedge Pred_{OT}$$
$$\implies \bigvee_{x\in X} \left(\forall j.\beta_j = \beta_{jpx} \wedge Pred_{OT_{px}} \wedge \alpha = \alpha_{px} \wedge Pred_{px}\{\!\{Post_{OT} \uplus Post_{OT_{px}}\}\!\}\right)$$

For any open transition OT_{px}, since $(t_p, u|Pred'_{p,u}) \in \mathcal{R}'$ there exists an indexed set of open transitions $OT_{pxy}^{y\in Y} \subseteq \mathcal{T}_3$:

$$\frac{\beta_{jpxy}^{j\in J_{pxy}}, Pred_{OT_{pxy}}, Post_{OT_{pxy}}}{u \xrightarrow{\alpha_{pxy}} u_{pxy}} \qquad (**)$$

such that $\forall y, J_{px} = J_{pxy}, (t_{px}, u_{pxy}|Pred_{pxy}) \in \mathcal{R}'$; and

$$Pred'_{p,u} \wedge Pred_{OT_{px}} \implies$$
$$\bigvee_{y\in Y} \left(\forall j.\beta_{jpx} = \beta_{jpxy} \wedge Pred_{OT_{pxy}} \wedge \alpha_{px} = \alpha_{pxy} \wedge Pred_{pxy}\{\!\{Post_{OT_{px}} \uplus Post_{OT_{pxy}}\}\!\}\right)$$

This is verified for each $p \in P$.
Overall, we have a family of open transitions $OT_{pxy}^{p\in P, x\in X, y\in Y} \subseteq \mathcal{T}_3$ that should simulate OT.

First, we have $\forall y, \forall x, \forall p, J_1 = J_{px} = J_{pxy}, (s', u_{pxy}|Pred'_{pxy}) \in \mathcal{R}''$ for some $Pred'_{pxy}$. Indeed for any p, x, and y, t_{px} relates s' and u_{pxy}, we have $(s', t_{px}|Pred_{px}) \in \mathcal{R}$ and $(t_{px}, u_{pxy}|Pred_{pxy}) \in \mathcal{R}'$. More precisely, $t_{px} \in (t'_p)^{p\in P'}$ where $(t'_p)^{p\in P'}$ and $P' \subseteq P$ is the set of states relating s' and u_{pxy} (the states used in the open transition must belong to the set of states ensuring the transitive relation). Additionally, for all p, x, y, $Pred_{px} \wedge Pred_{pxy} \implies Pred'_{pxy}$ (this is one element of the disjunction defining the predicate $Pred'_{pxy}$ relating s' and u_{pxy} in the definition of $\mathcal{R}''$).

One can notice that, as bisimulation predicates are used to relate states that belong to two different open automata, the free variables of these predicates that do not belong to the two related automata can safely be renamed to avoid any name clash. In practice, we can suppose that $Pred'_{pxy}$ does not contain the variables of $\mathcal{T}_2$ because it is used to relate states of $\mathcal{T}_1$ and $\mathcal{T}_3$. Indeed if $Pred'_{pxy}$ uses variables of $\mathcal{T}_2$, we can consider instead another predicate that is equivalent to $Pred'_{pxy}$ and does not contain the variables of $\mathcal{T}_2$ (this is safe according to the semantic interpretation of open automata and relations). Similarly, we can

suppose that $Pred_{px}$ contains no variable in $\mathcal{T}_3$, and $Pred_{pxy}$ contains no variable in $\mathcal{T}_1$.

Second, by definition of bisimulation we need (recall that $Pred_{s,u}$ is the original predicate relating s and u by definition of the transitive closure):
$Pred_{s,u} \wedge Pred_{OT} \implies$
$\bigvee_{x\in X}\bigvee_{y\in Y}\bigvee_{p\in P}\Big(\forall j.\beta_j = \beta_{jpxy} \wedge Pred_{OT_{pxy}} \wedge \alpha = \alpha_{pxy} \wedge Pred'_{pxy}\{\!\{Post_{OT} \uplus Post_{OT_{pxy}}\}\!\}\Big)$.

From (*) and (**) we have:
for all p, $Pred_{s,p} \wedge Pred_{OT} \wedge Pred_{p,u}$
$\implies \bigvee_{x\in X}\Big(\forall j.\beta_j = \beta_{jpx} \wedge Pred_{OT_{px}} \wedge \alpha = \alpha_{px} \wedge Pred_{px}\{\!\{Post_{OT} \uplus Post_{OT_{px}}\}\!\}\Big) \wedge Pred_{p,u}$
$\implies \bigvee_{x\in X}\Big(\forall j.\beta_j = \beta_{jpx} \wedge (Pred_{OT_{px}} \wedge Pred_{p,u}) \wedge \alpha = \alpha_{px} \wedge Pred_{px}\{\!\{Post_{OT} \uplus Post_{OT_{px}}\}\!\}\Big)$
$\implies \bigvee_{x\in X}\Big(\forall j.\beta_j = \beta_{jpx} \wedge \bigvee_{y\in Y}\Big(\forall j'.\beta_{j'px} = \beta_{j'pxy} \wedge Pred_{OT_{pxy}} \wedge \alpha_{px} = \alpha_{pxy}$
$\wedge\, Pred_{pxy}\{\!\{Post_{OT_{px}} \uplus Post_{OT_{pxy}}\}\!\}\Big) \wedge \alpha = \alpha_{px} \wedge Pred_{px}\{\!\{Post_{OT} \uplus Post_{OT_{px}}\}\!\}\Big)$
$\implies \bigvee_{x\in X}\bigvee_{y\in Y}\Big(\forall j, j'.\beta_j = \beta_{jpx} \wedge \beta_{j'px} = \beta_{j'pxy} \wedge \Big(Pred_{OT_{pxy}} \wedge \alpha = \alpha_{px} = \alpha_{pxy}$
$\wedge\, Pred_{pxy}\{\!\{Post_{OT_{px}} \uplus Post_{OT_{pxy}}\}\!\} \wedge Pred_{px}\{\!\{Post_{OT} \uplus Post_{OT_{px}}\}\!\}\Big)\Big)$

By construction, four substitution functions $\{\!\{\ }\!\}$ only have an effect on the variables of the open automaton they belong to, they also produce terms containing only variables of the open automaton they belong to. Finally, because of the domain of the substitution functions of the predicates, we have:
$Pred_{px}\{\!\{Post_{OT} \uplus Post_{OT_{px}}\}\!\} \wedge Pred_{pxy}\{\!\{Post_{OT_{px}} \uplus Post_{OT_{pxy}}\}\!\} \Leftrightarrow$
$Pred_{px}\{\!\{Post_{OT} \uplus Post_{OT_{px}} \uplus Post_{OT_{pxy}}\}\!\} \wedge Pred_{pxy}\{\!\{Post_{OT} \uplus Post_{OT_{px}} \uplus Post_{OT_{pxy}}\}\!\}$
$\implies Pred'_{pxy}\{\!\{Post_{OT} \uplus Post_{OT_{px}} \uplus Post_{OT_{pxy}}\}\!\} \Leftrightarrow$
$Pred'_{pxy}\{\!\{Post_{OT} \uplus Post_{OT_{pxy}}\}\!\}$

This allows us to conclude, with $Pred_{s,u} = \bigvee_{p\in P} Pred_{s,p} \wedge Pred_{p,u}$:
$Pred_{s,u} \wedge Pred_{OT}$
$\implies \bigvee_{p\in P}(Pred_{s,p} \wedge Pred_{p,u} \wedge Pred_{OT})$
$\implies \bigvee_{p\in P}\bigvee_{x\in X}\bigvee_{y\in Y}\Big(\forall j, j'.\beta_j = \beta_{jpx} \wedge \beta_{j'px} = \beta_{j'pxy} \wedge Pred_{OT_{pxy}}$
$\wedge\, \alpha = \alpha_{px} = \alpha_{pxy} \wedge Pred'_{pxy}\{\!\{Post_{OT} \uplus Post_{OT_{pxy}}\}\!\}\Big)$
$\implies \bigvee_{x\in X}\bigvee_{y\in Y}\bigvee_{p\in P}\Big(\forall j.\beta_j = \beta_{jpxy} \wedge Pred_{OT_{pxy}} \wedge \alpha = \alpha_{pxy} \wedge Pred'_{pxy}\{\!\{Post_{OT} \uplus Post_{OT_{pxy}}\}\!\}\Big)$

Concerning the other direction of bisimulation, it is sufficient to notice that the role of s and u in the definition of $\mathcal{R}''$ is symmetrical, and thus the proof is similar. □

Appendix A.2. Composition Lemmas

The proofs of the composition theorems for FH-bisimilarity rely on two main lemmas, dealing respectively with the decomposition of a composed behaviour between the context and the internal pNet, and with their recomposition.

Lemma 1*: Open transition decomposition*

Consider two pNets P and Q that are not pLTSs[13]. Let $\text{Leaves}(Q) = p_l^{l\in L_Q}$; suppose:

$$P[Q]_{j_0} \models \frac{\beta_j^{j\in J}, \mathit{Pred}, \mathit{Post}}{\lhd s_i^{i\in L} \rhd \xrightarrow{\alpha} \lhd s_i'^{i\in L} \rhd}$$

with $J \cap \text{Holes}(Q) \neq \emptyset$ or $\exists i \in L_Q.\, s_i \neq s_i'$, i.e. Q takes part in the reduction. Then, there exist α_Q, Pred', Pred'', Post', Post'' s.t.:

$$P \models \frac{\beta_j^{j\in(J\setminus\text{Holes}(Q))\cup\{j_0\}}, \mathit{Pred}', \mathit{Post}'}{\lhd s_i^{i\in L\setminus L_Q} \rhd \xrightarrow{\alpha} \lhd s_i'^{i\in L\setminus L_Q} \rhd}$$

$$\text{and} \quad Q \models \frac{\beta_j^{j\in J\cap\text{Holes}(Q)}, \mathit{Pred}'', \mathit{Post}''}{\lhd s_i^{i\in L_Q} \rhd \xrightarrow{\alpha_Q} \lhd s_i'^{i\in L_Q} \rhd}$$

and $\mathit{Pred} \iff \mathit{Pred}' \wedge \mathit{Pred}'' \wedge \alpha_Q = \beta_{j_0}$, $\mathit{Post} = \mathit{Post}' \uplus \mathit{Post}''$ where Post'' is the restriction of Post over variables $\mathit{vars}(Q)$.

Preliminary note: The introduction of fresh variables introduce alpha-conversion at many points of the proof; we only give major arguments concerning alpha-conversion to make the proof readable; in general, fresh variables appear in each transition inside terms β_j, v, and Pred.

Proof. Consider rule **Tr2** in Definition 9, applied to the pNet $P[Q]_{j_0}$.

$$\frac{\begin{array}{c}\text{Leaves}(\langle\!\langle P_m^{m\in I}, \overline{Sort}, SV_k^{k\in K}\rangle\!\rangle) = pLTS_l^{l\in L} \qquad k\in K \qquad SV_k = (\alpha'_m)^{m\in I_1\uplus I_2\uplus J} \to \alpha'[e_b] \\ \forall m\in I_1.P_m \models \dfrac{\beta_j^{j\in J_m}, \mathit{Pred}_m, \mathit{Post}_m}{\lhd s_i^{i\in L_m} \rhd \xrightarrow{\alpha_m} \lhd (s_i')^{i\in L_m} \rhd} \qquad \forall m\in I_2.P_m \models \dfrac{\emptyset, \mathit{Pred}_m, \mathit{Post}_m}{\lhd s_m \rhd \xrightarrow{\alpha_m} \lhd s'_m \rhd} \\ J' = \biguplus_{m\in I_1} J_m \uplus J \qquad \mathit{Pred} = \bigwedge_{m\in I_1\uplus I_2} \mathit{Pred}_m \wedge \mathit{Predsv}(SV_k, \alpha_m^{m\in I_1\uplus I_2}, \beta_j^{j\in J}, \alpha) \\ \forall i \in L\setminus\left(\biguplus_{m\in I_1} L_m \uplus I_2\right).\, s_i' = s_i \qquad \text{fresh}(\alpha'_m, \alpha', \beta_j, \alpha)\end{array}}{\langle\!\langle P_m^{m\in I}, \overline{Sort}, SV_k^{k\in K}\rangle\!\rangle \models \dfrac{\beta_j{}^{j\in J'}, \mathit{Pred}, \biguplus_{m\in I_1\uplus I_2} \mathit{Post}_m}{\lhd s_i^{i\in L} \rhd \xrightarrow{\alpha} \lhd (s_i')^{i\in L} \rhd}}\ \textbf{Tr2}$$

[13] A similar lemma can be proven for a pLTS Q.

We know each premise is *True* for $P[Q]_{j_0}$. $j_0 \in I_1$ because Q is not a pLTS. We try to prove the equivalent premise for P.

First, K and the synchronisation vector SV_k are unchanged (however j_0 passes from the set of sub-pNets to the set of holes). We have $\text{Leaves}(P[Q]_{j_0}) = \text{Leaves}(P) \uplus \text{Leaves}(Q)$.

Now focus on the OTs of the sub-pNets. For each $m \in I_1 \uplus I_2$ we have one of the two following OT:
either m in I_1

$$P_m \models \frac{\beta_j^{j \in J_m}, Pred_m, Post_m}{\lhd s_i^{i \in L_m} \rhd \xrightarrow{\alpha_m} \lhd (s'_i)^{i \in L_m} \rhd}$$

or, m in I_2

$$P_m \models \frac{\emptyset, Pred_m, Post_m}{\lhd s_m \rhd \xrightarrow{\alpha_m} \lhd s'_m \rhd}$$

Only elements of $(I_1 \uplus I_2) \backslash \{j_0\}$ are useful to assert the premise for reduction of P; the last one ensures the open transition for the pNet Q (note that Q is at place j_0, and by definition of the open transition for $P[Q]_{j_0}$, $L_{j_0} = L_Q$, and $J_{j_0} = J \cap \text{Holes}(Q)$):

$$Q \models \frac{\beta_j^{j \in J \cap \text{Holes}(Q)}, Pred_{j_0}, Post''}{\lhd s_i^{i \in L_Q} \rhd \xrightarrow{\alpha_{j_0}} \lhd (s'_i)^{i \in L_Q} \rhd}$$

This already ensures the second part of the conclusion of the lemma, i.e. the OT for Q if we choose $\alpha_Q = \alpha_{j_0}$ and $Pred'' = Pred_{j_0}$. Considering the OT of P we have another J' that is $J'_p = J' \setminus \text{Holes}(Q) \uplus \{j_0\}$; we denote $I'_1 = I_1 \setminus \{j_0\}$ the predicate is $Pred' = \bigwedge_{m \in I'_1 \uplus I_2} Pred_m \wedge Pred_{sv}(SV_k, \alpha_m^{m \in I'_1 \uplus I_2}, \beta_j^{j \in J \cup \{j_0\}}, \alpha)$
where

$$Pred_{sv}(SV_k, \alpha_m^{m \in I'_1 \uplus I_2}, \beta_j^{j \in J \cup \{j_0\}}, \alpha) \Leftrightarrow$$
$$\forall i \in I'_1 \uplus I_2.\, \alpha_i = \alpha'_i \wedge \forall j \in J \cup \{j_0\}.\, \beta_j = \alpha'_j \wedge \alpha = \alpha' \wedge e_b$$

Modulo renaming of fresh variables, this is identical to the predicate that occurs in the source open transition except $\alpha_{j_0} = \alpha'_{j_0}$ has been replaced by $\beta_{j_0} = \alpha'_{j_0}$. As $\alpha_{j_0} = \alpha_Q$ and β_{j_0} is free, we have $\beta_{j_0} = \alpha'_{j_0} \wedge \beta_{j_0} = \alpha_Q \iff \alpha_{j_0} = \alpha'_{j_0}$. Thus, $Pred \iff (Pred' \wedge Pred'') \wedge \alpha_Q = \beta_{j_0}$. Finally, Post into conditions of the context P and the pNet Q (they are built similarly as they only deal with leaves): $Post = Post' \uplus Post''$. This concludes the proof as we checked all the premises of the open transition for both P and Q. We obtain the following reduction by the rule **Tr2**:

$$\text{Leaves}(\langle\!\langle P_m^{m\in I\setminus\{j_0\}}, \overline{Sort}, SV_k^{k\in K}\rangle\!\rangle) = pLTS_l^{l\in L} \qquad k\in K \qquad SV_k = (\alpha'_m)^{m\in I_1\uplus I_2\uplus J} \to \alpha'[e_b]$$

$$\forall m\in I_1\setminus\{j_0\}.P_m \models \frac{\beta_j^{j\in J_m}, Pred_m, Post_m}{\lhd s_i^{i\in L_m}\rhd \xrightarrow{\alpha_m} \lhd (s'_i)^{i\in L_m}\rhd} \qquad \forall m\in I_2.P_m \models \frac{\emptyset, Pred_m, Post_m}{\lhd s_m\rhd \xrightarrow{\alpha_m} \lhd s'_m\rhd}$$

$$J' = \biguplus_{m\in I_1\setminus\{j_0\}} J_m \uplus J$$

$$Pred' = \bigwedge_{m\in I_1\uplus I_2\setminus\{j_0\}} Pred_m \wedge Pred_{sv}(SV_k, \alpha_m^{m\in I_1\uplus I_2\setminus\{j_0\}}, \beta_j^{j\in J\cup\{j_0\}}, \alpha)$$

$$\forall i\in L\setminus\left(\biguplus_{m\in I_1\setminus\{j_0\}} L_m \uplus I_2\right).\, s'_i = s_i \qquad \text{fresh}(\alpha'_m, \alpha', \beta_j, \alpha)$$

$$\langle\!\langle P_m^{m\in I\setminus\{j_0\}}, \overline{Sort}, SV_k^{k\in K}\rangle\!\rangle \models \frac{\beta_j{}^{j\in J\setminus \text{Holes}(Q)\uplus\{j_0\}}, Pred', \biguplus_{m\in I_1\setminus\{j_0\}\uplus I_2} Post_m}{\lhd s_i^{i\in L\setminus L_Q}\rhd \xrightarrow{\alpha} \lhd (s'_i)^{i\in L\setminus L_Q}\rhd}$$

□

In general, the actions that can be emitted by Q is a subset of the possible actions of the holes, and the predicate involving v_Q and the synchronisation vector is more restrictive than the one involving only the variable β_{j_0}. This has no impact on the previous proof and this restriction results from the composition of predicates.

Lemma 2: Open transition composition

Consider two pNets P and Q where P is not a pLTS. Suppose $j_0 \in J$ and:

$$P \models \frac{\beta_j^{j\in J}, Pred, Post}{\lhd s_i^{i\in L}\rhd \xrightarrow{\alpha} \lhd s'_i{}^{i\in L}\rhd} \qquad \text{and} \qquad Q \models$$

$$\frac{\beta_j^{j\in J_Q}, Pred', Post'}{\lhd s_i^{i\in L_Q}\rhd \xrightarrow{\alpha_Q} \lhd s'_i{}^{i\in L_Q}\rhd}$$

Then, we have:

$$P[Q]_{j_0} \models \frac{\beta_j^{(j\in J\setminus\{j_0\})\uplus J_Q}, Pred\wedge Pred'\wedge \alpha_Q = \beta_{j_0}, Post\uplus Post'}{\lhd s_i^{i\in L\uplus L_Q}\rhd \xrightarrow{\alpha} \lhd s'_i{}^{i\in L\uplus L_Q}\rhd}$$

Note that this does not mean that any two pNets can be composed and produce an open transition. Indeed, the predicate $Pred\wedge Pred'\wedge\alpha_Q = \beta_{j_0}$ will not be satisfiable if the action of α_Q cannot be matched with β_{j_0}. Note also that β_{j_0} is now only used as an intermediate term inside formulas: it does not appear neither as global action nor as an action of a hole.

PROOF. Let $P = \langle\!\langle P_m^{m\in I}, \overline{Sort}, SV_k^{k\in K}\rangle\!\rangle$. Consider first the open transition derived from P. Consider each premise of the open transition (constructed by **Tr2** rule in Definition 9).

$$\text{Leaves}(\langle\!\langle P_m^{m\in I}, \overline{Sort}, SV_k^{k\in K}\rangle\!\rangle) = pLTS_l^{l\in L} \qquad k\in K \qquad SV_k = (\alpha'_m)^{m\in I_1\uplus I_2\uplus J} \to \alpha'[e_b]$$

$$\forall m\in I_1.P_m \models \frac{\beta_j^{j\in J_m}, Pred_m, Post_m}{\lhd s_i^{i\in L_m}\rhd \xrightarrow{\alpha_m} \lhd (s'_i)^{i\in L_m}\rhd} \qquad \forall m\in I_2.P_m \models \frac{\emptyset, Pred_m, Post_m}{\lhd s_m\rhd \xrightarrow{\alpha_m} \lhd s'_m\rhd}$$

$$J' = \biguplus_{m\in I_1} J_m \uplus J \qquad Pred = \bigwedge_{m\in I_1\uplus I_2} Pred_m \wedge Pred_{sv}(SV_k, \alpha_m^{m\in I_1\uplus I_2}, \beta_j^{j\in J}, \alpha)$$

$$\forall i\in L\setminus\left(\biguplus_{m\in I_1} L_m \uplus I_2\right).\, s'_i = s_i \qquad \text{fresh}(\alpha'_m, \alpha', \beta_j, \alpha)$$

Tr2

$$\langle\!\langle P_m^{m\in I}, \overline{Sort}, SV_k^{k\in K}\rangle\!\rangle \models \frac{\beta_j{}^{j\in J'}, Pred, \biguplus_{m\in I_1\uplus I_2} Post_m}{\lhd s_i^{i\in L}\rhd \xrightarrow{\alpha} \lhd (s'_i)^{i\in L}\rhd}$$

We know each premise is *True* for P and try to prove the equivalent premise for $P[Q]_{j_0}$ (using the open transition of Q). $P[Q]_{j_0}$ exhibits a similar **Tr2** rule where K and the synchronisation vector are unchanged (j_0 is now in the set of sub-pNets); $SV_k = (\alpha'_j)^{j\in I\uplus\{j_0\}\uplus J} \to \alpha'[e_b]$. $\text{Leaves}(P[Q]_{j_0}) = \text{Leaves}(P) \uplus \text{Leaves}(Q)$. I and J are the set of leaves and holes of P, $I_1 \uplus I_2$ and J' are the sets of moving leaves and holes in the reduction of P. All sub-pNets of must be reduced, we need:

$$\forall m\in I_1\uplus\{j_0\}.P_m \models \frac{\beta_j^{j\in J_m}, Pred_m, Post_m}{\lhd s_i^{i\in L_m}\rhd \xrightarrow{\alpha_m} \lhd (s'_i)^{i\in L_m}\rhd}$$

$$\forall m\in I_2.P_m \models \frac{\emptyset, Pred_m, Post_m}{\lhd s_m\rhd \xrightarrow{\alpha_m} \lhd s'_m\rhd}$$

the sub-pNet at position j_0 is the one filled by Q (we define $P_{j_0} = Q$ and similarly $J_m = J_Q$, $Pred_m = Pred'$, $Post_m = Post'$,... are the elements of the OT of Q) which offers an open transition by hypothesis, the other open transitions are immediate consequence of the open transition that can be performed by P (premises of **Tr2**). The set of moving leaves is the union of the moving leaves in the open transition for P and the ones for Q; similarly the moving holes are the union of the moving holes, minus j_0: $J'_{PQ} = J \setminus \{j_0\} \uplus J_Q$. The predicate for the open transition is:
$Pred'' = \bigwedge_{m\in I_1\uplus I_2} Pred_m \wedge Pred' \wedge Pred(SV_k, v_i^{i\in I_1\uplus I_2} \uplus (j_0\mapsto v_Q), \beta_j^{j\in J}, \alpha)$.
By definition we have:
$Pred(SV_k, \alpha_i^{i\in I} \uplus (j_0\mapsto\alpha_Q), \beta_j^{j\in J}, v) \Leftrightarrow \forall i\in I.\,\alpha_i = \alpha'_i \wedge \forall j\in J.\,\beta_j = \alpha'_j \wedge \alpha = \alpha' \wedge \alpha_Q = \alpha_{j_0} \wedge e_b$, this is equivalent to $\forall i\in I.\,\alpha_i = \alpha'_i \wedge \forall j\in J.\,\beta_j = \alpha'_j \wedge \alpha = \alpha' \wedge \alpha_Q = \beta_{j_0} \wedge \beta_{j_0} = \alpha_{j_0} \wedge e_b$ and by definition of *Pred* (as obtained by applying **Tr2** rule), $Pred'' \iff Pred \wedge Pred' \wedge \alpha_Q = \beta_{j_0}$. The post-condition gathers

the post-conditions related to all the leaves: $\biguplus_{m\in I_1\cup\{j_0\}\uplus I_2} Post_m = Post \uplus Post'$.

Finally, the composed open transition can be built by **Tr2** rule as follows:

$$\frac{\begin{array}{c}\text{Leaves}(\langle\!\langle P_m^{m\in I\cup\{j_0\}}, \overline{Sort}, SV_k^{k\in K}\rangle\!\rangle) = pLTS_l^{l\in L} \qquad k\in K \qquad SV_k = (\alpha'_m)^{m\in I_1\uplus I_2\uplus J} \to \alpha'[e_b] \\ \forall m\in I_1\cup\{j_0\}.P_m \models \dfrac{\beta_j^{j\in J_m}, Pred_m, Post_m}{\lhd s_i^{i\in L_m}\rhd \xrightarrow{\alpha_m} \lhd (s'_i)^{i\in L_m}\rhd} \qquad \forall m\in I_2.P_m \models \dfrac{\emptyset, Pred_m, Post_m}{\lhd s_m\rhd \xrightarrow{\alpha_m} \lhd s'_m\rhd} \\ J'_{PQ} = J\setminus\{j_0\}\uplus J_Q \\ Pred'' = \bigwedge_{m\in I_1\uplus I_2} Pred_m \wedge Pred' \wedge Pred(SV_k, v_i^{i\in I_1\uplus I_2}\uplus(j_0\mapsto v_Q), \beta_j^{j\in J}, \alpha) \\ \forall i\in L\setminus\left(\biguplus_{m\in I_1\cup\{j_0\}} L_m\uplus I_2\right).s'_i = s_i \qquad \text{fresh}(\alpha'_m, \alpha', \beta_j, \alpha)\end{array}}{\langle\!\langle P_m^{m\in I}, \overline{Sort}, SV_k^{k\in K}\rangle\!\rangle \models \dfrac{\beta_j{}^{j\in J'_{PQ}}, Pred'', Post\uplus Post'}{\lhd s_i^{i\in L\uplus L_Q}\rhd \xrightarrow{\alpha} \lhd (s'_i)^{i\in L\uplus L_Q}\rhd}}$$

This provides the desired conclusion. □

Note that we also have the following lemma (trivial):

Lemma 3 (Open transition composition – inactive).
This lemma is the simple case where the pNet filling the hole is not involved in the transition. Suppose $j_0 \notin J$ and $L_Q = \text{Leaves}(Q)$:

$$P \models \frac{\beta_j^{j\in J}, Pred, Post}{\lhd s_i^{i\in L}\rhd \xrightarrow{\alpha} \lhd s'_i{}^{i\in L}\rhd}$$

Then, for any state $\lhd s_i^{i\in L_Q}\rhd$ of Q, we have:

$$P[Q]_{j_0} \models \frac{\beta_j^{j\in J}, Pred, Post}{\lhd s_i^{i\in L} \uplus s_i^{i\in L_Q}\rhd \xrightarrow{\alpha} \lhd s'_i{}^{i\in L} \uplus s_i^{i\in L_Q}\rhd}$$

The proof is trivial.

Appendix A.3. Proof of Theorem 3

Congruence: *Consider an open pNet: $P = \langle\!\langle P_i^{i\in I}, Sort_j^{j\in J}, \overline{SV}\rangle\!\rangle$. Let $j_0 \in J$ be a hole. Let Q and Q' be two FH-bisimilar pNets such that* $\text{Sort}(Q) = \text{Sort}(Q') \subseteq Sort_{j_0}$[14]. *Then $P[Q]_{j_0}$ and $P[Q']_{j_0}$ are FH-bisimilar.*

[14]Note that $\text{Sort}(Q) = \text{Sort}(Q')$ is ensured by strong bisimilarity.

PROOF. The proof of Theorem 3 exhibits classically a bisimulation relation for a composed system. It considers then an open transition of $P[Q]_{j_0}$ that should be simulated. It then uses Lemma 1 to decompose the open transition of $P[Q]_{j_0}$ and obtain an open transition of P and Q; the FH-bisimulation property can be applied to Q to obtain an equivalent family of open transitions of Q'; this family is then recomposed by Lemma 2 to build a set of open transitions of $P[Q']_{j_0}$ that will simulate the original one.

Let $\text{Leaves}(Q) = p_l^{l\in L}$, $\text{Leaves}(Q') = {p'}_l^{l\in L'}$, $\text{Leaves}(P) = p_l^{l\in L_P}$. Consider Q FH-bisimilar to Q'. It means that there is a relation $\mathcal{R}$ that is an FH-bisimulation between the open automata of the two pNets. We will consider the relation $\mathcal{R}' = \{(s,t|Pred_{s,t})|s = s' \uplus s'' \wedge t = t' \uplus s'' \wedge s'' \in \mathcal{S}_P \wedge (s',t'|Pred_{s,t}) \in \mathcal{R}\}$ where $\mathcal{S}_P$ is the set of states of the open automaton of P. We will prove that $\mathcal{R}'$ is an open FH-bisimulation. Consider a pair of FH-bisimilar states: $(\triangleleft s_i^{i\in L_P \uplus L} \triangleright, \triangleleft t_i^{i\in L'} \uplus s_i^{i\in L_P} \triangleright | Pred_{s,t}) \in \mathcal{R}'$. Consider an open transition OT of $P[Q]_{j_0}$.

$$P[Q]_{j_0} \models \frac{\beta_j^{j\in J}, Pred_{OT}, Post_{OT}}{\triangleleft s_i^{i\in L_P\uplus L} \triangleright \xrightarrow{\alpha} \triangleleft {s'}_i^{i\in L_P\uplus L} \triangleright}$$

Let $J' = J \setminus \text{Holes}(Q) \cup \{j_0\}$. By Lemma 1 we have:

$$P \models \frac{\beta_j^{j\in J'}, Pred', Post'}{\triangleleft s_i^{i\in L_P} \triangleright \xrightarrow{\alpha} \triangleleft {s'}_i^{i\in L_P} \triangleright}$$

$$Q \models \frac{\beta_j^{j\in J\cap \text{Holes}(Q)}, Pred'', Post''}{\triangleleft s_i^{i\in L} \triangleright \xrightarrow{\alpha_Q} \triangleleft {s'}_i^{i\in L} \triangleright}$$

and $Pred_{OT} \iff Pred' \wedge Pred'' \wedge \alpha_Q = \beta_{j_0}$, $Post_{OT} = Post' \uplus Post''$ ($Post''$ is the restriction of $Post$ over $vars(Q)$). As Q is FH-bisimilar to Q' and $(\triangleleft s_i^{i\in L} \triangleright, \triangleleft t_i^{i\in L'} \triangleright | Pred_{s,t}) \in \mathcal{R}$ there is a family OT'_x of open transitions of the automaton of Q' such that

$$Q' \models \frac{\beta_{jx}^{j\in J\cap \text{Holes}(Q)}, Pred_{OT_x}, Post_{OT_x}}{\triangleleft t_i^{i\in L'} \triangleright \xrightarrow{\alpha_x} \triangleleft t_{ix}^{i\in L'} \triangleright}$$

and $\forall x, (\triangleleft s_i^{i\in L} \triangleright, \triangleleft t_{ix}^{i\in L'} \triangleright | Pred_{sx}) \in \mathcal{R}$; and
$Pred_{s,t} \wedge Pred'' \implies$
$$\bigvee_{x\in X} \left(\forall j \in J \cap \text{Holes}(Q). \beta_j = \beta_{jx} \wedge Pred_{OT_x} \wedge \alpha_Q = \alpha_x \wedge Pred_{s,x}\{\!\{Post'' \uplus Post_{OT_x}\}\!\}\right)$$

We can now apply Lemma 2 on each of the OT'_x together with the transition of P and obtain a new family OT_x of open transitions (where for $i \in L_P$, $t_i = s_i$ and $t_{ix} = s'_i$, and for $j \in Holes(P)$, $\beta_{jx} = \beta_j$):

$$P[Q']_{j_0} \models \frac{\beta_{jx}^{j\in J}, Pred' \wedge Pred_{OT_x} \wedge \alpha_x = \beta_{j_0 x}, Post' \uplus Post_{OT_x}}{\triangleleft t_i^{i\in L'\uplus L_Q} \triangleright \xrightarrow{\alpha_x} \triangleleft t_{ix}^{i\in L'\uplus L_Q} \triangleright}$$

Observe that we used the fact that $J = (J \setminus \text{Holes}(Q) \cup \{j_0\}) \setminus \{j_0\} \cup (J \cap \text{Holes}(Q))$. Now we have to verify the conditions for the FH-bisimulation

between OT and OT_x. $\forall x, (\lhd {s'}_i^{\,i\in L_P \uplus L} \rhd, \lhd t_{ix}^{i\in L_P \uplus L'} \rhd | Pred_{s,x}) \in \mathcal{R}'$ (by definition of $\mathcal{R}'$) and in three steps we get:

$Pred_{s,t} \wedge Pred_{OT} \implies Pred_{s,t} \wedge Pred' \wedge Pred'' \wedge \alpha_Q = \beta_{j_0}$

$\Longrightarrow \bigvee_{x\in X} \Big(\forall j \in J \cap \text{Holes}(Q).\beta_j = \beta_{jx} \wedge Pred_{OT_x} \wedge \alpha_Q = \alpha_x \wedge Pred_{s,x}\{\!\{Post'' \uplus Post_{OT_x}\}\!\}\Big) \wedge$

$Pred' \wedge \alpha_Q = \beta_{j_0}$

$\Longrightarrow \bigvee_{x\in X} \Big(\forall j \in J \cap \text{Holes}(Q).\beta_j = \beta_{jx} \wedge Pred' \wedge Pred_{OT_x} \wedge \alpha_Q = \alpha_x \wedge \alpha_Q = \beta_{j_0 x} \wedge$

$Pred_{s,x}\{\!\{Post'' \uplus Post_{OT_x}\}\!\}\Big)$

Note that, β_{j_0} can be transformed into $\beta_{j_0 x}$ because of the implication hypothesis. The obtained formula reaches the goal except for two points:

- We need $\forall j \in J$ instead of $\forall j \in J \cap \text{Holes}(Q)$ but adding prerequisite on more variables does not change the validity of the formula (those variables are not used).
- Concerning the last term, we need $Pred_{sx}\{\!\{Post_{OT} \uplus (Post' \uplus Post_{OT_x})\}\!\}$, i.e. $Pred_{s,x}\{\!\{(Post' \uplus Post'') \uplus (Post' \uplus Post_{OT_x})\}\!\}$. We can conclude by observing that $Pred_{s,x}$ does not use any variable of P and thus the substitution $\{\!\{Post'\}\!\}$ has no effect on it.

Finally:

$Pred_{s,t} \wedge Pred_{OT} \implies$

$\bigvee_{x\in X} \Big(\forall j \in J.\beta_j = \beta_{jx} \wedge \big(Pred' \wedge Pred_{OT_x} \wedge \alpha_Q = \beta_{j_0 x}\big) \wedge \alpha_Q = \alpha_x \wedge Pred_{s,x}\{\!\{Post'' \uplus Post_{OT_x}\}\!\}\Big)$

This proves the condition of the FH-simulation, the other direction is similar. □

Appendix A.4. Proof of Theorem 4: Context equivalence

Consider two FH-bisimilar open pNets: $P = \langle\!\langle P_i^{i\in I}, Sort_j^{j\in J}, \overline{SV}\rangle\!\rangle$ *and* $P' = \langle\!\langle {P'}_i^{i\in I}, Sort_j^{j\in J}, \overline{SV'}\rangle\!\rangle$ *(recall they must have the same holes to be bisimilar). Let* $j_0 \in J$ *be a hole, and* Q *be a pNet such that* $\text{Sort}(Q) \subseteq Sort_{j_0}$. *Then* $P[Q]_{j_0}$ *and* $P'[Q]_{j_0}$ *are FH-bisimilar.*

Proof. The proof of Theorem 4 exhibits a bisimulation relation for a composed system. It then uses Lemma 1 to decompose the open transition of $P[Q]_{j_0}$ and obtain an open transition of P on which the FH-bisimulation property can be applied to obtain an equivalent family of open transitions of P'; this family is then recomposed by Lemma 2 to build a set of open transitions of $P'[Q]_{j_0}$ that will simulate the original one.

Let $\text{Leaves}(Q) = p_l^{l\in L_Q}$, $\text{Leaves}(P) = p_l^{l\in L}$, $\text{Leaves}(P') = {p'}_l^{l\in L'}$. Consider P FH-bisimilar to P'. It means that there is a relation $\mathcal{R}$ that is an FH-bisimulation between the open automata of the two pNets. We will consider the relation $\mathcal{R}' = \{(s,t|Pred_{s,t}) | s = s' \uplus s'' \wedge t = t' \uplus s'' \wedge s \in \mathcal{S}_Q \wedge (s',t'|Pred_{s,t}) \in \mathcal{R}\}$ where $\mathcal{S}_Q$ is the set of states of the open automaton of Q. We will prove

that $\mathcal{R}'$ is an open FH-bisimulation. Consider a pair of FH-bisimilar states: $(\lhd s_{1i}^{i\in L\uplus L_Q}\rhd, \lhd s_{2i}^{i\in L'} \uplus s_{1i}^{i\in L_Q} \rhd \,|Pred) \in \mathcal{R}'$. Consider an open transition OT of $P[Q]_{j_0}$.

$$P[Q]_{j_0} \models \frac{\beta_j^{j\in J}, Pred_{OT}, Post_{OT}}{\lhd s_i^{i\in L\uplus L_Q} \rhd \xrightarrow{\alpha} \lhd {s'}_i^{i\in L\uplus L_Q} \rhd}$$

Let $J' = J \setminus \text{Holes}(Q) \cup \{j_0\}$. By Lemma 1 we have:

$$P \models \frac{\beta_j^{j\in J'}, Pred', Post'}{\lhd s_1^{i\in L} \rhd \xrightarrow{\alpha} \lhd s'_i{}^{i\in L} \rhd}$$

$$Q \models \frac{\beta_j^{j\in J\cap \text{Holes}(Q)}, Pred'', Post''}{\lhd s_i^{i\in L_Q} \rhd \xrightarrow{\alpha_Q} \lhd {s'}_i^{i\in L_Q} \rhd}$$

and $Pred_{OT} \iff Pred' \wedge Pred'' \wedge \alpha_Q = \beta_{j_0}$, $Post_{OT} = Post' \uplus Post''$ ($Post''$ is the restriction of $Post$ over $vars(Q)$). As P is FH-bisimilar to P' and $(\lhd s_i^{i\in L}\rhd, \lhd t_i^{i\in L'} \rhd \,|Pred_{s,t}) \in \mathcal{R}$ there is a family OT'_x of open transitions of the automaton of P' such that

$$P' \models \frac{\beta_{jx}^{j\in J'}, Pred_{OT_x}, Post_{OT_x}}{\lhd t_i^{i\in L'} \rhd \xrightarrow{\alpha_x} \lhd t_{ix}^{i\in L'} \rhd}$$

and $\forall x, (\lhd s_i^{i\in L}\rhd, \lhd t_{ix}^{i\in L'} \rhd \,|Pred_{sx}) \in \mathcal{R}$; and
$Pred_{s,t} \wedge Pred' \Longrightarrow \bigvee_{x\in X} \Big(\forall j \in J'.\beta_j = \beta_{jx} \wedge Pred_{OT_x} \wedge \alpha = \alpha_x \wedge Pred_{s,x}\{\!\{Post' \uplus Post_{OT_x}\}\!\}\Big)$

We can now apply Lemma 2 on each of the OT'_x together with the transition of Q and obtain a new family OT_x of open transitions (where for $i \in L_Q$, $t_i = s_i$ and $t_{ix} = s'_i$, and for $j \in Holes(Q)$, $b_{jx} = b_j$):

$$P'[Q]_{j_0} \models \frac{\beta_{jx}^{j\in J}, Pred_{OT_x} \wedge Pred'' \wedge \alpha_Q = \beta_{j_0 x}, Post_{OT_x} \uplus Post''}{\lhd t_i^{i\in L'\uplus L_Q} \rhd \xrightarrow{\alpha_x} \lhd t_{ix}^{i\in L'\uplus L_Q} \rhd}$$

Observe that $J = (J \setminus \text{Holes}(Q) \cup \{j_0\}) \setminus \{j_0\} \cup (J \cap \text{Holes}(Q))$. Now we have to verify the conditions for the FH-bisimulation between OT and OT_x. $\forall x, (\lhd {s'}_i^{i\in L\uplus L_Q}\rhd, \lhd t_{ix}^{i\in L'\uplus L_Q} \rhd \,|Pred_{s,x}) \in \mathcal{R}'$ (by definition of $\mathcal{R}'$) and in four steps we get:

$$\begin{aligned} Pred_{s,t} \wedge Pred_{OT} &\Longrightarrow Pred_{s,t} \wedge Pred' \wedge Pred'' \wedge \alpha_Q = \beta_{j_0} \\ &\Longrightarrow \bigvee_{x\in X} \Big(\forall j \in J'.\beta_j = \beta_{jx} \wedge Pred_{OT_x} \wedge \alpha_Q = \beta_{j_0} \wedge \alpha = \alpha_x \wedge Pred_{s,x}\{\!\{Post' \uplus \\ &Post_{OT_x}\}\!\}) \wedge Pred''\Big) \\ &\Longrightarrow \bigvee_{x\in X} (\forall j \in J'.\beta_j = \beta_{jx} \wedge \big(Pred_{OT_x} \wedge Pred'' \wedge \alpha_Q = \beta_{j_0 x}\big) \wedge \alpha = \alpha_x \wedge \\ &Pred_{s,x}\{\!\{Post' \uplus Post_{OT_x}\}\!\}) \end{aligned}$$

The obtained formula reaches the goal except for two points:

- We need $\forall j \in J$ instead of $\forall j \in J'$ with $J' = J \setminus \text{Holes}(Q) \cup \{j_0\}$ but the formula under the quantifier does not depend on b_{j_0} now (thanks to the

substitution). Concerning Holes(Q), adding prerequisite on more variables does not change the validity of the formula (those variables are not used).

- We need $Pred_{s,x}\{\!\{Post_{OT} \uplus (Post_{OT_x} \uplus Post'')\}\!\}$, i.e., $Pred_{s,x}\{\!\{(Post' \uplus Post'') \uplus (Post_{OT_x} \uplus Post'')\}\!\}$. We can conclude by observing that $Pred_{s,x}$ does not use any variable of Q and thus the substitution involving $Post''$ has no effect.

This proves the condition of the FH-simulation, the other direction is similar. □

Appendix B. Weak FH-bisimilarity lemmas and proofs

We define a quantified composition operator for effects, i.e. *Post* elements of the open transitions. We use $\bigodot_{i=n}^{1} Post_i$ to denote $Post_n \odot \ldots \odot Post_0$. By convention $\bigodot_{i=0}^{1} Post_i$ is the identity.

Appendix B.1. Weak FH-bisimilarity is an equivalence

In this section, we first define two alternative definitions, one for weak open transition, one for weak bisimulation. We use these two alternative definitions to show that weak bisimulation is an equivalence, we will also re-use these alternative definitions in the proofs of the theorems in next sections.

Lemma 4 (Alternative definition of weak open transitions). *Let $A = \langle\!\langle J, \mathcal{S}, s_0, V_1, \mathcal{T} \rangle\!\rangle$ be an open automaton and $\langle\!\langle J, \mathcal{S}, s_0, V_2, \mathcal{WT} \rangle\!\rangle$ be the weak open automaton derived from A. The two following statements are equivalent*

1. *Either $\alpha = \tau \wedge \overline{\gamma} = \emptyset \wedge Pred = True \wedge Post = \mathrm{Id}(s) \wedge s = s'$; or there exist $\overline{\beta_{1i}}$, $\overline{\beta_{2i}}$, and $\overline{\beta_{3i}}$, $Pred_{1i}$, $Pred_{3i}$, $Post_{1i}$, and $Pred_2$, $Post_2$, $n \geq 0$, $m \geq 0$ s.t.*[15]*:*

$$\forall i \in [1..n]. \overset{\overline{\beta_{1i}}, Pred_{1i}, Post_{1i}}{\cdots\cdots\cdots\cdots\cdots\cdots} \in \mathcal{T} \quad \wedge \quad \overset{\overline{\beta_2}, Pred_2, Post_2}{\cdots\cdots\cdots\cdots\cdots\cdots} \in \mathcal{T} \quad \wedge$$
$$s_{1i} \xrightarrow{\tau} s_{1(i+1)} \qquad\qquad s_2 \xrightarrow{\alpha} s'_2$$

$$\forall i \in [1..m]. \overset{\overline{\beta_{3i}}, Pred_{3i}, Post_{3i}}{\cdots\cdots\cdots\cdots\cdots\cdots} \in \mathcal{T}$$
$$s_{3i} \xrightarrow{\tau} s_{3(i+1)}$$

[15] $n = 0$ (resp. $m = 0$) corresponds to the case where there is no τ transition before (resp. after) the transition α.

2. *there exist* $\overline{\gamma}$, *Pred*, *Post s.t.*

$$\frac{\overline{\gamma}, Pred, Post}{s \overset{\alpha'}{\Rightarrow} s'} \in \mathcal{WT}$$

where

$$\alpha' = \alpha\{\!\{\bigodot_{j=n}^{1} Post_{1j}\}\!\}$$

$$s = s_{11} \wedge s_{1(n+1)} = s_2 \wedge s'_2 = s_{31} \wedge s_{3(m+1)} = s'$$

$$\overline{\gamma} = \bigoplus_{i=1}^{n} (\overline{\beta_{1i}}\{\!\{\bigodot_{j=i-1}^{1} Post_{1j}\}\!\})^{\nabla} \oplus (\overline{\beta_2}\{\!\{\bigodot_{j=n}^{1} Post_{1j}\}\!\})^{\nabla} \oplus \bigoplus_{i=0}^{m} (\overline{\beta_{3i}}\{\!\{\bigodot_{j=i-1}^{1} Post_{3j} \odot Post_2 \odot \bigodot_{j=n}^{1} Post_{1j}\}\!\})^{\nabla}$$

$$Pred = \bigwedge_{i=1}^{n} Pred_{1i}\{\!\{\bigodot_{j=i-1}^{1} Post_{1j}\}\!\} \wedge Pred_2\{\!\{\bigodot_{j=n}^{1} Post_{1j}\}\!\} \wedge \left(\bigwedge_{i=1}^{m} Pred_{3i}\{\!\{\bigodot_{j=i-1}^{1} Post_{3j} \odot Post_2 \odot \bigodot_{j=n}^{1} Post_{1j}\}\!\}\right)$$

$$Post = \bigodot_{j=m}^{1} Post_{3j} \odot Post_2 \odot \bigodot_{j=n}^{1} Post_{1j}$$

Proof. ($\Rightarrow$) We present an induction on n and m, focusing on the incrementation on n: we prove that the property is valid for $m = 0$, $n = 0$ apply a first induction proof for going from n to $n + 1$, a similar induction can be applied to go from m to $m + 1$ (omitted).

- The base case there is one transition, so $n = 0$ and $m = 0$, we have:

$$\frac{\overline{\beta}, Pred, Post}{s \overset{\alpha}{\rightarrow} s'} \in \mathcal{T}$$

by rule **WT2** we can directly conclude the implication:

$$\frac{\overline{\beta}, Pred, Post}{s \overset{\alpha}{\rightarrow} s'} \in \mathcal{T} \Rightarrow \frac{(\overline{\beta})^{\nabla}, Pred, Post}{s \overset{\alpha}{\Rightarrow} s'} \in \mathcal{WT}$$

- For the inductive step, first we have by induction hypothesis that the formula holds for some lengths m and n. Induction step is to infer that formula holds for transitions of length $n + 1$. We consider the case $n' = n + 1$. We want to prove (1) $\Rightarrow$ (2) in the lemma, and in (1) we focus on the case where there is a set of open transitions (this is the case: $s \neq s'$).

In other words, we consider the sequence of $(n+m+2)$ open transitions:

$$\Big(\forall i \in [1..n+1].\frac{\overline{\beta_{1i}}, Pred_{1i}, Post_{1i}}{s_{1i} \xrightarrow{\tau} s_{1(i+1)}} \in \mathcal{T} \quad \wedge \quad \frac{\overline{\beta_2}, Pred_2, Post_2}{s_2 \xrightarrow{\alpha} s_2'} \in \mathcal{T} \quad \wedge$$

$$\forall i \in [1..m].\frac{\overline{\beta_{3i}}, Pred_{3i}, Post_{3i}}{s_{3i} \xrightarrow{\tau} s_{3(i+1)}} \in \mathcal{T}\Big)$$

By recurrence hypothesis we suppose that (1) $\Rightarrow$ (2) holds for n and m (compared to the line above, we remove the first τ transition). We have:

$$\Big(\forall i \in [2..n+1].\frac{\overline{\beta_{1i}}, Pred_{1i}, Post_{1i}}{s_{1i} \xrightarrow{\tau} s_{1(i+1)}} \in \mathcal{T} \wedge \frac{\overline{\beta_2}, Pred_2, Post_2}{s_2 \xrightarrow{\alpha} s_2'} \in \mathcal{T} \wedge$$

$$\forall i \in [1..m].\frac{\overline{\beta_{3i}}, Pred_{3i}, Post_{3i}}{s_{3i} \xrightarrow{\tau} s_{3(i+1)}} \in \mathcal{T}\Big) \Rightarrow \frac{\overline{\gamma}, Pred, Post}{s'' \stackrel{\alpha'}{\Longrightarrow} s'} \in \mathcal{WT}$$

where

$$s'' = s_{12} \wedge s_{1(n+2)} = s_2 \wedge s_2' = s_{31} \wedge s_{3(m+1)} = s'$$

$$\alpha' = \alpha\{\!\{ \overset{2}{\underset{j=n+1}{\odot}} Post_{1j} \}\!\}$$

$$\overline{\gamma} = \bigoplus_{i=2}^{n+1} (\overline{\beta_{1i}}\{\!\{ \overset{2}{\underset{j=i-1}{\odot}} Post_{1j} \}\!\})^{\nabla} \oplus (\overline{\beta_2}\{\!\{ \overset{2}{\underset{j=n+1}{\odot}} Post_{1j} \}\!\})^{\nabla} \oplus$$
$$\bigoplus_{i=1}^{m} (\overline{\beta_{3i}}\{\!\{ \overset{1}{\underset{j=i-1}{\odot}} Post_{3j} \odot Post_2 \odot \overset{1}{\underset{j=n}{\odot}} Post_{1j} \}\!\})^{\nabla}$$

$$Pred = \bigwedge_{i=2}^{n+1} Pred_{1i}\{\!\{ \overset{2}{\underset{j=i-1}{\odot}} Post_{1j} \}\!\} \wedge Pred_2\{\!\{ \overset{2}{\underset{j=n+1}{\odot}} Post_{1j} \}\!\} \wedge$$
$$\Big(\bigwedge_{i=1}^{m} Pred_{3i}\{\!\{ \overset{1}{\underset{j=i-1}{\odot}} Post_{3j} \odot Post_2 \odot \overset{1}{\underset{j=n+1}{\odot}} Post_{1j} \}\!\} \Big)$$

$$Post = \overset{1}{\underset{j=m}{\odot}} Post_{3j} \odot Post_2 \odot \overset{2}{\underset{j=n+1}{\odot}} Post_{1j}$$

We need to prove that by adding the following open transition the implication remains true:

$$\frac{\overline{\beta_{11}}, Pred_{11}, Post_{11}}{s_{11} \xrightarrow{\tau} s_{12}} \in \mathcal{T}$$

First by using rule **WT2** we have:

$$\frac{\overline{\beta_{11}}, Pred_{11}, Post_{11}}{s_{11} \xrightarrow{\tau} s_{12}} \in \mathcal{T} \Rightarrow \frac{(\overline{\beta_{11}})^{\nabla}, Pred_{11}, Post_{11}}{s_{11} \stackrel{\tau}{\Rightarrow} s_{12}} \in \mathcal{WT}$$

On the other hand, by rule **WT1** we have the following weak open transition:

$$\frac{\emptyset, True, \mathrm{Id}(s')}{s' \stackrel{\tau}{\Rightarrow} s'} \in \mathcal{WT}$$

Finally by applying rule **WT3** on the above weak open transitions:

$$\frac{\begin{array}{c} \dfrac{(\overline{\beta_{11}})^{\nabla}, Pred_{11}, Post_1}{s_{11} \stackrel{\tau}{\Rightarrow} s_{12}} \in \mathcal{WT} \\ \dfrac{\overline{\gamma}, Pred, Post}{s'' \stackrel{\alpha'}{\Rightarrow} s'} \in \mathcal{WT} \qquad \dfrac{\emptyset, True, \mathrm{Id}(s'')}{s' \stackrel{\tau}{\Rightarrow} s'} \in \mathcal{WT} \\ \overline{\gamma''} = (\overline{\beta_{11}})^{\nabla} \oplus \overline{\gamma}\{\!\{Post_1\}\!\} \qquad Pred'' = Pred_{11} \wedge Pred\{\!\{Post_{11}\}\!\} \\ Post'' = \mathrm{Id}(s'') \odot Post \odot Post_{11} \qquad \alpha'' = \alpha'\{\!\{Post_{11}\}\!\} \end{array}}{\dfrac{\overline{\gamma''}, Pred'', Post''}{s_{11} \stackrel{\alpha''}{\Longrightarrow} s'} \in \mathcal{WT}} \mathbf{WT3}$$

where we obtain the conclusion of the lemma, as required with the following assertions (derived from previous assertions):

$$s_{11} = s_{11} \wedge s_{1(n+2)} = s_2 \wedge s_2' = s_{31} \wedge s_{3(m+1)} = s'$$

$$\alpha'' = \alpha\{\!\{ \bigodot_{j=n+1}^{2} Post_{1j} \}\!\}\{\!\{Post_{11}\}\!\} = \alpha\{\!\{ \bigodot_{j=n+1}^{1} Post_{1j} \}\!\}$$

$$\overline{\gamma''} = \bigoplus_{i=1}^{n+1} (\overline{\beta_{1i}}\{\!\{ \bigodot_{j=i-1}^{1} Post_{1j} \}\!\})^{\nabla} \oplus (\overline{\beta_2}\{\!\{ \bigodot_{j=n+1}^{1} Post_{1j} \}\!\})^{\nabla} \oplus \bigoplus_{i=1}^{m} (\overline{\beta_{3i}}\{\!\{ \bigodot_{j=i-1}^{1} Post_{3j} \odot Post_2 \odot \bigodot_{j=n}^{1} Post_{1j} \}\!\})^{\nabla}$$

$$Pred = \bigwedge_{i=1}^{n+1} Pred_{1i}\{\!\{ \bigodot_{j=i-1}^{1} Post_{1j} \}\!\} \wedge Pred_2\{\!\{ \bigodot_{j=n+1}^{1} Post_{1j} \}\!\} \wedge \left(\bigwedge_{i=1}^{m} Pred_{3i}\{\!\{ \bigodot_{j=i-1}^{1} Post_{3j} \odot Post_2 \odot \bigodot_{j=n+1}^{1} Post_{1j} \}\!\} \right)$$

$$Post = \bigodot_{j=m}^{1} Post_{3j} \odot Post_2 \odot \bigodot_{j=n+1}^{1} Post_{1j}$$

The right part of the disjunction, i.e.

$$\Big(\alpha = \tau \wedge \overline{\gamma} = \emptyset \wedge \mathit{Pred} = \mathit{True} \wedge \mathit{Post} = \mathrm{Id}(s) \wedge s = s'\Big)$$

is handled trivially by rule **WT1**.

($\Leftarrow$) We proceed by structural induction on the rules building the weak transition (as described in the original definition). The recurrence hypothesis being that the original definition implies the characterization (1), with the conditions stated at the bottom of the theorem. We consider the different rules:

- Case rule **WT1**. We have:

$$\frac{\overset{\cdots\cdots\cdots}{\emptyset,\ \mathit{True},\ \mathrm{Id}(s)}}{s \overset{\tau}{\Rightarrow} s} \in \mathcal{WT}$$

We can directly conclude by the right part of the disjunction the following:

$$\frac{\overset{\cdots\cdots\cdots}{\emptyset,\ \mathit{True},\ \mathrm{Id}(s)}}{s \overset{\tau}{\Rightarrow} s} \in \mathcal{WT} \Rightarrow \Big(\alpha = \tau \wedge \overline{\gamma} = \emptyset \wedge \mathit{Pred} = \mathit{True} \wedge \mathit{Post} = \mathrm{Id}(s) \wedge s = s'\Big)$$

- Case rule **WT2**. We have:

$$\frac{\overset{\cdots\cdots\cdots}{\overline{\gamma},\ \mathit{Pred},\ \mathit{Post}}}{s \overset{\alpha}{\Rightarrow} s'} \in \mathcal{WT} \Rightarrow \frac{\overset{\cdots\cdots\cdots}{\overline{\beta},\ \mathit{Pred},\ \mathit{Post}}}{s \overset{\alpha}{\rightarrow} s'} \in \mathcal{T}$$

where $\overline{\gamma} = (\overline{\beta})^{\nabla}$.

These two cases above prove the implication with $n = 0$ and $m = 0$.

- Case rule **WT3**. We have:

$$\frac{\begin{array}{c}\overset{\cdots\cdots\cdots}{\overline{\gamma_1},\ \mathit{Pred}_1,\ \mathit{Post}_1} \\ s \overset{\tau}{\Rightarrow} s' \end{array} \in \mathcal{WT} \qquad \frac{\overset{\cdots\cdots\cdots}{\overline{\gamma_2},\ \mathit{Pred}_2,\ \mathit{Post}_2}}{s' \overset{\alpha}{\Rightarrow} s''} \in \mathcal{WT} \qquad \frac{\overset{\cdots\cdots\cdots}{\overline{\gamma_3},\ \mathit{Pred}_3,\ \mathit{Post}_3}}{s'' \overset{\tau}{\Rightarrow} s'''} \in \mathcal{WT}}{\begin{array}{c}\mathit{Pred} = \mathit{Pred}_1 \wedge \mathit{Pred}_2\{\!\{\mathit{Post}_1\}\!\} \wedge \mathit{Pred}_3\{\!\{\mathit{Post}_2 \odot \mathit{Post}_1\}\!\} \\ \overline{\gamma} = \overline{\gamma_1} \oplus \overline{\gamma_2}\{\!\{\mathit{Post}_1\}\!\} \oplus \overline{\gamma_3}\{\!\{\mathit{Post}_2 \odot \mathit{Post}_1\}\!\} \qquad \alpha' = \alpha\{\!\{\mathit{Post}_1\}\!\} \\ \hline \overset{\cdots\cdots\cdots\cdots\cdots}{\overline{\gamma},\ \mathit{Pred},\ \mathit{Post}_3 \odot \mathit{Post}_2 \odot \mathit{Post}_1} \\ s \overset{\alpha'}{\Rightarrow} s''' \end{array} \in \mathcal{WT}}$$

1. By induction hypothesis this means each tau weak open transition can be written as a series of n_1 tau open transitions such $n_1 = n+m+1$, hence by simplification we have (strictly speaking, by induction we

might also have the case $\alpha = \tau \wedge \overline{\gamma} = \emptyset \wedge \ldots$ but in this case, rule **WT1** allows us to obtain a similar reduction with $n_1 = 1$):

$$\frac{\overline{\gamma_1}, Pred_1, Post_1}{s \overset{\tau}{\Rightarrow} s'} \in \mathcal{WT} \Rightarrow \forall i \in [1..n_1]. \frac{\overline{\beta_{1i}}, Pred_{1i}, Post_{1i}}{s_{1i} \xrightarrow{\tau} s_{1(i+1)}} \in \mathcal{T}$$

where

$$s = s_{11} \wedge s_{1(n_1+1)} = s', \quad \overline{\gamma_1} = \bigoplus_{i=1}^{n_1} (\overline{\beta_{1i}} \{\!\{ \bigodot_{j=i-1}^{1} Post_{1j} \}\!\})^{\nabla}$$

$$Pred_1 = \bigwedge_{i=1}^{n_1} (Pred_{1i} \{\!\{ \bigodot_{j=i-1}^{1} Post_{ij} \}\!\}), \quad Post_1 = \bigodot_{i=n_1}^{1} \{\!\{Post_{1i}\}\!\}$$

2. Similarly, a series of m_1 open transitions such that $m_1 = n + m + 1$ can be simplified as follows:

$$\frac{\overline{\gamma_3}, Pred_3, Post_3}{s'' \overset{\tau}{\Rightarrow} s'''} \in \mathcal{WT} \Rightarrow \forall i \in [1..m_1]. \frac{\overline{\beta_{3i}}, Pred_{3i}, Post_{3i}}{s_{3i} \xrightarrow{\tau} s_{3(i+1)}} \in \mathcal{T}$$

where

$$s'' = s_{31} \quad \wedge \quad s_{3(m_1+1)} = s''' \quad \wedge$$

$$\overline{\gamma_3} = \bigoplus_{i=1}^{m_1} (\overline{\beta_{3i}} \{\!\{ \bigodot_{j=i-1}^{1} Post_{3j} \}\!\})^{\nabla} \quad \wedge$$

$$Pred_3 = \bigwedge_{i=1}^{m_1} (Pred_{3i} \{\!\{ \bigodot_{j=i-1}^{1} Post_{3j} \}\!\}) \quad \wedge \quad Post_3 = \bigodot_{i=m_1}^{1} \{\!\{Post_{3i}\}\!\}$$

3. Concerning the middle reduction, by induction hypothesis there exists a set of open transitions in $\mathcal{T}$ such that:

$$\frac{\overline{\gamma_2}, Pred_2, Post_2}{s' \overset{\alpha'}{\Rightarrow} s''} \in \mathcal{WT} \Rightarrow \left(\forall i \in [1..n_2]. \frac{\overline{\beta_{2i}}, Pred_{2i}, Post_{2i}}{s_{2i} \xrightarrow{\tau} s_{2(i+1)}} \in \mathcal{T} \wedge \right.$$

$$\left. \frac{\overline{\beta'}, Pred', Post'}{s_2 \xrightarrow{\alpha''} s'_2} \in \mathcal{T} \wedge \forall i \in [1..m_2]. \frac{\overline{\beta'_{2i}}, Pred'_{2i}, Post'_{2i}}{s'_{2i} \xrightarrow{\tau} s'_{2(i+1)}} \in \mathcal{T} \right)$$

where

$$
\begin{aligned}
s' = s_{21} \wedge s_{2(n_2+1)} = s_2 \wedge s'_2 = s'_{21} \wedge s'_{2(m_2+1)} = s'' \\
\alpha'' =& \alpha' \{\!\{ \bigodot_{j=n_2}^{1} Post_{2j} \}\!\} \\
\overline{\gamma_2} =& \bigoplus_{i=1}^{n_2} (\overline{\beta_{2i}} \{\!\{ \bigodot_{j=i-1}^{1} Post_{2j} \}\!\})^{\nabla} \oplus (\overline{\beta'} \{\!\{ \bigodot_{j=n_2}^{1} Post_{2j} \}\!\})^{\nabla} \oplus \\
& \bigoplus_{i=1}^{m_2} (\overline{\beta'_{2i}} \{\!\{ \bigodot_{j=i-1}^{1} Post'_{2j} \odot Post' \odot \bigodot_{j=n_2}^{1} Post_{2j} \}\!\})^{\nabla} \\
Pred_2 =& \bigwedge_{i=1}^{n_2} Pred_{2i} \{\!\{ \bigodot_{j=i-1}^{1} Post_{2j} \}\!\} \wedge Pred' \{\!\{ \bigodot_{j=n_2}^{1} Post_{2j} \}\!\} \wedge \\
& \Big(\bigwedge_{i=1}^{m_2} Pred'_{2i} \{\!\{ \bigodot_{j=i-1}^{1} Post'_{2j} \odot Post' \odot \bigodot_{j=n_2}^{1} Post_{2j} \}\!\} \Big) \\
Post_2 =& \bigodot_{j=m_2}^{1} Post'_{2j} \odot Post' \odot \bigodot_{j=n_2}^{1} Post_{2j}
\end{aligned}
$$

Therefore, we can deduce that we have:

$$
\frac{\overline{\gamma}, Pred, Post}{s \overset{\alpha}{\Rightarrow} s'} \in \mathcal{WT} \Rightarrow \Bigg(\forall i \in [1..(n_1+n_2)]. \frac{\overline{\beta_{4i}}, Pred_{4i}, Post_{4i}}{s_{4i} \xrightarrow{\tau} s_{4(i+1)}} \in \mathcal{T} \wedge
$$

$$
\frac{\overline{\beta'}, Pred', Post'}{s_2 \xrightarrow{\alpha''} s'_2} \in \mathcal{T} \wedge \forall i \in [1..(m_1+m_2)]. \frac{\overline{\beta_{5i}}, Pred_{5i}, Post_{5i}}{s_{5i} \xrightarrow{\tau} s_{5(i+1)}} \in \mathcal{T} \Bigg)
$$

such that

$$
s_{4i} = \begin{cases} s_{1i} & \text{if } i \leq n_1 \\ s_{2i-n_1} & \text{if } i > n_1 \end{cases} \qquad s_{5i} = \begin{cases} s_{3i} & \text{if } i \leq m_1 \\ s'_{2i-m_1} & \text{if } i > m_1 \end{cases}
$$

and similarly for $Pred_{4i}$, $Pred_{5i}$, $Post_{4i}$, and $Post_{5i}$.

Also, we have the following assertions:

$$s = s_{41} \ \wedge \ s_{4(n_1+n_2+1)} = s_2 \ \wedge \ s'_2 = s_{51} \ \wedge \ s_{5(m_1+m_2+1)} = s'$$

$$\alpha'' = \alpha \{\!\{ \bigodot_{j=n_1+n_2}^{1} Post_{4j} \}\!\}$$

$$\overline{\gamma} = \bigoplus_{i=1}^{n_1+n_2} (\overline{\beta_{4i}} \{\!\{ \bigodot_{j=i-1}^{1} Post_{4j} \}\!\})^{\nabla} \oplus (\overline{\beta'} \{\!\{ \bigodot_{j=n_1+n_2}^{1} Post_{4j} \}\!\})^{\nabla} \oplus \bigoplus_{i=1}^{m_1+m_2} (\overline{\beta_{5i}} \{\!\{ \bigodot_{j=i-1}^{1} Post'_{5i} \odot Post' \odot \bigodot_{j=n_1+n_2}^{1} Post_{4i} \}\!\})^{\nabla}$$

$$Pred = \bigwedge_{i=1}^{n_1+n_2} Pred_{4i} \{\!\{ \bigodot_{j=i-1}^{1} Post_{4j} \}\!\} \wedge Pred' \{\!\{ \bigodot_{j=n_1+n_2}^{1} Post_{4j} \}\!\} \wedge \bigwedge_{i=1}^{m_1+m_2} Pred_{5i} \{\!\{ \bigodot_{j=i-1}^{1} Post_{5j} \odot Post' \odot \bigodot_{j=n_1+n_2}^{1} Post_{4j} \}\!\}$$

$$Post = \bigodot_{j=m_1+m_2}^{1} Post_{5j} \odot Post' \odot \bigodot_{j=n_1+n_2}^{1} Post_{4j}$$

This concludes the inductive step, showing that the decomposition expressed by the $\Leftarrow$ direction of the lemma is always possible with the right side conditions. □

Lemma 5 (Alternative definition of weak bisimulation). *The definition of weak bisimulation given in Definition 14 is equivalent to the following one:*

Let $A_1 = \langle\!\langle J, \mathcal{S}_1, s_0, V_1, \mathcal{T}_1 \rangle\!\rangle$ *and* $A_2 = \langle\!\langle J, \mathcal{S}_2, t_0, V_2, \mathcal{T}_2 \rangle\!\rangle$ *be open automata;* $\langle\!\langle J, \mathcal{S}_1, s_0, V_1, \mathcal{WT}_1 \rangle\!\rangle$ *and* $\langle\!\langle J, \mathcal{S}_2, t_0, V_2, \mathcal{WT}_2 \rangle\!\rangle$ *be the weak open automaton derived from* A_1 *and* A_2 *respectively. For any states* $s \in \mathcal{S}_1$ *and* $t \in \mathcal{S}_2$ *such that* $(s, t | Pred_{s,t}) \in \mathcal{R}$*, we have:*

- *For any open transition* WOT *in* $\mathcal{WT}_1$*:*

$$\frac{\gamma_j^{j \in J'}, Pred_{OT}, Post_{OT}}{s \overset{\alpha}{\Rightarrow} s'}$$

there exists an indexed set of weak open transitions $WOT_x^{x \in X} \subseteq \mathcal{WT}_2$*:*

$$\frac{\gamma_{jx}^{j \in J_x}, Pred_{OT_x}, Post_{OT_x}}{t \overset{\alpha_x}{\Rightarrow} t_x}$$

such that $\forall x, J' = J_x, (s', t_x | Pred_{s',x}) \in \mathcal{R}$*; and*

$$Pred_{s,t} \wedge Pred_{OT} \Longrightarrow \bigvee_{x \in X} (\forall j \in J_x . \gamma_j = \gamma_{jx} \wedge Pred_{OT_x} \wedge \alpha = \alpha_x \wedge Pred_{s',x} \{\!\{ Post_{OT} \uplus Post_{OT_x} \}\!\})$$

- *and symmetrically any open transition from WOT in $\mathcal{WT}_2$ can be covered by a set of weak transitions from s in $\mathcal{WT}_1$.*

PROOF. Note that Definition 14 is a particular case of the definition above, thus we only need to prove one direction of the equivalence between the two definitions, namely:
($\Rightarrow$) We prove that Definition 14 implies the definition above. In other words, suppose that $Pred_{s,t} \in \mathcal{R}$ and suppose that the following statement holds:

$$\frac{\gamma_j^{j\in J'}, Pred_{OT}, Post_{OT}}{s \overset{\alpha}{\Rightarrow} s'} \in \mathcal{WT}_1$$

Moreover, by using Lemma 4 we know that:

$$\frac{\gamma_j^{j\in J'}, Pred_{OT}, Post_{OT}}{s \overset{\alpha}{\Rightarrow} s'} \in \mathcal{WT}_1 \Rightarrow \Bigg(\forall i \in [1..n].\frac{\beta_{1ij}^{j\in J'_1}, Pred_{1i}, Post_{1i}}{s_{1i} \xrightarrow{\tau} s_{1(i+1)}} \in \mathcal{T}_1 \wedge \frac{\beta_{2j}^{j\in J'_2}, Pred_2, Post_2}{s_{20} \xrightarrow{\alpha'} s_{21}} \in \mathcal{T}_1 \wedge \forall i \in [1..m].\frac{\beta_{3ij}^{j\in J'_3}, Pred_{3i}, Post_{3i}}{s_{3i} \xrightarrow{\tau} s_{3(i+1)}} \in \mathcal{T}_1\Bigg)$$

where

$$s = s_{11} \wedge s_{1(n+1)} = s_{20} \wedge s_{21} = s_{31} \wedge s_{3(m+1)} = s'$$

$$\alpha = \alpha' \{\!\{ \bigodot_{j=n}^{1} Post_{1j} \}\!\}$$

$$\gamma_j^{j\in J'} = \bigoplus_{i=1}^{n} (\overline{\beta_{1i}} \{\!\{ \bigodot_{j=i-1}^{1} Post_{1j} \}\!\})^{\nabla} \oplus (\overline{\beta_2} \{\!\{ \bigodot_{j=n}^{1} Post_{1j} \}\!\})^{\nabla} \oplus \bigoplus_{i=1}^{m} (\overline{\beta_{3i}} \{\!\{ \bigodot_{j=i-1}^{1} Post_{3j} \odot Post_2 \odot \bigodot_{j=n}^{1} Post_{1j} \}\!\})^{\nabla}$$

$$Pred_{OT} = \bigwedge_{i=1}^{n} Pred_{1i} \{\!\{ \bigodot_{j=i-1}^{1} Post_{1j} \}\!\} \wedge Pred_2 \{\!\{ \bigodot_{j=n}^{1} Post_{1j} \}\!\} \wedge \Big(\bigwedge_{i=1}^{m} Pred_{3i} \{\!\{ \bigodot_{j=i-1}^{1} Post_{3j} \odot Post_2 \odot \bigodot_{j=n}^{1} Post_{1j} \}\!\} \Big)$$

$$Post_{OT} = \bigodot_{j=m}^{1} Post_{3j} \odot Post_2 \odot \bigodot_{j=n}^{1} Post_{1j}$$

For the sake of simplicity, we prove the rule in the restricted case where n and m are equal to 1, hence a single tau open transition will be considered on each side of the potentially visible one. The proof may be easily generalized to the multiple tau open transitions by using the same reasoning and **WT3** rule. Consider each open transition separately:

1. For the first open transition in $\mathcal{T}_1$:

$$\frac{\beta_{1j}^{j\in J_1'}, Pred_1, Post_1}{s_{11} \xrightarrow{\tau} s_{12}}$$

by hypothesis we have $(s, t|Pred_{s,t}) \in \mathcal{R}$ and $s = s_{11}$. Thus, by Definition 14 we can deduce there exists an indexed set of weak open transitions $WOT_a^{a\in A} \subseteq \mathcal{WT}_2$:

$$\frac{\gamma_{ja}^{j\in J_a}, Pred_{OT_a}, Post_{OT_a}}{t \xRightarrow{\alpha_{1a}} u_a}$$

such that $\forall a, J_a = \{j \in J_1' | \beta_{1j} \neq \tau\}, (s_{12}, u_a|Pred_{s_{11},a}) \in \mathcal{R}$ and
$Pred_{s,t} \wedge Pred_1 \implies$
$\bigvee_{a\in A} \left(\forall j \in J_a.(\beta_{1j})^{\nabla} = \gamma_{ja} \wedge Pred_{OT_a} \wedge \alpha_{1a} = \tau \wedge Pred_{s_{11},a}\{\!\{Post_1 \uplus Post_{OT_a}\}\!\}\right)$

Note that, because $\mathbb{E} \cap \mathbb{A} = \emptyset$ (actions and expressions are disjoint) and $\alpha_{1a} = \tau$ we have directly (α_{1a} cannot be a variable, and cannot contain expressions/variables because τ has no parameter):

$$\frac{\gamma_{ja}^{j\in J_a}, Pred_{OT_a}, Post_{OT_a}}{t \xRightarrow{\tau} u_a}$$

2. Concerning the middle open transition in $\mathcal{T}_1$:

$$\frac{\beta_{2j}^{j\in J_2'}, Pred_2, Post_2}{s_{20} \xrightarrow{\alpha'} s_{21}}$$

we have $(s_{11}, u_a|Pred_{s_{11},a}) \in \mathcal{R}$ and $s_{11} = s_{20}$. Again by Definition 14 we can deduce there exists an indexed set of weak open transitions $WOT_b^{b\in B} \subseteq \mathcal{WT}_2$:

$$\frac{\gamma_{jb}^{j\in J_b}, Pred_{OT_b}, Post_{OT_b}}{u_a \xRightarrow{\alpha_{2b}} v_b}$$

such that $\forall b, J_b = \{j \in J_2' | \beta_{2j} \neq \tau\}, (s_{21}, v_b|Pred_{s_{21},b}) \in \mathcal{R}$;
$Pred_{s_{11},a} \wedge Pred_2 \implies$
$\bigvee_{b\in B} \left(\forall j \in J_b.(\beta_{2j})^{\nabla} = \gamma_{jb} \wedge Pred_{OT_b} \wedge \alpha' = \alpha_{2b} \wedge Pred_{s_{21},b}\{\!\{Post_2 \uplus Post_{OT_b}\}\!\}\right)$

3. Similarly to the case 1, we consider the third open transition in $\mathcal{T}_1$:

$$\frac{\beta_{3j}^{j\in J_3'}, Pred_3, Post_3}{s_{31} \xrightarrow{\tau} s_{32}} \in \mathcal{T}$$

From previous case, we have $(s_{21}, v_b | Pred_{s_{21},b}) \in \mathcal{R}$, and we have $s_{21} = s_{31}$. Then, by Definition 14 there exists an indexed set of weak open transitions $WOT_a^{c\in C} \subseteq \mathcal{WT}_2$:

$$\frac{\gamma_{jc}^{j\in J_c}, Pred_{OT_c}, Post_{OT_c}}{v_b \overset{\tau}{\Rightarrow} w_c}$$

such that $\forall c, J_c = \{j \in J_3' | \beta_{3j} \neq \tau\}, (s_{31}, w_c | Pred_{s_{31},c}) \in \mathcal{R}$ and

$$Pred_{s_{21},b} \wedge Pred_3 \implies \bigvee_{c\in C} \left(\forall j \in J_c.(\beta_{3j})^{\nabla} = \gamma_{jc} \wedge Pred_{OT_c} \wedge Pred_{s_{31},c}\{\!\{Post_3 \uplus Post_{OT_c}\}\!\}\right)$$

Based on cases described above by applying **WT3** rule on the resulting WOTs we have:

$$\frac{\gamma_{ja}^{j\in J_a}, Pred_{OT_x}, Post_{OT_a} \quad \gamma_{jb}^{j\in J_b}, Pred_{OT_y}, Post_{OT_b} \quad \gamma_{jc}^{j\in J_c}, Pred_{OT_c}, Post_{OT_c}}{\begin{array}{c} t \overset{\tau}{\Rightarrow} u_a \quad u_a \overset{\alpha_2}{\Rightarrow} v_b \quad v_b \overset{\tau}{\Rightarrow} w_c \\ \overline{\gamma'} = \gamma_{ja}^{j\in J_a} \oplus \gamma_{jb}^{j\in J_b}\{\!\{Post_{OT_a}\}\!\} \oplus \gamma_{jc}^{j\in J_c}\{\!\{Post_{OT_b} \odot Post_{OT_a}\}\!\} \\ Pred = Pred_{OT_a} \wedge Pred_{OT_b}\{\!\{Post_{OT_a}\}\!\} \wedge Pred_{OT_c}\{\!\{Post_{OT_b} \odot Post_{OT_a}\}\!\} \\ Post = Post_{OT_c} \odot Post_{OT_b} \odot Post_{OT_a} \quad \alpha'' = \alpha_2\{\!\{Post_{OT_a}\}\!\} \end{array}}$$

$$\frac{\overline{\gamma}, Pred, Post}{t \overset{\alpha''}{\Rightarrow} w_c}$$

It remains to be proven that the following statement holds:
$Pred_{s,t} \wedge Pred \implies \bigvee_{x\in X} \left(\forall j \in J.\gamma_j' = \gamma_j \wedge Pred \wedge \alpha = \alpha'' \wedge Pred_{s',x}\{\!\{Post_{OT} \uplus Post\}\!\}\right)$
We have:
$Pred_{OT} = Pred_1 \wedge Pred_2\{\!\{Post_1\}\!\} \wedge Pred_3\{\!\{Post_2 \odot Post_1\}\!\}$
$Post_{OT} = Post_3 \odot Post_2 \odot Post_1$
Moreover, we have the following statement:
$Pred_{s,t} \wedge Pred_1 \implies \bigvee_{a\in A} \left(\forall j \in J_a.(\beta_{1j})^{\nabla} = \gamma_{ja} \wedge Pred_{OT_a} \wedge Pred_{s_{11},a}\{\!\{Post_1 \uplus Post_{OT_a}\}\!\}\right)$
With the conjunction of the predicate $Pred_2\{\!\{Post_1\}\!\}$ on both sides of the im-

plication, we get:

$$Pred_{s,t} \wedge Pred_1) \wedge Pred_2\{\!\{Post_1\}\!\} \implies \bigvee_{a\in A}\big(\forall j \in J_a.(\beta_{1j})^{\nabla} = \gamma_{ja} \wedge Pred_{OT_a} \wedge Pred_{s_{11},a}\{\!\{Post_1 \uplus Post_{OT_a}\}\!\} \wedge Pred_2\{\!\{Post_1 \uplus Post_{OT_a}\}\!\}\big)$$

Note that on the right side of the implication we added the substitution of $Post_{OT_x}$ without affecting the validity of the statement, because the domain of the substitution function $Post_{OT_x}$ is disjoint from the others. Hence a little rewriting gives:

$$Pred_{s,t} \wedge Pred_1) \wedge Pred_2\{\!\{Post_1\}\!\} \implies \bigvee_{a\in A}\big(\forall j \in J_a.(\beta_{1j})^{\nabla} = \gamma_{ja} \wedge Pred_{OT_a} \wedge (Pred_{s_{11},a} \wedge Pred_2)\{\!\{Post_1 \uplus Post_{OT_a}\}\!\}\big)$$

By replacing the inner predicate $(Pred_{s_{11},u_a} \wedge Pred_2)$ by the conclusion of the statement given in case 2, the formula becomes:

$$Pred_{s,t} \wedge Pred_1) \wedge Pred_2\{\!\{Post_1\}\!\} \implies \bigvee_{a\in A}\Big(\forall j \in J_a.(\beta_{1j})^{\nabla} = \gamma_{ja} \wedge Pred_{OT_a} \wedge \Big(\bigvee_{b\in B}(\forall j \in J_b.(\beta_{2j})^{\nabla} = \gamma_{jb} \wedge Pred_{OT_b} \wedge \alpha' = \alpha_{2b} \wedge Pred_{s_{21},b}\{\!\{Post_2 \uplus Post_{OT_b}\}\!\})\{\!\{Post_1 \uplus Post_{OT_a}\}\!\}\Big)\Big)$$

This can be rewritten into:

$$Pred_{s,t} \wedge Pred_1) \wedge Pred_2\{\!\{Post_1\}\!\} \implies \bigvee_{a\in A}\bigvee_{b\in B}\Big(\forall j \in J_a.(\beta_{1j})^{\nabla} = \gamma_{ja} \wedge \forall j \in J_b.(\beta_{2j})^{\nabla} = \gamma_{jb}\{\!\{Post_1 \uplus Post_{OT_a}\}\!\} \wedge Pred_{OT_a} \wedge Pred_{OT_b}\{\!\{Post_1 \uplus Post_{OT_a}\}\!\} \wedge (\alpha' = \alpha_{2b})\{\!\{Post_1 \uplus Post_{OT_a}\}\!\} \wedge Pred_{s_{21},b}\{\!\{Post_2 \odot Post_1 \uplus Post_{OT_b} \odot Post_{OT_a}\}\!\}\Big)$$

Since $Post_1$ does not act on γ_{jb}, nor on $Pred_{OT_b}$ and α_2. As well $Post_{OT_a}$ does not act on α', nor on β_{2j} the formula can be simplified as follows:

$$Pred_{s,t} \wedge Pred_1) \wedge Pred_2\{\!\{Post_1\}\!\} \implies \bigvee_{a\in A}\bigvee_{b\in B}\Big(\forall j \in J_a.(\beta_{1j})^{\nabla} = \gamma_{ja} \wedge \forall j \in J_b.(\beta_{2j})^{\nabla}\{\!\{Post_1\}\!\} = \gamma_{jb}\{\!\{Post_{OT_a}\}\!\} \wedge Pred_{OT_a} \wedge Pred_{OT_b}\{\!\{Post_{OT_a}\}\!\} \wedge \alpha'\{\!\{Post_1\}\!\} = \alpha_{2b}\{\!\{Post_{OT_a}\}\!\} \wedge Pred_{s_{21},b}\{\!\{Post_2 \odot Post_1 \uplus Post_{OT_b} \odot Post_{OT_a}\}\!\}\Big)$$

Finally, the conjunction with the term $Pred_3\{\!\{Post_2 \odot Post_1\}\!\}$ of the both sides of the implication and rewriting, we get:

$$Pred_{s,t} \wedge Pred_1) \wedge Pred_2\{\!\{Post_1\}\!\} \wedge Pred_3\{\!\{Post_2 \odot Post_1\}\!\} \implies$$
$$\bigvee_{a\in A}\bigvee_{b\in B}\Big(\forall j \in J_a.(\beta^1_j)^{\nabla} = \gamma_{ja} \wedge \forall j \in J_b.(\beta_{2j})^{\nabla}\{\!\{Post_1\}\!\} = \gamma_{jb}\{\!\{Post_{OT_a}\}\!\} \wedge Pred_{OT_a} \wedge$$
$$Pred_{OT_b}\{\!\{Post_{OT_a}\}\!\} \wedge \alpha'\{\!\{Post_1\}\!\} = \alpha_{2b}\{\!\{Post_{OT_a}\}\!\} \wedge (Pred_{s_{21},v_b} \wedge Pred_3)$$
$$\{\!\{Post_2 \odot Post_1 \uplus Post_{OT_b} \odot Post_{OT_a}\}\!\}\Big)$$

Again note that because of the domain of the substitution function is independent from some predicates and expressions, we removed $Post_1$ and we added the term $Post_{OT_b} \odot Post_{OT_a}$ in the substitution of the right side of the implication. Finally, by replacing the predicate $(Pred_{s_{21},b} \wedge Pred_3)$ by the conclusion of the implication given in case 3, we get:

$$Pred_{s,t} \wedge \underbrace{Pred_1 \wedge Pred_2\{\!\{Post_1\}\!\} \wedge Pred_3\{\!\{Post_2 \odot Post_1\}\!\}}_{Pred_{OT}} \implies$$
$$\bigvee_{a\in A}\bigvee_{b\in B}\bigvee_{c\in C}\Big(\forall j \in J_a.(\beta_{1j})^{\nabla} = \gamma_{ja} \wedge \forall j \in J_b.(\beta_{2j}\{\!\{Post_{OT_a}\}\!\})^{\nabla} = \gamma_{jb}\{\!\{Post_{OT_a}\}\!\} \wedge$$
$$\forall j \in J_c.(\beta_{3j}\{\!\{Post_2 \odot Post_1\}\!\})^{\nabla} = \gamma_{jc}\{\!\{Post_{OT_b} \odot Post_{OT_a}\}\!\} \wedge$$
$$\underbrace{Pred_{OT_a} \wedge Pred_{OT_b}\{\!\{Post_{OT_a}\}\!\} \wedge Pred_{OT_c}\{\!\{Post_{OT_b} \odot Post_{OT_a}\}\!\}}_{Pred} \wedge$$
$$\underbrace{\alpha'\{\!\{Post_1\}\!\}}_{\alpha} = \underbrace{\alpha_{2b}\{\!\{Post_{OT_a}\}\!\}}_{\alpha''}$$
$$\wedge\, Pred_{s_{31},c}\{\!\{\underbrace{Post_3 \odot Post_2 \odot Post_1}_{Post_{OT}} \uplus \underbrace{Post_{OT_c} \odot Post_{OT_b} \odot Post_{OT_a}}_{Post}\}\!\}\Big)$$

The three for all statements (on J_a, J_b and J_c) can be concatenated using $\oplus$, the list union lifted to indexed sets (if $\gamma = \gamma'$ and $\gamma'' = \gamma'''$ then $\gamma \oplus \gamma'' = \gamma' \oplus \gamma'''$).

$$\forall j \in J_a \uplus J_b \uplus J_c.(\beta_{1j})^{\nabla} \oplus (\beta_{2j}\{\!\{Post_{OT_a}\}\!\})^{\nabla} \oplus (\beta_{3j}\{\!\{Post_2 \odot Post_1\}\!\})^{\nabla} =$$
$$\gamma_{ja} \oplus \gamma_{jb}\{\!\{Post_{OT_a}\}\!\} \oplus \gamma_{jc}\{\!\{Post_{OT_b} \odot Post_{OT_a}\}\!\}$$

We have $s_{31} = s'$, so can rewrite the formula:
$Pred_{s,t} \wedge Pred_{OT} \implies \bigvee_{a\in A}\bigvee_{b\in B}\bigvee_{c\in C}\Big(\forall j \in J.\gamma'_j = \gamma_j \wedge Pred \wedge \alpha = \alpha'' \wedge Pred_{s',c}\{\!\{Post_{OT} \uplus Post\}\!\}\Big)$

All the combinations of elements in A, B, and C provide a set X of weak open transitions (each combination of one transition in A, one in B, and one in C provides one weak open transition in the set X, i.e. each $x \in X$ corresponds

to a triple $(a, b, c) \in A \times B \times C$); this defines a set of weak open transitions indexed over X; each such open transition leads to a w_c that we call t_x. This re-indexing allows us to conclude:

$$Pred_{s,t} \wedge Pred_{OT} \implies \bigvee_{x \in X} \Big(\forall j \in J. \gamma'_j = \gamma_j \wedge Pred \wedge \alpha = \alpha'' \wedge Pred_{s',x} \{\!\{ Post_{OT} \uplus Post \}\!\} \Big)$$

Theorem 6. *Weak FH-bisimilarity is an equivalence.* Suppose $\mathcal{R}$ is a weak FH-bisimulation. Then $\mathcal{R}$ is an equivalence, that is, $\mathcal{R}$ is reflexive, symmetric and transitive.

With the above lemma, we can use the same technique as for Theorem 1 to prove that a weak FH-bisimilarity is an equivalence. Indeed, we essentially use the same proof-scheme the main difference concerns β and γ. Indeed, while the schema of the proof of transitivity was not directly applicable on the definition of weak bisimulation, Lemma 5 provides a characterization of weak bisimulation similar to the definition of strong bisimulation, and thus the same proof scheme is directly applicable.

Appendix B.2. Composition properties

This section gives decomposition/composition lemmas and their proofs, these are the equivalent of the composition lemmas for open transitions, but applied to weak open automata.

Lemma 6 (Weak open transition decomposition). *Let* $\text{Leaves}(Q) = pLTS_l^{l \in L_Q}$*; suppose*[16]*:*

$$P[Q]_{j_0} \models \frac{\gamma_j^{j \in J}, Pred, Post}{\lhd s_i^{i \in L} \rhd \overset{\alpha}{\Rightarrow} \lhd s'^{i \in L}_i \rhd}$$

with $J \cap \text{Holes}(Q) \neq \emptyset$ *or* $\exists i \in L_Q. s_i \neq s'_i$*, i.e. Q takes part in the reduction. Then there exist n, Pred′, Post′, and for all* $p \in [1..n]$ *there exist* β_p*,* α_p*,* $Pred_p$*,* $Post_p$ *and a family* $\gamma_{pj}^{j \in J_p}$ *and for all* $p \in [1..n+1]$ $s_{pi}^{i \in L_Q}$*. s.t.:*

[16]Note that the hypotheses of the lemma imply that Q is not a pLTS but a similar lemma can be proven for a pLTS Q.

$$P \models \frac{\gamma_j^{j\in(J_p\setminus \mathrm{Holes}(Q))\cup\{j_0\}}, Pred', Post'}{\lhd s_i^{i\in L\setminus L_Q} \rhd \overset{\alpha}{\Rightarrow} \lhd s_i'^{i\in L\setminus L_Q}\rhd} \qquad \text{and} \qquad \gamma_{j_0} = [\beta_0..\beta_n]$$

$$\text{and for all } p \in [1..n] \quad Q \models \frac{\gamma_{pj}^{j\in J_p}, Pred_p, Post_p}{\lhd s_{pi}^{i\in L_Q} \rhd \overset{\alpha_p}{\Longrightarrow} \lhd s_{(p+1)i}^{i\in L_Q}\rhd}$$

$$\text{such that} \qquad \bigcup_{p=1}^{n} J_p = J\cap \mathrm{Holes}(Q),\ \gamma_j^{j\in J\cap \mathrm{Holes}(Q)} = \bigoplus_{p=1}^{n} (\gamma_{pj}^{j\in J_p})\{\!\{ \bigodot_{i=p-1}^{1} Post_i \}\!\},$$

$$Pred \iff Pred' \wedge \bigwedge_{p=1}^{n} (\alpha_p\{\!\{ \bigodot_{i=p-1}^{1} Post_i \}\!\} = \beta_p \wedge Pred_p\{\!\{ \bigodot_{i=p-1}^{1} Post_i \}\!\}),$$

$$Post = Post' \uplus \bigodot_{p=n}^{1} Post_p, \text{ and } \quad \forall i \in L_Q.\, s_{(n+1)i} = s_i' \wedge s_{1i} = s_i$$

where for any p, $Post_p$ only acts upon variables $vars(Q)$.

Proof. Suppose that we have:

$$P[Q]_{j_0} \models \frac{\gamma_j^{j\in J}, Pred, Post}{\lhd s_l^{l\in L} \rhd \overset{\alpha}{\Rightarrow} \lhd s_l'^{l\in L}\rhd}$$

By Lemma 4 this implies the following:

$$\forall p \in [1..m_1]\ P[Q]_{j_0} \models \frac{\overline{\beta_{1p}}, Pred_{1p}, Post_{1p}}{\lhd s_{pl}^{l\in L} \rhd \overset{\tau}{\rightarrow} \lhd s_{(p+1)l}^{l\in L}\rhd}, \qquad P[Q]_{j_0} \models \frac{\overline{\beta_2}, Pred_2, Post_2}{\lhd t_l^{l\in L} \rhd \overset{\alpha'}{\longrightarrow} \lhd t'^{l\in L}_l\rhd}$$

$$\text{and } \forall p \in [1..m_2]\ P[Q]_{j_0} \models \frac{\overline{\beta_{3p}}, Pred_{3p}, Post_{3p}}{\lhd u_{pl}^{l\in L} \rhd \overset{\tau}{\rightarrow} \lhd u_{(p+1)l}^{l\in L}\rhd}$$

where

$$\forall l \in L.\, s_l = s_{1l} \wedge s_{(m_1+1)l} = t_l \wedge t'_l = u_{1l} \wedge u_{(m_2+1)l} = s'_l$$

$$\alpha = \alpha' \{\!\{ \bigodot_{j=m_1}^{1} Post_{1j} \}\!\}$$

$$\gamma_j^{j\in J} = \bigoplus_{i=1}^{m_1} (\overline{\beta_{1i}} \{\!\{ \bigodot_{j=i-1}^{1} Post_{1j} \}\!\})^{\nabla} \oplus (\overline{\beta_2} \{\!\{ \bigodot_{j=m_1}^{1} Post_{1j} \}\!\})^{\nabla} \oplus \bigoplus_{i=1}^{m_2} (\overline{\beta_{3i}} \{\!\{ \bigodot_{j=i-1}^{1} Post_{3j} \odot Post_2 \odot \bigodot_{j=m_1}^{1} Post_{1j} \}\!\})^{\nabla}$$

$$Pred = \bigwedge_{i=1}^{m_1} Pred_{1i} \{\!\{ \bigodot_{j=i-1}^{1} Post_{1j} \}\!\} \wedge Pred_2 \{\!\{ \bigodot_{j=m_1}^{1} Post_{1j} \}\!\} \wedge \bigwedge_{i=1}^{m_2} Pred_{3i} \{\!\{ \bigodot_{j=i-1}^{1} Post_{3j} \odot Post_2 \odot \bigodot_{j=m_1}^{1} Post_{1j} \}\!\}$$

$$Post = \bigodot_{j=m_2}^{1} Post_{3j} \odot Post_2 \odot \bigodot_{j=m_1}^{1} Post_{1j}$$

We can apply Lemma 1 on each OT:

1. For each open transition OT_p in the form $(\overline{\beta_{1p}} = \beta_{1pj}^{j\in J_{1p}})$:

   $$P[Q]_{j_0} \models \frac{\beta_{1pj}^{j\in J_{1p}}, Pred_{1p}, Post_{1p}}{\lhd s_{pl}^{l\in L} \rhd \xrightarrow{\tau} \lhd s_{(p+1)l}^{l\in L} \rhd}$$

   If Q moves then we obtain by Lemma 1:
   $P \models \dfrac{(\beta_{1pj})^{j\in(J_{1p}\setminus \mathrm{Holes}(Q))\cup\{j_0\}}, Pred'_{1p}, Post'_{1p}}{\lhd s_{pl}^{l\in L\setminus L_Q} \rhd \xrightarrow{\tau} \lhd (s_{(p+1)l})^{l\in L\setminus L_Q} \rhd}$ and
   $Q \models \dfrac{(\beta_{1pj})^{j\in J_{1p}\cap \mathrm{Holes}(Q)}, Pred''_{1p}, Post''_{1p}}{\lhd s_{pl}^{l\in L_Q} \rhd \xrightarrow{\alpha_{1p}} \lhd s_{(p+1)l}^{l\in L_Q} \rhd}$
   such that
   $Pred_{1p} \Longleftrightarrow Pred'_{1p} \wedge Pred''_{1p} \wedge \alpha_{1p} = \beta_{1pj_0}$, $Post_{1p} = Post'_{1p} \uplus Post''_{1p}$ where $Post''_{1p}$ is the restriction of $Post_{1p}$ over $vars(Q)$.
   Else Q does not move and we have:
   $P \models \dfrac{(\beta_{1pj})^{j\in(J_{1p}\setminus \mathrm{Holes}(Q))}, Pred'_{1p}, Post'_{1p}}{\lhd (s_{pl})^{l\in L\setminus L_Q} \rhd \xrightarrow{\tau} \lhd (s_{(p+1)l})^{l\in L\setminus L_Q} \rhd}$ and $\lhd (s_p)_l^{l\in L_Q} \rhd = \lhd (s_{(p+1)l})^{l\in L_Q} \rhd$

2. Similarly, we have similar open transitions on states u_{pl} (for the final τ transitions).

3. Finally, for the open transition in the form $(\overline{\beta_2} = \beta_{2j}^{j\in J_2})$:

$$P[Q]_{j_0} \models \frac{\beta_{2j}^{j\in J_2}, Pred_2, Post_2}{\lhd t_l^{l\in L_Q} \rhd \xrightarrow{\alpha'} \lhd (t'_l)^{l\in L_Q} \rhd}$$

If Q moves then we obtain by Lemma 1:

$P \models \dfrac{(\beta_{2j})^{j\in(J_2\setminus \mathrm{Holes}(Q))\cup\{j_0\}}, Pred'_2, Post'_2}{\lhd t_l^{l\in L\setminus L_Q} \rhd \xrightarrow{\alpha'} \lhd t_l'^{l\in L\setminus L_Q} \rhd}$ and

$Q \models \dfrac{(\beta_{2j})^{j\in J_2\cap \mathrm{Holes}(Q)}, Pred''_2, Post''_2}{\lhd t_l^{l\in L_Q} \rhd \xrightarrow{\alpha_{21}} \lhd t_l'^{l\in L_Q} \rhd}$

such that $Pred_2 \iff Pred'_2 \wedge Pred''_2 \wedge \alpha_{20} = \beta_{2j_0}$, $Post_2 = Post'_2 \uplus Post''_2$ where $Post''_2$ is the restriction of $Post_2$ over variables $vars(Q)$.

Else Q does not move and we have:

$P \models \dfrac{(\beta_{2j})^{j\in(J_2\setminus \mathrm{Holes}(Q))\cup\{j_0\}}, Pred'_2, Post'_2}{\lhd t_l^{l\in L\setminus L_Q} \rhd \xrightarrow{\alpha'} \lhd (t_l)'^{l\in L\setminus L_Q} \rhd}$ and $\lhd t_l^{l\in L_Q} \rhd = \lhd (t'_l)^{l\in L_Q} \rhd$

By using Lemma 4, and denoting $J = \bigcup_{i=1}^{m_1} J_{1i} \cup J_2 \cup \bigcup_{i=1}^{m_2} J_{3i}$, we can conclude from cases (1), (2) and (3) that we have:

$$P \models \frac{(\gamma'_j)^{j\in(J\setminus \mathrm{Holes}(Q))\cup\{j_0\}}, Pred', Post'}{\lhd s_l^{l\in L\setminus L_Q} \rhd \xRightarrow{\alpha''} \lhd s_l'^{l\in L\setminus L_Q} \rhd}$$

where $\alpha'' = \alpha' \{\!\{ \overset{1}{\underset{j=m_1}{\odot}} Post'_{1j} \}\!\}$

On the other hand, we have:

$\alpha = \alpha' \{\!\{ \overset{1}{\underset{j=m_1}{\odot}} Post'_{1j} \}\!\} \{\!\{ \overset{1}{\underset{j=m_1}{\odot}} Post''_{1j} \}\!\} = \alpha'' \{\!\{ \overset{1}{\underset{j=m_1}{\odot}} Post''_{1j} \}\!\}.$

As $\{\!\{ \overset{1}{\underset{j=m_1}{\odot}} Post''_{1j} \}\!\}$ has no effect on variables of P and thus on variables of α'', so we have $\alpha = \alpha''$.

$$\begin{aligned}
&\forall l \in L.\, s_l = s_{1l} \wedge s_{(m_1+1)l} = t_l \wedge t'_l = u_{1l} \wedge u_{(m_2+1)l} = s'_l \\
(\gamma'_j)^{j\in(J\setminus \mathrm{Holes}(Q))\cup\{j_0\}} =& \bigoplus_{i=1}^{m_1} (\overline{\beta_{1i}} \{\!\{ \overset{1}{\underset{j=i-1}{\odot}} Post'_{1j} \}\!\})^{\nabla} \oplus (\overline{\beta_2} \{\!\{ \overset{1}{\underset{j=m_1}{\odot}} Post'_{1j} \}\!\})^{\nabla} \oplus \\
& \bigoplus_{i=1}^{m_2} (\overline{\beta_{3i}} \{\!\{ \overset{1}{\underset{j=i-1}{\odot}} Post'_{3j} \odot Post'_2 \odot \overset{1}{\underset{j=m_1}{\odot}} Post'_{1j} \}\!\})^{\nabla}
\end{aligned}$$

$$Pred' = \bigwedge_{i=1}^{m_1} Pred'_{1i}\{\!\{ \bigodot_{j=i-1}^{1} Post'_{1j} \}\!\} \wedge Pred'_2\{\!\{ \bigodot_{j=m_1}^{1} Post'_{1j} \}\!\} \wedge$$
$$\bigwedge_{i=1}^{m_2} Pred'_{3i}\{\!\{ \bigodot_{j=i-1}^{1} Post'_{3j} \odot Post'_2 \odot \bigodot_{j=m_1}^{1} Post'_{1j} \}\!\}$$
$$Post' = \bigodot_{j=m_2}^{1} Post'_{3j} \odot Post'_2 \odot \bigodot_{j=m_1}^{1} Post'_{1j}$$

Note that for all $j \in J \setminus \text{Holes}(Q)$, $\gamma'_j = \gamma_j$ because for all l $Post'_{1l}$ coincides with $Post_{1l}$ on the variables of β_{1ij}, and similarly for $Post'_2$ and $Post'_{3l}$.

We introduce the following predicate (we will need it for reasoning about the global predicate and will reason about it along the proof):

$$Pred_\beta = \bigwedge_{p=1}^{m_1} (\beta_{1pj_0} = \alpha_{1p})\{\!\{ \bigodot_{j=p-1}^{1} Post_{1j} \}\!\} \wedge (\beta_{2j_0} = \alpha_{21})\{\!\{ \bigodot_{j=m_1}^{1} Post_{1j} \}\!\} \wedge$$
$$\bigwedge_{p=1}^{m_2} (\beta_{3pj_0} = \alpha_{3p})\{\!\{ \bigodot_{j=p-1}^{1} Post_{3j} \odot Post_2 \odot \bigodot_{j=m_1}^{1} Post_{1j} \}\!\}$$

Concerning Q, we reduce the sequence of OTs to a path for which it moves in all steps. In other words, if Q does not move at step q, then we have $\lhd s_{ql}^{l\in L_Q} \rhd = \lhd s_{(q+1)l}^{l\in L_Q} \rhd$, then we skip the state $\lhd s_{(q+1)l}^{l\in L_Q} \rhd$, i.e. we rename all the following states $\lhd s_{pl}^{l\in L_Q} \rhd$ where $p \geq q+1$ into $\lhd s_{(p-1)l}^{l\in L_Q} \rhd$. Note that self-loops where Q does an action but stays at the same state are not removed. We proceed in the same way for states named u. To simplify the proof, we suppose that in case 3, Q moves, else transition 3 of Q should be skipped and the last s_{pl} are equal to the first u_{1l}. So we have:

$$\forall p \in [1..n_1]\ Q \models \frac{(\beta'_{1pj})^{j\in J\cap \text{Holes}(Q)}, Pred''_{1p}, Post''_{1p}}{\lhd s_{pl}^{l\in L_Q} \rhd \xrightarrow{\alpha_{1p}} \lhd s_{(p+1)l}^{l\in L_Q} \rhd},$$

$$Q \models \frac{(\beta'_{2j})^{j\in J\cap \text{Holes}(Q)}, Pred''_2, Post''_2}{\lhd t_l^{l\in L_Q} \rhd \xrightarrow{\alpha_{21}} \lhd t'^{l\in L_Q}_l \rhd} \text{ and}$$

$$\forall p \in [1..n_2]\ Q \models \frac{(\beta'_{3pj})^{j\in J\cap \text{Holes}(Q)}, Pred''_{3p}, Post''_{3p}}{\lhd u_{pl}^{l\in L_Q} \rhd \xrightarrow{\alpha_{3p}} \lhd u_{(p+1)l}^{l\in L_Q} \rhd}$$

such that $n_1 \leq m_1$ and $n_2 \leq m_2$.

By renaming all state names (s, u and t) with the same state name v. We have:

$$\forall p \in [1..(n_1+n_2+1)]\ Q \models \frac{\beta_{pj}^{j\in J\cap \text{Holes}(Q)}, Pred''_p, Post''_p}{\lhd v_{pl}^{l\in L_Q} \rhd \xrightarrow{\alpha'_p} \lhd v_{(p+1)l}^{l\in L_Q} \rhd}$$

In this equation, and using case 1 above for all $k \in [1 \ldots n_1]$ there is a $p \in [1..m_1]$ such that $\alpha_{1p} = \alpha'_k$ (following the re-indexing done in the removal of steps where Q does not move), we know that $Pred_{1p}$ contains the predicate $(\alpha_{1p} = \beta_{1pj_0})$.

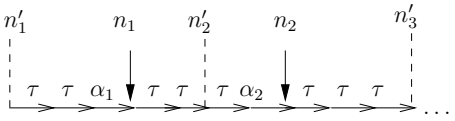


Figure B.10: Composition of the subsequences

Because β_{1pj_0} only contains variables of P and α'_k only variables of Q, we have:

$$(\alpha_{1p} = \beta_{1pj_0})\{\!\{ \overset{1}{\underset{j=p-1}{\odot}} Post_{1j} \}\!\} \iff \alpha_{1p}\{\!\{ \overset{1}{\underset{j=p-1}{\odot}} Post''_{1j} \}\!\} = \beta_{1pj_0}\{\!\{ \overset{1}{\underset{j=p-1}{\odot}} Post'_{1j} \}\!\}$$
$$\iff \alpha'_k\{\!\{ \overset{1}{\underset{j=k-1}{\odot}} Post''_{j} \}\!\} = \beta_{1pj_0}\{\!\{ \overset{1}{\underset{j=p-1}{\odot}} Post'_{1j} \}\!\}$$

We can obtain similar equations for α'_{n_1+1} related with β_{2j_0} and the α'_k for $k \geq n_1 + 2$ related with β_{3pj_0} for some p. Note that the substitutions are however more complex in the other cases. Overall we obtain (we skip here the details about the three cases 1, 2, and 3 above that all fall into the same equation because of the re-indexing we perform):

$$Pred_\beta \iff \gamma_{j_0} = ([\alpha'_p\{\!\{ \overset{1}{\underset{j=p-1}{\odot}} Post''_{j} \}\!\} | p \in [1..n_1 + n_2 + 1]])^\nabla \tag{B.1}$$

Let us consider the sequence of (n_1+n_2+1) actions α'_p some of them may be non-observable (they are τ transitions). By considering the sequence of τ and non-τ actions we split the sequence of actions into $n+1$ sub-sequences, such that each sub-sequence is a sequence of actions containing only one observable action that will be named α_p, and possibly many non-observable (τ) ones.

We can decompose each of the $n+1$ sub-sequences in the following way (see Figure B.10). For $k \in [0..n]$ the position of the k^{th} visible action is n_k. For $l \in [1..n]$, n'_l is any index between n_{l-1} and n_l, additionally $n'_1 = 1$ and $n'_{n+1} = n_1 + n_2 + 1$. We obtain n sub-sequences made of the following OTs, for all $k \in [1..n]$:

$$\forall p \in [n'_k..(n_k-1)]\ Q \models \frac{\beta_{pj}^{j\in J\cap \text{Holes}(Q)}, Pred''_p, Post''_p}{\lhd v_{pl}^{l\in L_Q} \rhd \xrightarrow{\tau} \lhd v_{(p+1)l}^{l\in L_Q} \rhd},$$

$$Q \models \frac{\beta_{n_k j}^{j\in J\cap \text{Holes}(Q)}, Pred''_{n_k}, Post''_{n_k}}{\lhd v_{n_k l}^{l\in L_Q} \rhd \xrightarrow{\alpha'_{n_k}} \lhd v_{(n_k+1)l}^{l\in L_Q} \rhd} \text{ and}$$

$$\forall p \in [(n_k+1)..(n'_{k+1}-1)]\ Q \models \frac{\beta_{pj}^{j\in J\cap \text{Holes}(Q)}, Pred''_p, Post''_p}{\lhd v_{pl}^{l\in L_Q} \rhd \xrightarrow{\tau} \lhd v_{(p+1)l}^{l\in L_Q} \rhd}$$

Thereafter, by Lemma 4 we can deduce the following weak open transition:

$$Q \models \frac{(\gamma_{kj})^{j\in J\cap \text{Holes}(Q)}, Pred_k, Post_k}{\lhd v_{kl}^{l\in L_Q} \rhd \overset{\alpha_k}{\Longrightarrow} \lhd (v'_{kl})^{l\in L_Q} \rhd}$$

with:

$$\forall l \in L_Q . v_{kl} = v_{(n'_k)l} \wedge v'_{kl} = v_{(n'_k)l}$$

$$\alpha_k = \alpha'_{n_k} \{\!\{ \bigodot_{j=n_k-1}^{n'_k} Post''_j \}\!\}$$

$$\gamma_{kj}^{j\in J\cap \mathrm{Holes}(Q)} = \bigoplus_{i=n'_k}^{(n_k-1)} (\beta_{ij}^{j\in J\cap \mathrm{Holes}(Q)} \{\!\{ \bigodot_{l=i-1}^{n'_k} Post''_l \}\!\})^{\nabla} \oplus (\beta_{n_k j}^{j\in J\cap \mathrm{Holes}(Q)} \{\!\{ \bigodot_{l=n_k-1}^{n'_k} Post''_l \}\!\})^{\nabla} \oplus$$
$$\bigoplus_{i=n_k+1}^{n'_{k+1}-1} (\beta_{ij}^{j\in J\cap \mathrm{Holes}(Q)} \{\!\{ \bigodot_{l=i-1}^{n_k+1} Post''_l \odot Post''_{n_k} \odot \bigodot_{l=n_k-1}^{n'_k} Post''_l \}\!\})^{\nabla}$$

$$Pred_k = \bigwedge_{i=n'_k}^{n_k-1} Pred''_i \{\!\{ \bigodot_{j=i-1}^{n'_k} Post''_j \}\!\} \wedge Pred''_{n_k} \{\!\{ \bigodot_{j=n_k-1}^{n'_k} Post''_j \}\!\} \wedge$$
$$\bigwedge_{i=n_k+1}^{n'_{k+1}-1} Pred''_i \{\!\{ \bigodot_{j=i-1}^{n'_k} Post''_j \odot Post''_{n_k} \odot \bigodot_{j=n_k-1}^{n'_k} Post''_j \}\!\}$$

$$Post_k = \bigodot_{j=n'_{k+1}-1}^{n'_k} Post''_j$$

Note that for all $k \in [1..n-1]$, $v'_{kl} = v_{(k+1)l}$, $v_{1l} = s_{1l} = s_l$, and $v'_{nl} = v_{(n_1+n_2+3)l} = u_{(n_2+1)l} = s'_l$.

By definition of $Post_k$, we have $\bigodot_{j=n'_k-1}^{1} Post''_j = \bigodot_{j=k-1}^{1} Post_j$. Consequently, we have:

$$\alpha'_{n_k} \{\!\{ \bigodot_{j=n_k-1}^{1} Post''_j \}\!\} = \alpha'_{n_k} \{\!\{ \bigodot_{j=n_k-1}^{n'_k} Post''_j \odot \bigodot_{j=n'_k-1}^{1} Post''_j \}\!\} = \alpha_k \{\!\{ \bigodot_{j=n'_k-1}^{1} Post''_j \}\!\} = \alpha_k \{\!\{ \bigodot_{j=k-1}^{1} Post_j \}\!\}$$

From equation B.1, we obtain the following equation (we recall that the actions α_k are the actions α'_p that are observable):

$$Pred_\beta \iff \gamma_{j_0} = \bigoplus_{p=1}^{p=n_1+n_2+2} (\alpha'_p \{\!\{ \bigodot_{j=p-1}^{1} Post_j \}\!\})^{\nabla}$$
$$\iff \gamma_{j_0} = [\alpha_p \{\!\{ \bigodot_{j=p-1}^{1} Post_j \}\!\} | p \in [1..n]]$$

We need now to show that the set of WOT obtained above verifies the

conditions of the lemma, i.e. it is a set of WOT of the form:

$$Q \models \frac{\gamma_{pj}^{j\in J_p}, Pred_p, Post_p}{\lhd s_{pi}^{i\in L_Q} \rhd \overset{\alpha_p}{\Longrightarrow} \lhd s_{(p+1)i}^{i\in L_Q} \rhd}$$

with

$$\bigcup_{p=1}^{n} J_p = J \cap \mathrm{Holes}(Q) \qquad \text{trivial}$$

$$\gamma_j^{j\in J\cap \mathrm{Holes}(Q)} = \bigoplus_{p=1}^{n} (\gamma_{pj}^{j\in J_p}) \{\!\{ \bigodot_{i=p-1}^{1} Post_i \}\!\}$$

Indeed we have:

$$\gamma_j^{j\in J} = \bigoplus_{i=1}^{m_1} (\overline{\beta_{1i}} \{\!\{ \bigodot_{j=i-1}^{1} Post_{1j} \}\!\})^{\nabla} \oplus (\overline{\beta_2} \{\!\{ \bigodot_{j=m_1}^{1} Post_{1j} \}\!\})^{\nabla} \oplus$$
$$\bigoplus_{i=1}^{m_2} (\overline{\beta_{3i}} \{\!\{ \bigodot_{j=i-1}^{1} Post_{3j} \odot Post_2 \odot \bigodot_{j=m_1}^{1} Post_{1j} \}\!\})^{\nabla}$$

And thus, because $\beta_{pj}^{j\in J\cap \mathrm{Holes}(Q)}$ are equal to the concatenation of $(\beta'_{1pj})^{j\in J\cap \mathrm{Holes}(Q)}$, $(\beta'_{2j})^{j\in J\cap \mathrm{Holes}(Q)}$, and $(\beta'_{3pj})^{j\in J\cap \mathrm{Holes}(Q)}$ (re-indexed because we skipped some transitions), and additionally $(\beta'_{1pj})^{j\in J\cap \mathrm{Holes}(Q)}$, $(\beta'_{2j})^{j\in J\cap \mathrm{Holes}(Q)}$, and $(\beta'_{3pj})^{j\in J\cap \mathrm{Holes}(Q)}$ are identical to the hole labels $\beta_{1kj}^{j\in J\cap \mathrm{Holes}(Q)}$, $\beta_{2j}^{j\in J\cap \mathrm{Holes}(Q)}$, and $\beta_{3kj}^{j\in J\cap \mathrm{Holes}(Q)}$ (re-indexed) when Q moves[17]. We can assert a similar equality on post-conditions, i.e. between $Post''_p$ and $Post''_{1k}$, $Post''_2$, $Post''_{3k}$ where $Post''_{1p}$ is the restriction of $Post_{1p}$ over $vars(Q)$ (see initial decomposition, case 1, 2, and 3 above). Overall, we have $\forall i \in L_Q.\, s_{(n+1)i} = s'_i \wedge s_{0i} = s_i$ (see above):

$$\gamma_j^{j\in J\cap \mathrm{Holes}(Q)} = \bigoplus_{i=1}^{m_1} (\overline{(\beta'_{1i})} \{\!\{ \bigodot_{j=i-1}^{1} Post''_{1j} \}\!\})^{\nabla} \oplus (\overline{\beta'_2} \{\!\{ \bigodot_{j=m_1}^{1} Post''_{1j} \}\!\})^{\nabla} \oplus$$
$$\bigoplus_{i=1}^{m_2} (\overline{\beta'_{3i}} \{\!\{ \bigodot_{j=i-1}^{1} Post''_{3j} \odot Post''_2 \odot \bigodot_{j=m_1}^{1} Post''_{1j} \}\!\})^{\nabla}$$
$$= \bigoplus_{k=1}^{n} \left(\bigoplus_{i=n'_k}^{n'_{k+1}-1} (\beta_{ij}^{j\in J\cap \mathrm{Holes}(Q)} \{\!\{ \bigodot_{j=i-1}^{n'_k} Post''_j \bigodot_{j=n'_k-1}^{1} Post''_j \}\!\})^{\nabla} \right)$$
$$= \bigoplus_{k=1}^{n} \left(\gamma_{kj}^{j\in J\cap \mathrm{Holes}(Q)} \{\!\{ \bigodot_{j=k-1}^{1} Post_k \}\!\} \right)$$

[17] more precisely, when Q moves either $\beta_{1kj}^{j\in J\cap \mathrm{Holes}(Q)}$ is not empty and thus $(\beta'_{1pj})^{j\in J\cap \mathrm{Holes}(Q)} = \beta_{1kj}^{j\in J\cap \mathrm{Holes}(Q)}$, or both are empty if the holes of Q perform no action.

Next, we have:

$$Pred = Pred' \wedge \bigwedge_{p=1}^{n} \left((\alpha_p \{\!\{ \bigodot_{i=p-1}^{1} Post_i \}\!\}) = \beta_p \wedge (Pred_p \{\!\{ \bigodot_{i=p-1}^{1} Post_i \}\!\}) \right)$$

Indeed we have:

$$\begin{aligned}
Pred &\iff \bigwedge_{i=1}^{m_1} Pred_{1i} \{\!\{ \bigodot_{j=i-1}^{1} Post_{1j} \}\!\} \wedge Pred_2 \{\!\{ \bigodot_{j=m_1}^{1} Post_{1j} \}\!\} \wedge \bigwedge_{i=1}^{m_2} Pred_{3i} \{\!\{ \bigodot_{j=i-1}^{1} Post_{3j} \odot Post_2 \odot \bigodot_{j=m_1}^{1} Post_{1j} \}\!\} \\
&\iff \bigwedge_{i=1}^{m_1} \left(Pred'_{1i} \wedge Pred''_{1i} \wedge \alpha_{1i} = \beta_{1ij_0} \right) \{\!\{ \bigodot_{j=i-1}^{1} Post_{1j} \}\!\} \\
&\quad \wedge \left(Pred'_2 \wedge Pred''_2 \wedge \alpha_{21} = \beta_{2j_0} \right) \{\!\{ \bigodot_{j=m_1}^{1} Post_{1j} \}\!\} \\
&\quad \wedge \bigwedge_{i=1}^{m_2} \left(Pred'_{3i} \wedge Pred''_{3i} \wedge \alpha_{3i} = \beta_{3ij_0} \right) \{\!\{ \bigodot_{j=i-1}^{0} Post_{3j} \odot Post_2 \odot \bigodot_{j=m_1}^{1} Post_{1j} \}\!\} \\
&\iff \bigwedge_{i=1}^{m_1} \left((Pred'_{1i} \{\!\{ \bigodot_{j=i-1}^{1} Post'_{1j} \}\!\}) \wedge (Pred''_{1i} \{\!\{ \bigodot_{j=i-1}^{1} Post''_{1j} \}\!\}) \wedge (\alpha_{1i} = \beta_{1ij_0}) \{\!\{ \bigodot_{j=i-1}^{1} Post_{1j} \}\!\} \right) \\
&\quad \wedge \left(Pred'_2 \{\!\{ \bigodot_{j=m_1}^{1} Post_{1j} \}\!\} \wedge Pred''_2 \{\!\{ \bigodot_{j=m_1}^{1} Post_{1j} \}\!\} \wedge (\alpha_{21} = \beta_{2j_0}) \{\!\{ \bigodot_{j=m_1}^{1} Post_{1j} \}\!\} \right) \\
&\quad \wedge \bigwedge_{i=1}^{m_2} \left(Pred'_{3i} \{\!\{ \bigodot_{j=i-1}^{1} Post_{3j} \odot Post_2 \odot \bigodot_{j=m_1}^{1} Post_{1j} \}\!\} \wedge Pred''_{3i} \{\!\{ \bigodot_{j=i-1}^{1} Post_{3j} \odot Post_2 \odot \bigodot_{j=m_1}^{1} Post_{1j} \}\!\} \right. \\
&\quad \left. \wedge (\alpha_{3i} = \beta_{3ij_0}) \{\!\{ \bigodot_{j=i-1}^{1} Post_{3j} \odot Post_2 \odot \bigodot_{j=m_1}^{1} Post_{1j} \}\!\} \right) \\
&\iff Pred' \wedge \bigwedge_{k=1}^{n} Pred_k \{\!\{ \bigodot_{j=k-1}^{1} Post_j \}\!\} \wedge Pred_\beta \\
&\iff Pred' \wedge \bigwedge_{k=1}^{n} Pred_k \{\!\{ \bigodot_{j=k-1}^{1} Post_j \}\!\} \wedge (\gamma_{j_0} = [\alpha_i \{\!\{ \bigodot_{j=i-1}^{1} Post_j \}\!\} | i \in [1..n]])
\end{aligned}$$

which is exactly what is needed with $\gamma_{j_0} = [\beta_0..\beta_n]$.

Finally we have $Post = Post' \uplus \bigodot_{p=n}^{1} Post_p$ because

$$\begin{aligned} Post &= \bigodot_{j=m_2}^{1} Post_{3j} \odot Post_2 \odot \bigodot_{j=m_1}^{1} Post_{1j} \\ &= \bigodot_{j=m_2}^{1} Post'_{3j} \odot Post'_2 \odot \bigodot_{j=m_1}^{1} Post'_{1j} \uplus \bigodot_{j=m_2}^{1} Post''_{3j} \odot Post''_2 \odot \bigodot_{j=m_1}^{1} Post''_{1j} \\ &= Post' \uplus \bigodot_{j=n'_{n+1}-1}^{1} Post''_j \end{aligned}$$

Which concludes because we have $\bigodot_{j=n'_k-1}^{1} Post''_j = \bigodot_{j=k-1}^{1} Post_j$. □

Lemma 7 (Weak open transition composition). *Suppose that we have a weak open automaton such that the WOTs cannot observe silent actions (see Definition 11). Suppose $j_0 \in J$ and:*

$$P \models \frac{\beta_j^{j\in J}, Pred, Post}{\triangleleft s_i^{i\in L} \triangleright \xrightarrow{\alpha} \triangleleft (s'_i)^{i\in L} \triangleright} \quad \text{and} \quad Q \models \frac{\overline{\gamma}, Pred_Q, Post_Q}{\triangleleft s_i^{i\in L_Q} \triangleright \xRightarrow{\alpha_Q} \triangleleft (s'_i)^{i\in L_Q} \triangleright}$$

Let $Pred' = Pred \wedge (\beta_{j_0} = \alpha_Q \wedge Pred_Q)$ *and* $Post' = Post \uplus Post_Q$ *Then, we have:*

$$P[Q]_{j_0} \models \frac{\overline{\gamma} \uplus (\beta_j^{j\in J\setminus\{j_0\}})^{\nabla}, Pred', Post'}{\triangleleft s_i^{i\in L\uplus L_Q} \triangleright \overset{\alpha}{\Rightarrow} \triangleleft (s'_i)^{i\in L\uplus L_Q} \triangleright}$$

Proof. By Lemma 4 we can decompose the WOT of Q into a series of $k+1$ and $k'+1$ tau open transitions and an α'_Q open transition (observable or not depending on α_Q):

$$\forall h \in [1..k].Q \models \frac{\overline{\beta_{1h}}, Pred_{1h}, Post_{1h}}{\triangleleft (s_{1h}) \triangleright \xrightarrow{\tau} \triangleleft (s_{1(h+1)}) \triangleright}, \quad Q \models \frac{\overline{\beta_2}, Pred_2, Post_2}{\triangleleft s_{21} \triangleright \xrightarrow{\alpha'_Q} \triangleleft s_{22} \triangleright},$$

$$\text{and} \quad \forall h \in [1..k'].Q \models \frac{\overline{\beta_{3h}}, Pred_{3h}, Post_{3h}}{\triangleleft (s_{3h}) \triangleright \xrightarrow{\tau} \triangleleft (s_{3(h+1)}) \triangleright}$$

such that

$$s_i^{i\in L_Q} = s_{11} \wedge s_{1(k+1)i} = s_{21} \wedge s_{22} = s_{31} \wedge s_{3(k'+1)i} = {s'}_i^{i\in L_Q}$$

$$\alpha_Q = \alpha'_Q \{\!\{ \bigodot_{j=k}^{1} Post_{1j} \}\!\}$$

$$\overline{\gamma} = \bigoplus_{h=1}^{k} (\overline{\beta_{1h}} \{\!\{ \bigodot_{j=h-1}^{1} Post_{1j} \}\!\})^{\nabla} \oplus (\overline{\beta_2} \{\!\{ \bigodot_{j=k}^{1} Post_{1j} \}\!\})^{\nabla} \oplus$$
$$\bigoplus_{h=1}^{k'} (\overline{\beta_{3h}} \{\!\{ \bigodot_{j=h-1}^{1} Post_{3j} \odot Post_2 \odot \bigodot_{j=k}^{1} Post_{1j} \}\!\})^{\nabla}$$

$$Pred_Q = \bigwedge_{h=01}^{k} Pred_{1h} \{\!\{ \bigodot_{j=h-1}^{1} Post_{1j} \}\!\} \wedge Pred_2 \{\!\{ \bigodot_{h=k}^{1} Post_{1h} \}\!\} \wedge$$
$$\bigwedge_{h=1}^{k'} Pred_{3h} \{\!\{ \bigodot_{j=h-1}^{1} Post_{3j} \odot Post_2 \odot \bigodot_{h=k}^{1} Post_{1h} \}\!\}$$

$$Post_Q = \bigodot_{h=k'}^{1} Post_{3h} \odot Post_2 \odot \bigodot_{j=k}^{1} Post_{1j}$$

1. For the first k open tau transitions, by Definition 11 P can necessarily make a tau open transition if the hole indexed j_0 makes a tau action. So by Lemma 2 we obtain k open transitions in the form:
$$P[Q]_{j_0} \models \frac{\overline{\beta_{1h}}, Pred_{1h}, Post_{1h}}{\lhd s_{1h} \uplus s_i^{i \in L} \rhd \xrightarrow{\tau} \lhd s_{1(h+1)} \uplus s_i^{i \in L} \rhd}$$
2. For the possibly observable open transition. By Lemma 2 with the lemma hypotheses we obtain:
$$P[Q]_{j_0} \models \frac{{\beta_j}^{(j \in J \setminus \{j_0\})} \uplus \overline{\beta_2}, Pred \wedge Pred_2 \wedge \alpha_Q = \beta_{j_0}, Post \uplus Post_2}{\lhd s_i^{i \in L} \uplus s_{21} \rhd \xrightarrow{\alpha} \lhd {s'}_i^{i \in L} \uplus s_{22} \rhd}$$
3. We proceed in the same way as the first item for k' last weak open transitions, and we obtain k' open tau transitions.

Using Lemma 4, from cases (1), (2) and (3) we get:
$$P[Q]_{j_0} \models \frac{\overline{\gamma_c}, Pred_c, Post_c}{\lhd s_{11} \uplus s_i^{i \in L} \rhd \stackrel{\alpha'}{\Rightarrow} \lhd {s'}_i^{i \in L} \uplus s_{3(k'+1)} \rhd}$$

where $\alpha' = \alpha \{\!\{ \bigodot_{j=k}^{1} Post_{1j} \}\!\}$ and $\alpha = \alpha'$ because $Post_{1j}$ acts on variables of Q and α contains only variables of P.

$$\overline{\gamma_c} = \bigoplus_{h=1}^{k} (\overline{\beta_{1h}} \{\!\{ \bigodot_{i=h-1}^{1} Post_{1i} \}\!\})^{\nabla} \oplus (({\beta_j}^{(j \in J \setminus \{j_0\})} \uplus \overline{\beta_2}) \{\!\{ \bigodot_{i=k}^{1} Post_{1i} \}\!\})^{\nabla} \oplus$$
$$\bigoplus_{h=1}^{k'} (\overline{\beta_{3h}} \{\!\{ \bigodot_{i=h-1}^{1} Post_{3i} \odot Post_2 \odot \bigodot_{i=k}^{1} Post_{1i} \}\!\})^{\nabla}$$
$$= \overline{\gamma} \uplus ({\beta_j}^{\in J \setminus \{j_0\}})^{\nabla} \quad \text{because } Post_{1j} \text{ does not act on variables of } \beta_j.$$

$$Pred_c = \bigwedge_{h=1}^{k} Pred_{1h}\{\!\{ \bigodot_{i=h-1}^{1} Post_{1i} \}\!\} \wedge \big(Pred \wedge Pred_2 \wedge \alpha'_Q = \beta_{j_0}\big) \{\!\{ \bigodot_{i=k}^{1} Post_{1i} \}\!\} \wedge$$
$$\bigwedge_{h=1}^{k'} Pred_{3h}\{\!\{ \bigodot_{i=h-1}^{1} Post_{3i} \odot (Post \uplus Post_2) \odot \bigodot_{i=k}^{1} Post_{1i} \}\!\}$$
$$Post_c = (\bigodot_{i=k'}^{1} Post_{3i}) \odot (Post \uplus Post_2) \odot \bigodot_{i=k}^{1} Post_{1i}$$

Note that we have $s_i^{i\in L_Q} = s_{11} \wedge s_{1(k+1)i} = s_{21} \wedge s_{22} = s_{31} \wedge s_{3(k'+1)i} = {s'}_i^{i\in L_Q}$.

Note also that $Post$ only acts on variables of P while $Post_{1i}$ only acts on variables of Q. We conclude on predicate and posts as follows[18]:

$$\begin{aligned} Pred_c &= Pred_Q \wedge Pred \{\!\{ \bigodot_{i=k}^{1} Post_{1i} \}\!\} \wedge (\alpha'_Q = \beta_{j_0}) \{\!\{ \bigodot_{i=k}^{1} Post_{1i} \}\!\} \\ &= Pred_Q \wedge Pred \wedge \alpha_Q = \beta_{j_0} \\ Post_c &= Post \uplus Post_Q \end{aligned}$$

□

Lemma 8 (Weak open transition composition). *Suppose that we have a weak open automaton such that the WOTs cannot observe silent actions (see Definition 11). Suppose $j_0 \in J$ and $\gamma_{j_0} = [\beta_0..\beta_n]$ and additionally:*

$$P \models \frac{\gamma_j^{j\in J}, Pred, Post}{\lhd s_i^{i\in L} \rhd \overset{\alpha}{\Rightarrow} \lhd {s'}_i^{i\in L} \rhd} \quad \text{and } \forall p \in [1..n] \quad Q \models \frac{\gamma_{pj}^{j\in J_p}, Pred_p, Post_p}{\lhd s_{pi}^{i\in L_Q} \rhd \overset{\alpha_p}{\Rightarrow} \lhd s_{(p+1)i}^{i\in L_Q} \rhd}$$

Let

$$J_Q = \bigcup_{p=1}^{n} J_p \qquad \forall i \in L_Q.\, s_i = s_{0i} \quad \wedge \quad s'_i = s_{(n+1)i}$$

$$\forall j \in J_p,\, \gamma_j = \bigoplus_{p=1}^{n} \gamma_{pj} \{\!\{ \bigodot_{k=p}^{1} Post_k \}\!\}$$

$$Pred' = Pred \wedge \bigwedge_{p=1}^{n} (\alpha_p = \beta_p \wedge Pred_p) \{\!\{ \bigodot_{i=p-1}^{1} Post_i \}\!\} \qquad Post' = Post \uplus \bigodot_{p=n}^{1} Post_p$$

Then, we have:

$$P[Q]_{j_0} \models \frac{\gamma_j^{j\in(J\setminus\{j_0\})\uplus J_Q}, Pred', Post'}{\lhd s_i^{i\in L\uplus L_Q} \rhd \overset{\alpha}{\Rightarrow} \lhd {s'}_i^{i\in L\uplus L_Q} \rhd}$$

[18] $Post_{1i}$ only has an effect on variables of Q and thus does not modify $Pred$ or β_{j_0}.

Proof. Suppose we have:

$$P \models \frac{\gamma_j^{j\in J}, Pred, Post}{\lhd s_i^{i\in L} \rhd \overset{\alpha}{\Rightarrow} \lhd s_i'^{i\in L} \rhd}$$

By Lemma 4 this implies the following:

$$\forall p \in [1\,..\,m_1].P \models \frac{\beta_{1pj}^{j\in J_{1p}}, Pred_{1p}, Post_{1p}}{\lhd (s_{1pi})^{i\in L} \rhd \xrightarrow{\tau} \lhd (s_{1(p+1)i})^{i\in L} \rhd},$$

$$P \models \frac{\beta_{2j}^{j\in J_2}, Pred_2, Post_2}{\lhd (s_{21i})^{i\in L} \rhd \xrightarrow{\alpha'} \lhd (s_{22i})^{i\in L} \rhd}$$

and $$\forall p \in [1\,..\,m_2].P \models \frac{\beta_{3pj}^{j\in J_{3p}}, Pred_{3p}, Post_{3p}}{\lhd (s_{3pi})^{i\in L} \rhd \xrightarrow{\tau} \lhd (s_{3(p+1)i})^{i\in L} \rhd}$$

where:

$$\forall i \in L.\ \ s_i = s_{11i} \wedge s_{1(m_1+1)i} = s_{21i} \wedge s_{22i} = s_{31i} \wedge s_{3(m_2+1)i} = s_i'$$

$$\alpha = \alpha' \{\!\{ \bigodot_{j=m_1}^{1} Post_{1j} \}\!\}$$

$$\gamma_j^{j\in J} = \bigoplus_{i=1}^{m_1} (\beta_{1ij}^{j\in J_{1p}} \{\!\{ \bigodot_{k=i-1}^{1} Post_{1k} \}\!\})^{\nabla} \oplus (\beta_{2j}^{j\in J_2} \{\!\{ \bigodot_{k=m_1}^{10} Post_{1k} \}\!\})^{\nabla} \oplus \bigoplus_{i=1}^{m_2} (\beta_{3ij}^{j\in J_{3p}} \{\!\{ \bigodot_{k=i-1}^{1} Post_{3k} \odot Post_2 \odot \bigodot_{k=m_1}^{1} Post_{1k} \}\!\})^{\nabla}$$

$$Pred = \bigwedge_{p=1}^{m_1} \left(Pred_{1p} \{\!\{ \bigodot_{j=p-1}^{1} Post_{1j} \}\!\} \right) \wedge Pred_2 \{\!\{ \bigodot_{p=m_1}^{1} Post_{1p} \}\!\} \wedge \bigwedge_{p=1}^{m_2} Pred_{3p} \{\!\{ \bigodot_{j=p-1}^{1} Post_{3j} \odot Post_2 \odot \bigodot_{p=m_1}^{1} Post_{1p} \}\!\}$$

$$Post = \bigodot_{p=m_2}^{1} Post_{3p} \odot Post_2 \odot \bigodot_{j=m_1}^{1} Post_{1j}$$

Note that, for $l \in \{1, 3\}$ if $\beta_{lpj_0} = \tau$, then, because of Definition 11, P necessarily makes a τ open transition and remains in the same state, e.g. $s_{1pi} = s_{1(p+1)i}$. Thus without loss of generality, we can bypass such an open transition and obtain another decomposition of the WOT without the open transition that requires $\beta_{lpj_0} = \tau$. We can thus suppose that for all p and l we have $\beta_{lpj_0} \neq \tau$ or $j_0 \notin J_{1p}$. To avoid a special case, we suppose that the hole j_0 moves during the OT α', i.e. $\beta_{2j_0} = \beta_m$ for some m. Additionally, $\beta_m \neq \tau$, else we would have $\alpha = \alpha' = \tau$ and the α' OT could be also removed from the reduction, leading to a particular and simpler case.

We introduce $n_i^{i\in[1..m-1]}$, and $(n_i')^{i\in[m+1..n]}$ the indices of the steps in which the hole j_0 moves in the 3 sets of OTs above (β_m is the action that matches the hole j_0 in the OT α'), in other words, we have for all i, $\beta_{1n_ij_0}$ a visible action,

as additionally:

$$\begin{aligned}\gamma_{j_0} &= [\beta_0..\beta_n] \\ &= \bigoplus_{\substack{i=1 \\ j_0 \in J_{1i}}}^{m_1} (\beta_{1ij_0} \{\!\{ \bigodot_{k=i-1}^{1} Post_{1k} \}\!\})^{\nabla} \oplus (\beta_{2j_0} \{\!\{ \bigodot_{k=m_1}^{1} Post_{1k} \}\!\})^{\nabla} \oplus \\ &\qquad \bigoplus_{\substack{i=0 \\ j_0 \in J_{3i}}}^{m_2} (\beta_{3ij_0} \{\!\{ \bigodot_{k=i-1}^{1} Post_{3k} \odot Post_2 \odot \bigodot_{k=m_1}^{1} Post_{1k} \}\!\})^{\nabla}\end{aligned}$$

We have, by definition of n_i and n'_i:

$$\forall i \in [1\,..\,m-1], \beta_{1n_ij_0} \{\!\{ \bigodot_{k=n_i-1}^{1} Post_{1k} \}\!\} = \beta_i, \qquad \beta_{2j_0} \{\!\{ \bigodot_{k=m_1}^{1} Post_{1k} \}\!\} = \beta_m, \text{ and}$$

$$\forall i \in [m+1\,..\,n], \beta_{3n'_ij_0} \{\!\{ \bigodot_{k=n'_i-1}^{1} Post_{3k} \odot Post_2 \odot \bigodot_{k=m_1}^{1} Post_{1k} \}\!\} = \beta_i$$

Now, we compose OTs for each of the case above (depending on the OT of P):

1. For the first τ OTs, i.e. $p \in [1\,..\,m_1]$. We have:
   Either there is i such that $p = n_i$, and thus β_i and β_{1pj_0} are defined. In this case by Lemma 7, we have:

$$P[Q]_{j_0} \models \frac{\overline{\gamma'_{1p}}, Pred'_{1p}, Post'_{1p}}{\triangleleft s_{1pj}^{j\in L} \uplus s_{ij}^{j\in L_Q} \triangleright \overset{\tau}{\Rightarrow} \triangleleft s_{1(p+1)j}^{j\in L} \uplus s_{(i+1)j}^{j\in L_Q} \triangleright}$$

   with

$$\overline{\gamma'_{1p}} = \gamma_{ij}^{j\in J_i} \uplus (\beta_{1pj}^{j\in J_{1p}\setminus\{j_0\}})^{\nabla} \qquad Pred'_{1p} = Pred_{1p} \wedge (\beta_{1pj_0} = \alpha_i \wedge Pred_i)$$

$$Post'_{1p} = Post_{1p} \uplus Post_i$$

   Or $j_0 \notin \mathrm{dom}(\beta_{1p})$ and Q does not move in the composed reduction. In this case there is no i such that $p = n_i$, but there is i such that $p \in]n_i..n_{i+1}[$, and

$$P[Q]_{j_0} \models \frac{\overline{\beta_{1p}}, Pred_{1p}, Post_{1p}}{\triangleleft s_{1pj}^{j\in L} \uplus s_{ij}^{j\in L_Q} \triangleright \overset{\tau}{\rightarrow} \triangleleft s_{1(p+1)j}^{j\in L} \uplus s_{ij}^{j\in L_Q} \triangleright}$$

   and thus we also have a weak OT by Definition 13 (rule (**WT2**)):

$$P[Q]_{j_0} \models \frac{\overline{\gamma'_{1p}}, Pred'_{1p}, Post'_{1p}}{\triangleleft s_{1pj}^{j\in L} \uplus s_{ij}^{j\in L_Q} \triangleright \overset{\tau}{\Rightarrow} \triangleleft s_{1(p+1)j}^{j\in L} \uplus s_{ij}^{j\in L_Q} \triangleright}$$

   with $\overline{\gamma'_{1p}} = \overline{(\beta_{1p})^{\nabla}}, Pred'_{1p} = Pred_{1p}, Post'_{1p} = Post_{1p}$

2. Similarly, for the middle OT with label α:

$$P[Q]_{j_0} \models \frac{\overline{\gamma'_2}, Pred'_2, Post'_2}{\lhd (s_{21j})^{j\in L} \uplus (s_{mj})^{j\in L_Q} \rhd \overset{\alpha'}{\Rightarrow} \lhd (s_{22j})^{j\in L} \uplus (s_{(m+1)j})^{j\in L_Q} \rhd}$$

with

$$\overline{\gamma'_2} = \gamma_{mj}^{j\in J_m} \uplus (\beta_{2j}^{j\in J_2\setminus\{j_0\}})^{\nabla} \qquad Pred'_2 = Pred_2 \wedge (\beta_{2j_0} = \alpha_m \wedge Pred_m)$$

$$Post'_2 = Post_2 \uplus Post_m$$

3. For the last τ OTs, i.e. $p \in [1\,..\,m_2]$. We have similarly to the first case: Either there is i such that $p = n'_i$, and thus β_i and β_{1pj_0} are defined. In this case by Lemma 7, we have:

$$P[Q]_{j_0} \models \frac{\overline{\gamma'_{3p}}, Pred'_{3p}, Post'_{3p}}{\lhd s_{3pj}^{j\in L} \uplus s_{ij}^{j\in L_Q} \rhd \overset{\tau}{\Rightarrow} \lhd s_{3(p+1)j}^{j\in L} \uplus s_{(i+1)j}^{j\in L_Q} \rhd}$$

with

$$\overline{\gamma'_{3p}} = \gamma_{ij}^{j\in J_i} \uplus (\beta_{3pj}^{j\in J_{3p}\setminus\{j_0\}})^{\nabla} \qquad Pred'_{3p} = Pred_{3p} \wedge (\beta_{3pj_0} = \alpha_i \wedge Pred_i)$$

$$Post'_{3p} = Post_{3p} \uplus Post_i$$

Or $j_0 \notin \mathrm{dom}(\beta_{3p})$ and Q does not move in the composed reduction. In this case there is no i such that $p = n'_i$, but there is i such that $p \in]n'_i..n'_{i+1}[$, and

$$P[Q]_{j_0} \models \frac{\overline{\beta_{3p}}, Pred_{3p}, Post_{3p}}{\lhd s_{3pj}^{j\in L} \uplus s_{ij}^{j\in L_Q} \rhd \overset{\tau}{\rightarrow} \lhd s_{3(p+1)j}^{j\in L} \uplus s_{ij}^{j\in L_Q} \rhd}$$

and thus we also have a weak OT by definition 13 (rule **WT2**):

$$P[Q]_{j_0} \models \frac{\overline{\gamma'_{3p}}, Pred'_{3p}, Post'_{3p}}{\lhd s_{3pj}^{j\in L} \uplus s_{ij}^{j\in L_Q} \rhd \overset{\tau}{\Rightarrow} \lhd s_{3(p+1)j}^{j\in L} \uplus s_{ij}^{j\in L_Q} \rhd}$$

with $\overline{\gamma'_{3p}} = \overline{\beta_{3p}}, Pred'_{3p} = Pred_{3p}, Post'_{3p} = Post_{3p}$

By definition of weak open transition (Definition 13, rule **WT3**), we obtain:

$$P[Q]_{j_0} \models \frac{\overline{\gamma'}, Pred'', Post''}{\lhd s_{10j}^{j\in L} \uplus s_{0j}^{j\in L_Q} \rhd \overset{\alpha''}{\Rightarrow} \lhd s_{3(m_2+1)j}^{j\in L} \uplus s_{(n+1)j}^{j\in L_Q} \rhd}$$

where

$$\alpha'' = \alpha' \{\!\{ \bigodot_{j=m_1}^{1} Post'_{1j} \}\!\}$$

$$\overline{\gamma'} = \bigoplus_{i=1}^{m_1} \overline{\gamma'_{1i}} \{\!\{ \bigodot_{k=i-1}^{1} Post'_{1k} \}\!\} \oplus \overline{\gamma'_2} \{\!\{ \bigodot_{k=m_1}^{1} Post'_{1k} \}\!\} \oplus \bigoplus_{i=1}^{m_2} \overline{\gamma'_{3i}} \{\!\{ \bigodot_{k=i-1}^{1} Post'_{3k} \odot Post'_2 \odot \bigodot_{k=n}^{1} Post'_{1k} \}\!\}$$

$$Pred'' = \bigwedge_{i=1}^{m_1} Pred'_{1i} \{\!\{ \bigodot_{j=i-1}^{1} Post'_{1j} \}\!\} \wedge Pred'_2 \{\!\{ \bigodot_{j=m_1}^{1} Post'_{1j} \}\!\} \wedge \bigwedge_{i=1}^{m_2} Pred'_{3i} \{\!\{ \bigodot_{j=i-1}^{1} Post'_{3j} \odot Post'_2 \odot \bigodot_{j=m_2}^{1} Post'_{1j} \}\!\}$$

$$Post'' = \bigodot_{j=m_2}^{1} Post'_{3j} \odot Post'_2 \odot \bigodot_{j=m_1}^{1} Post'_{1j}$$

However it must be noticed that in steps 1 and 3, we have two kinds of WOTs with different signatures (depending on whether Q moves or not). It is still possible to glue them together in a global rule with two more terms for *Pred* and *Post* terms. This global merge is possible because the post-conditions of P only act on variables of P and those of Q on variables of Q (for example $Post_i$ has no effect on $Pred_{1p}$ and thus does not need to be taken into account when dealing with WOTs where Q does not move).

We now compare each element of the obtained WOT with the conclusion of the lemma:

$$\begin{aligned} \alpha'' &= \alpha' \{\!\{ \bigodot_{j=m_1}^{1} Post'_{1j} \}\!\} \\ &= \alpha' \{\!\{ \bigodot_{j=m_1}^{1} Post_{1j} \}\!\} \qquad \alpha' \text{ only contains variables of } P \text{ untouched by } Post_i \\ &= \alpha \end{aligned}$$

For $\overline{\gamma'}$ we distinguish elements in the holes of P and of Q.

First suppose $j \in J \setminus \{j_0\}$ we have $\gamma'_j = \gamma_j$ because $Post'_{ij}$ has no effect on variables of P and on β_{1pj}, consequently we have:

$$\gamma'_j = \bigoplus_{i=1}^{m_1} (\beta_{1ij} \{\!\{ \bigodot_{k=i-1}^{1} Post_{1k} \}\!\})^{\nabla} \oplus (\beta_{2j} \{\!\{ \bigodot_{k=m_1}^{1} Post_{1k} \}\!\})^{\nabla} \oplus \bigoplus_{i=1}^{m_2} (\beta_{3ij} \{\!\{ \bigodot_{k=i-1}^{1} Post_{3k} \odot Post_2 \odot \bigodot_{k=m_1}^{1} Post_{1k} \}\!\})^{\nabla}$$

Second, when $j \in J_t$ for some t, γ'_j is the concatenation of elements of γ'_{1ij}, γ'_{2j}, γ'_{3ij} that are not empty. By construction the concatenation of these elements is γ_{tj}, for $t \in [0..n]$. $Post_{ik}$ has no effect on γ_{tj} but $Post_k$ has. We obtain:

$$\begin{aligned}\gamma'_j =& \bigoplus_{i=1}^{m_1} \gamma'_{1ij} \{\!\{ \bigodot_{k=i-1}^{1} Post'_{1k} \}\!\} \oplus \gamma'_{2j} \{\!\{ \bigodot_{k=m_1}^{1} Post'_{1k} \}\!\} \oplus \\ & \bigoplus_{i=1}^{m_2} \gamma'_{3ij} \{\!\{ \bigodot_{k=i-1}^{1} Post'_{3k} \odot Post'_2 \odot \bigodot_{k=n}^{1} Post'_{1k} \}\!\} \\ =& \bigoplus_{t=1}^{n} \gamma_{tj} \{\!\{ \bigodot_{k=t-1}^{1} Post_k \}\!\}\end{aligned}$$

Concerning predicates, we also separate predicates on P from predicates on Q, and from the equality on the action filling the hole:

$$\begin{aligned}Pred'' =& \Big(\bigwedge_{i=1}^{m_1} Pred'_{1i} \{\!\{ \bigodot_{j=i-1}^{1} Post'_{1j} \}\!\} \wedge Pred'_2 \{\!\{ \bigodot_{j=m_1}^{1} Post'_{1j} \}\!\} \\ & \wedge \bigwedge_{i=1}^{m_3} Pred'_{3i} \{\!\{ \bigodot_{j=i-1}^{1} Post'_{3j} \odot Post'_2 \odot \bigodot_{j=m_1}^{1} Post'_{1j} \}\!\} \Big) \\ =& \Big(\bigwedge_{p=1}^{m_1} Pred_{1p} \{\!\{ \bigodot_{j=p-1}^{1} Post_{1j} \}\!\} \wedge Pred_2 \{\!\{ \bigodot_{p=m_1}^{1} Post_{1p} \}\!\} \wedge \bigwedge_{p=1}^{m_2} Pred_{3p} \{\!\{ \bigodot_{j=p-1}^{1} Post_{3j} \otimes Post_2 \otimes \bigodot_{p=m_1}^{1} Post_{1p} \}\!\} \Big) \\ & \wedge \bigwedge_{t=1}^{n} Pred_t \bigodot_{i=t-1}^{1} Post_i \wedge \Big(\bigwedge_{i=1}^{m-1} (\beta_{1 n_i j_0} = \alpha_i) \{\!\{ \bigodot_{j=n_i-1}^{1} Post'_{1j} \}\!\} \wedge (\beta_{2 j_0} = \alpha_m) \{\!\{ \bigodot_{j=m_1}^{1} Post'_{1j} \}\!\} \\ & \wedge \bigwedge_{i=m+1}^{n} (\beta_{3 n'_i j_0} = \alpha_i) \{\!\{ \bigodot_{j=n'_i-1}^{1} Post'_{3j} \odot Post'_2 \odot \bigodot_{j=m_1}^{1} Post'_{1j} \}\!\} \Big) \\ =& \Big(\bigwedge_{p=1}^{m_1} Pred_{1p} \{\!\{ \bigodot_{j=p-1}^{1} Post_{1j} \}\!\} \Big) \wedge Pred_2 \{\!\{ \bigodot_{p=m_1}^{1} Post_{1p} \}\!\} \wedge \bigwedge_{p=1}^{m_2} Pred_{3p} \{\!\{ \bigodot_{j=p-1}^{1} Post_{3j} \otimes Post_2 \otimes \bigodot_{p=m_1}^{1} Post_{1p} \}\!\} \Big) \\ & \wedge \bigwedge_{t=1}^{n} Pred_t \bigodot_{i=t-1}^{1} Post_i \wedge \Big(\bigwedge_{i=1}^{m-1} (\beta_i = \alpha_i) \{\!\{ \bigodot_{j=i-1}^{1} Post_j \}\!\} \wedge (\beta_m = \alpha_m) \{\!\{ \bigodot_{j=m}^{1} Post_j \}\!\} \\ & \wedge \bigwedge_{i=m+1}^{n} (\beta_i = \alpha_i) \{\!\{ \bigodot_{j=i-1}^{m} Post_j \odot Post_m \odot \bigodot_{j=m-1}^{1} Post_j \}\!\} \Big) \\ =& \, Pred\end{aligned}$$

Finally, concerning post-conditions:

$$\begin{aligned} Post'' &= \bigodot_{j=m_2}^{1} Post'_{3j} \odot Post'_2 \odot \bigodot_{j=m_1}^{1} Post'_{1j} \\ &= \left(\bigodot_{j=m_2}^{1} Post'_{3j} \odot Post'_2 \odot \bigodot_{j=m_1}^{1} Post'_{1j} \right) \uplus \bigodot_{j=n}^{1} Post_j \\ &= Post \uplus \bigodot_{j=n}^{1} Post_j \end{aligned}$$

This allows us to conclude concerning the lemma. □

Theorem 7. *Congruence.* Consider an open pNet: $P = \langle\!\langle P_i^{i\in I}, Sort_j^{j\in J}, \overline{SV} \rangle\!\rangle$. Let $j_0 \in J$ be a hole. Let Q and Q' be two weak FH-bisimilar pNets such that $\mathrm{Sort}(Q) = \mathrm{Sort}(Q') = Sort_{j_0}$. Then $P[Q]_{j_0}$ and $P[Q']_{j_0}$ are weak FH-bisimilar.

Proof. Consider Q weak FH-bisimilar to Q'. It means that there exists an FH-bisimulation $\mathcal{R}_{Q,Q'}$ relating the two pNets Q and Q'. We define a relation $\mathcal{R}$ relating states of $P[Q]_{j_0}$ with states of $P[Q']_{j_0}$:

$$\mathcal{R} = \{(\triangleleft S_P \uplus S_Q \triangleright, \triangleleft S_P \uplus S_{Q'} \triangleright, Pred_{Q,Q'}) | \, (S_Q, S_{Q'}, Pred_{Q,Q'}) \in \mathcal{R}_{Q,Q'}\}$$

To prove weak FH-bisimilarity of $P[Q]_{j_0}$ and $P[Q']_{j_0}$, we consider an open transition OT of $P[Q]_{j_0}$, and an equivalent state of $P[Q']_{j_0}$, and we try to find a family of WOT of $P[Q']_{j_0}$ that simulates OT. Consider an OT of $P[Q]_{j_0}$ it is of the form (notations introduced to prepare the decomposition):

$$P[Q]_{j_0} \models \frac{\beta_j^{j\in(J_P \uplus J_Q)}, Pred_P \wedge Pred_Q, Post_P \uplus Post_Q}{\triangleleft S_P \uplus S_Q \triangleright \xrightarrow{\alpha} \triangleleft S'_P \uplus S'_Q \triangleright}$$

By the decomposition lemma for OTs (Lemma 1), we obtain the 2 following OTs (equality side-conditions have been unlined for clarity):

$$P \models \frac{\beta_j^{j\in J_P} \uplus (j_0 \mapsto \alpha_Q), Pred_P, Post_Q}{\triangleleft S_P \triangleright \xrightarrow{\alpha} \triangleleft S'_P \triangleright} \quad \text{and} \quad Q \models \frac{\beta_j^{j\in J_Q}, Pred_Q, Post_Q}{\triangleleft S_Q \triangleright \xrightarrow{\alpha_Q} \triangleleft S'_Q \triangleright}$$

By definition of $\mathcal{R}$ we have $(S_Q, S_{Q'} | Pred_{Q,Q'}) \in \mathcal{R}_{Q,Q'}$. And thus, by definition of weak FH-bisimulation, there exists a family of weak open transitions WOT_x:

$$\frac{\gamma_{jx}^{j\in J_Q}, Pred_{Q'x}, Post_{Q'x}}{\triangleleft S_{Q'} \triangleright \overset{\alpha_x}{\Longrightarrow} \triangleleft S'_{Q'x} \triangleright}$$

where

$$\forall x. \, (S'_Q, S'_{Q'x} | Pred_{Q,Q'x}) \in \mathcal{R}_{Q,Q'}$$

and

$$Pred_{Q,Q'} \wedge Pred_Q \implies$$
$$\Big(\bigvee_{x \in X} (\forall j \in J_Q.\, (\beta_j)^\nabla = \gamma_{jx}) \Rightarrow (Pred_{Q_x} \wedge \alpha_Q = \alpha_x \wedge Pred_{Q,Q'x}\{\!\{Post_{Q'x} \uplus Post_Q\}\!\}) \Big)$$

Composing the OT of P with the WOTs of Q' by Lemma 7 we obtain:

$$P[Q']_{j_0} \models \frac{(\beta_j^{j \in J_P})^\nabla \uplus \gamma_{jx}^{j \in J_Q}, Pred_P \wedge Pred_{Q'x}, Post_P \uplus Post_{Q'x}}{\triangleleft S_P \uplus S_{Q'} \triangleright \overset{\alpha}{\Rightarrow} \triangleleft S'_P \uplus S'_{Q'x} \triangleright}$$

with $\bigvee_{x \in X} \big(\forall j \in J_Q.\, (\beta_j)^\nabla = \gamma_{jx} \implies \alpha_Q = \alpha_x\big)$ that ensures that the open transitions can be recomposed when the OT fires.

Side conditions necessary to prove weak-FH bisimulations are:

$$\forall x.\, (S'_P \uplus S'_Q, S'_P \uplus S'_{Q'x} | Pred_{Q,Q'x}) \in \mathcal{R}$$

which is true, and

$$Pred_{Q,Q'} \wedge Pred_P \wedge Pred_Q \implies$$
$$\Big(\bigvee_{x \in X} (\forall j \in J_Q.\, (\beta_j)^\nabla = \gamma_{jx} \wedge \forall j \in J_P.\, (\beta_j)^\nabla = (\beta_j)^\nabla) \Rightarrow$$
$$(Pred_P \wedge Pred_{Q'x} \wedge \alpha = \alpha \wedge Pred_{Q,Q'x}\{\!\{Post_P \uplus Post_{Q'x} \uplus Post_Q\}\!\}) \Big)$$

We conclude by observing that $Post_P$ has no effect on variables of Q and Q', and thus on $Pred_{Q,Q'x}$. □

Theorem 8. *Context equivalence.* Consider two FH-bisimilar open pNets: $P = \langle\!\langle P_i^{i \in I}, Sort_j^{j \in J}, \overline{SV} \rangle\!\rangle$ and $P' = \langle\!\langle {P'}_i^{i \in I}, Sort_j^{j \in J}, \overline{SV'} \rangle\!\rangle$ (recall they must have the same holes to be bisimilar). Let $j_0 \in J$ be a hole, and Q be a pNet such that $\mathrm{Sort}(Q) = Sort_{j_0}$. Then $P[Q]_{j_0}$ and $P'[Q]_{j_0}$ are FH-bisimilar.

Proof. Consider P weak FH-bisimilar to P'. There exists an FH-bisimulation $\mathcal{R}_{P,P'}$ relating P and P'. We define a relation $\mathcal{R}$ relating states of $P[Q]_{j_0}$ with states of $P'[Q]_{j_0}$:

$$\mathcal{R} = \{(\triangleleft S_P \uplus S_Q \triangleright, \triangleleft S_{P'} \uplus S_Q \triangleright, Pred_{P,P'}) |\, (S_P, S_{P'}, Pred_{P,P'}) \in \mathcal{R}_{P,P'}\}$$

To prove weak FH-bisimilarity of $P[Q]_{j_0}$ and $P'[Q]_{j_0}$, we consider an open transition OT of $P[Q]_{j_0}$, and an equivalent state of $P'[Q]_{j_0}$, and we try to find a family of WOT of $P'[Q]_{j_0}$ that simulates OT. Consider an OT of $P[Q]_{j_0}$ it is of the form (notations introduced to prepare the decomposition):

$$P[Q]_{j_0} \models \frac{\beta_j^{j \in (J_P \uplus J_Q)}, Pred_P \wedge Pred_Q \wedge Pred, Post_P \uplus Post_Q}{\triangleleft S_P \uplus S_Q \triangleright \overset{\alpha}{\rightarrow} \triangleleft S'_P \uplus S'_Q \triangleright}$$

By the decomposition lemma for OTs (Lemma 1), we obtain the 2 following OTs (equality side-conditions have been unlined for clarity):

$$P \models \frac{\beta_j^{j\in J_P} \uplus (j_0 \mapsto \alpha_Q), Pred_P, Post_Q}{\lhd S_P \rhd \xrightarrow{\alpha} \lhd S'_P \rhd} \quad \text{and} \quad Q \models \frac{\beta_j^{j\in J_Q}, Pred_Q, Post_Q}{\lhd S_Q \rhd \xrightarrow{\alpha_Q} \lhd S'_Q \rhd}$$

With $Pred \iff \alpha_Q = \beta_{j_0}$

By definition of $\mathcal{R}$ we have $(S_P, S_{P'}, Pred_{P,P'}) \in \mathcal{R}_{P,P'}$. And thus, by definition of weak FH-bisimulation, there exists a family of weak open transitions WOT_x:

$$\frac{\gamma_{jx}^{j\in J_P \uplus \{j_0\}}, Pred_{P'x}, Post_{P'x}}{\lhd S_{P'} \rhd \overset{\alpha_x}{\Longrightarrow} \lhd S'_{P'x} \rhd}$$

where

$$\forall x.\, (S'_P, S'_{P'x}, Pred_{P,P'x}) \in \mathcal{R}_{P,P'}$$

and

$$Pred_{P,P'} \wedge Pred_P \implies \Big(\bigvee_{x\in X} (\forall j \in J_P.\, (\beta_j)^\nabla = \gamma_{jx} \wedge (\alpha_Q)^\nabla = \gamma_{j_0}) \Rightarrow (Pred_{P'_x} \wedge \alpha = \alpha_x \wedge Pred_{P,P'x}\{\!\{Post_{P'x} \uplus Post_P\}\!\}) \Big)$$

We here need a special case of Lemma 8 where the inner pNet Q does a simple OT. This is just a particular case of the theorem but where notations get simplified because the inner pNet does a single transition. This way we can compose the WOTs of P' with the OT of Q and obtain, with $\gamma_{j_0} = [\beta]$:

$$P'[Q]_{j_0} \models \frac{(\beta_j^{j\in J_Q})^\nabla \uplus \gamma_{jx}^{j\in J_P}, Pred_{P'x} \wedge Pred_Q \wedge \alpha_Q = \beta, Post_{P'x} \uplus Post_Q}{\lhd S_{P'} \uplus S_Q \rhd \overset{\alpha_x}{\Longrightarrow} \lhd S'_{P'x} \uplus S'_Q \rhd}$$

Side conditions necessary to prove weak FH-bisimulations are:

$$\forall x.\, (S'_P \uplus S'_Q, S'_{P'x} \uplus S'_Q, Pred_{P,P'x}) \in \mathcal{R}$$

which is true, and

$$Pred_{P,P'} \wedge Pred_P \wedge Pred_Q Pred \implies \Big(\bigvee_{x\in X} (\forall j \in J_P.\, (\beta_j)^\nabla = \gamma_{jx} \wedge \forall j \in J_Q.\, (\beta_j)^\nabla = (\beta_j)^\nabla) \Rightarrow (Pred_{P'x} \wedge Pred_Q \wedge \alpha_Q = \beta \wedge \alpha_x = \alpha \wedge Pred_{P,P'x}\{\!\{Post_{P'x} \uplus Post_P \uplus Post_Q\}\!\}) \Big)$$

We conclude by observing that $Post_Q$ has no effect on variables of P and P', and thus on $Pred_{P,P'x}$ and $Pred$ leading to the conclusion about $\alpha_Q = \beta$. $\square$

Appendix C. Full details of the Simple Protocol Example

The first piece of code is the textual definition of the *SimpleProtocolSpec* pNet, that was drawn in Figure 2, page 14. This code should be intuitive enough to read, with the following language conventions, that brings some user-friendly features, mapped by the editor into pure pNet constructs.

- Constants of any type (including Action) must be declared as "const". They are used either as functions with argument, as typically `in(msg)`, or constants without argument, typically as "tau()".
- Variables can be declared as global variables of a pLTS (e.g. `m_msg` in `PerfectBuffer`), or a pNet Node in the case of synchronisation vector variables (e.g. `p_a`), or as input variables in a pLTS, as `?msg` in `PerfectBuffer`.
- The variables in the guards of synchronisation vectors (e.g. in SV1) do not need to be explicitly quantified: by convention, all variables in a guard that do not appear inside the vector actions will be recognised as bound by a *forall* quantifier inside the guard.
- The tools will check that everything is correctly declared, that variables are used properly and do not conflict between different objects, that vectors have coherent length, etc.

```
SimpleProtocolSpec:
import "Data_Alg.algp"
root SimpleProtocolSpec
const in, out:Action
const p_send, q_recv: Action
const tau:Action

pLTS PerfectBuffer
initial b0
vars ?m:Data
vars b_msg:Data b_ec:Nat

state b0
transition in(m) -> b1 {b_msg:=m, b_ec:=0}

state b1
transition out(b_msg, b_ec) -> b0
transition synchro(tau())  -> b1 {b_ec:=b_ec+1}

pNet SimpleProtocolSpec
holes P,Q
subnets P,PerfectBuffer,Q
vars p_a,q_b:Action m:Data ec:Nat

vector SV0 <p_send(m),in(m),_>->synchro(in(m))
vector SV1 <p_a,_,_>->p_a [p_a != p_send(x)]
vector SV2 <_,out(m,ec),q_recv(m,ec)>->synchro(out(m,ec))
vector SV3 <_,_,q_b>->q_a [q_b != q_recv(x,y)]
```

The corresponding generated Open Automaton was given in Figure 4, page 21.

Next is the code for the *SimpleProtocolImpl* pNet:

```
SimpleProtocolImpl:
  import "Data_Alg.algp"
  root SimpleProtocolImpl
const in,out:Action
const tau,p_send,q_recv,m_recv,m_send,m_error: Action
const s_recv,s_send,s_ack,s_error,r_recv,r_ack,r_send: Action

pLTS Sender
  initial s0
  vars ?m:Data
  vars  s_msg:Data s_ec:Nat
state s0
  transition s_recv(m) -> s1 {s_msg:=m, s_ec:=0}
state s1
  transition s_send(s_msg, s_ec) -> s2
state s2
  transition s_ack() -> s0
  transition s_error() -> s1 {s_ec:=s_ec+1}

pLTS Medium
  initial m0
  vars ?m:Data ?ec:Nat
  vars m_msg:Data m_ec:Nat
state m0
  transition m_recv(m,ec) -> m1 {m_msg:=m, m_ec:=ec}
state m1
  transition m_send(m_msg, m_ec) -> m0
  transition synchro(tau()) -> m2
state m2
  transition m_error() -> m0

pLTS Receiver
  initial r0
  vars ?m:Data ?ec:Nat
  vars r_msg:Data r_ec:Nat
state r0
  transition r_recv(m,ec) -> r1 {r_msg:=msg, r_ec:=ec}
state r1
  transition r_send(r_msg, r_ec) -> r2
state r2
  transition r_ack() -> r0

pNet SimpleProtocol
  subnets Sender,Medium,Receiver
  vars m:Data c:Nat
vector SV0 <s_recv(m),_, _>->in(m)
vector SV1 <s_send(m,ec),m_recv(m,ec),_>->synchro(tau())
vector SV2 <_,m_send(m,ec),r_recv(m,ec)>->synchro(tau())
vector SV3 <s_ack(),_,r_ack()>->synchro(tau())
vector SV4 <s_error(),m_error(),_>->synchro(tau())
vector SV5 <_,_,r_send(m,ec)>->out(m,ec)

pNet SimpleProtocolImpl
  holes P,Q
  subnets P,SimpleProtocol,Q
  vars p_a,q_a:Action  m:Data c:Nat
vector SV0 <p_send(m),in(m),_>->synchro(in(m))
vector SV1 <p_a,_,_>->p_a [p_a != p_send(x)]
vector SV2 <_,out(m,ec),q_recv(m,ec)>->synchro(out(m,ec))
vector SV3 <_,_,q_b>->q_b [q_b != q_recv(x,y)]
```

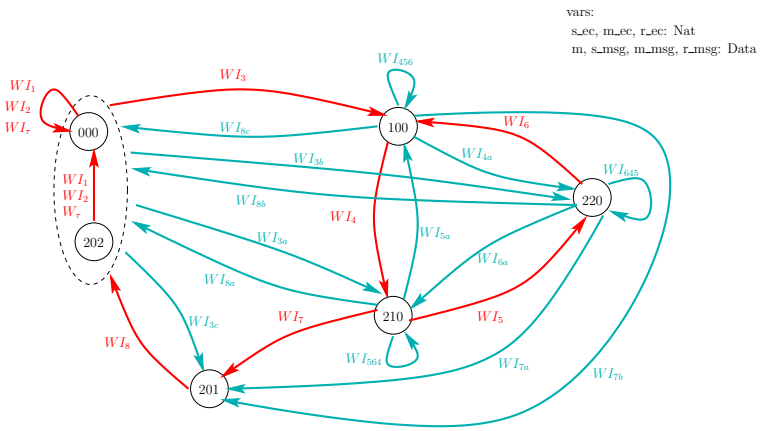


Figure C.11: Weak Open Automaton for *SimpleProtocolImpl*

In Figure C.11 we recall the weak open automaton of *SimpleProtocolImpl*. This drawing is based on the observation that states 202 and 000 are only linked by a "pure τ" transition, and have exactly the same possible behaviours. In this configuration we can guarantee that they are weak bisimilar, and we have merged their (incoming and outgoing) transitions in the figure. We denote this equivalence class of states as $\{000, 202\}$.

Full details of the weak transitions is listed here:

In the first 3 weak transitions, S denotes the set of all global states.

$$W_\tau = \frac{\{\}, True, ()}{S \overset{\tau}{\Rightarrow} S}$$

$$WI_1 = \frac{\{\texttt{P}\mapsto\texttt{p-a}\}, [\forall \texttt{x.p-a} \neq \texttt{p-send(x)}], ()}{S \overset{\texttt{p-a}}{\Longrightarrow} S}$$

$$WI_2 = \frac{\{\texttt{Q}\mapsto\texttt{q-b}\}, [\forall \texttt{x,y.q-b} \neq \texttt{q-recv(x,y)}], ()}{S \overset{\texttt{q-b}}{\Longrightarrow} S}$$

All the following transitions are parameterised by an integer $n \in \texttt{Nat}$, meaning they stand for the corresponding (infinite) set of weak OTs. In some cases, this set is further restricted (see e.g. $WI_{7b}(n)$), in which cases we have added an explicit quantifier.

$$WI_3(n) = \frac{\{\texttt{P}\mapsto\texttt{p-send(m)}\}, True, (\texttt{s\_msg} \leftarrow \texttt{m}, \texttt{s\_ec} \leftarrow n)}{\{000, 202\} \overset{\texttt{in(m)}}{\Longrightarrow} 100}$$

$$WI_{3a}(n) = \frac{\{\texttt{P}\mapsto\texttt{p-send(m)}\}, True, (\texttt{m\_msg} \leftarrow \texttt{m}, \texttt{m\_ec} \leftarrow n, \texttt{s\_ec} \leftarrow n)}{\{000, 202\} \overset{\texttt{in(m)}}{\Longrightarrow} 210}$$

$$WI_{3b}(n) = \frac{\{\texttt{P}\mapsto\texttt{p-send(m)}\}, True, (\texttt{s\_ec} \leftarrow n)}{\{000, 202\} \overset{\texttt{in(m)}}{\Longrightarrow} 220}$$

$$WI_{3c}(n) = \frac{\{\texttt{P}\mapsto\texttt{p-send(m)}\}, True, (\texttt{r\_msg} \leftarrow \texttt{m}, \texttt{r\_ec} \leftarrow n)}{\{000, 202\} \xRightarrow{\texttt{in(m)}} 201}$$

$$WI_4(n) = \frac{\{\}, True, (\texttt{m\_msg} \leftarrow \texttt{s\_msg}, \texttt{m\_ec} \leftarrow \texttt{s\_ec} + n, \texttt{s\_ec} \leftarrow \texttt{s\_ec} + n)}{100 \stackrel{\tau}{\Rightarrow} 210}$$

$$WI_{4a}(n) = \frac{\{\}, True, (\texttt{s\_ec} \leftarrow \texttt{s\_ec} + n)}{100 \stackrel{\tau}{\Rightarrow} 220}$$

$$WI_5(n) = \frac{\{\}, True, (\texttt{s\_ec} \leftarrow \texttt{s\_ec} + n)}{210 \stackrel{\tau}{\Rightarrow} 220}$$

$$WI_{5a}(n) = \frac{\{\}, True, (\texttt{s\_ec} \leftarrow \texttt{s\_ec} + 1 + n)}{210 \stackrel{\tau}{\Rightarrow} 100}$$

$$WI_6(n) = \frac{\{\}, True, (\texttt{s\_ec} \leftarrow \texttt{s\_ec} + 1 + n)}{220 \stackrel{\tau}{\Rightarrow} 100}$$

$$WI_{6a}(n) = \frac{\{\}, True, (\texttt{m\_msg} \leftarrow \texttt{s\_msg}, \texttt{m\_ec} \leftarrow \texttt{s\_ec} + 1 + n, \texttt{s\_ec} \leftarrow \texttt{s\_ec} + 1 + n)}{220 \stackrel{\tau}{\Rightarrow} 210}$$

Because

$$\begin{aligned} Post_{6a} = \ & post_4 \odot post^*_{456} \odot post_6 \\ = \ & ((\texttt{m\_msg} \leftarrow \texttt{s\_msg}, \texttt{m\_ec} \leftarrow \texttt{s\_ec}) \odot (\texttt{s\_ec} \leftarrow \texttt{s\_ec} + n)) \odot (\texttt{s\_ec} \leftarrow \texttt{s\_ec} + 1) \\ = \ & (\texttt{m\_msg} \leftarrow \texttt{s\_msg}, \texttt{m\_ec} \leftarrow (\texttt{s\_ec} + 1) + n, \texttt{s\_ec} \leftarrow (\texttt{s\_ec} + 1) + n) \end{aligned}$$

$$WI_{456*}(n) = \frac{\{\}, True, (\texttt{s\_ec} \leftarrow \texttt{s\_ec} + n)}{100 \stackrel{\tau}{\Rightarrow} 100}$$

$$WI_{564*}(n) = \frac{\{\}, True, (\texttt{m\_msg} \leftarrow \texttt{s\_msg}, \texttt{s\_ec} \leftarrow \texttt{s\_ec} + 1 + n, \texttt{m\_ec} \leftarrow \texttt{s\_ec} + 1 + n)}{210 \stackrel{\tau}{\Rightarrow} 210}$$

$$WI_{645*}(n) = \frac{\{\}, True, (\texttt{s\_ec} \leftarrow \texttt{s\_ec} + 1 + n)}{220 \stackrel{\tau}{\Rightarrow} 220}$$

$$WI_7(n) = \frac{\{\}, True, (\texttt{r\_msg} \leftarrow \texttt{s\_msg}, \texttt{r\_ec} \leftarrow \texttt{s\_ec} + n)}{210 \stackrel{\tau}{\Rightarrow} 201}$$

$$WI_{7a}(n) = \frac{\{\}, True, (\texttt{r\_msg} \leftarrow \texttt{s\_msg}, \texttt{r\_ec} \leftarrow \texttt{m\_ec} + n)}{220 \stackrel{\tau}{\Rightarrow} 201}$$

$$\forall n \geq 1. WI_{7b}(n) = \frac{\{\}, True, (\texttt{r\_msg} \leftarrow \texttt{m\_msg}, \texttt{r\_ec} \leftarrow \texttt{s\_ec} + n)}{100 \stackrel{\tau}{\Rightarrow} 201}$$

$$WI_8 = \frac{\{\texttt{Q}\mapsto\texttt{q-recv(r1-msg,r1-ec)}\}, True, ()}{201 \xRightarrow{\texttt{out(r1-msg,r1-ec)}} \{202, 000\}}$$

$$\forall n \geq 1. WI_{8a}(n) = \frac{\{\texttt{Q}\mapsto\texttt{q-recv(m\_msg,m\_ec} + n)\}, True, ()}{210 \xRightarrow{\texttt{out(m\_msg,m\_ec}+n\texttt{)}} \{202, 000\}}$$

$$\forall n \geq 1. WI_{8b}(n) = \frac{\{\texttt{Q}\mapsto\texttt{q-recv(s\_msg,s\_ec} + n)\}, True, ()}{220 \xRightarrow{\texttt{out(s\_msg,m\_ec}+n\texttt{)}} \{202, 000\}}$$

$$\forall n \geq 1. WI_{8c}(n) = \frac{\{\texttt{Q}\mapsto\texttt{q-recv(s\_msg,s\_ec}+n)\}, True, ()}{100 \xrightarrow{\texttt{out(s\_msg,s\_ec}+n)} \{202,000\}}$$

Then for all τ transitions above we have a similar WOT that include a non-τ move from an external action of P or Q, like for example:

$$WI_4P(n) = \frac{\{\texttt{P}\mapsto\texttt{p-a}\}, [\forall \texttt{x.p-a} \neq \texttt{p-send(x)}], (\texttt{m\_msg} \leftarrow \texttt{s\_msg}, \texttt{m\_ec} \leftarrow \texttt{s\_ec}+n, \texttt{s\_ec} \leftarrow \texttt{s\_ec}+n)}{100 \xrightarrow{\texttt{p-a}} 210}$$

$$\text{and } WI_4Q(n) = \frac{\{\texttt{Q}\mapsto\texttt{q-b}\}, [\forall \texttt{x,y.q-b} \neq \texttt{q-recv(x,y)}], (\texttt{m\_msg} \leftarrow \texttt{s\_msg}, \texttt{m\_ec} \leftarrow \texttt{s\_ec}+n, \texttt{s\_ec} \leftarrow \texttt{s\_ec}+n)}{100 \xrightarrow{\texttt{q-b}} 210}$$

but also e.g.:

$$WI_{456*}P(n) = \frac{\{\texttt{P}\mapsto\texttt{p-a}\}, [\forall \texttt{x.p-a} \neq \texttt{p-send(x)}], (\texttt{s\_msg} \leftarrow \texttt{s\_msg}, \texttt{s\_ec} \leftarrow \texttt{s\_ec}+n)}{100 \xrightarrow{\texttt{p-a}} 100}$$

The following table give a summary of WOTs, when sharing their names as much as possible.

WOT name	Pairs of source states and target states	# WOTs
WI_1 WI_2 WI_τ	$\{(s,s) \mid s \in \texttt{States of WOA}\} \cup \{(202,000)\}$	21
$WI_3(n)$	{(202,100),(000,100)}	2
$WI_{3a}(n)$	{(202,210),(000,210)}	2
$WI_{3b}(n)$	{(202,220),(000,220)}	2
$WI_{3c}(n)$	{(202,201),(000,201)}	2
$WI_4(n)$ $WI_4P(n)$ $WI_4Q(n)$	{(100,210)}	3
$WI_{4a}(n)$ $WI_4aP(n)$ $WI_4aQ(n)$	{(100,220)}	3
$WI_{456*}(n)$ $WI_{456*}P(n)$ $WI_{456*}Q(n)$	{(100,100)}	3
$WI_5(n)$ $WI_5P(n)$ $WI_5Q(n)$	{(210,220)}	3
$WI_{5a}(n)$ $WI_{5a}P(n)$ $WI_{5a}Q(n)$	{(210,100)}	3
$WI_{564*}(n)$ $WI_{564*}P(n)$ $WI_{564*}Q(n)$	{(210,210)}	3
$WI_6(n)$ $WI_6P(n)$ $WI_6Q(n)$	{(220,100)}	3
$WI_{6a}(n)$ $WI_{6a}P(n)$ $WI_{6a}Q(n)$	{(220,210)}	3
$WI_{645*}(n)$ $WI_{645*}P(n)$ $WI_{645*}Q(n)$	{(220,220)}	3
$WI_7(n)$ $WI_7P(n)$ $WI_7Q(n)$	{(210,201)}	3
$WI_{7a}(n)$ $WI_{7a}P(n)$ $WI_{7a}Q(n)$	{(220,201)}	3
$WI_{7b}(n)$ $WI_{7b}P(n)P$ $WI_{7b}Q(n)$	{(100,201)}	3
$WI_8(n)$	{(201,202),(201,000)}	2
$WI_{8a}(n)$	{(210,202),(210,000)}	2
$WI_{8b}(n)$	{(220,202),(220,000)}	2
$WI_{8c}(n)$	{(100,202),(100,000)}	2

That makes a total of 73 WOTs in the open automaton for *SimpleProtocolImpl*.

Appendix C.1. Details of the FH-bisimulation checking

We recall here the relation $\mathcal{R}$ that is the candidate for our weak FH-bisimulation relation:

SimpleProtocolSpec states	*SimpleProtocolImpl* states	Predicate
b0	000	True
b0	202	True
b1	100	`b_msg = s_msg` $\wedge$ `b_ec = s_ec`
b1	210	`b_msg = m_msg` $\wedge$ `b_ec = m_ec`
b1	220	`b_msg = s_msg` $\wedge$ `b_ec = s_ec`
b1	201	`b_msg = r_msg` $\wedge$ `b_ec = r_ec`

Consider the first triple <b0, 000, True>, we have to prove the following 6 properties, in which $OT << WOT$ means that the (strong) open transition OT is covered, in the sense of Definition 14, by the weak transition WOT (it could be a set, but this will not be used here):

$$SS_1 << WI_1 \qquad SI_1 << WS_1$$
$$SS_2 << WI_2 \qquad SI_2 << WS_2$$
$$SS_3 << WI_3 \qquad SI_3 << WS_3$$

Note that if we were using the alternative weak FH-bisimulation relation from Appendix B.1, Lemma 4, that is checking strong FH-bisimulation between the corresponding weak automaton, we would have a more transitions coverage to examine, as we have 4 weak transitions for $b0$ in the *SimpleProtocolSpec* weak automaton, and 7 WOTs (including 4 parameterised WOTs) from 000 in the *SimpleProtocolImpl* automaton.

Preliminary remarks:

- Both pNets trivially verify the "non-observability" condition: the only vectors having τ as an action of a sub-net are of the form "$< -, \tau, - >\rightarrow \tau$".
- We must take care of variable name conflicts: in our example, the variables of the 2 systems already have different names, but the action parameters occurring in the transitions (m, msg, ec) are the same, that is not correct. Recall that we disambiguate the reference to the variable m into $m1$ for *SimpleProtocolSpec* and $m2$ for *SimpleProtocolImpl*.

In our running example in page 34, we have shown the proof for one of the transitions of $(\text{b0}, 202, True)$, namely that SS_3 is covered by $WI_3(0)$. We give here another example with $SS_1 << WI_1$, from the first triple $(\text{b0}, 000, True)$. It includes less trivial predicates in the OTs:

$$SS_1 = \frac{\{\texttt{P}\mapsto\texttt{p-a1}\}, [\forall \texttt{m1.p-a1} \neq \texttt{p-send(m1)}], ()}{\text{b0} \xrightarrow{\texttt{p-a1}} \text{b0}}$$

$$WI_1 = \frac{\{\texttt{P}\mapsto\texttt{p-a2}\}, [\forall \texttt{m2.p-a2} \neq \texttt{p-send(m2)}], ()}{000 \overset{\texttt{p-a2}}{\Longrightarrow} 000}$$

Let us check formally the conditions:

- Their sets of active (non-silent) holes is the same: $J' = J_x = \{\texttt{P}\}$.
- Triple $(\text{b0}, 000, True)$ is in $\mathcal{R}$.
- The verification condition

$$\begin{aligned}\forall fv_{OT}.\{&Pred \wedge Pred_{OT} \implies \\ &\bigvee_{x\in X}\big[\exists fv_{OT_x}.\big(\forall j \in J_x.(\beta_j)^{\nabla} = \gamma_{jx} \wedge Pred_{OT_x} \wedge \alpha = \alpha_x \wedge \\ &\qquad Pred_{s',t_x}\{\!\{Post_{OT} \uplus Post_{OT_x}\}\!\}\big)\big]\}\end{aligned}$$

  Gives us:

  $\forall \texttt{p-a1}.\{True \wedge \forall \texttt{m1.p-a1} \neq \texttt{p-send(m1)}$
  $\implies \exists \texttt{p-a2}.(\texttt{p-a1} = \texttt{p-a2} \wedge \forall \texttt{m2.p-a2} \neq \texttt{p-send(m2)} \wedge \texttt{p-a1} = \texttt{p-a2} \wedge True\}$

  That is trivially true, choosing `p-a2=p-a1` for each given `p-a1`.

All others pairs from this set are just as easily proven true.