Compositional Equivalences Based on Open pNets
Rabéa Ameur-Boulifa, Ludovic Henrio, Eric Madelaine

To cite this version:

HAL Id: hal-03894031
https://hal.science/hal-03894031v2
Submitted on 12 Dec 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License
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 Méditerrannée, 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 $T$, and denote as $\Sigma$ the signature of the data and action constructors. Within $T$, we distinguish a set of data expressions $E$, including a set of boolean expressions $B$ ($B \subseteq E$), and a set of action expressions called the action algebra $A$, with $A \subseteq T, E \cap A = \emptyset$; naturally action terms will use data expressions as sub-terms$^1$. The function $\text{vars}(t)$ identifies the set of variables in a term $t \in T$.

$^1$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 \( a, b, \ldots \). 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 \( p\text{LTS} \) and are defined as follows:

\[
\begin{align*}
\alpha \in A & \quad ::= \quad a(p_1, \ldots, p_n) & \quad \text{action terms} \\
p_i & \quad ::= \quad ?x \mid e_i & \quad \text{parameters (input var or expression)} \\
e_i & \quad ::= \quad \text{Value} \mid x \mid \text{op}(e_1, \ldots, e_n) & \quad \text{Expressions}
\end{align*}
\]

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: \( p_i = ?x \Rightarrow \forall j \neq i. x \notin \text{vars}(p_j) \). 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 \( a(?x_1, \ldots, ?x_n) \) for inputs, \( a(v_1, \ldots, v_n) \) 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 \( A_S \), it will be used in synchronisation vectors of \( p\text{Nets} \):

\[
\alpha \in A_S \quad ::= \quad a(e_1, \ldots, e_n)
\]

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 \( p\text{Nets} \) 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: \( t^i \in I \) 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, \( t^i \in I \) defines first \( I \) the set over which the family is indexed, and then \( t_i \) the elements of the family. For example \( t^i \in (3) \) 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 \( t^i \in I \)” 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 “\( x \in A^i \in I \)” to mean \( \exists i \in I. x = A_i \). To simplify equations, an indexed set can be denoted \( \mathcal{I} \) instead of \( t^i \in I \) when \( I \) is irrelevant or clear from the context.
The disjoint union on sets is \( \sqcup \) and we only use \( A \sqcup B \) when \( A \) and \( B \) are disjoint. We extend it to union of indexed sets provided they are indexed over disjoint families; \( \sqcup \) 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 \( \text{dom}(A \setminus B) = \text{dom}(A) \setminus \text{dom}(B) \).

Substitutions. This article also uses substitutions. Applying a substitution inside a term 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 \( \text{Post} \) 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. These partial functions are sometimes called substitution functions in the following. Thus, \( \{ \{ \text{Post} \} \} \) is the operation that applies, in a parallel manner, the substitution defined by the partial function \( \text{Post} \). \( \circ \) is a composition operator on these partial functions, such that for any term \( t \) we have: \( t \{ \{ \text{Post} \circ \text{Post}' \} \} = (t \{ \{ \text{Post}' \} \}) \{ \{ \text{Post} \} \} \). This property must also be valid when the substitution does not operate on all variables. We thus define a composition operation as follows:

\[
(x_k \leftarrow e_k)^{k \in K} \circ (x'_{k'} \leftarrow e'_{k'})^{k' \in K'} = (x_k \leftarrow e_k \{ \{ x'_{k'} \leftarrow e'_{k'} \} \}^{k' \in K'})^{k \in K} \cup (x'_{k'} \leftarrow e'_{k'})^{k' \in K''}
\]

where \( K'' = \{ k' \in K' \mid x'_{k'} \notin \{ x_k \}^{k \in K} \} \)

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.

![Diagram](image)

It consists of one pNet (Definition 2 below) with two holes and a sub-net. 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 \( pLTS \equiv \langle S, s_0, V, \rightarrow \rangle \) 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 \( \alpha \in A \) is a parameterised action, \( e_b \in B \) is a guard, and the variables \( x_j \) (that are pairwise distinct) are assigned the expressions \( e_j \in E \). If \( s \xrightarrow{\langle \alpha, e_b, (x_j := e_j)^{j \in J} \rangle} s' \in \rightarrow \) then \( vars(\alpha) \setminus inv(\alpha) \subseteq V \), \( vars(e_b) \subseteq V \cup vars(\alpha) \), and \( \forall j \in J. \ (vars(e_j) \subseteq V \cup inv(\alpha) \land x_j \in V) \).

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 com-
position 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. Synchroni-
sation 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

\[
P \triangleq \text{pLTS} \mid \langle \langle P_i \in I, \text{Sort}_j \in J, SV_k \in K \rangle \rangle
\]

We denote \( \text{vars}(P) \) the set of variables used by the pLTSs inside \( P \) and \( \text{Sort}(P) \) 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 \( P_i \in I \) is the family of sub-pNets indexed over \( I \).
- \( J \) is a set of indices, called holes. \( I \) and \( J \) are disjoint: \( I \cap J = \emptyset, I \cup J \neq \emptyset \).
- \( \text{Sort}_j \subseteq A_S \) is a set of action terms, denoting the sort of hole \( j \).
- \( SV_k \) is a set of synchronisation vectors.

\[
\forall k \in K, SV_k = \alpha_i \in I_k \cup J_k \rightarrow \alpha'_k \in \text{Sort}_k \text{ where } \alpha'_k \in A_S, I_k \subseteq I, J_k \subseteq J,
\]
\[
\forall i \in I_k, \alpha_i \in \text{Sort}(P_i), \forall j \in J_k, \alpha_j \in \text{Sort}_j, \text{ and Sort}_{|\alpha_i|} \subseteq \bigcup_{l \in I_k \cup J_k} \text{vars}(\alpha_l).
\]

The global action of a vector \( SV_k\) is \( \alpha'_k \). \( e_k \in \mathbb{B} \) is a guard associated to the
vector such that \( \text{vars}(e_k) \subseteq \bigcup_{l \in I_k \cup J_k} \text{vars}(\alpha_l) \).

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 \( A_S \) it can
  perform, where each action signature is an action label plus the arity of
  the action.

\[
\text{Sort}(\langle \langle S, s_0, V, \rightarrow \rangle \rangle) = \{\text{Sort}(\alpha) | s \xrightarrow{\langle \alpha, e_0, (x_j := e_j) \rangle \in I} s' \in \rightarrow \}
\]

\[
\text{Sort}(\langle \langle P, \text{Sort}, SV \rangle \rangle) = \{\text{Sort}(\alpha') | \pi \rightarrow \alpha' \in \text{SV} \}
\]

\[
\text{Sort}(\alpha(p_1, ..., p_n)) = (\alpha, n)
\]
• The set of variables of a pNet $P$, denoted $\text{vars}(P)$ 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):

$$\text{Holes}(\llbracket S, s_0, V, \rightarrow \rrbracket) = \emptyset$$
$$\text{Holes}(\llbracket P_i \in I, \text{Sort} \in J, SV \rrbracket) = J \cup \bigcup_{i \in I} \text{Holes}(P_i)$$
$$\forall i \in I. \text{Holes}(P_i) \cap J = \emptyset$$
$$\forall i_1, i_2 \in I. i_1 \neq i_2 \Rightarrow \text{Holes}(P_{i_1}) \cap \text{Holes}(P_{i_2}) = \emptyset$$

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

$$\text{Leaves}(\llbracket S, s_0, V, \rightarrow \rrbracket) = \emptyset$$
$$\text{Leaves}(\llbracket I \in I, \text{Sort} \in J, SV \rrbracket) = \bigsqcup_{i \in I} \text{Leaves}(P_i) \cup \{i \mapsto P_i | P_i \text{ is a pLTS}\}$$

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 sort.

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 pLTSs 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

---

3A more complex compatibility relation could be defined, but this is out of the scope of this article.
operational semantics (SOS rules) 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: \( P = \langle \langle P_i \in I, \text{Sort}_j \in J, SV \rangle \rangle \) can be (partially) filled by providing a pNet \( Q \) to fill one of its holes. Suppose \( j_0 \in J \) and \( \text{Sort}(Q) \subseteq \text{Sort}_{j_0} \), then:

\[
P [Q]_{j_0} = \langle \langle P_i \in I \cup \{j_0 \rightarrow Q\}, \text{Sort}_j \in J \setminus \{j_0\}, SV \rangle \rangle
\]

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$(m), in(m), _ > → in(m)$ 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
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 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 $\text{Sort}_j \in J$, a set $V$ of variables, and a set of states $\mathcal{S}$ is a structure of the form:

$$
\beta_j \in J' \quad \text{Pred}, \text{Post} \\
\begin{array}{c}
\delta \\
\Rightarrow \\
\alpha \\
\end{array}
\quad s \xrightarrow{\alpha} s'
$$

where $J' \subseteq J$ is the set of holes involved in the transition; $s, s' \in \mathcal{S}$ are states of the automaton; and $\beta_j$ is a transition of the hole $j$, with $\text{Sort}(\beta_j) \subseteq \text{Sort}_j$. $\alpha$ is the resulting action of this open transition. $\text{Pred}$ is a predicate, $\text{Post}$ 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 $\beta_j$ and $\alpha$. 

Figure 1: pNet structure of the example and its specification expressed as a pNet called $\text{SimpleProtocolSpec}$.
The assignments are applied simultaneously because the variables in $V$ can be in both sides ($x_k$s 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 \( A = \langle J, S, s_0, V, T \rangle \) where:

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

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 \( \text{Pred} \) for each partial function \( \text{Post} \), if \( V \cap \text{dom}(\text{Post}) = \emptyset \), we have:

\[
\begin{align*}
\forall \alpha, \beta, \text{Pred}^{'}, \text{Post}^{'} \quad \overline{s, \text{Pred}^{'}, \text{Post}^{'}} \in T \\
\implies \overline{s \{ \text{Post} \}, \text{Pred} \{ \text{Post} \} \land \text{Pred}, \text{Post} \circ \text{Post}^{'}} \in T
\end{align*}
\]

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 \( \text{Pred} \) and \( \text{Pred}^{'} \) can be denoted \( \text{Pred} = \text{Pred}^{'} \), 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 \( \langle J, S_1, s_0, V_1, T_1 \rangle \) and \( \langle J, S_2, t_0, V_2, T_2 \rangle \) has the signature \( S_1 \times S_2 \rightarrow \mathbb{B} \). 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 \( R = \{(s, t|P_{s,t})\} \), where: 1) For any pair \((s, t) \in S_1 \times S_2\), there is a single \((s, t|P_{s,t}) \in R\) stating that \( s \) and \( t \) are related if \( P_{s,t} \) is True, i.e. the states are related when the value of the automata variables satisfy the predicate \( P_{s,t} \). 2) The free variables of \( P_{s,t} \) belong to \( V_1 \) and \( V_2 \), i.e. \( \text{vars}(P_{s,t}) \subseteq V_1 \cup V_2 \). FH-bisimulation is defined formally:

**Definition 7 (Strong FH-bisimulation).** Suppose \( A_1 = \langle J, S_1, s_0, V_1, T_1 \rangle \) and \( A_2 = \langle J, S_2, t_0, V_2, T_2 \rangle \) are open automata with identical holes of the same sort, with disjoint sets of variables \( (V_1 \cap V_2 = \emptyset) \).

Then \( R \) is an FH-bisimulation if and only if for all states \( s \in S_1 \) and \( t \in S_2 \), \((s, t|P_{s,t}) \in R\), we have the following:

- For any open transition \( OT \) in \( T_1 \):

\[\text{In this article, we denote } \beta_{jx} \text{ a double indexed set, instead of the classical } \beta_{j,x}. \text{ Indeed the standard notation would be too heavy in our case.} \]
there exists an indexed set of open transitions
\( OT_x \subseteq \mathcal{T}_2 \):}

\[
\beta_{j \in J_x}, \text{Pred}_{OT_x}, \text{Post}_{OT_x}
\]

such that \( J_x = J \) and there exists some \( \text{Pred}_{s', t} \) such that
\( (s', t | \text{Pred}_{s', t}) \in R \) and

\[
\text{Pred}_{s, t} \wedge \text{Pred}_{OT} = \implies \bigvee_{x \in X} (\forall j. \beta_j = \beta_{j \in J_x} \wedge \alpha = \alpha_x \wedge \text{Pred}_{s', t} \{\text{Post}_{OT} \psi \text{Post}_{OT_x}\})
\]

- 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, \( \text{Pred}_{s', t} \{\text{Post}_{OT} \psi \text{Post}_{OT_x}\} \) applies in parallel the substitution defined by the partial functions \( \text{Post}_{OT} \) and \( \text{Post}_{OT_x} \) (parallelism is crucial inside each \( \text{Post} \) set but not between \( \text{Post}_{OT} \) and \( \text{Post}_{OT_x} \) 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 \( R \) such that \( (s, t | \text{Pred}) \in R \), and additionally \( \text{Pred}' \Rightarrow \text{Pred} \) and it does not mean that there is a FH-bisimulation \( R' \) such that \( (s, t | \text{Pred}') \), indeed pair of states such that \( \text{Pred} \) is true and \( \text{Pred}' \) 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.

\[
\beta, \text{True}, \{y \leftarrow x\}
\]

Bisimulation should allow it to be matched by the two following ones (depending on the value of \( x \)), to prove that the relation \( R = \{(s_1, s_2, \text{True}), (s'_1, s'_2, \text{True})\} \) is a bisimulation.

\[
\beta, x \geq 0, \{y \leftarrow x\} \\
\beta, x < 0, \{y \leftarrow x\}
\]

\[
\begin{align*}
\beta, x \geq 0, \{y \leftarrow x\} & \quad \beta, x < 0, \{y \leftarrow x\} \\
S_2 \xrightarrow{\alpha(x)} S'_2 & \quad S_2 \xrightarrow{\alpha(x)} S'_2
\end{align*}
\]

\(^5\)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 \( \text{Pred}_{s,t} \land \text{Pred}_{\text{OT}} \) 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 \( R \) 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 \( A \). Then it is decidable whether the relation \( R \) 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 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) = \(\{\langle s_{i0}^L \rangle \mid \forall i \in L. s_i \in S_i\}\). A pLTS being its own single leave: States(\(\langle\langle S, s_0, V, \rightarrow\rangle\rangle\)) = \(\{s \mid s \in S\}\).

The initial state is defined as: \(\text{InitState}(P) = \langle s_{i0}^L \rangle\).

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 \(\text{Pred}_{sv}\) 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:

\[
\text{Pred}_{sv}\left(\left((\alpha'_i)^{i \in I}, (\beta'_j)^{j \in J} \rightarrow \alpha'[e_b]\right), \alpha_i^{i \in I}, \beta_j^{j \in J}, \alpha\right) \iff \\
\forall i \in I. \alpha_i = \alpha'_i \land \forall j \in J. \beta_j = \beta'_j \land \alpha = \alpha' \land e_b
\]

Somehow, this predicate entails a verification of satisfiability in the sense that if the predicate \(\text{Pred}_{sv}\) 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:
\[ s \xrightarrow{(\alpha, e_b, (x_j \leftarrow e_j)^{j \in J})} s' \xrightarrow{\emptyset, e_b, \{x_j \leftarrow e_j\}^{j \in J}} \]

\[ \langle S, s_0, \rightarrow \rangle \models \emptyset, e_b, \{x_j \leftarrow e_j\}^{j \in J} \]

\[ \langle s, s' \rangle \xrightarrow{\emptyset, e_b, \{x_j \leftarrow e_j\}^{j \in J}} \]

\[ \langle S, s_0, \rightarrow \rangle \models \emptyset, e_b, \{x_j \leftarrow e_j\}^{j \in J} \]

\[ Tr1 \]

and

\[ \text{Leaves}((\langle P_m \in I, \text{Sort}, SV_k \in K \rangle) = pLTSl_k \]

\[ SV_k = (\alpha'_m)^{m \in I} \xrightarrow{\bigcup_{m \in I} J_m \cup J} (s'_i) ^{i \in L} \]

\[ \forall m \in I_1, P_m \models \beta^j \in J_m, \text{Pred}_m, \text{Post}_m \]

\[ \langle s_i \xrightarrow{\alpha_m} (s'_i)^{i \in L} \rangle \]

\[ \forall m \in I_2, P_m \models \emptyset, \text{Pred}_m, \text{Post}_m \]

\[ \text{Pred} = \bigcup_{m \in I_1 \cup I_2} \text{Pred}_m \wedge \text{Pred}_{SV}(SV_k, \alpha'_m, \alpha^j \in J, \alpha) \]

\[ \forall i \in L \setminus \bigcup_{m \in I_1 \cup I_2} L_m \cup I_2 \]

\[ s'_i = s_i \quad \text{fresh}(\alpha'_m, \alpha^j \in J, \alpha) \]

\[ \langle \bigcup_{m \in I_1 \cup I_2} \text{Post}_m \rangle \]

\[ Tr2 \]

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

\[ OT_1 = \{ P \xrightarrow{p-\text{send}(m)}, [m \leftarrow m'], (s_{\text{msg}} \leftarrow m) \] 

\[ \langle s_0, m_0, r_0 \xrightarrow{\text{in}(m')} \rangle \xrightarrow{s_1, m_0, r_0} \]

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 \( s_{\text{msg}} \).

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 \( \alpha \), \( \text{fresh}(\alpha) \) means all variables in \( \alpha \) are fresh.

**Definition 9 (Semantics of open pNets).** The semantics of a pNet \( P \) is an open automaton \( A = \langle \text{Holes}(P), \text{States}(P), \text{InitState}(P), \text{vars}(P), T \rangle \) where \( T \) is the smallest set of open transitions such that \( T = \{ OT \mid P \models OT \} \) and \( P \models OT \) is defined by the rules in Figure 3.
The rule \textbf{Tr1} for a pLTS checks that the guard is verified and transforms assignments into post-conditions.

The rule \textbf{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 \textbf{Tr1}), and \(I_1\) for the sub-pNets that are not pLTSs (with open transitions obtained by rule \textbf{Tr2}), \(J\) is the set of holes involved in the transition.\footnote{Formally, if \(SV_k = (\alpha')_{m \in M} \rightarrow \alpha'_{\varepsilon_k}\) 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 \textbf{Tr2} but the rule would be more difficult to read.}

A key to understand \textbf{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 \textbf{Tr1} and \textbf{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].

![Diagram](image_url)

Figure 4: Open automaton for \textit{SimpleProtocolSpec}
Example

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

Figure 5 shows the open automaton of SimpleProtocolImpl from Figure 2. In this drawing, we have short labels for states, representing \( S_{000} \), \( M_{0} \), \( R_{0} \). Note that open transitions are denoted \( SI_i \) and tau open transition by \( SI_\tau \). 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 \( \tau \) 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 \( \tau \) actions (\( SI_4 \), \( SI_5 \) and \( SI_6 \)) 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 \( P[Q]_{000} \), 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. Let $\text{Leaves}(Q) = \{l_i \in L_Q\}$ and suppose:

$$P \mid_{Q_{j_0}} = \beta_{j_0} \in J \wedge \text{Pred}, \text{Post} \wedge \alpha_i \rightarrow s'_i \in L_Q$$

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 $\alpha_Q, \text{Pred}', \text{Post}'', \text{Post}''$ s.t.:

$$P \mid_{Q_{j_0}} = \beta_{j_0} \in J \setminus \text{Holes}(Q) \cup \{j_0\} \wedge \text{Pred}', \text{Post}'$$

and $Q \mid_{j} = \beta_{j} \in J \setminus \text{Holes}(Q) \wedge \text{Pred}'' \wedge \alpha_Q \rightarrow s'_i \in L_Q$.

Note that this does not mean that any two pNets can be composed and produce an open transition. Indeed, the predicate $\text{Pred} \wedge \text{Pred}' \wedge \alpha_Q = \beta_{j_0}$ is often not satisfiable, in particular if the action $\alpha_Q$ cannot be matched with $\beta_{j_0}$. Note also that $\beta_{j_0}$ 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.

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.
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
\[ P = \langle \langle P_i \in I, \text{Sort}_j \in J, SV \rangle \rangle. \]
Let \( j_0 \in J \) be a hole. Let \( Q \) and \( Q' \) be two FH-bisimilar pNets such that \( \text{Sort}(Q) = \text{Sort}(Q') = \text{Sort}_{j_0}. \) Then \( P[Q]_{j_0} \) and \( P[Q']_{j_0} \) are FH-bisimilar.

**Theorem 4 (Context equivalence).** Consider two open pNets
\[ P = \langle \langle P_i \in I, \text{Sort}_j \in J, SV \rangle \rangle \] and \( P' = \langle \langle P'_i \in I, \text{Sort}'_j \in J, SV' \rangle \rangle \) 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 \( \text{Sort}(Q) = \text{Sort}_{j_0}. \) 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 \( \mathbf{i} \) 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 \( \tau \) 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

---

8Note that \( \text{Sort}(Q) = \text{Sort}(Q') \) 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 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 \( A = \langle J, S, s_0, V, T \rangle \) cannot observe τ actions if and only if for all \( j \) in \( J \) and \( s \) in \( S \) we have:

1. \[
\begin{array}{c}
(j \mapsto \tau), \text{True}, \text{Id}(V) \\
\hline
s \xrightarrow{\tau} s
\end{array}
\in T
\]

and

2. for all \( \beta_j, J, \alpha, s, s', \text{Pred}, \text{Post} \) such that

\[
\begin{array}{c}
\beta_j \in J, \text{Pred}, \text{Post} \\
\hline
s \xrightarrow{\alpha} s'
\end{array}
\in T
\]

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

\[
\alpha = \tau \land s = s' \land \text{Pred} = \text{True} \land \text{Post} = \text{Id}(V) \land J = \{ j \}
\]

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 $\tau$ transitions too (because these $\tau$ actions cannot be observed). This possible synchronisation of $\tau$ actions would not be a problem as condition 1 still ensures that each process can do a $\tau$ 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 $\gamma$ 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 $i$, $\gamma_i \oplus \gamma'_{i}$ concatenates the sequences of actions at index $i$ of $\gamma_i$ and the one at index $i$ of $\gamma'_{i}$ $[a]$ 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 $(\beta)^\triangledown$ as the mapping $\beta$ with only observable actions of the holes in $I$, but where each element is either empty or a list of length 1:

$$(\beta|_{i \in I})^\triangledown = [eta_i]_{i \in I'}$$

where $I' = \{i | i \in I \land \beta_i \neq \tau\}$

As an example the $(\beta)^\triangledown$ built from the transition $OT_1$ in Example 3 page 19 is $p \rightarrow [p-send(m)]$. Remark that in our simple example no $\tau$ transition involves any visible action from a hole, so we have no $\beta$ 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 $\tau$ transitions of the holes before or after any open transition freely. This property justifies the fact that we can abstract away from $\tau$ 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 $Sort_j$ and a set of states $S$ is a structure of the form:

$$\gamma|_{j \in J}, Pred, Post$$

$$(\text{pred}, \text{post})$$

$s \buildrel \gamma \over \rightarrow s'$

---

9One of the two sequences is empty when $i \notin \text{dom}(\gamma_i)$ or $i \notin \text{dom}(\gamma_i)$. 26
\( s \xrightarrow{\tau} s \in \mathcal{WT} \)

**WT1**

\[
\begin{align*}
\exists \gamma_j \subseteq J, s, s' \in S \quad \text{and} \quad \gamma_j \text{ is a list of transitions of the hole } j, \text{ with each element of the list in } \text{Sort}_j. \quad \alpha \text{ is an action label denoting the resulting action of this open transition.} \\
\text{Pred and Post are defined similarly to Definition 5. We use } \mathcal{WT} \text{ to range over sets of weak open transitions.} \\
\text{A weak open automaton } \langle J, S, s_0, V, WT \rangle \text{ is similar to an open automaton except that } \mathcal{WT} \text{ is a set of weak open transitions over } J \text{ and } S. \\
\text{A weak open transition labelled } \alpha \text{ can be seen as a sequence of open transitions that are all labelled } \tau \text{ except one that is labelled } \alpha; \text{ however conditions on predicates, effects, and states must be verified for this sequence to be fired.} \\
\text{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 } \tau \text{ saturation: we add } \tau \text{ open transitions before or after another open transition, regardless of whether it is observable or not.} \\
\text{Definition 13 (Building a weak open automaton).} \\
\text{Let } A = \langle J, S, s_0, V, T \rangle \text{ be an open automaton. The weak open automaton derived from } A \text{ is an open automaton } \langle J, S, s_0, V, WT \rangle \text{ where } WT \text{ is derived from } T \text{ by saturation, applying the rules of Figure 6.} \\
\text{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 } \tau \text{ transitions before or after any weak open transition. This rule carefully composes predicates, effects, and actions of the holes. Indeed, predicate } \texttt{Pred}_2 \text{ manipulates variables of } s_1 \text{ that result from the first weak }
\end{align*}
\]

\[
\begin{align*}
\exists \gamma_j \subseteq J, s, s' \in S \quad \text{and} \quad \gamma_j \text{ is a list of transitions of the hole } j, \text{ with each element of the list in } \text{Sort}_j. \quad \alpha \text{ is an action label denoting the resulting action of this open transition.} \\
\text{Pred and Post are defined similarly to Definition 5. We use } \mathcal{WT} \text{ to range over sets of weak open transitions.} \\
\end{align*}
\]

\[
\begin{align*}
\exists \gamma_j \subseteq J, s, s' \in S \quad \text{and} \quad \gamma_j \text{ is a list of transitions of the hole } j, \text{ with each element of the list in } \text{Sort}_j. \quad \alpha \text{ is an action label denoting the resulting action of this open transition.} \\
\text{Pred and Post are defined similarly to Definition 5. We use } \mathcal{WT} \text{ to range over sets of weak open transitions.} \\
\end{align*}
\]
open transition. Their values thus depend on the initial state but also on the effect (as a substitution function $\text{Post}_1$) of the first weak open transition. In the same manner, $\text{Pred}_3$ must be applied the substitution defined by the composition $\text{Post}_2 \circ \text{Post}_1$. 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 $\{\}, \text{True}, ()$ as $W_T$. The weak open transition shown here is the transition delivering the result of the algorithm to hole $Q$ by applying rules: $\text{WT1}, \text{WT2},$ and $\text{WT3}$. First rule $\text{WT1}$ adds a $W_T$ loop on each state. Rule $\text{WT2}$ transforms each 2 OTs into WOTs. Then consider application of Rule $\text{WT3}$ on a sequence of 3 WOTs. The result will be: $\{\}, \text{True}, (b_{ec} \leftarrow b_{ec} + 2)$. We can iterate this construction an arbitrary number of times, getting for any natural number $n$ a weak open transition: $\emptyset, \text{True}, (\text{ec} \leftarrow \text{ec} + n)$. Finally, applying again $\text{WT3}$, and using the central open transition having $\text{out}(b_{msg}, b_{ec})$ as $\alpha$, we get the resulting weak open transition between $b_1$ and $b_0$ (as shown in Figure 7). Applying the substitutions finally yields the weak transitions family $\mathcal{WS}_7$ 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.

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 WI_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.
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:

\[
\begin{align*}
\text{post}_3 &= (s_{\text{msg}} \leftarrow m, s_{\text{ec}} \leftarrow 0) \\
\text{post}_4 &= (m_{\text{msg}} \leftarrow s_{\text{msg}}, m_{\text{ec}} \leftarrow s_{\text{ec}}) \\
\text{post}_5 &= () \\
\text{post}_6 &= (s_{\text{ec}} \leftarrow s_{\text{ec}} + 1)
\end{align*}
\]

Then the effect of one single weak open transition \( WI_3 \) is:

\[
\begin{align*}
\text{post}_{456} &= \text{post}_6 \circ \text{post}_5 \circ \text{post}_4 \equiv (s_{\text{ec}} \leftarrow s_{\text{ec}} + 1)
\end{align*}
\]

So if we denote \( post_{456+} \) any iteration of this loop, we get

\[
\begin{align*}
\text{post}_{456+} &= (s_{\text{ec}} \leftarrow s_{\text{ec}} + n) \quad \text{for any } n \geq 0,
\end{align*}
\]

and the \( Post \) of the weak OT \( WI_3 \) is:

\[
\begin{align*}
\text{Post}_3 &= post_{456+} \circ post_3 \equiv (s_{\text{msg}} \leftarrow m, s_{\text{ec}} \leftarrow n), \forall n \geq 0 \quad \text{and } \text{Post} \text{ of } WI_3
\end{align*}
\]

is:

\[
\begin{align*}
\text{post}_4 \circ \text{post}_{456+} \circ post_3 &= (m_{\text{msg}} \leftarrow m, m_{\text{ec}} \leftarrow n), \forall n \geq 0.
\end{align*}
\]

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 \( WI_3 \) when a silent action has 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 \( \text{Nat} \).

\[
\begin{align*}
WI_3 &= \{ (\forall x. p - a), [\forall x. p - a \neq p - send(x)], () \} \\
\text{sends} &\xrightarrow{\text{in}(m)} 100
\end{align*}
\]

\[
\begin{align*}
\forall n \geq 0. WI_3(n) &= \{ (\forall x. p - send(m)), \text{True}, (s_{\text{msg}} \leftarrow m, s_{\text{ec}} \leftarrow n) \} \\
&\xrightarrow{\text{in}(m)} 100 \\
\forall n \geq 0. WI_4(n) &= \{ \}, \text{True}, (s_{\text{msg}} \leftarrow m_{\text{msg}}, m_{\text{ec}} \leftarrow s_{\text{ec}} + n, s_{\text{ec}} \leftarrow s_{\text{ec}} + n) \\
&\xrightarrow{100} 210
\end{align*}
\]

\[
\begin{align*}
\forall n \geq 0. WI_{456}(n) &= \{ \}, \text{True}, (s_{\text{ec}} \leftarrow s_{\text{ec}} + n) \\
&\xrightarrow{100} 210
\end{align*}
\]

The \( Post \) of the weak OT \( WI_{6a} \) is:

\[
\begin{align*}
\text{Post}_{6a} &= post_4 \circ post_{456+} \circ post_6 \\
&= (m_{\text{msg}} \leftarrow s_{\text{msg}}, m_{\text{ec}} \leftarrow s_{\text{ec}}) \circ (s_{\text{ec}} \leftarrow s_{\text{ec}} + n) \circ (s_{\text{ec}} \leftarrow s_{\text{ec}} + 1) \\
&= (m_{\text{msg}} \leftarrow s_{\text{msg}}, m_{\text{ec}} \leftarrow s_{\text{ec}} + 1 + n, s_{\text{ec}} \leftarrow s_{\text{ec}} + 1 + n)
\end{align*}
\]

So we get:

\[
\begin{align*}
\forall n \geq 0. WI_{6a}(n) &= \{ \}, \text{True}, (s_{\text{ec}} \leftarrow s_{\text{ec}} + 1 + n, s_{\text{ec}} \leftarrow s_{\text{ec}} + 1 + n) \\
&\xrightarrow{220} 210
\end{align*}
\]

\[\text{Footnote}\]

\[\text{Footnote}\]
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 $A_1 = \langle J, S_1, s_0, V_1, T_1 \rangle$ and $A_2 = \langle J, S_2, t_0, V_2, T_2 \rangle$ be open automata with disjoint sets of variables. Let $\langle J, S_1, s_0, V_1, W T_1 \rangle$ and $\langle J, S_2, t_0, V_2, W T_2 \rangle$ be the weak open automata derived from $A_1$ and $A_2$ respectively. Let $R$ a relation over $S_1$ and $S_2$, as in Definition 7.

Then $R$ is a weak FH-bisimulation iff for any states $s \in S_1$ and $t \in S_2$ such that $(s, t | Pred_{s,t}) \in R$, we have the following:

- For any open transition $OT$ in $T_1$:

  \[ \forall j \in J', \, Pred_{OT}, \, Post_{OT} \]

  \[ s \overset{\alpha}{\rightarrow} s' \]

  there exists an indexed set of weak open transitions $WOT_{x \in X} \subseteq WT_2$:

  \[ \forall j \in J_x, \, Pred_{OT_x} \]

  \[ t \overset{\alpha}{\rightarrow} t_x \]
such that \( \forall x. \{ j \in J' \mid \beta_j \neq \tau \} = J_x, (s', t_x | \text{Pred}_{s', t_x}) \in \mathcal{R} \); and

\[
\text{Pred}_{s,t} \land \text{Pred}_{\text{OT}} \implies \\
\bigvee_{x \in X} \left( (\forall j \in J_x. (\beta_j)^x = \gamma_{jx} \land \text{Pred}_{\text{OT}x} \land \alpha = \alpha_x \land \text{Pred}_{s', t_x} \{ \text{Post}_{\text{OT} \cup Post_{\text{OT}x}} \}) \right)
\]

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

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:

\[
\forall f_{\text{OT}}. \left\{ \text{Pred}_{s,t} \land \text{Pred}_{\text{OT}} \implies \\
\bigvee_{x \in X} \left[ \exists f_{\text{OT}x}. (\forall j \in J_x. (\beta_j)^x = \gamma_{jx} \land \text{Pred}_{\text{OT}x} \land \alpha = \alpha_x \land \text{Pred}_{s', t_x} \{ \text{Post}_{\text{OT} \cup Post_{\text{OT}x}} \}) \right] \right\}
\]

where \( f_{\text{OT}} \) denotes the set of free variables of all expressions in \( \text{OT} \).
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 $\tau$ transition, and the global visible action is a $\tau$. 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 $\tau$ from a hole it cannot synchronise it with another action nor emit an action that is not $\tau$. This is formalised as follows:

**Definition 15 (Non-observability of silent actions for pNets).**
A pNet $\langle P_i \in I, \text{Sort}_j \in J, SV \rangle$ cannot observe silent actions if it verifies:

$\forall i \in I \uplus J. (i \rightarrow \tau) \rightarrow \tau[\text{True}] \in SV$ and

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

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 2, 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 $\langle J, S, s_0, V, WT \rangle$ 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 $WOT \in WT$.

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 theorems 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 $P = \langle\langle P_i \in I^i, \text{Sort}_j \in J^j, SV \rangle\rangle$. Let $j_0 \in J$ be a hole. Let $Q$ and $Q'$ be two weak FH-bisimilar pNets such that $\text{Sort}(Q) = \text{Sort}(Q') \subseteq \text{Sort}_{j_0}$. 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 $P = \langle\langle P_i \in I^i, \text{Sort}_j \in J^j, SV \rangle\rangle$ and $P' = \langle\langle P'_i \in I^i, \text{Sort}_j \in J^j, SV' \rangle\rangle$ 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 $\text{Sort}(Q) \subseteq \text{Sort}_{j_0}$. 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 $\tau$ then the $+$ operator can observe the $\tau$ 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 $\tau$ 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 $\tau$ as an action of a sub-net are of the form "$<-, \tau, -> \tau>"$.

\[11\] Note that $\text{Sort}(Q) = \text{Sort}(Q')$ is ensured by weak FH-bisimilarity.
<table>
<thead>
<tr>
<th>SimpleProtocolSpec states</th>
<th>SimpleProtocolImpl states</th>
<th>Predicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>b0</td>
<td>000</td>
<td>True</td>
</tr>
<tr>
<td>b0</td>
<td>202</td>
<td>True</td>
</tr>
<tr>
<td>b1</td>
<td>100</td>
<td>b_msg = s_msg ∧ b_ec = s_ec</td>
</tr>
<tr>
<td>b1</td>
<td>210</td>
<td>b_msg = m_msg ∧ b_ec = m_ec</td>
</tr>
<tr>
<td>b1</td>
<td>220</td>
<td>b_msg = s_msg ∧ b_ec = s_ec</td>
</tr>
<tr>
<td>b1</td>
<td>201</td>
<td>b_msg = r_msg ∧ b_ec = r_ec</td>
</tr>
</tbody>
</table>

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 \((m, \text{msg}, \text{ec})\) 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 \(\text{SimpleProtocolSpec}\), and \(m₂\) for \(\text{SimpleProtocolImpl}\).

Now consider the relation \(R\) defined by the triples in Table 1. Checking that \(R\) 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 \(SS₃\) from state \(b₀\). Its easy to guess that it will correspond to \(WI₃(0)\) of state 202 (and equivalently state 000, see Figure 9):

\[
SS₃ = \begin{array}{llll}
100 & \{P \rightarrow p\text{-send}(m₁)\}, True,(b\_msg \leftarrow m₁, b\_ec \leftarrow 0) \\
000 & \{P \rightarrow p\text{-send}(m₂)\}, True,(s\_msg \leftarrow m₂, s\_ec \leftarrow 0) \\
b₁ & \text{in}(m₁) \Rightarrow b₁ \\
000 & \Rightarrow 100
\end{array}
\]

Let us check formally the conditions:

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

\[
\forall f_{\text{OT}} . \{ \text{Pred} \land \text{Pred}_{\text{OT}} \} \implies \bigvee_{x \in \x} \exists f_{\text{OT}_x} . \\
(\forall j \in J_x . (\beta_j) = \gamma_{jx} \land \text{Pred}_{\text{OT}_x} \land \alpha = \alpha_x \land \text{Pred}_{\text{OT}_x} \land \text{Post}_{\text{OT}_x} \land \text{Post}_{\text{OT}_x})
\]

This gives us:

\[
\forall m₁ . \{ \text{True} \land \text{True} \} \implies \exists m₂ . \\
([p\text{-send}(m₁)] = [p\text{-send}(m₂)] \land \text{True} \land \text{in}(m₁) = \text{in}(m₂) \land
\]

35
\((b_{msg} = s_{msg} \land b_{ec} = s_{ec}) |\{(b_{msg} \leftarrow m_1, b_{ec} \leftarrow 0) \cup (s_{msg} \leftarrow m_2, s_{ec} \leftarrow 0)\}\}\)

That is reduced to:

\(\forall m_1.\exists m_2. (p\text{-send}(m_1) = p\text{-send}(m_2) \land in(m_1) = in(m_2) \land m_1 = m_2 \land 0 = 0)\)

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.

38
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 naïve 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


Appendix A. Proof on FH-bisimulation

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

Suppose \( R \) is an FH-bisimilarity. Then \( R \) is an equivalence, that is, \( R \) 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 \( T_1, T_2, T_3 \) and states \( s, t, u \) in those automata. Suppose we have \( R \) an FH-bisimulation relation between states of \( T_1 \) and of \( T_2 \); members of \( R \) are of the form \((s, t | \text{Pred}_{s,t})\). Suppose we also have \( R' \) an FH-bisimulation relation between states of \( T_2 \) and of \( T_3 \); members of \( R' \) are of the form \((t, u | \text{Pred}_{t,u})\).

Let \( R'' \) be the relation:

\[
R'' = \{(s, u | \text{Pred}_{s,u}) \mid \text{Pred}_{s,u} = \bigvee_{(s, t | \text{Pred}_{s,t}) \in R} \bigwedge_{(t, u | \text{Pred}_{t,u}) \in R'} \text{Pred}_{s,t} \land \text{Pred}_{t,u}\}
\]

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 \( R \) relates \( s \) and \( t \), and \( R' \) 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 \( R'' \) is an FH-bisimulation. Consider \((s, u | \text{Pred}_{s,u}) \in R''\). Then there is a set of states of \( T_2 \) relating \( s \) and \( u \), let \((t_p)_{p \in P} \) be this family. We have \( \text{Pred}_{s,u} = \bigvee_{p \in P} \text{Pred}_{s,p} \land \text{Pred}_{p,u} \).

For any \( p \in P \) by definition of \( R'' \), \((s, t_p | \text{Pred}_{s,p}) \in R \), and \((t_p, u | \text{Pred}'_{p,u}) \in R' \). We have the following by definition of bisimulation: For any open transition \( OT \) in \( T_1 \) originating from \( s \).

\[
\beta_{j \in J_1} \text{Pred}_j, \text{Post}_j \quad \alpha \rightarrow \beta_{j \in J_1} \text{Pred}_j, \text{Post}_j \quad \alpha \rightarrow \beta_{j \in J_1} \text{Pred}_j, \text{Post}_j
\]

---

124 We omit the constraints stating that each \( s, t, u \) is in the states of \( T_1, T_2, T_3 \) for the sake of readability.
There exists an indexed set of open transitions $OT_{px} \subseteq T_2$:

$$
\beta_j^{px}, \text{Pred}_{OT_{px}}, \text{Post}_{OT_{px}}, t_p \xrightarrow{\alpha_{px}} t_{px} \ 
\text{such that } \forall x, J_1 = J_{px}, (s', t_{px}|\text{Pred}_{px}) \in R; \text{ and }
$$

$$
\text{Pred}_{s,p} \land \text{Pred}_{OT} 
\implies \bigvee_{x \in X} \left( \forall j. \beta_j = \beta_{jpx} \land \text{Pred}_{OT_{px}} \land \alpha = \alpha_{px} \land \text{Pred}_{px} \left\{ [\text{Post}_{OT} \cup \text{Post}_{OT_{px}}] \right\} \right)
$$

For any open transition $OT_{px}$, since $(t_p, u|\text{Pred}'_{p,u}) \in R'$ there exists an indexed set of open transitions $OT_{pxy} \subseteq T_3$:

$$
\beta_j^{pxy}, \text{Pred}_{OT_{pxy}}, \text{Post}_{OT_{pxy}}, u \xrightarrow{\alpha_{pxy}} u_{pxy} \ 
\text{such that } \forall y, J_1 = J_{pxy}, (t_{px}, u_{pxy}|\text{Pred}_{pxy}) \in R'; \text{ and }
$$

$$
\text{Pred}'_{p,u} \land \text{Pred}_{OT_{px}} \implies \bigvee_{y \in Y} \left( \forall j. \beta_j = \beta_{jpxy} \land \text{Pred}_{OT_{pxy}} \land \alpha = \alpha_{pxy} \land \text{Pred}_{pxy} \left\{ [\text{Post}_{OT_{px}} \cup \text{Post}_{OT_{pxy}}] \right\} \right)
$$

This is verified for each $p \in P$.

Overall, we have a family of open transitions $OT_{pxy}^{p \in P, x \in X, y \in Y} \subseteq T_3$ that should simulate $OT$.

First, we have $\forall y, \forall x, \forall p, J_1 = J_{pxy}, (s', t_{pxy}|\text{Pred}_{pxy}) \in R''$ for some $\text{Pred}_{pxy}'$. Indeed for any $p$, $x$, and $y$, $t_{px}$ relates $s'$ and $u_{pxy}$, we have $(s', t_{px}|\text{Pred}_{px}) \in R$ and $(t_{px}, u_{pxy}|\text{Pred}_{pxy}) \in R'$. 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 $p$, $x$, $y$, $\text{Pred}_{px} \land \text{Pred}_{pxy} \implies \text{Pred}_{pxy}'$ (this is one element of the disjunction defining the predicate $\text{Pred}_{pxy}'$ relating $s'$ and $u_{pxy}$ in the definition of $R''$).

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 $\text{Pred}_{pxy}'$ does not contain the variables of $T_2$ because it is used to relate states of $T_1$ and $T_3$. Indeed if $\text{Pred}_{pxy}'$ uses variables of $T_2$, we can consider instead another predicate that is equivalent to $\text{Pred}_{pxy}'$ and does not contain the variables of $T_2$ (this is safe according to the semantic interpretation of open automata and relations). Similarly, we can
suppose that \( \text{Pred}_{px} \) contains no variable in \( T_3 \), and \( \text{Pred}_{pxy} \) contains no variable in \( T_1 \).

Second, by definition of bisimulation we need (recall that \( \text{Pred}_{s,u} \) is the original predicate relating \( s \) and \( u \) by definition of the transitive closure):

\[
\text{Pred}_{s,u} \land \text{Pred}_{OT} \implies \\
\bigvee_{x \in X} \bigvee_{y \in Y} \bigvee_{p \in P} \left( \forall j. \beta_j = \beta_{jpx} \land \text{Pred}_{pxy} \land \alpha = \alpha_{pxy} \land \text{Pred}'_{pxy} \langle \text{Post}_{OT} \cup \text{Post}_{OT_{pxy}} \rangle \right).
\]

From (*) and (**) we have:

for all \( p \), \( \text{Pred}_{s,p} \land \text{Pred}_{OT} \land \text{Pred}_{p,u} \)

\[
\implies \bigvee_{x \in X} \bigvee_{y \in Y} \bigvee_{p \in P} \left( \forall j. \beta_j = \beta_{jpx} \land (\text{Pred}_{OT_{px}} \land \text{Pred}_{p,u}) \land \alpha = \alpha_{px} \land \text{Pred}_{px} \langle \text{Post}_{OT} \cup \text{Post}_{OT_{px}} \rangle \right)
\]

\[
\implies \bigvee_{x \in X} \bigvee_{y \in Y} \bigvee_{p \in P} \left( \forall j. \beta_j = \beta_{jpx} \land \bigvee_{p \in P} \left( \forall j'. \beta_{j'px} = \beta_{j'pxy} \land \text{Pred}_{pxy} \land \alpha = \alpha_{pxy} \land \text{Pred}_{pxy} \langle \text{Post}_{OT} \cup \text{Post}_{OT_{pxy}} \rangle \right) \land \alpha = \alpha_{px} \land \text{Pred}_{px} \langle \text{Post}_{OT} \cup \text{Post}_{OT_{px}} \rangle \right)
\]

\[
\implies \bigvee_{x \in X} \bigvee_{y \in Y} \left( \forall j. \beta_j \land \beta_{jpx} \land \beta_{j'px} = \beta_{j'pxy} \land \text{Pred}_{pxy} \langle \text{Post}_{OT} \cup \text{Post}_{OT_{pxy}} \rangle \right)
\]

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:

\[
\text{Pred}_{px} \langle \text{Post}_{OT} \cup \text{Post}_{OT_{px}} \rangle \land \text{Pred}_{pxy} \langle \text{Post}_{OT} \cup \text{Post}_{OT_{pxy}} \rangle \iff \\
\text{Pred}_{px} \langle \text{Post}_{OT} \cup \text{Post}_{OT_{px}} \rangle \land \text{Pred}_{pxy} \langle \text{Post}_{OT} \cup \text{Post}_{OT_{pxy}} \rangle \iff \\
\text{Pred}_{pxy} \langle \text{Post}_{OT} \cup \text{Post}_{OT_{px}} \rangle \iff \\
\text{Pred}_{pxy} \langle \text{Post}_{OT} \cup \text{Post}_{OT_{px}} \rangle
\]

This allows us to conclude, with \( \text{Pred}_{s,u} = \bigvee_{p \in P} \text{Pred}_{s,p} \land \text{Pred}_{p,u} \):

\[
\text{Pred}_{s,u} \land \text{Pred}_{OT} \implies \bigvee_{p \in P} \left( \text{Pred}_{s,p} \land \text{Pred}_{p,u} \land \text{Pred}_{OT} \right)
\]

\[
\implies \bigvee_{p \in P} \bigvee_{x \in X} \bigvee_{y \in Y} \bigvee_{p \in P} \left( \forall j, j'. \beta_j = \beta_{jpx} \land \beta_{j'px} = \beta_{j'pxy} \land \text{Pred}_{pxy} \langle \text{Post}_{OT} \cup \text{Post}_{OT_{pxy}} \rangle \right)
\]

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.

\[\square\]
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 pLTS\textsuperscript{13}. Let $\text{Leaves}(Q) = P_{i \in L_Q}^e$; suppose:

$$P[Q]_{j_0} = \mid_{\begin{array}{c} \beta_{j}^{\exists J, \text{Pred}, \text{Post}} \\ \langle s^i_{l} \in L_Q \rangle \end{array}} \alpha_{\langle s^i_{l} \in L_Q \rangle}$$

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 $\alpha_Q, \text{Pred}', \text{Pred}'', \text{Post}', \text{Post}''$ s.t.:

$$P = \mid_{\begin{array}{c} \beta_{j}^{\exists J \setminus \text{Holes}(Q) \cup \{j_0\}, \text{Pred}', \text{Post}'} \\ \langle s^i_{l} \in L_Q \rangle \end{array}} \alpha_{\langle s^i_{l} \in L_Q \rangle}$$

and $Q = \mid_{\begin{array}{c} \beta_{j}^{\exists J \setminus \text{Holes}(Q), \text{Pred}'', \text{Post}''} \\ \langle s^i_{l} \in L_Q \rangle \end{array}} \alpha_{\langle s^i_{l} \in L_Q \rangle}$

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

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 $\beta_j, \nu$, and $\text{Pred}$.

Proof. Consider rule $\text{Tr2}$ in Definition $\text{I}$ applied to the pNet $P[Q]_{j_0}$.

Leaves($\langle P_{m \in I}, \text{Sort}, SV_{k \in K}^e \rangle$) = pLTS\textsuperscript{13} $k \in K$ $SV_k = (\alpha_m)_{m \in I_1 \equiv I_2 \equiv J} \rightarrow \alpha'_{[e_0]}$

$$\forall m \in I_1, \hspace{1em} P_{m} = \mid_{\begin{array}{c} \beta_{j}^{\exists J_m, \text{Pred}_m, \text{Post}_m} \\ \langle s^i_{l} \in L_m \rangle \end{array}} \alpha_{\langle s^i_{l} \in L_m \rangle} \hspace{1em} \forall m \in I_2, \hspace{1em} P_{m} = \mid_{\begin{array}{c} \emptyset, \text{Pred}_m, \text{Post}_m \\ \langle s^i_{l} \in L_m \rangle \end{array}} \alpha_{\langle s^i_{l} \in L_m \rangle}$$

$$J' = \bigcup_{m \in I_1} J_m \uplus J \hspace{1em} \text{Pred} = \bigcup_{m \in I_1 \equiv I_2} \text{Pred}_m \land \text{Pred}_m (SV_k, \alpha_m, \alpha'_m, \beta_{j}^{\exists J}, \alpha)$$

$$\bigcup_{m \in I_1 \equiv I_2} \text{Pred}_m \land \text{Pred}_m (SV_k, \alpha_m, \alpha'_m, \beta_{j}^{\exists J}, \alpha)$$

$$\forall i \in L \bigcup_{m \in I_1 \equiv I_2} \text{ Pred}_m (SV_k, \alpha_m, \alpha'_m, \beta_{j}^{\exists J}, \alpha)$$

$$\forall i \in L \bigcup_{m \in I_1 \equiv I_2} \text{ Pred}_m (SV_k, \alpha_m, \alpha'_m, \beta_{j}^{\exists J}, \alpha)$$

\text{Tr2}

$$\mid_{\begin{array}{c} \beta_{j}^{\exists J', \text{Pred}, \bigcup_{m \in I_1 \equiv I_2} \text{Post}_m} \\ \langle s^i_{l} \in L_Q \rangle \end{array}} \alpha_{\langle s^i_{l} \in L_Q \rangle}$$

\textsuperscript{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 \(\text{Leaves}(P[Q]_{j_0}) = \text{Leaves}(P) \cup \text{Leaves}(Q)\).

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

either \(m\) in \(I_1\)

\[
P_m \models \beta_j \in J_m \quad S^m_{\alpha}, \quad \text{Pred}_m \quad \text{Post}_m
\]

or, \(m\) in \(I_2\)

\[
P_m \models \emptyset, \quad \text{Pred}_m \quad \text{Post}_m
\]

Only elements of \((I_1 \cup I_2) \setminus \{j_0\}\) 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 \(J_{j_0} = J \cap \text{Holes}(Q)\)):

\[
Q \models \beta_j \in J \cap \text{Holes}(Q) \quad S^0_{\alpha}, \quad \text{Pred}_{j_0} \quad \text{Post}''
\]

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 \(\text{Pred}'' = \text{Pred}_{j_0}\). Considering the OT of \(P\) we have another \(J'\) that is \(J'_p = J' \setminus \text{Holes}(Q) \cup \{j_0\}\); we denote \(I'_1 = I_1 \setminus \{j_0\}\) the predicate is \(\text{Pred}' = \bigwedge_{m \in I'_1 \cup I_2} \text{Pred}_m \land \text{Pred}_{SV_k} (SV_k, \alpha_m \in \{i\}_w I_2, \beta_j \in J \cup \{j_0\}, \alpha)\)

where

\[
\text{Pred}_{SV_k} (SV_k, \alpha_m \in \{i\}_w I_2, \beta_j \in J \cup \{j_0\}, \alpha) \leftrightarrow \\
\forall i \in I'_1 \cup I_2, \alpha_i = \alpha'_i \land \forall j \in J \cup \{j_0\}, \beta_j = \alpha'_j \land \alpha = \alpha' \land e_0
\]

Modulo renaming of fresh variables, this is identical to the predicate that occurs in the source open transition except \(\alpha_{j_0} = \alpha'_0\) has been replaced by \(\beta_{j_0} = \alpha'_{j_0}\). As \(\alpha_{j_0} = \alpha_Q\) and \(\beta_{j_0}\) is free, we have \(\beta_{j_0} = \alpha'_0 \land \beta_{j_0} = \alpha_Q \leftrightarrow \alpha_{j_0} = \alpha_{j_0}\). Thus, \(\text{Pred}' \leftrightarrow (\text{Pred}' \land \text{Pred}'' \land \alpha_Q = \beta_{j_0})\). Finally, Post into conditions of the context \(P\) and the pNet \(Q\) (they are built similarly as they only deal with leaves): \(\text{Post} = \text{Post}' \lor \text{Post}''\). 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:
Leaves(\(\{P_m^{m\in I\setminus\{j_0\}}, SV_k^{k\in K}\}\)) = pLTS_{m\in L} k \in K \quad SV_k = (\alpha_m)^{m\in I_1 \cup I_2} \Rightarrow \alpha'_m\)

\(\forall m\in I_1 \setminus \{j_0\}, P_m = \beta_j^{\in J_m}, Pred_m, Post_m \quad \forall m\in I_2, P_m = \emptyset, Pred_m, Post_m \quad J' = \bigcup_{m\in I_1 \setminus \{j_0\}} J_m \uplus J \quad Pred' = \bigwedge_{m\in I_1 \cup I_2 \setminus \{j_0\}} Pred_m \land Pred_{sv}(SV_k, \alpha_m^{m\in I_1 \cup I_2 \setminus \{j_0\}}, \beta_j^{\in J \cup \{j_0\}}, \alpha) \quad \forall i \in L \setminus \left(\biguplus_{m\in I_1 \setminus \{j_0\}} L_m \uplus I_2\right), s'_i = s_i \quad fresh(\alpha'_m, \alpha', \beta_j, \alpha) \)

\(\langle P_m^{m\in I\setminus\{j_0\}}, SV_k^{k\in K} \rangle = \langle P_m^{m\in I\setminus\{j_0\}}, SV_k^{k\in K} \rangle \quad \alpha \quad \langle s'_i \in L \rangle \quad \alpha \quad s'_i \in L \quad \alpha \quad s'_i \in L \quad \alpha \quad s'_i \in L \)

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 \(\beta_{j_0}\). 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:

\[P = \beta_j^{\in J}, Pred, Post\]

\[Q = \beta_j^{\in J}, Pred', Post'\]

Then, we have:

\[P[Q]_{j_0} = \beta_j^{\in J \setminus \{j_0\} \cup Q}, Pred \land Pred' \land \alpha_Q = \beta_j, Post \uplus Post'\]

Note that this does not mean that any two pNets can be composed and produce an open transition. Indeed, the predicate \(Pred \land Pred' \land \alpha_Q = \beta_{j_0}\) will not be satisfiable if the action of \(\alpha_Q\) cannot be matched with \(\beta_{j_0}\). Note also that \(\beta_{j_0}\) 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 \( P = \langle P_m \in L, \overline{\text{Sort}}, S^k_v \subseteq K \rangle \). Consider first the open transition derived from \( P \). Consider each premise of the open transition (constructed by Tr2 rule in Definition 9).

Leaves(\( \langle P_m \in L, \overline{\text{Sort}}, S^k_v \subseteq K \rangle \) = \( pLTS_{L}^{k \in L} \quad k \in K \) \( SV_k = (\alpha_m')_{m \in I \rightarrow I_2 \cup J} \rightarrow \alpha'[c_b] \)

\[
\forall m \in I_1, P_m = \text{Pred}_m, \text{Post}_m \\
\forall m \in I_2, P_m = \emptyset, \text{Pred}_m, \text{Post}_m
\]

\[
J' = \bigcup_{m \in I_1} J_m \sqcup J \\
\text{Pred} = \bigwedge_{m \in I_1 \cup I_2} \text{Pred}_m \land \text{Pred}_{sv}(SV_k, \alpha_m \in I_1 \cup I_2, \beta_{j}^{\in J}, \alpha)
\]

\[
\forall i \in L \setminus \left( \bigcup_{m \in I_1} L_m \cup I_2 \right), s_i = s_i \quad \text{fresh}(\alpha_m', \beta_j', \alpha)
\]

\[
\langle P_m \in L, \overline{\text{Sort}}, S^k_v \subseteq K \rangle = \bigwedge_{m \in I_1 \cup I_2} \text{Pred}_m, \bigcup_{m \in I_1 \cup I_2} \text{Post}_m
\]

We know each premise is \( \text{True} \) 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): \( SV_k = (\alpha_j')_{j \in I \cup \{ j_0 \} \cup J} \rightarrow \alpha'[c_b] \). Leaves(\( P[Q]_{j_0} \) = Leaves(\( P \) \( \cup \) Leaves(\( Q \) ). \( I \) and \( J \) are the sets of leaves and holes of \( P \), \( I_1 \cup I_2 \) and \( J' \) are the sets of moving leaves and holes in the reduction of \( P \). All sub-pNets of must be reduced, we need:

\[
\forall m \in I_1 \cup \{ j_0 \}, P_m = \text{Pred}_m, \text{Post}_m \\
\forall m \in I_2, P_m = \emptyset, \text{Pred}_m, \text{Post}_m
\]

the sub-pNet at position \( j_0 \) is the one filled by \( Q \) (we define \( P_{j_0} = Q \) and similarly \( J_m = J_Q \), \( \text{Pred}_m = \text{Pred}' \), \( \text{Post}_m = \text{Post}' \) 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 \( j_0 \): \( J_{P,Q}' = J \setminus \{ j_0 \} \sqcup J_Q \). The predicate for the open transition is:

\[
\text{Pred}'' = \bigwedge_{m \in I_1 \cup I_2} \text{Pred}_m \land \text{Pred}' \land \text{Pred}(SV_k, v_i \in I \cup I_2 \cup (j_0 \rightarrow v_Q), \beta_{j}^{\in J}, \alpha)
\]

By definition we have:

\[
\text{Pred}(SV_k, \alpha_i \in I \cup (j_0 \rightarrow \alpha_Q), \beta_{j}^{\in J}, v) \leftrightarrow \forall i \in I. \alpha_i = \alpha'_i \land \forall j \in J. \beta_j = \beta'_j \land \alpha = \alpha' \land \alpha_Q = \beta_{j_0} \land c_b, \text{this is equivalent to} \forall i \in I. \alpha_i = \alpha'_i \land \forall j \in J. \beta_j = \beta'_j \land \alpha = \alpha' \land \alpha_Q = \beta_{j_0} \land \beta_{j_0} = \alpha_{j_0} \land c_b \text{ and by definition of Pred (as obtained by applying Tr2 rule), Pred''} \leftrightarrow \text{Pred } \land \text{Pred}' \land \alpha_Q = \beta_{j_0}. \text{ The post-condition gathers}
\]

51
Finally, the composed open transition can be built by the post-conditions related to all the leaves:

\[ \biguplus_{m \in I_1 \cup \{j_0\} \cup I_2} Post_m = Post \uplus Post'. \]

Finally, the composed open transition can be built by the post-conditions related to all the leaves:

\[ \biguplus_{m \in I_1 \cup \{j_0\} \cup I_2} Post_m = Post \uplus Post'. \]

This provides the desired conclusion. \(\square\)

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 \(L_Q = \text{Leaves}(Q)\):

\[
P \models_{\beta_j \in J^\prime \cup J} \text{Pred}, Post
\]

Then, for any state \(\langle s_i^L \rangle\) of \(Q\), we have:

\[
P[Q]_{j_0} \models_{\beta_j \in J^\prime \cup J} \text{Pred}, Post
\]

The proof is trivial.

**Appendix A.3. Proof of Theorem**

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

\[\text{Note that } \text{Sort}(Q) = \text{Sort}(Q') \text{ is ensured by strong bisimilarity.}\]
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 \( \text{Leaves}(Q) = p_i^{t_i \in L} \), \( \text{Leaves}(Q') = p_i^{t_i \in L'} \), \( \text{Leaves}(P) = p_i^{t_i \in L_P} \). Consider \( Q \) FH-bisimilar to \( Q' \). It means that there is a relation \( R \) that is an FH-bisimulation between the open automata of the two pNets. We will consider the relation \( \mathcal{R}' = \{(s, t | \text{Pred}_{s,t}) | s = s' \cup s'' \land t = t' \cup s'' \in S_P \land (s', t' | \text{Pred}_{s,t}) \in \mathcal{R} \} \) where \( S_P \) 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: \((q_i^{t_i \in L} \cup 0, q_i^{t_i \in L'} \cup 0) \in \mathcal{R} \). Consider an open transition \( OT \) of \( P[Q]_{j_0} \).

\[
P[Q]_{j_0} = \frac{P[Q][\beta_{j}^{l \in J} = J^{\text{Post}_{\text{OT}}}, Post_{\text{OT}}}_{\langle s_i \in L \cup L_{P} \rangle \overset{\alpha}{\rightarrow} \langle t_i \in L_{P} \rangle}
\]

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

\[
P[Q][\beta_{j}^{l \in J} = J^{\text{Post}'}, Post']_{\langle s_i \in L \cup L_{P} \rangle \overset{\alpha}{\rightarrow} \langle t_i \in L_{P} \rangle}
\]

and \( \text{Pred}_{\text{OT}} \iff \text{Pred}' \land \text{Pred}'' \land \alpha_{Q} = \beta_{j_0}, \) \( \text{Post}_{\text{OT}} = \text{Post}' \cup \text{Post}'' \) (\( \text{Post}'' \) is the restriction of \( \text{Post} \) over \( \text{vars}(Q) \)). As \( Q \) is FH-bisimilar to \( Q' \) and \((q_i^{t_i \in L} \cup 0, q_i^{t_i \in L'} \cup 0) \in \mathcal{R} \) there is a family \( OT'_{x} \) of open transitions of the automaton of \( Q' \) such that

\[
Q' = \frac{\beta_{j_{x}^{l \in J_{x} \setminus \text{Holes}(Q)}}^{l \in J_{x} \setminus \text{Holes}(Q)} = J_{x}^{\text{Post}_{\text{OT}_{x}}}, Post_{\text{OT}_{x}}}_{\langle q_i^{t_i \in L'} \cup 0 \rangle \overset{\alpha_{x}}{\rightarrow} \langle t_i^{l \in L}' \cup 0 \rangle}
\]

and \( \forall x, (q_i^{t_i \in L} \cup 0, q_i^{t_i \in L'} \cup 0) \in \mathcal{R} \); and \( \text{Pred}_{s,t} \land \text{Pred}'' \implies \bigvee_{x \in X} \left( \forall j \in J \cap \text{Holes}(Q), \beta_{j} = \beta_{j_{x}} \land \text{Pred}_{s,t} \land \alpha_{Q} = \alpha_{x} \land \text{Pred}_{s,t} \{ \text{Post}'' \cup \text{Post}_{\text{OT}_{x}} \} \right) \)

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 \( t_{ix} = s_{ix} \), and for \( j \in \text{Holes}(P), \beta_{jx} = \beta_{j} \)):

\[
P[Q]_{j_0} = \frac{P[Q][\beta_{j}^{l \in J} = J^{\text{Pred}' \land \text{Pred}_{\text{OT}_{x}} \land \alpha_{x} = \beta_{j_{0}x}, Post'_x \cup Post_{\text{OT}_{x}}}, Post_{\text{OT}_{x}}]}{\langle q_i^{t_i \in L} \cup 0 \rangle \overset{\alpha_{x}}{\rightarrow} \langle t_i^{l \in L} \cup 0 \rangle}
\]

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 OTx. ∀x, (\langle s, i \in L_p \rangle \Delta, \langle i \in L_p \rangle \Delta) | Pred_{s,x} \in R' (by definition of R') and in three steps we get:

\[
\text{Pred}_{s,t} \land \text{Pred}_{0} \implies \text{Pred}_{s,t} \land \text{Pred}' \land \text{Pred}'' \land \alpha Q = \beta_{0}\)
\[
\implies \bigvee_{x \in \chi} \left( \forall j \in J \cap \text{Holes}(Q), \beta_j = \beta_{0x} \land \text{Pred}''_{0} \land \alpha Q = \alpha_{x} \land \text{Pred}_{s,x} \{\langle \text{Post}'' \cup \text{Post}_{0}\rangle \} \right) \land \text{Pred}' \land \alpha Q = \beta_{0x} \land \text{Pred}_{s,x} \{\langle \text{Post}'' \cup \text{Post}_{0}\rangle \} \right)
\]

Note that, \(\beta_{0x}\) can be transformed into \(\beta_{j0x}\) 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 \(\text{Pred}_{s,x} \{\langle \text{Post}_{0} \cup (\text{Post}' \cup \text{Post}_{0})\rangle\}\), i.e. \(\text{Pred}_{s,x} \{\langle \text{Post}' \cup \text{Post}'' \rangle \cup (\text{Post}' \cup \text{Post}_{0})\}\). We can conclude by observing that \(\text{Pred}_{s,x}\) does not use any variable of \(P\) and thus the substitution \(\{\langle \text{Post}'\rangle\}\) has no effect on it.

Finally:

\[
\text{Pred}_{s,t} \land \text{Pred}_{0} \implies \bigvee_{x \in \chi} \left( \forall j \in J, \beta_j = \beta_{0x} \land \left( \text{Pred}' \land \text{Pred}_{0} \land \alpha Q = \beta_{0x} \land \alpha Q = \alpha_{x} \land \text{Pred}_{s,x} \{\langle \text{Post}'' \cup \text{Post}_{0}\rangle \} \right) \right)
\]

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

\[\square\]


Consider two FH-bisimilar open pNets: \(P = \langle P_{i}^{\in I}, Sort_{i}^{\in J}, SV_{i} \rangle\) and \(P' = \langle P_{i}^{\in I}, Sort_{i}^{\in J}, SV_{i} \rangle\) (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 \(\text{Sort}(Q) \subseteq \text{Sort}_{j_{0}}\). Then \(P[Q]_{j_{0}}\) and \(P'[Q]_{j_{0}}\) are FH-bisimilar.

\textbf{PROOF.} The proof of Theorem 4 exhibits a bisimulation relation for a composed system. It then uses Lemma 3 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 \(\text{Leaves}(Q) = P_{i}^{\in L_{Q}}, \text{Leaves}(P) = P_{i}^{\in L}, \text{Leaves}(P') = P_{i}^{\in L'}\). Consider \(P\) FH-bisimilar to \(P'\). It means that there is a relation \(R\) that is an FH-bisimulation between the open automata of the two pNets. We will consider the relation \(R' = \{(s,t) | \text{Pred}_{s,t} \} | s = s' \cup s'' \land t = t' \cup s'' \land s \in S_{Q} \land (s', t') | \text{Pred}_{s,t} \in R\}\) where \(S_{Q}\) is the set of states of the open automaton of \(Q\). We will prove
that \( R' \) is an open FH-bisimulation. Consider a pair of FH-bisimilar states: 
\[
\langle s_i^{LQ}, s_t^{L'} \rangle \cup \langle s_i^{LQ}, s_t^{L'} \rangle \models \text{Pred} \in R'.
\]
Consider an open transition \( OT \) of \( P[Q]_{j_0} \):
\[
P[Q]_{j_0} \models \frac{\beta_j \in J, \text{Pred}_{OT}, \text{Post}_{OT}}{\langle s_i^{LQ}, s_t^{L'} \rangle \models \text{Pred} \in R'}
\]
Let \( J' = J \setminus \text{Holes}(Q) \cup \{j_0\} \). By Lemma 1, we have:
\[
P \models \frac{\beta_j \in J', \text{Pred}', \text{Post}'}{\langle s_i^{L}, s_t^{L'} \rangle \models \text{Pred} \in R'}
\]
and \( \text{Pred}_{OT} \iff \text{Pred}' \land \text{Pred}'' \land \alpha_Q = \beta_{j_0}, \text{Post}_{OT} = \text{Post}' \cup \text{Post}'' \) (\( \text{Post}'' \) is the restriction of \( \text{Post} \) over \( \text{vars}(Q) \)). As \( P \) is FH-bisimilar to \( P' \) and \( \langle s_i^{L}, s_t^{L'} \rangle \models \text{Pred}_{s,t} \) then \( \text{Post}_{s,t} \) is an open FH-bisimulation. Consider a pair of FH-bisimilar states:
\[
\langle s_i^{L}, s_t^{L'} \rangle \models \text{Pred} \in R'.
\]
\[\text{Holes}(Q) \cup \{j_0\} \]
\[\text{Post}_{s,t} \]
\[\alpha = \alpha_Q = \beta_{j_0}, \text{Post}_{s,t} \]}
\]
We can now apply Lemma 2 on each of the \( \text{OT}' \) together with the transition of \( Q \) and obtain a new family \( \text{OT}' \) of open transitions (where for \( i \in L_Q, t_i = s_i \) and \( t_{ix} = s_{ix} \) and for \( j \in \text{Holes}(Q), b_j = b_j \)):
\[
P'[Q]_{j_0} \models \frac{\beta_j \in J, \text{Pred}_{OT}, \text{Post}_{OT}}{\langle t_i^{L'}, t_{ix}^{L'} \rangle \models \text{Pred} \in R'}
\]
Observe that \( J = (J \setminus \text{Holes}(Q) \cup \{j_0\}) \cup (\{j_0\}) \cup (J \cap \text{Holes}(Q)) \). Now we have to verify the conditions for the FH-bisimulation between \( OT \) and \( \text{OT}' \).
\[\forall x, \langle s_i^{L}, s_t^{L'} \rangle \models \text{Pred}_{s,x} \in R' \] (by definition of \( R' \)) and in four steps we get:
\[
\text{Pred}_{s,t} \land \text{Pred}_{OT} \implies \text{Pred}_{s,t} \land \text{Pred}' \land \text{Pred}'' \land \alpha_Q = \beta_{j_0}
\]
\[\implies \bigvee_{s \in X} \left( \forall j \in J'. \beta_j = \beta_{jx} \land \text{Pred}_{OT} \land \alpha_Q = \beta_{j_0} \land \alpha = \alpha_x \land \text{Pred}_{s,x} \right) \land \text{Post}' \lor \text{Post}_{OT} \}
\]
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 $\text{Pred}_{s,x}(\text{Post}_{OT} \cup (\text{Post}_{OT} \not\cup \text{Post}''))$, i.e., $\text{Pred}_{s,x}(\{\text{Post}' \not\cup \text{Post}''\}) \cup (\text{Post}_{OT} \not\cup \text{Post}'')$. We can conclude by observing that $\text{Pred}_{s,x}$ does not use any variable of $Q$ and thus the substitution involving $\text{Post}''$ has no effect.

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

\[\square\]

Appendix B. Weak FH-bisimilarity lemmas and proofs

We define a quantified composition operator for effects, i.e. $\text{Post}$ elements of the open transitions. We use $\otimes_{i=0}^{1} \text{Post}_{i}$ to denote $\text{Post}_{n} \odot \ldots \odot \text{Post}_{0}$. By convention $\otimes_{i=0}^{1} \text{Post}_{i}$ 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 $A = \langle J, S, s_{0}, V_{1}, T \rangle$ be an open automaton and $\langle J, S, s_{0}, V_{2}, W_{T} \rangle$ be the weak open automaton derived from $A$. The two following statements are equivalent

1. Either $\alpha = \tau \wedge \gamma = \emptyset \wedge \text{Pred} = \text{True} \wedge \text{Post} = \text{Id}(s) \wedge s = s'$; or
   there exist $\beta_{1i}, \beta_{2i},$ and $\beta_{3i}$, $\text{Pred}_{1i}, \text{Pred}_{3i}, \text{Post}_{1i}$, and $\text{Pred}_{2}, \text{Post}_{2}$, $n \geq 0, m \geq 0$ s.t.$^{15}$

\[
\forall i \in [1..n]. \overline{\beta_{1i}, \text{Pred}_{1i}, \text{Post}_{1i}}_{s_{1i}}^{s_{1(i+1)}} \in T \wedge \overline{\beta_{2i}, \text{Pred}_{2}, \text{Post}_{2}}_{s_{2i}}^{s_{2(i+1)}} \in T \wedge
\]

\[
\forall i \in [1..m]. \overline{\beta_{3i}, \text{Pred}_{3i}, \text{Post}_{3i}}_{s_{3i}}^{s_{3(i+1)}} \in T
\]

$^{15}n = 0$ (resp. $m = 0$) corresponds to the case where there is no $\tau$ transition before (resp. after) the transition $\alpha$. 

56
2. there exist \( \gamma, \text{Pred}, \text{Post} \) s.t.

\[
\begin{align*}
\gamma, \text{Pred}, \text{Post} & \in W\mathcal{T} \\
\alpha' & \Rightarrow s' \in \mathcal{T}
\end{align*}
\]

where

\[
\alpha' = \alpha \{ \{ \text{Post}_1 \} \}_{j=n}^1
\]

\[
s = s_{11} \land s_{1(n+1)} = s_2 \land s' = s_{31} \land s_{3(m+1)} = s'
\]

\[
\gamma = \bigoplus_{i=1}^m (\beta_{1i} \{ \{ \text{Post}_1 \} \} \gamma) \oplus (\beta_{2i} \{ \{ \text{Post}_1 \} \} \gamma) \oplus \\
\bigoplus_{i=0}^m (\beta_{3i} \{ \{ \text{Post}_2 \circ \text{Post}_3 \} \} \gamma)
\]

\[
\text{Pred} = \bigwedge_{i=1}^m (\text{Pred}_{1i} \{ \{ \text{Post}_1 \} \} \wedge \text{Pred}_{2i} \{ \{ \text{Post}_1 \} \} \wedge \\
\bigwedge_{i=1}^m (\text{Pred}_{3i} \{ \{ \text{Post}_2 \circ \text{Post}_3 \} \} \gamma))
\]

\[
\text{Post} = \bigodot_{j=n}^m \text{Post}_3 \circ \text{Post}_2 \circ \bigodot_{j=n}^1 \text{Post}_1
\]

**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:

\[
\begin{align*}
\beta, \text{Pred}, \text{Post} & \in \mathcal{T} \\
\alpha & \Rightarrow s' \in \mathcal{T}
\end{align*}
\]

by rule \(\text{WT2}\) we can directly conclude the implication:

\[
\begin{align*}
\beta, \text{Pred}, \text{Post} & \in \mathcal{T} \Rightarrow (\beta) \gamma, \text{Pred}, \text{Post} \\
\alpha & \Rightarrow s' \in W\mathcal{T}
\end{align*}
\]

- 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:

\[
\left( \forall i \in [1..n + 1]. \; \beta_{3i}^{1}, \text{Pred}_{3i}, \text{Post}_{3i} \right) \in \mathcal{T} \land \left( \forall i \in [1..m]. \; \beta_{3i}^{2}, \text{Pred}_{3i}, \text{Post}_{3i} \right) \in \mathcal{T}
\]

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

\[
\left( \forall i \in [2..n + 1]. \; \beta_{3i}^{1}, \text{Pred}_{3i}, \text{Post}_{3i} \right) \in \mathcal{T} \land \left( \forall i \in [1..m]. \; \beta_{3i}^{2}, \text{Pred}_{3i}, \text{Post}_{3i} \right) \in \mathcal{T} \Rightarrow s'' = s_1 \land s_3 = s' \in \mathcal{W}
\]

where

\[
s'' = s_1 \land s_3 = s_1 \land s_3 = s' = s''
\]

\[
\alpha' = \alpha \bigoplus_{j=n+1}^{2} \text{Post}_{1j}
\]

\[
\gamma = \bigoplus_{i=1}^{n+1} (\beta_{3i}^1 \bigoplus_{j=i-1}^{2} \text{Post}_{1j}) \oplus (\beta_{2i}^2 \bigoplus_{j=n+1}^{2} \text{Post}_{1j})
\]

\[
\bigoplus_{i=1}^{m} (\beta_{3i}^1 \bigoplus_{j=i-1}^{1} \text{Post}_{3j} \circ \text{Post}_{2j} \bigoplus_{j=n}^{1} \text{Post}_{1j})
\]

\[
\text{Pred} = \bigwedge_{i=2}^{n+1} \text{Pred}_{1i} \bigoplus_{j=i-1}^{2} \text{Post}_{1j} \land \bigwedge_{j=n+1}^{2} \text{Post}_{2j} \bigoplus_{j=n+1}^{2} \text{Post}_{1j}
\]

\[
\text{Post} = \bigoplus_{j=m}^{1} \text{Post}_{3j} \circ \text{Post}_{2j} \bigoplus_{j=n+1}^{2} \text{Post}_{1j}
\]

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

\[
\beta_{11}^{1}, \text{Pred}_{11}, \text{Post}_{11} \in \mathcal{T}
\]
First by using rule \textbf{WT2} we have:

\[
\begin{array}{c}
\frac{\overline{\beta}_{11}, \text{Pred}_{11}, \text{Post}_{11}}{s_1 \xrightarrow{s_2} s_2} \\
\overline{\beta}_{11} \in \mathcal{T} \Rightarrow (\overline{\beta}_{11})^\triangledown, \text{Pred}_{11}, \text{Post}_{11} \in \mathcal{WT}
\end{array}
\]

On the other hand, by rule \textbf{WT1} we have the following weak open transition:

\[
\begin{array}{c}
\frac{\emptyset, \text{True}, \text{Id}(s')}{s' \xrightarrow{s'} s'} \\
\emptyset \in \mathcal{WT}
\end{array}
\]

Finally by applying rule \textbf{WT3} on the above weak open transitions:

\[
\begin{array}{c}
\frac{(\overline{\beta}_{11})^\triangledown, \text{Pred}_{11}, \text{Post}_{11}}{s_1 \xrightarrow{s_2} s_2} \\
\overline{\beta}_{11}, \text{Pred}, \text{Post} \in \mathcal{WT} \Rightarrow (\overline{\beta}_{11})^\triangledown, \text{Pred}_{11}, \text{Post}_{11} \in \mathcal{WT}
\end{array}
\]

\[
\begin{array}{c}
\frac{\overline{\gamma}'', \text{Pred}'', \text{Post}''}{s_1 \xrightarrow{s'} s'} \\
\overline{\gamma}'' = (\overline{\beta}_{11})^\triangledown \oplus \overline{\beta}_{11} \in \mathcal{WT}
\end{array}
\]

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

\[
s_1 = s_1 \wedge s_{1(n+2)} = s_2 \wedge s'_2 = s_{31} \wedge s_{3(m+1)} = s'
\]

\[
\alpha'' = \alpha \left\{ \bigwedge_{j=n+1}^{m} \text{Post}_{1j} \right\} \{ \text{Post}_{11} \} = \alpha' \left\{ \bigwedge_{j=n+1}^{m} \text{Post}_{1j} \right\}
\]

\[
\overline{\gamma}'' = \bigoplus_{i=1}^{n} (\overline{\beta}_{1i} \left\{ \bigwedge_{j=i+1}^{m} \text{Post}_{ij} \right\})^\triangledown \oplus (\overline{\beta}_{2i} \left\{ \bigwedge_{j=i+1}^{m} \text{Post}_{ij} \right\})^\triangledown \oplus \bigoplus_{i=1}^{m} (\overline{\beta}_{3i} \left\{ \bigwedge_{j=i+1}^{m} \text{Post}_{ij} \right\})^\triangledown
\]

\[
\text{Pred} = \bigwedge_{i=1}^{m} \text{Pred}_{1i} \left\{ \bigwedge_{j=i+1}^{m} \text{Post}_{ij} \right\} \wedge \text{Pred}_{2i} \left\{ \bigwedge_{j=i+1}^{m} \text{Post}_{ij} \right\} \wedge \left( \bigwedge_{i=1}^{m} \text{Pred}_{3i} \left\{ \bigwedge_{j=i+1}^{m} \text{Post}_{ij} \right\} \right)
\]

\[
\text{Post} = \bigvee_{j=m}^{n+1} \text{Post}_{3j} \bigvee_{j=n+1}^{m} \text{Post}_{2j} \bigvee_{j=n+1}^{m} \text{Post}_{1j}
\]
The right part of the disjunction, i.e.

\[(\alpha = \tau \land \overline{\gamma} = \emptyset \land \text{Pred} = \text{True} \land \text{Post} = \text{Id}(s) \land s = s')\]

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:

  \[
  \emptyset, \text{True}, \text{Id}(s) \xrightarrow{s} s \in \mathcal{WT}
  \]

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

  \[
  \emptyset, \text{True}, \text{Id}(s) \xrightarrow{s} s \in \mathcal{WT} \Rightarrow (\alpha = \tau \land \overline{\gamma} = \emptyset \land \text{Pred} = \text{True} \land \text{Post} = \text{Id}(s) \land s = s')
  \]

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

  \[
  \overline{\gamma}, \text{Pred}, \text{Post} \xrightarrow{s} s' \in \mathcal{T} \Rightarrow \overline{\beta}, \text{Pred}, \text{Post} \xrightarrow{s} s'' \in \mathcal{T}
  \]

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

  These two cases above prove the implication with \(n = 0\) and \(m = 0\).

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

  \[
  \overline{\gamma_1}, \text{Pred}_1, \text{Post}_1 \xrightarrow{s} s' \in \mathcal{WT} \quad \overline{\gamma_2}, \text{Pred}_2, \text{Post}_2 \xrightarrow{s'} s'' \in \mathcal{WT} \quad \overline{\gamma_3}, \text{Pred}_3, \text{Post}_3 \xrightarrow{s''} s''' \in \mathcal{WT}
  \]

  \[
  \text{Pred} = \text{Pred}_1 \land \text{Pred}_2 \{\text{Post}_1\} \land \text{Pred}_3 \{\text{Post}_2 \odot \text{Post}_1\}
  \]

  \[
  \overline{\gamma} = \overline{\gamma_1} \oplus \overline{\gamma_2} \{\text{Post}_1\} \oplus \overline{\gamma_3} \{\text{Post}_2 \odot \text{Post}_1\}
  \]

  \[
  \alpha' = \alpha \{\text{Post}_1\}
  \]

  \[
  \overline{\gamma}, \text{Pred}, \text{Post}_3 \odot \text{Post}_2 \odot \text{Post}_1 \xrightarrow{s} s''' \in \mathcal{WT}
  \]

  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 \( \alpha = \tau \land \gamma = \emptyset \land \ldots \) but in this case, rule \( \text{WT1} \) allows us to obtain a similar reduction with \( n_1 = 1 \):

\[
\overline{s_1}, \text{Pred}_1, \text{Post}_1 \quad \overline{s} \Rightarrow s' \in \mathcal{WT} \Rightarrow \forall i \in [1..n_1], \overline{s_{1,i}}, \text{Pred}_{1,i}, \text{Post}_{1,i} \quad s_{1,i} \Rightarrow s_{1(i+1)}
\]

where

\[
s = s_{11} \land s_{1(n_1+1)} = s', \quad \overline{\gamma_1} = \bigoplus_{i=1}^{n_1} (\bigotimes_{j=i-1}^{1} \text{Post}_{1,j})
\]

\[
\text{Pred}_1 = \bigwedge_{i=1}^{n_1} (\text{Pred}_{1,i} \bigcirc \text{Post}_{1,i}), \quad \text{Post}_1 = \bigcup_{i=n_1}^{1} \text{Post}_{1,i}
\]

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

\[
\overline{s_3}, \text{Pred}_3, \text{Post}_3 \quad \overline{s''} \Rightarrow s''' \in \mathcal{WT} \Rightarrow \forall i \in [1..m_1], \overline{s_{3,i}}, \text{Pred}_{3,i}, \text{Post}_{3,i} \quad s_{3,i} \Rightarrow s_{3(i+1)}
\]

where

\[
s'' = s_{31} \land \ldots \land s_{3(m_1+1)} = s''' \land \overline{\gamma_3} = \bigoplus_{i=1}^{m_1} (\bigotimes_{j=i-1}^{1} \text{Post}_{3,j})
\]

\[
\text{Pred}_3 = \bigwedge_{i=1}^{m_1} (\text{Pred}_{3,i} \bigcirc \text{Post}_{3,i}) \land \text{Post}_3 = \bigcup_{i=m_1}^{1} \text{Post}_{3,i}
\]

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

\[
\overline{s_2}, \text{Pred}_2, \text{Post}_2 \quad \overline{s'} \Rightarrow s'' \in \mathcal{WT} \Rightarrow (\forall i \in [1..n_2], \overline{s_{2,i}}, \text{Pred}_{2,i}, \text{Post}_{2,i} \quad s_{2,i} \Rightarrow s_{2(i+1)}
\]

\[
\overline{s_2}, \text{Pred}', \text{Post}' \quad \overline{s_2'} \Rightarrow s_2'' \in \mathcal{T} \land \forall i \in [1..m_2], \overline{s_{2,i}}, \text{Pred}_{2,i}', \text{Post}_{2,i}' \quad s_{2,i} \Rightarrow s_{2(i+1)}
\]
where
\[ s' = s_{21} \land s_{2(n_2+1)} = s_2 \land s_{2}^\prime = s_{21} \land s_{2(m_2+1)} = s'' \]
\[ \alpha'' = \alpha' \left( \bigcap_{j=n_2}^{1} \text{Post}_{2j} \right) \]
\[ \overline{\gamma}_2 = \bigoplus_{i=1}^{n_2} \left( \bigcap_{j=i-1}^{1} \text{Post}_{2j} \right) \bigoplus \left( \left( \bigcap_{j=i-1}^{1} \text{Post}_{2j} \right) \bigcap \left( \bigcap_{j=n_2}^{1} \text{Post}_{2j} \right) \right) \]
\[ \text{Pred}_2 = \bigwedge_{i=1}^{n_2} \left( \bigcap_{j=i-1}^{1} \text{Pred}_{2i} \right) \bigwedge \left( \bigcap_{j=n_2}^{1} \text{Pred}_{2j} \right) \]
\[ \text{Post}_2 = \bigcap_{j=m_2}^{1} \text{Post}_{2j} \bigcap \left( \bigcap_{j=n_2}^{1} \text{Post}_{2j} \right) \]

Therefore, we can deduce that we have:

\[ \pi, \text{Pred, Post} \rightarrow s \in \mathcal{W}T \Rightarrow \left( \forall i \in [1..(n_1+n_2)]. \right) \]

\[ \overline{s}_{4i} \rightarrow s_{4i} \]

\[ \overline{s}_{5i} \rightarrow s_{5(i+1)} \]

such that
\[ s_{4i} = \begin{cases} 
  s_{1i} & \text{if } i \leq n_1 \\
  s_{2i-n_1} & \text{if } i > n_1 
\end{cases} \]

\[ s_{5i} = \begin{cases} 
  s_{3i} & \text{if } i \leq m_1 \\
  s_{2i-m_1} & \text{if } i > m_1 
\end{cases} \]

and similarly for \( \text{Pred}_{4i}, \text{Pred}_{5i}, \text{Post}_{4i}, \) and \( \text{Post}_{5i}. \)
Lemma 5 (Alternative definition of weak bisimulation).

The definition of weak bisimulation given in Definition \[\text{14}\] is equivalent to the following one:

Let $A_1 = \langle J, S_1, s_0, V_1, T_1 \rangle$ and $A_2 = \langle J, S_2, t_0, V_2, T_2 \rangle$ be open automata; \(\langle J, S_1, s_0, V_1, W T_1 \rangle\) and \(\langle J, S_2, t_0, V_2, W T_2 \rangle\) be the weak open automaton derived from $A_1$ and $A_2$ respectively. For any states $s \in S_1$ and $t \in S_2$ such that $(s, t | \text{Pred}_{s,t}) \in R$, we have:

- For any open transition $WOT$ in $WT_1$:

$$\gamma_{j}^{\in J^{'}, \text{Pred}_{OT}, \text{Post}_{OT}} \quad s \Rightarrow s'$$

there exists an indexed set of weak open transitions $WOT_x^{\in X} \subseteq WT_2$:

$$\gamma_{j}^{\in J^{'}, \text{Pred}_{OT_x}, \text{Post}_{OT_x}} \quad t \Rightarrow t_x$$

such that $\forall x, J' = J_x, (s', t_x | \text{Pred}_{x,t}) \in R$; and

$$\text{Pred}_{s,t} \land \text{Pred}_{OT} \implies \bigvee_{x \in X} (\forall j \in J_x \; \gamma_{j} = \gamma_{j}^{\in X} \land \text{Pred}_{x,t} \land \alpha = \alpha_x \land \text{Pred}_{x,t} \land \text{Post}_{OT} \land \text{Post}_{OT_x} \}$$
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 $\operatorname{Pred}_{s,t} \in \mathcal{R}$ and suppose that the following statement holds:

\[
\gamma_{j}^{j'}, \operatorname{Pred}_{\text{OT}}, \operatorname{Post}_{\text{OT}} \quad \xrightarrow{s} \quad s' \in \mathcal{WT}_1
\]

Moreover, by using Lemma 4 we know that:

\[
\gamma_{j}^{j'}, \operatorname{Pred}_{\text{OT}}, \operatorname{Post}_{\text{OT}} \quad \xrightarrow{s} \quad s' \in \mathcal{WT}_1 \Rightarrow \left( \forall i \in [1..n], \beta_{2j}^{j'}, \operatorname{Pred}_{1i}, \operatorname{Post}_{3i} \quad \xrightarrow{s_{20} \to s_{21}} \right) \wedge \left( \forall i \in [1..m], \beta_{3j}^{j'}, \operatorname{Pred}_{3i}, \operatorname{Post}_{3i} \quad \xrightarrow{s_{3i} \to s_{3(i+1)}} \right)
\]

where

\[
s = s_{11} \land s_{1(n+1)} = s_{20} \land s_{21} = s_{31} \land s_{3(m+1)} = s'
\]

\[
\alpha = \alpha' \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{i=1}^{n} \begin{array}{c}
\bigoplus_{j=i-1}^{1} \begin{array}{c}
\bigoplus_{i=1}^{m} \begin{array}{c}
\bigoplus_{j=i-1}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array}{c}
\bigoplus_{j=n}^{1} \begin{array{ 
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 \( T_1 \):

\[
\begin{align*}
\beta_j & \in J_1', \text{ Pred}_1, \text{ Post}_1 \\
\vdots \\
\tau & \rightarrow \ s_{12}
\end{align*}
\]

by hypothesis we have \((s_1, t|\text{Pred}_{s_1,t}) \in \mathcal{R}\) and \( s = s_{11} \). Thus, by Definition 14 we can deduce there exists an indexed set of weak open transitions \( WOT^a \subseteq \mathcal{W}_T \):

\[
\begin{align*}
\gamma_j & \in J_1', \text{ Pred}_{OPT_1}, \text{ Post}_{OPT_1} \\
\vdots \\
\tau & \rightarrow \ u_a
\end{align*}
\]

such that \( \forall a, J_a = \{ j \in J_1' | \beta_j \neq \tau \}, (s_{12}, u_a|\text{Pred}_{s_{11},a}) \in \mathcal{R} \) and

\[
\text{Pred}_{s_1,t} \land \text{Pred}_1 \implies \bigvee_{a \in A} \{ (\forall j \in J_a.(\beta_j)^\forall = \gamma_j \land \text{Pred}_{OPT_a} \land \alpha_{1a} = \tau \land \text{Pred}_{s_{11},a}\{\{\text{Post}_1 \sqcup \text{Post}_{OPT_a}\}\} \}
\]

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

\[
\begin{align*}
\gamma_j & \in J_1', \text{ Pred}_{OPT_1}, \text{ Post}_{OPT_1} \\
\vdots \\
\tau & \rightarrow \ u_a
\end{align*}
\]

2. Concerning the middle open transition in \( T_1 \):

\[
\begin{align*}
\beta_j & \in J_2', \text{ Pred}_2, \text{ Post}_2 \\
\vdots \\
\tau & \rightarrow \ s_{21}
\end{align*}
\]

we have \((s_{11}, u_a|\text{Pred}_{s_{11},a}) \in \mathcal{R}\) and \( s_{11} = s_{20} \). Again by Definition 14 we can deduce there exists an indexed set of weak open transitions \( WOT^b \subseteq \mathcal{W}_T \):

\[
\begin{align*}
\gamma_j & \in J_b, \text{ Pred}_{OPT_b}, \text{ Post}_{OPT_b} \\
\vdots \\
u_a & \rightarrow \ v_b
\end{align*}
\]

such that \( \forall b, J_b = \{ j \in J_2' | \beta_{2j} \neq \tau \}, (s_{21}, v_b|\text{Pred}_{s_{21},b}) \in \mathcal{R}; \)

\[
\text{Pred}_{s_{11},a} \land \text{Pred}_2 \implies \bigvee_{b \in B} \{ (\forall j \in J_b.(\beta_{2j})^\forall = \gamma_j \land \text{Pred}_{OPT_b} \land \alpha' = \alpha_{2b} \land \text{Pred}_{s_{21},b}\{\{\text{Post}_2 \sqcup \text{Post}_{OPT_b}\}\} \}
\]

65
3. Similarly to the case 1, we consider the third open transition in $T_1$:

$$\beta_{3j} \in J, \text{Pred}, \text{Post} \in T$$

$\sigma_{31} \xrightarrow{\tau} \sigma_{32}$

From previous case, we have $(s_{21}, v_b | \text{Pred}_{s_{21},b}) \in R$, and we have $s_{21} = s_{31}$.

Then, by Definition 13, there exists an indexed set of weak open transitions $\text{WOT}_{s} \subseteq \text{WT}_2$:

$$\gamma_{c} \in J_c, \text{Pred}_{OT_c}, \text{Post}_{OT_c}$$

$$v_b \Rightarrow w_c$$

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

$$\text{Pred}_{s_{21},b} \land \text{Pred}_3 \implies \bigvee_{c \in C} (\forall j \in J_c, \beta_{3j}^{\gamma} = \gamma_{jc} \land \text{Pred}_{OT_c} \land \text{Pred}_{s_{31},c} \{\text{Post}_3 \cup \text{Post}_{OT_c}\})$$

Based on cases described above by applying $\text{WT}_3$ rule on the resulting $\text{WOT}$s we have:

$$\gamma_{ja} \in J_a, \text{Pred}_{OT_a}, \text{Post}_{OT_a}$$

$$\gamma_{jb} \in J_b, \text{Pred}_{OT_b}, \text{Post}_{OT_b}$$

$$\gamma_{jc} \in J_c, \text{Pred}_{OT_c}, \text{Post}_{OT_c}$$

$$l \Rightarrow u_a$$

$$u_a \Rightarrow v_b$$

$$v_b \Rightarrow w_c$$

$$\gamma = \gamma_{ja} \oplus \gamma_{jb} \oplus \gamma_{jc} \oplus \text{Post}_{OT_a} \cup \text{Post}_{OT_b} \cup \text{Post}_{OT_c}$$

$$\text{Pred} = \text{Pred}_{OT_a} \land \text{Pred}_{OT_b} \land \text{Pred}_{OT_c} \land \text{Post}_{OT_a} \cup \text{Post}_{OT_b} \cup \text{Post}_{OT_c}$$

$$\text{Post} = \text{Post}_{OT_a} \cup \text{Post}_{OT_b} \cup \text{Post}_{OT_c}$$

$$\gamma^{''} = \gamma_{ja} \land \text{Pred} \land \alpha = \alpha^{''} \land \text{Pred}_{s_{31},a} \{\text{Post}_3 \cup \text{Post}_{OT_a}\}$$

It remains to be proven that the following statement holds:

$$\text{Pred}_{s,t} \land \text{Pred} \implies \bigvee_{x \in X} (\forall j \in J, \gamma_j = \gamma_j \land \text{Pred} \land \alpha = \alpha^{''} \land \text{Pred}_{s',x} \{\text{Post}_{OT} \cup \text{Post}\})$$

We have:

$$\text{Pred}_{OT} = \text{Pred}_1 \land \text{Pred}_2 \{\text{Post}_1\} \land \text{Pred}_3 \{\text{Post}_2 \cup \text{Post}_1\}$$

$$\text{Post}_{OT} = \text{Post}_3 \cup \text{Post}_2 \cup \text{Post}_1$$

Moreover, we have the following statement:

$$\text{Pred}_{s,t} \land \text{Pred}_1 \implies \bigvee_{a \in A} (\forall j \in J, \beta_{1j}^{\gamma} = \gamma_{ja} \land \text{Pred}_{OT_a} \land \text{Pred}_{s_{31},a} \{\text{Post}_3 \cup \text{Post}_{OT_a}\})$$

With the conjunction of the predicate $\text{Pred}_2 \{\text{Post}_1\}$ on both sides of the im-
application, we get:

\[
Pred_{s,t} \land \Pred_4 \land \Pred_2 \{\text{Post}_1\} \implies \\
\bigvee_{a \in A} \left(\forall j \in J_a. (\beta_{1j})^\top = \gamma_{ja} \land \Pred_{OT_a} \land \\
\Pred_{s11.a} \{\text{Post}_1 \cup \text{Post}_{OT_a}\} \land \Pred_2 \{\text{Post}_1 \cup \text{Post}_{OT_a}\}\right)
\]

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

\[
Pred_{s,t} \land \Pred_4 \land \Pred_2 \{\text{Post}_1\} \implies \\
\bigvee_{a \in A} \left(\forall j \in J_a. (\beta_{1j})^\top = \gamma_{ja} \land \Pred_{OT_a} \land \Pred_{s11.a} \land \Pred_2 \{\text{Post}_1 \cup \text{Post}_{OT_a}\}\right)
\]

By replacing the inner predicate \Pred_{s11.a} \land \Pred_2 by the conclusion of the statement given in case 2, the formula becomes:

\[
Pred_{s,t} \land \Pred_4 \land \Pred_2 \{\text{Post}_1\} \implies \\
\bigvee_{a \in A} \left(\forall j \in J_a. (\beta_{1j})^\top = \gamma_{ja} \land \Pred_{OT_a} \land \left(\bigvee_{b \in B} \left(\forall j \in J_b. (\beta_{2j})^\top = \gamma_{jb} \land \Pred_{OT_b} \land \alpha' = \alpha_{2b} \land \\
\Pred_{s21.b} \{\text{Post}_2 \cup \text{Post}_{OT_b}\} \{\text{Post}_1 \cup \text{Post}_{OT_b}\}\right)\right)\right)
\]

This can be rewritten into:

\[
Pred_{s,t} \land \Pred_4 \land \Pred_2 \{\text{Post}_1\} \implies \\
\bigvee_{a \in A} \bigvee_{b \in B} \left(\forall j \in J_a. (\beta_{1j})^\top = \gamma_{ja} \land \forall j \in J_b. (\beta_{2j})^\top = \gamma_{jb} \land \Pred_{OT_a} \land \\
\Pred_{OT_b} \{\text{Post}_1 \cup \text{Post}_{OT_b}\} \land \alpha' = \alpha_{2b} \land \Pred_{s21,b} \{\text{Post}_2 \cup \text{Post}_{OT_b} \cup \text{Post}_{OT_a}\}\right)
\]

Since Post does not act on \gamma_{jb}, nor on \Pred_{OT_b} and \alpha_2. As well \Pred_{OT_b} does not act on \alpha', nor on \beta_{2j}, the formula can be simplified as follows:

\[
Pred_{s,t} \land \Pred_4 \land \Pred_2 \{\text{Post}_1\} \implies \\
\bigvee_{a \in A} \bigvee_{b \in B} \left(\forall j \in J_a. (\beta_{1j})^\top = \gamma_{ja} \land \forall j \in J_b. (\beta_{2j})^\top \{\text{Post}_1\} = \gamma_{jb} \land \Pred_{OT_a} \land \\
\Pred_{OT_b} \{\text{Post}_{OT_b}\} \land \alpha' \{\text{Post}_1\} = \alpha_{2b} \land \Pred_{s21,b} \{\text{Post}_2 \cup \text{Post}_1 \cup \text{Post}_{OT_b} \cup \text{Post}_{OT_a}\}\right)
\]
Finally, the conjunction with the term \( \text{Pred}_t \{ \text{Post}_1 \} \) of the both sides of the implication and rewriting, we get:

\[
\text{Pred}_{s,t} \land \text{Pred}_t \{ \text{Post}_1 \} \land \text{Pred}_t \{ \text{Post}_2 \land \text{Post}_1 \} \implies \\
\bigvee_{a \in A} \bigvee_{b \in B} \bigvee_{c \in C} \left( \forall j \in J_a. (\beta_{1j})^\gamma = \gamma_ja \land \forall j \in J_b. (\beta_{2j})^\gamma = \gamma_jb \{ \text{Post}_{OT_a} \} \land \right.
\]
\[\left. \text{Pred}_{OT_a} \{ \text{Post}_{OT_a} \} \land \alpha' \{ \text{Post}_1 \} = \alpha_{26} \{ \text{Post}_{OT_a} \} \land (\text{Pred}_{s_{21,31}} \land \text{Pred}_t) \\
\left. \{ \text{Post}_2 \land \text{Post}_1 \} \lor \text{Post}_{OT_a} \land \text{Post}_{OT_b} \right) \right)
\]

Again note that because of the domain of the substitution function is independent from some predicates and expressions, we removed \( \text{Post}_{OT_a} \) and \( \text{Post}_{OT_b} \) in the substitution of the right side of the implication. Finally, by replacing the predicate \( \text{Pred}_{s_{21,31}} \land \text{Pred}_t \) by the conclusion of case 3, we get:

\[
\text{Pred}_{s,t} \land \text{Pred}_t \land \text{Pred}_t \{ \text{Post}_1 \} \land \text{Pred}_t \{ \text{Post}_2 \land \text{Post}_1 \} \implies \\
\text{Pred}_{OT} \bigoplus \bigoplus \left( \forall j \in J_a. (\beta_{1j})^\gamma = \gamma_ja \land \forall j \in J_b. (\beta_{2j})^\gamma = \gamma_jb \{ \text{Post}_{OT_a} \} \land \right.
\]
\[\left. \text{Pred}_{OT_a} \land \text{Pred}_{OT_b} \{ \text{Post}_{OT_a} \} \land \text{Pred}_{OT_c} \{ \text{Post}_{OT_b} \} \right) \land \text{Pred}_{s_{31}} \{ \text{Post}_3 \lor \text{Post}_2 \lor \text{Post}_1 \} \land \text{Post}_{OT} \land \text{Post}_{OT_b} \land \text{Post}_{OT_c} \left( \text{Post} \right)
\]

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 \( \gamma \oplus \gamma'' = \gamma' \oplus \gamma''' \)).

\[
\forall j \in J_a \cup J_b \cup J_c. (\beta_{1j})^\gamma \oplus (\beta_{2j})^\gamma \{ \text{Post}_{OT_a} \} \oplus (\beta_{3j})^\gamma \{ \text{Post}_{OT_b} \} = \gamma_{ja} \oplus \gamma_{jb} \{ \text{Post}_{OT_a} \} \cup \gamma_{jc} \{ \text{Post}_{OT_b} \} \cup \{ \text{Post}_{OT_c} \}
\]

We have \( s_{31} = s' \), so can rewrite the formula:

\[
\text{Pred}_{s,t} \land \text{Pred}_{OT} \implies \bigvee_{a \in A} \bigvee_{b \in B} \bigvee_{c \in C} \left( \forall j \in J. \gamma_j' = \gamma_j \land \text{Pred} \land \alpha = \right.
\]
\[\left. \text{Pred}_{s'} \{ \text{Post}_{OT} \lor \text{Post} \} \right)
\]

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:

\[
\text{Pred}_{s,t} \land \text{Pred}_{OT} \implies \bigvee_{x \in X} \left( \forall j \in J, \gamma'_j = \gamma_j \land \alpha = \alpha'' \land \text{Pred}_{s',x} \left\{ \text{Post}_{OT} \right\} \right)
\]

**Theorem 6.** Weak FH-bisimilarity is an equivalence. Suppose \(R\) is a weak FH-bisimulation. Then \(R\) is an equivalence, that is, \(R\) 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 \(\beta\) and \(\gamma\). 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 \(\text{Leaves}(Q) = pLT_{\mathcal{S}_i^{L_Q}}\); suppose \(^{16}\)

\[
P[Q]_{j_0} = \epsilon_{s_i^{L_Q}} \text{Pred}, \text{Post} \quad <s_i^{L_Q} \alpha > <s_i' \in \mathcal{L}_Q >
\]

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 \(n, \text{Pred}', \text{Post}'\), and for all \(p \in [1..n]\) there exist \(\beta_p, \alpha_p, \text{Pred}_p, \text{Post}_p\) and a family \(\gamma_{j_p}^{L_P}\) and for all \(p \in [1..n+1]\) \(s_{p \epsilon L_Q}^{i}\) 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 \(Q\).
such that

\[ \bigcup_{p=1}^{n} J_p = J \cap \text{Holes}(Q), \gamma_j \in J \cap \text{Holes}(Q) = \bigoplus_{p=1}^{n} (\gamma_{j,p}^{i} \bigoplus_{i=p-1}^{1} \text{Post}_i) \],

\[ \text{Pred} \iff \bigwedge_{p=1}^{n} (\alpha_p \bigoplus_{i=p-1}^{1} \text{Post}_i) = \beta_p \land \bigwedge_{p=1}^{n} (\alpha_p \bigoplus_{i=p-1}^{1} \text{Post}_i) \],

\[ \text{Post} = \text{Post}' \cup \bigcup_{p=n}^{1} \text{Post}_p, \text{ and } \forall i \in L. s_{(i+1)n} = s_i \land s_{n+1} = s_i \]

where for any \( p \), Post \(_p\) only acts upon variables \( \text{vars}(Q) \).

**Proof.** Suppose that we have:

\[ P[Q]_{j_0} \models \begin{array}{c} \forall i \in L \setminus L_Q \gg \alpha \gg s_i' \end{array}, \gamma_{j_0} = [\beta_0 .. \beta_n] \]

By Lemma 4 this implies the following:

\[ \forall p \in \{1 \ldots m_1\} P[Q]_{j_0} \models \begin{array}{c} \beta_{1p} \land \bigwedge_{i=p-1}^{1} \text{Post}_i \end{array}, \quad P[Q]_{j_0} \models \begin{array}{c} \beta_{2p} \land \bigwedge_{i=p-1}^{1} \text{Post}_i \end{array} \]

and \( \forall p \in \{1 \ldots m_2\} P[Q]_{j_0} \models \begin{array}{c} \beta_{3p} \land \bigwedge_{i=p-1}^{1} \text{Post}_i \end{array} \]

where
We can apply Lemma 1 on each $OT_P$ in the form $(\beta_{1p} = \beta_{1p}')$

1. For each open transition $OT_P$ in the form $(\beta_{1p} = \beta_{1p}')$

   $$P[Q]_{ \beta_{1p} } = \beta_{1p}' \land \text{Pred}_{1p} \land \text{Post}_{1p}$$

   If $Q$ moves then we obtain by Lemma 1

   $$P \models (\beta_{1p})' \in (J_{1p} \setminus \text{Holes}(Q)) \cup \{J_0\}, \text{Pred}_{1p}' \land \text{Post}_{1p}'$$

   and

   $$Q \models (\beta_{1p})' \in (J_{1p} \setminus \text{Holes}(Q)), \text{Pred}_{1p}' \land \text{Post}_{1p}'$$

   such that

   $\text{Pred}_{1p} \iff \text{Pred}_{1p}' \land \text{Pred}_{1p}'' \land \alpha_{1p} = \beta_{1p,0}$, $\text{Post}_{1p} = \text{Post}_{1p}' \sqcup \text{Post}_{1p}''$ where $\text{Post}_{1p}''$ is the restriction of $\text{Post}_{1p}$ over $\text{vars}(Q)$.

   Else $Q$ does not move and we have:

   $$P \models (\beta_{1p})' \in (J_{1p} \setminus \text{Holes}(Q)), \text{Pred}_{1p}' \land \text{Post}_{1p}'$$

   and

   $$Q \models (\beta_{1p})' \in (J_{1p} \setminus \text{Holes}(Q)) \cup \{J_0\}, \text{Pred}_{1p}' \land \text{Post}_{1p}'$$

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

3. Finally, for the open transition in the form $(\beta_{2j} = \beta_{2j}')$:
we have

On the other hand, we have:

By using Lemma 4, and denoting \( J \) \( \alpha \) \( j \)

\[
\alpha' = \bigoplus_{i=1}^{m_1} \left( \bigoplus_{j=m_1+1}^{1} \right) \text{Post}^{1}_{1j} \bigoplus \left( \bigoplus_{i=1}^{m_2} \right) \text{Post}^{2}_{1j} \bigoplus \left( \bigoplus_{j=m_1}^{1} \right) \text{Post}^{2}_{1j}
\]

such that \( \text{Pred}_2 \iff J_2 \not\in \text{Holes}(Q) \), \( \text{Pred}', \text{Post}'_2 \)

\[
Q_{\text{Post}} = \bigoplus_{i=1}^{m_2} \left( \bigoplus_{j=m_1+1}^{1} \right) \text{Post}'_2 \bigoplus \left( \bigoplus_{i=1}^{m_2} \right) \text{Post}^{2}_2 
\]

where \( \text{Post}''_2 \) is the restriction of \( \text{Post}_2 \) over variables \( \text{vars}(Q) \).

Else \( Q \) does not move and we have:

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:

\[
\alpha'' = \bigoplus_{j=m_1}^{1} \left( \bigoplus_{i=1}^{m_1} \right) \text{Post}_{1j} \bigoplus \left( \bigoplus_{i=1}^{m_2} \right) \text{Post}^{2}_{1j} \bigoplus \left( \bigoplus_{j=m_1}^{1} \right) \text{Post}^{2}_{1j}
\]

where \( \alpha'' = \alpha' \bigoplus \left( \bigoplus_{j=m_1}^{1} \right) \text{Post}_{1j} \bigoplus \left( \bigoplus_{i=1}^{m_2} \right) \text{Post}^{2}_{1j} \bigoplus \left( \bigoplus_{j=m_1}^{1} \right) \text{Post}^{2}_{1j} \).

On the other hand, we have:

\[
\alpha = \alpha' \bigoplus \left( \bigoplus_{j=m_1}^{1} \right) \text{Post}_{1j} \bigoplus \left( \bigoplus_{i=1}^{m_2} \right) \text{Post}^{2}_{1j} \bigoplus \left( \bigoplus_{j=m_1}^{1} \right) \text{Post}^{2}_{1j}
\]

As \( \bigoplus_{j=m_1}^{1} \left( \bigoplus_{i=1}^{m_1} \right) \text{Post}_{1j} \bigoplus \left( \bigoplus_{i=1}^{m_2} \right) \text{Post}^{2}_{1j} \bigoplus \left( \bigoplus_{j=m_1}^{1} \right) \text{Post}^{2}_{1j} \) has no effect on variables of \( P \) and thus on variables of \( \alpha'' \), so we have \( \alpha = \alpha'' \).
\( P_{\beta} = \bigwedge_{i=1}^{m_1} \big( (\beta_{1p_{j0}} = \alpha_{1p}) \big) \big( \bigcirc Post_{1j}^i \big) \wedge (\beta_{2\rightarrow} = \alpha_{21}) \big( \bigcirc Post_{1j}^i \big) \wedge \bigwedge_{j=m_1+1}^{m_2} \big( (\beta_{3p_{j0}} = \alpha_{3p}) \big) \big( \bigcirc Post_{3j}^i \big) \bigcirc Post_{1j}^i \big) \big) \wedge \bigwedge_{j=m_1+1}^{m_2} \big( \bigcirc Post_{3j}^i \big) \bigcirc Post_{1j}^i \big) \big) \)

Note that for all \( j \in J \setminus \text{Holes}(Q), \gamma_j' = \gamma_j \) because for all \( l Post_{1l}^i \) coincides with \( Post_{1l}^i \) on the variables of \( \beta_{1j}, \) and similarly for \( Post_{2}^i \) and \( Post_{3}^i. \)

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

\[
\begin{align*}
\text{Pred}_{\beta} &= \bigwedge_{p=1}^{m_1} ( (\beta_{1p_{j0}} = \alpha_{1p}) \big) \big( \bigcirc Post_{1j}^i \big) \wedge (\beta_{2\rightarrow} = \alpha_{21}) \big( \bigcirc Post_{1j}^i \big) \wedge \bigwedge_{j=m_1+1}^{m_2} ( (\beta_{3p_{j0}} = \alpha_{3p}) \big) \big( \bigcirc Post_{3j}^i \big) \bigcirc Post_{1j}^i \big) \big) \wedge \bigwedge_{j=m_1+1}^{m_2} \big( \bigcirc Post_{3j}^i \big) \bigcirc Post_{1j}^i \big) \big) \big) 
\end{align*}
\]

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 \( \langle \alpha_{pq} \rangle_{\ell \in L_Q} = \langle \alpha_{pq} \rangle_{s(\ell_{(q+1)}i)} \), i.e. we rename all the following states \( \langle \alpha_{pq} \rangle_{\ell \in L_Q} \) where \( p \geq q + 1 \) into \( \langle \alpha_{pq} \rangle_{\ell \in L_Q} \). 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 \( s_p \) are equal to the first \( s_{1l}. \) So we have:

\[
\forall p \in [1..m_1] \ Q \models (\beta_{1p}^i)_{\ell \in \text{Holes}(Q), \big( \text{Pred}_{\beta}^i \big) \big( \bigcirc Post_{1j}^i \big) \wedge \bigwedge_{j=m_1+1}^{m_2} \big( \bigcirc Post_{3j}^i \big) \bigcirc Post_{1j}^i \big) \big) \big) \big) \big) \big) \big) \big) \big) 
\]

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:

\[
\forall p \in [1..(n_1 + n_2 + 1)] \ Q \models (\beta_{p}^i)_{\ell \in \text{Holes}(Q), \big( \text{Pred}_{\beta}^i \big) \big( \bigcirc Post_{1j}^i \big) \wedge \bigwedge_{j=m_1+1}^{m_2} \big( \bigcirc Post_{3j}^i \big) \bigcirc Post_{1j}^i \big) \big) \big) \big) \big) \big) \big) \big) \big) 
\]

In this equation, and using case 1 above for all \( k \in [1..n_1] \) there is a \( p \in [1..m_1] \) such that \( \alpha_{kp} = \alpha_k' \) (following the re-indexing done in the removal of steps where \( Q \) does not move), we know that \( \text{Pred}_{1p} \) contains the predicate \( \alpha_{kp} = \beta_{1p_{j0}}. \)
Because $\beta_{1p,j_0}$ only contains variables of $P$ and $\alpha'_k$ only variables of $Q$, we have:

\[
(\alpha_1 = \beta_{1p,j_0}) \iff (\bigcirc_{j=p}^{1} Post_{1j}) = (\bigcirc_{j=p}^{1} Post'_{1j})
\]

\[
\iff (\bigcirc_{j=k-1}^{1} Post''_{j}) = (\bigcirc_{j=p}^{1} Post'_{1j})
\]

We can obtain similar equations for $\alpha'_{n_1+1}$ related with $\beta_{2p,j_0}$ and the $\alpha'_k$ for $k \geq n_1 + 2$ related with $\beta_{3p,j_0}$ 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):

\[
Pred_\beta \iff \gamma_{j_0} = (\bigcirc_{j=p}^{1} Post''_{j})|p \in [1..n_1 + n_2 + 1])^\vee \tag{B.1}
\]

Let us consider the sequence of $(n_1 + n_2 + 1)$ actions $\alpha'_p$, some of them may be non-observable (they are $\tau$ transitions). By considering the sequence of $\tau$ and non-$\tau$ 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 $\alpha_p$, and possibly many non-observable ($\tau$) 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 $k \in [1..n]$:

\[
\forall p \in [n_k..(n_{k+1} - 1)] \quad Q \models \begin{array}{c}
\beta_{pj} \in Holes(Q) \\
\alpha_{n_k} \quad \alpha_{n_{k+1}}
\end{array}
\quad \begin{array}{c}
\bigcirc_{q \in L_Q} \quad \alpha_{n_k} \\
\bigcirc_{q \in L_Q} \quad \alpha_{n_{k+1}}
\end{array}
\quad \begin{array}{c}
\alpha_{n_k} \quad \tau \quad \alpha_{n_{k+1}}
\end{array}
\]

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

\[
Q \models (\gamma_{j_k}) \in Holes(Q), \quad Pred_k, \quad Post_k
\]

\[
\begin{array}{c}
\alpha_k \\
\alpha_k
\end{array}
\quad \begin{array}{c}
\bigcirc_{q \in L_Q} \quad \alpha_k \\
\bigcirc_{q \in L_Q} \quad \alpha_k
\end{array}
\quad \begin{array}{c}
\tau \\
\tau
\end{array}
\]

\[
\begin{array}{c}
(n_k) \in L_Q \\
(n_k)
\end{array}
\]

\[
74
\]
with:

\[ \forall l \in LQ, v_{kl} = v_{(n'_k)l} \land v_{kl'} = v_{(n'_k)l} \]

\[ \alpha_k = \alpha_{n_k} \left\{ \bigotimes_{j=n_k-1}^{n'_k} \text{Post}_j^\prime \right\} \]

\[ \gamma_{kj}^{i \in J \cap \text{Holes}(Q)} = \bigoplus_{i=n'_k}^{n'_{k+1}-1} \left( \beta_{ij}^{i \in J \cap \text{Holes}(Q)} \left\{ \bigotimes_{l=i-1}^{n'_k} \text{Post}_l^\prime \right\} \right) \oplus \left( \beta_{l}^{i \in J \cap \text{Holes}(Q)} \left\{ \bigotimes_{l=n_k-1}^{n'_k} \text{Post}_l^\prime \right\} \right) \]

\[ \text{Pred}_k = \bigwedge_{i=n'_k}^{n'_{k+1}-1} \text{Pred}_i \left\{ \bigotimes_{j=n_k-1}^{n'_k} \text{Post}_j^\prime \right\} \land \text{Pred}_i \left\{ \bigotimes_{j=n_k-1}^{n'_k} \text{Post}_j^\prime \right\} \land \bigwedge_{i=n'_k}^{n'_{k+1}-1} \text{Pred}_i \left\{ \bigotimes_{j=n_{k+1}-1}^{n'_k} \text{Post}_j^\prime \right\} \]

\[ \text{Post}_k = \bigotimes_{j=n'_k}^{n'_{k+1}-1} \text{Post}_j^\prime \]

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

By definition of Post\(_k\), we have \( \bigotimes_{j=n'_k}^{n'_{k+1}-1} \text{Post}_j^\prime = \bigotimes_{j=n_k-1}^{n'_k} \text{Post}_j^\prime \). Consequently, we have:

\[ \alpha'_{n_k} \left\{ \bigotimes_{j=n_k-1}^{n'_k} \text{Post}_j^\prime \right\} = \alpha'_{n_k} \left\{ \bigotimes_{j=n_k-1}^{n'_k} \text{Post}_j^\prime \right\} = \alpha_k \left\{ \bigotimes_{j=n'_k}^{n'_{k+1}-1} \text{Post}_j^\prime \right\} = \alpha_k \left\{ \bigotimes_{j=n_k-1}^{n'_k} \text{Post}_j^\prime \right\} \]

From equation B.1 we obtain the following equation (we recall that the actions \( \alpha_k \) are the actions \( \alpha'_p \) that are observable):

\[ \text{Pred}_j \iff \gamma_{j0} = \bigoplus_{p=1}^{p=n_1+n_2+2} \left( \alpha_p \left\{ \bigotimes_{j=p-1}^{1} \text{Post}_j \right\} \right)^\vee \]

\[ \iff \gamma_{j0} = [\alpha_p \left\{ \bigotimes_{j=p-1}^{1} \text{Post}_j \right\} | p \in [1..n]] \]

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:

\[
Q = \left( \gamma_{p_j} \in J, \text{Pred}_{p_j}, \text{Post}_{p_j} \right)_{\text{p_j} \in L_Q} \Rightarrow \left( \sigma_{(p+1)} \right)
\]

with

\[
\bigcup_{p=1}^{n} J_p = J \cap \text{Holes}(Q) \quad \text{trivial}
\]

\[
\gamma_{j}^{\in J \cap \text{Holes}(Q)} = \bigoplus_{p=1}^{n} (\gamma_{p_j}^{\in J_p} \{ \bigcirc \} \text{Post}_{p_j})
\]

Indeed we have:

\[
\gamma_{j}^{\in J} = \bigoplus_{i=1}^{m_1} (\beta_{1i}^{\in J_1} \{ \bigcirc \} \text{Post}_{1i})^\lor \bigoplus_{j=1}^{m_1} (\beta_{2j}^{\in J_2} \{ \bigcirc \} \text{Post}_{2j})^\lor \bigoplus_{i=1}^{m_2} (\beta_{3i}^{\in J_3} \{ \bigcirc \} \text{Post}_{3i})^\lor \bigoplus_{j=1}^{m_2} (\beta_{4j}^{\in J_4} \{ \bigcirc \} \text{Post}_{4j})^\lor
\]

And thus, because \( \beta_{1j}^{\in J \cap \text{Holes}(Q)} \) are equal to the concatenation of \((\beta_{1p_j}^{'}_{j})^{\in J \cap \text{Holes}(Q)}, (\beta_{2j}^{'}_{j})^{\in J \cap \text{Holes}(Q)}, \) and \((\beta_{3p_j}^{'}_{j})^{\in J \cap \text{Holes}(Q)} \) (re-indexed because we skipped some transitions), and additionally \((\beta_{1p_j}^{''}_{j})^{\in J \cap \text{Holes}(Q)}, (\beta_{2j}^{''}_{j})^{\in J \cap \text{Holes}(Q)}, \) and \((\beta_{3p_j}^{''}_{j})^{\in J \cap \text{Holes}(Q)} \) (re-indexed) when \( Q \) moves \(^{17}\)

We can assert a similar equality on post-conditions, i.e. between Post\(_p\), Post\(_{1k_j}\), Post\(_{2k_j}\), Post\(_{3k_j}\) where Post\(_{1p}\) is the restriction of Post\(_p\) over vars\(_Q\) (see initial decomposition, case 1, 2, and 3 above). Overall, we have \( \forall i \in L_Q, s_{(n+1)i} = s_i' \land s_{0i} = s_i \) (see above):

\[
\gamma_{j}^{\in J \cap \text{Holes}(Q)} = \bigoplus_{i=1}^{m_1} (\beta_{1i}^{\in J_1} \{ \bigcirc \} \text{Post}_{1i}')^\lor \bigoplus_{j=1}^{m_1} (\beta_{2j}^{\in J_2} \{ \bigcirc \} \text{Post}_{2j}')^\lor \bigoplus_{i=1}^{m_2} (\beta_{3i}^{\in J_3} \{ \bigcirc \} \text{Post}_{3i}')^\lor \bigoplus_{j=1}^{m_2} (\beta_{4j}^{\in J_4} \{ \bigcirc \} \text{Post}_{4j}')^\lor
\]

\[
= \bigoplus_{k=1}^{n_k} (\beta_{k}^{\in J \cap \text{Holes}(Q)} \{ \bigcirc \} \text{Post}_{k})
\]

\(^{17}\)more precisely, when \( Q \) moves either \( \beta_{1j}^{\in J \cap \text{Holes}(Q)} \) is not empty and thus \( \beta_{1p_j}^{'}_{j} \) \( \in J \cap \text{Holes}(Q) = \beta_{1k_j}^{'} \), or both are empty if the holes of \( Q \) perform no action.
Next, we have:

\[
P_{\text{pred}} = P_{\text{pred}}' \land \bigwedge_{p=1}^{n} \left( (\alpha_{p} \bowtie \bigcirc_{i=p-1}^{1} P_{\text{post}_{i}}) = \beta_{p} \land (\bigcirc_{i=p-1}^{1} P_{\text{post}_{i}}) \right)
\]

Indeed we have:

\[
P_{\text{pred}} \iff \bigwedge_{i=1}^{m_{1}} \left( P_{\text{pred}}_{1} \land P_{\text{pred}}'_{1} \land \alpha_{1} = \beta_{1j0} \right) \left( \bigcirc_{j=1-i}^{1} P_{\text{post}_{j}} \right)
\]

\[
\land \left( P_{\text{pred}}'_{2} \land P_{\text{pred}}''_{2} \land \alpha_{2} = \beta_{2j0} \right) \left( \bigcirc_{j=1-i}^{1} P_{\text{post}_{j}} \right)
\]

\[
\land \bigwedge_{i=1}^{m_{2}} \left( P_{\text{pred}}_{31} \land P_{\text{pred}}'_{1} \land \alpha_{3} = \beta_{3j0} \right) \left( \bigcirc_{j=1-i}^{1} P_{\text{post}_{j}} \right)
\]

\[
\land \left( P_{\text{pred}}''_{2} \left( \bigcirc_{j=1-i}^{1} P_{\text{post}_{j}} \right) \land (\alpha_{21} = \beta_{2j0}) \left( \bigcirc_{j=1-i}^{1} P_{\text{post}_{j}} \right) \right)
\]

\[
\land \left( P_{\text{pred}}_{31} \left( \bigcirc_{j=1-i}^{1} P_{\text{post}_{j}} \right) \land P_{\text{pred}}''_{1} \left( \bigcirc_{j=1-i}^{1} P_{\text{post}_{j}} \right) \land (\alpha_{31} = \beta_{3j0}) \left( \bigcirc_{j=1-i}^{1} P_{\text{post}_{j}} \right) \right)
\]

\[
\iff \text{P}_{\text{pred}}'' \land \bigwedge_{k=1}^{n} \left( P_{\text{pred}}_{4k} \left( \bigcirc_{j=1-k}^{1} P_{\text{post}_{j}} \right) \right)
\]

\[
\iff \text{P}_{\text{pred}}'' \land \bigwedge_{k=1}^{n} \left( P_{\text{pred}}_{4k} \left( \bigcirc_{j=1-k}^{1} P_{\text{post}_{j}} \right) \land (\gamma_{j0} = [\alpha_{i} \left( \bigcirc_{j=1-i}^{1} P_{\text{post}_{j}} \right)] i \in [1..n]) \right)
\]

which is exactly what is needed with \( \gamma_{j0} = [\beta_{0}..\beta_{n}] \).
Finally we have $Post = Post' \uplus \bigoplus_{p=n}^{1} Post_p$ because

$$Post = \bigoplus_{j=m_2}^{1} Post_{3j} \circ Post_2 \circ \bigoplus_{j=m_1}^{1} Post_{1j}$$

$$= \bigoplus_{j=m_2}^{1} Post'_{3j} \circ Post'_{2j} \circ \bigoplus_{j=m_1}^{1} Post'_{1j}$$

$$= Post' \uplus \bigoplus_{j=n'_{k+1}}^{1} Post''_j$$

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

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

$$P \models \beta_{j \in J}, Pred, Post \frac{\beta_{j \in L_Q} \models \alpha_{Q} \leftarrow \langle s'_i \rangle \in L_Q}{\langle s_i \rangle \models \alpha_{Q} \leftarrow \langle s'_i \rangle \in L_Q}$$

Let $Pred' = Pred \wedge (\beta_{j_0} = \alpha_{Q} \wedge Pred_{Q})$ and $Post' = Post \uplus Post_{Q}$ Then, we have:

$$P'[Q]_{j_0} \models \beta_{j \in J \setminus \{j_0\}} \uplus \alpha_{Q} \leftarrow \langle s'_i \rangle \in L_Q$$

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 $\alpha'_{Q}$ open transition (observable or not depending on $\alpha_{Q}$):

$$\forall h \in [1, k].Q \models \beta_{3h}, Pred_{3h}, Post_{3h} \frac{\langle s_{1h} \rangle \models \alpha_{Q} \leftarrow \langle s_{1(h+1)} \rangle}{\langle s_{21} \rangle \models \alpha_{Q} \leftarrow \langle s_{22} \rangle}$$

and $\forall h \in [1, k'].Q \models \beta_{2h}, Pred_{2h}, Post_{2h}$

such that

$$s_{i} \in L_Q = s_{11} \wedge s_{1(k+1)} = s_{21} \wedge s_{22} = s_{11} \wedge s_{3(k+1)}$$

$$\alpha_{Q} = \alpha'_{Q} \left\{ \bigoplus_{j=k}^{1} Post_{1j} \right\}$$

78
where \( \alpha \) and \( \alpha_1 \). For the first item:

By Lemma 2 with the lemma hypotheses we obtain:

\[
\gamma = \prod_{h=1}^{k} (\exists_{i=k} \text{Post}_{i+1}) \cap (\exists_{j=h-1} \text{Post}_j) \cup \prod_{i=k}^{1} (\exists_{i=k} \text{Post}_i) \cup \prod_{h=1}^{k'} (\exists_{i=k} \text{Post}_{i+1} \cap \text{Post}_2 \cup \exists_{j=h-1} \text{Post}_j) \cup \prod_{i=k}^{1} (\exists_{i=k} \text{Post}_i) \cup \prod_{h=1}^{k'} (\exists_{i=k} \text{Post}_{i+1} \cap \text{Post}_2 \cup \exists_{j=h-1} \text{Post}_j)
\]

\[
\text{Pred}_Q = \bigwedge_{h=0}^{1} \text{Pred}_{ih} \prod_{j=h-1}^{1} \text{Post}_j \land \text{Pred}_{ih} \prod_{j=h-1}^{1} \text{Post}_j \bigwedge_{h=k}^{1} \text{Pred}_{ih} \prod_{j=h-1}^{1} \text{Post}_j \prod_{h=k}^{1} \text{Post}_j
\]

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:

\[
P[Q]_{j_0} = \prod_{h=1}^{k} (\exists_{i=k} \text{Post}_{i+1}) \land \text{Pred}_{ih} \prod_{j=h-1}^{1} \text{Post}_j \land \text{Pred}_{ih} \prod_{j=h-1}^{1} \text{Post}_j \bigwedge_{h=k}^{1} \text{Pred}_{ih} \prod_{j=h-1}^{1} \text{Post}_j \prod_{h=k}^{1} \text{Post}_j
\]

2. For the possibly observable open transition. By Lemma 2 with the lemma hypotheses we obtain:

\[
P[Q]_{j_0} = \prod_{h=1}^{k} (\exists_{i=k} \text{Post}_{i+1}) \land \text{Pred}_{ih} \prod_{j=h-1}^{1} \text{Post}_j \land \text{Pred}_{ih} \prod_{j=h-1}^{1} \text{Post}_j \bigwedge_{h=k}^{1} \text{Pred}_{ih} \prod_{j=h-1}^{1} \text{Post}_j \prod_{h=k}^{1} \text{Post}_j
\]

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:

\[
P[Q]_{j_0} = \prod_{h=1}^{k} (\exists_{i=k} \text{Post}_{i+1}) \land \text{Pred}_{ih} \prod_{j=h-1}^{1} \text{Post}_j \land \text{Pred}_{ih} \prod_{j=h-1}^{1} \text{Post}_j \bigwedge_{h=k}^{1} \text{Pred}_{ih} \prod_{j=h-1}^{1} \text{Post}_j \prod_{h=k}^{1} \text{Post}_j
\]

where \( \alpha' = \alpha_{j=k} \prod_{j=k}^{1} \text{Post}_{i+1} \) and \( \alpha = \alpha' \) because \( \text{Post}_j \) acts on variables of \( Q \) and \( \alpha \) contains only variables of \( P \).

\[
\tau = \bigoplus_{h=1}^{k} (\exists_{i=k} \text{Post}_{i+1}) \cap (\exists_{j=h-1} \text{Post}_j) \cup \bigoplus_{i=k}^{1} (\exists_{i=k} \text{Post}_i) \cup \bigoplus_{h=1}^{k'} (\exists_{i=k} \text{Post}_{i+1} \cap \text{Post}_2 \cup \exists_{j=h-1} \text{Post}_j) \cup \bigoplus_{i=k}^{1} (\exists_{i=k} \text{Post}_i) \cup \bigoplus_{h=1}^{k'} (\exists_{i=k} \text{Post}_{i+1} \cap \text{Post}_2 \cup \exists_{j=h-1} \text{Post}_j)
\]

\[
= \bigoplus_{h=1}^{k} (\exists_{i=k} \text{Post}_{i+1}) \cap (\exists_{j=h-1} \text{Post}_j) \cup \bigoplus_{i=k}^{1} (\exists_{i=k} \text{Post}_i) \cup \bigoplus_{h=1}^{k'} (\exists_{i=k} \text{Post}_{i+1} \cap \text{Post}_2 \cup \exists_{j=h-1} \text{Post}_j)
\]

\[
\because \text{Post}_{i+1} \text{ does not act on variables of } \beta_j.
\]
\[
\begin{align*}
\text{Pred}_c &= \bigwedge_{h=1}^{k} \text{Pred}_1 \bigcirc \text{Post}_{1i} \land \left( \text{Pred} \land \text{Pred}_2 \land \alpha'_Q = \beta_{j_0} \right) \bigcirc \text{Post}_{1i} \land \\
&\bigwedge_{h=1}^{k'} \text{Pred}_3 \bigcirc \text{Post}_{3i} \land \text{Post}_2 \bigcirc \text{Post}_{1i} \\
\text{Post}_c &= \bigcirc \text{Post}_{1i} \land \text{Post}_2 \bigcirc \text{Post}_{1i} \\
\end{align*}
\]

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

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

\[
\begin{align*}
\text{Pred}_c &= \text{Pred}_Q \land \text{Pred}_1 \bigcirc \text{Post}_{1i} \land \left( \alpha'_Q = \beta_{j_0} \right) \bigcirc \text{Post}_{1i} \\
&= \text{Pred}_Q \land \text{Pred}_1 \land \alpha_Q = \beta_{j_0} \\
\text{Post}_c &= \text{Post}_Q \lor \text{Post}_{1i} \\
\end{align*}
\]

\[\square\]

**Lemma 8 (Weak open transition composition).** Suppose that we have a weak open automaton such that the WOTs cannot observe silent actions (see Definition 18). Suppose \( j_0 \in J \) and \( \gamma_{j_0} = [\beta_0 \ldots \beta_n] \) and additionally:

\[
\begin{align*}
P &= \bigcirc \text{Pred}_j \bigcirc \text{Post} \quad \forall j \in J \\
&\quad \bigcirc s_{ij}^{1 \in L_P} \Rightarrow \bigcirc s_{ij}^{1 \in L_P} \\
&\land \forall p \in [1..n] \quad Q &= \bigcirc \text{Pred}_p \bigcirc \text{Post}_p \quad \forall s_{ij}^{1 \in L_P} \Rightarrow \bigcirc s_{ij}^{1 \in L_P} \\
&\quad \forall \alpha \in [1..n] \\
\end{align*}
\]

Let

\[
J_Q = \bigcup_{p=1}^{n} J_p \\
\forall i \in L_Q, s_i = s_{0i} \land s'_i = s_{(n+1)i} \\
\forall j \in J_p, \gamma_j = \bigoplus_{p=1}^{n} \gamma_{jp} \bigcirc \text{Post}_k \\
\]

\[
\begin{align*}
\text{Pred}' &= \text{Pred} \land \bigwedge_{p=1}^{n} (\alpha_p = \beta_p \land \text{Pred}_p) \bigcirc \text{Post}_i \\
&\land \text{Post}' = \text{Post} \lor \bigcirc \text{Post}_p \\
\end{align*}
\]

Then, we have:

\[
P[Q]_{j_0} \models \bigcirc \text{Pred}' \bigcirc \text{Post}'
\]

\[18\] Post_{1i} only has an effect on variables of \( Q \) and thus does not modify \( \text{Pred} \) or \( \beta_{j_0} \).
Proof. Suppose we have:

\[ P \models \beta_j^{i \in J}, \text{Pred, Post} \]

\[ <s_i \in L_\alpha > \rightarrow < s_j^{i \in L_\alpha >} \]

By Lemma \[ \ref{lemma} \] this implies the following:

\[ \forall p \in [1..m_1]. P \models \beta^{i \in J_1p}, \text{Pred_1p, Post_1p} \]

\[ <s_{1pi} \in L_\alpha > \rightarrow < s_{1(p+1)i} \in L_\alpha > \]

\[ \beta^{i \in J_2}, \text{Pred_2, Post_2} \]

\[ <\ (s_{21i}) \in L_\alpha > \rightarrow < \ (s_{22i}) \in L_\alpha > \]

and \[ \forall p \in [1..m_2]. P \models \beta^{i \in J_3p}, \text{Pred_3p, Post_3p} \]

\[ <s_{3pi} \in L_\alpha > \rightarrow < s_{3(p+1)i} \in L_\alpha > \]

where:

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

\[ \alpha = \alpha' \bigoplus_{j=m_1}^{m_2} \text{Post}_{ij} \]

\[ \gamma_j = \bigoplus_{i=1}^{m_1} \left( \beta_{ij}^{i \in J_1} \bigoplus_{k=1}^{m_1} \text{Post}_{ik} \right) \]

\[ \bigoplus_{i=1}^{m_2} \left( \beta_{ij}^{i \in J_2} \bigoplus_{k=1}^{m_1} \text{Post}_{ik} \right) \]

\[ \bigoplus_{i=1}^{m_2} \left( \beta_{ij}^{i \in J_3} \bigoplus_{k=1}^{m_1} \text{Post}_{ik} \right) \]

\[ \text{Pred} = \bigwedge_{p=1}^{m_1} \bigwedge_{j=p-1}^{m_2} \text{Pred}_p \bigwedge_{j=1}^{m_1} \text{Pred}_1 \bigwedge_{j=m_1}^{m_2} \text{Pred}_3 \]

\[ \emptyset_{m_1} \]

\[ \text{Post} = \bigwedge_{p=1}^{m_1} \bigwedge_{j=p-1}^{m_2} \text{Post}_j \bigwedge_{j=m_1}^{m_2} \text{Post}_3 \]

Note that, for \( l \in \{1,3\} \) if \( \beta_{lpj_0} = \tau \), then, because of Definition \[ \ref{definition} \] \( P \) necessarily makes a \( \tau \) 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 \( \beta_{lpj_0} = \tau \) or \( j_0 \notin J_1p \). To avoid a special case, we suppose that the hole \( j_0 \) moves during the OT \( \alpha' \), i.e. \( \beta_{2j_0} = \beta_m \) for some \( m \). Additionally, \( \beta_m = \tau \), else we would have \( \alpha = \alpha' = \tau \) and the \( \alpha' \) OT could be also removed from the reduction, leading to a particular and simpler case.

We introduce \( n_i^{[1..m_1-1]} \), and \( (n_i')^{[m_1+1..n]} \) the indices of the steps in which the hole \( j_0 \) moves in the 3 sets of OTs above (\( \beta_m \) is the action that matches the hole \( j_0 \) in the OT \( \alpha' \), in other words, we have for all \( i \), \( \beta_{1n_i,j_0} \) a visible action,
as additionally:

\[\gamma_{j_0} = [\beta_0 \ldots \beta_n] \]

\[= \bigoplus_{i=0}^{m_2} (\beta_{3n_{j_0}} [\bigodot Post_{3k} \odot Post_{2} \odot Post_{1k}])^{\top} \]

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

\[\forall i \in [1 \ldots m-1], \beta_{1n_{j_0}} [\bigodot Post_{1k}] = \beta_i, \quad \beta_{2n_{j_0}} [\bigodot Post_{1k}] = \beta_m, \text{ and}\]

\[\forall i \in [m+1 \ldots n], \beta_{3n'_{j_0}} [\bigodot Post_{3k} \odot Post_{2} \odot Post_{1k}] = \beta_i\]

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

1. For the first \( \tau \) OTs, i.e. \( p \in [1 \ldots m_1] \). We have:
   Either there is \( i \) such that \( p = n_i \), and thus \( \beta_i \) and \( \beta_{1p_{j_0}} \) are defined. In this case by Lemma 7 we have:
   \[P[Q]_{j_0} = \overline{\gamma_{1p}}, \quad \text{Pred}_1, \text{Post}_1, \text{Pred}_1' = \text{Pred}_1 \wedge (\beta_{1p_{j_0}} = \alpha_i \wedge \text{Pred}_i)\]
   \[\text{Post}_1' = \text{Post}_1 \uplus \text{Post}_i\]
   Or \( j_0 \notin \text{dom}(\beta_{1p}) \) 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 \( p \in [n_i \ldots n_{i+1}] \), and
   \[P[Q]_{j_0} = \overline{\beta_{1p}}, \text{Pred}_1, \text{Post}_1, \text{Pred}_1' = \text{Pred}_1 \wedge (\beta_{1p_{j_0}} = \alpha_i \wedge \text{Pred}_i)\]
   \[\text{Post}_1' = \text{Post}_1 \uplus \text{Post}_i\]

and thus we also have a weak OT by Definition 13 (rule (WT2)):
2. Similarly, for the middle OT with label $\alpha$:

$$P[Q]_{j_0} \models \overrightarrow{\gamma_2}, \text{Pred}_2', \text{Post}_2'$$

with

$$\overrightarrow{\gamma_2} = \gamma_{m_j} \cup (\beta_{2j})^\nu_2 \ \text{with} \ \text{Pred}_2' = \text{Pred}_2 \land (\beta_{2j} = \alpha_m \land \text{Pred}_m)$$

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

3. For the last $\tau$ 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 $\beta_i$ and $\beta_{1p,j_0}$ are defined. In this case by Lemma 7 we have:

$$P[Q]_{j_0} \models \overrightarrow{\gamma_3}, \text{Pred}_3', \text{Post}_3'$$

with

$$\overrightarrow{\gamma_3} = \gamma_{i_j} \cup (\beta_{3j})^\nu_3 \ \text{with} \ \text{Pred}_3' = \text{Pred}_3 \land (\beta_{3j} = \alpha_i \land \text{Pred}_i)$$

and thus we also have a weak OT by definition 13 (rule WT3), we obtain:

$$P[Q]_{j_0} \models \overrightarrow{\gamma}, \text{Pred}'', \text{Post}''$$

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

$$P[Q]_{j_0} \models \overrightarrow{\gamma}, \text{Pred}'', \text{Post}''$$
where

\[
\alpha'' = \alpha' \left\langle \begin{array}{c}
1 \\
j \in m_1
\end{array} \right\rangle \ \\
\mathcal{\gamma} = \bigoplus_{i=1}^{m_1} \mathcal{\gamma}_i \left\langle \begin{array}{c}
1 \\
k = i - 1
\end{array} \right\rangle \ \\
\bigoplus_{i=1}^{m_2} \mathcal{\gamma}_i \left\langle \begin{array}{c}
1 \\
k = m_1
\end{array} \right\rangle \ \\
Pred'' = \bigwedge_{i=1}^{m_1} \left\langle \begin{array}{c}
1 \\
j \in [j_0]
\end{array} \right\rangle \ \\
\bigwedge_{i=1}^{m_2} \left\langle \begin{array}{c}
1 \\
k = [i_1, i_2]
\end{array} \right\rangle \ \\
\bigwedge_{i=1}^{m_1} \left\langle \begin{array}{c}
1 \\
j \in [j_0]
\end{array} \right\rangle \ \\
Post'' = \bigvee_{j=m_2}^{j=m_1} \left\langle \begin{array}{c}
1 \\
j \in [j_0]
\end{array} \right\rangle \ \\
\bigvee_{j=m_1}^{j=m_2} \left\langle \begin{array}{c}
1 \\
j \in [j_0]
\end{array} \right\rangle \ \\
\bigvee_{j=m_1}^{j=m_2} \left\langle \begin{array}{c}
1 \\
j \in [j_0]
\end{array} \right\rangle \ \\
\bigvee_{j=m_1}^{j=m_2} \left\langle \begin{array}{c}
1 \\
j \in [j_0]
\end{array} \right\rangle
\]

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:

\[
\alpha'' = \alpha' \left\langle \begin{array}{c}
1 \\
j \in m_1
\end{array} \right\rangle \\
\alpha' \text{ only contains variables of } P \text{ untouched by } Post_i \\
\alpha
\]

For \(\mathcal{\gamma}\) 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 \(Post_i\) has no effect on variables of \(P\) and on \(\beta_{1pj}\), consequently we have:

\[
\gamma_j' = \bigoplus_{i=1}^{m_1} (\beta_{1ij} \left\langle \begin{array}{c}
1 \\
k = i - 1
\end{array} \right\rangle \bigvee \bigoplus_{k=m_1}^{k=m_1} \left\langle \begin{array}{c}
1 \\
k = m_1
\end{array} \right\rangle) \ \\
\bigoplus_{i=1}^{m_2} (\beta_{2ij} \left\langle \begin{array}{c}
1 \\
k = i - 1
\end{array} \right\rangle \bigvee \bigoplus_{k=m_1}^{k=m_1} \left\langle \begin{array}{c}
1 \\
k = m_1
\end{array} \right\rangle) \ \\
\bigoplus_{i=1}^{m_1} (\beta_{3ij} \left\langle \begin{array}{c}
1 \\
k = i - 1
\end{array} \right\rangle \bigvee \bigoplus_{k=m_1}^{k=m_1} \left\langle \begin{array}{c}
1 \\
k = m_1
\end{array} \right\rangle)
\]
Second, when \( j \in J \) for some \( t \), \( \gamma_j' \) is the concatenation of elements of \( \gamma_{1ij}, \gamma_{2ij}, \gamma_{3ij} \) that are not empty. By construction the concatenation of these elements is \( \gamma_tj \), for \( t \in [0..n] \). \( Post_{tk} \) has no effect on \( \gamma_tj \) but \( Post_k \) has. We obtain:

\[
\gamma' = \sum_{i=1}^{m_1} \gamma'_{1ij} \{ 1 \} Post'_{1k} \oplus \gamma'_{2ij} \{ 1 \} Post'_{1k} \oplus \sum_{i=1}^{m_2} \gamma'_{3ij} \{ 1 \} Post'_{3k} \oplus Post'_{2} \oplus Post'_{1k} \]

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

\[
Pred'' = \left( \bigwedge_{i=1}^{m_1} \text{Pred}^{'}_{11} \{ 1 \} Post^{'}_{1j} \bigwedge_{j=1}^{m_1} \text{Pred}^{'}_{1j} \{ 1 \} Post^{'}_{1j} \right)
\]

\[
\wedge \bigwedge_{i=1}^{m_3} \text{Pred}^{'}_{3i} \{ 1 \} Post^{'}_{3j} \bigwedge_{j=1}^{m_3} \text{Pred}^{'}_{3j} \{ 1 \} Post^{'}_{3j} \right)
\]

\[
= \left( \bigwedge_{j=1}^{m_1} \text{Pred}^{'}_{1j} \{ 1 \} Post^{'}_{1j} \bigwedge_{j=1}^{m_1} \text{Pred}^{'}_{2j} \{ 1 \} Post^{'}_{1j} \right)
\]

\[
\bigwedge_{j=1}^{m_2} \text{Pred}^{'}_{3j} \{ 1 \} Post^{'}_{3j} \bigwedge_{j=1}^{m_2} \text{Pred}^{'}_{3j} \{ 1 \} Post^{'}_{3j} \right)
\]

\[
= \left( \bigwedge_{j=1}^{m_1} \text{Pred}^{'}_{1j} \{ 1 \} Post^{'}_{1j} \bigwedge_{j=1}^{m_1} \text{Pred}^{'}_{1j} \{ 1 \} Post^{'}_{1j} \right)
\]

\[
\bigwedge_{j=1}^{m_2} \text{Pred}^{'}_{3j} \{ 1 \} Post^{'}_{3j} \bigwedge_{j=1}^{m_2} \text{Pred}^{'}_{3j} \{ 1 \} Post^{'}_{3j} \right)
\]

\[
= \left( \bigwedge_{j=1}^{m_1} \text{Pred}^{'}_{1j} \{ 1 \} Post^{'}_{1j} \bigwedge_{j=1}^{m_1} \text{Pred}^{'}_{1j} \{ 1 \} Post^{'}_{1j} \right)
\]

\[
\bigwedge_{j=1}^{m_2} \text{Pred}^{'}_{3j} \{ 1 \} Post^{'}_{3j} \bigwedge_{j=1}^{m_2} \text{Pred}^{'}_{3j} \{ 1 \} Post^{'}_{3j} \right)
\]

\[
= \left( \bigwedge_{j=1}^{m_1} \text{Pred}^{'}_{1j} \{ 1 \} Post^{'}_{1j} \bigwedge_{j=1}^{m_1} \text{Pred}^{'}_{1j} \{ 1 \} Post^{'}_{1j} \right)
\]

\[
\bigwedge_{j=1}^{m_2} \text{Pred}^{'}_{3j} \{ 1 \} Post^{'}_{3j} \bigwedge_{j=1}^{m_2} \text{Pred}^{'}_{3j} \{ 1 \} Post^{'}_{3j} \right)
\]

\[
= \text{Pred}
\]
Finally, concerning post-conditions:

\[
P_{\text{Post}}'' = \bigoplus_{j=m_2}^{1} \text{Post}'_j \circ \text{Post}_j' \bigoplus_{j=m_1}^{1} \text{Post}''_j
\]

This allows us to conclude concerning the lemma.

**Theorem 7** *Congruence.* Consider an open pNet: \( P = \langle P_i^e \rangle, \text{Sort}_j^{e \in I}, SV \rangle \). Let \( j_0 \in J \) be a hole. Let \( Q \) and \( Q' \) be two weak FH-bisimilar pNets such that \( \text{Sort}(Q) = \text{Sort}(Q') = \text{Sort}_{j_0} \). 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 \( R_{Q,Q'} \) relating the two pNets \( Q \) and \( Q' \). We define a relation \( R \) relating states of \( P[Q]_{j_0} \) with states of \( P[Q']_{j_0} \):

\[ R = \{ \langle S_P \cup S_Q \rangle, \langle S_P \cup S_Q' \rangle, \text{Pred}_{Q,Q'}, \text{Post}_{Q,Q'} \} \subseteq R_{Q,Q'} \]

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):

\[ P[Q]_{j_0} \models \beta_j^{e \in I} (P \cup P_{j_0} \cup S_Q \cup S_{Q'} \cup \text{Pred}_{Q,Q'} \cup \text{Post}_{Q,Q'}) \]

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

\[ P \models \beta_j^{e \in I} (P \cup S_P \cup S_Q \cup S_{Q'} \cup \text{Pred}_{Q,Q'}) \]

and

\[ Q \models \beta_j^{e \in I} (Q \cup S_Q \cup S_{Q'} \cup \text{Pred}_{Q,Q'}) \]

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

\[ \forall x. (S_Q', S_{Q'} | \text{Pred}_{Q,Q'}) \in R_{Q,Q'} \]
and

$$P_{Q, Q'} \wedge \text{Pred}_Q \Rightarrow \left( \bigvee_{x \in X} (\forall j \in J_Q. (\beta_j)^X = \gamma_{jx}) \Rightarrow (\text{Pred}_Q \wedge \alpha_Q = \alpha_x \wedge \text{Pred}_{Q, Q'} \langle \langle \text{Post}_P \cup \text{Post}_{Q'} \rangle \rangle) \right)$$

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

$$P'[Q'_j]_0 \models \langle \langle \beta_j^{x \in J_P} \cup \gamma_j^{x \in J_Q}, \text{Pred}_P \wedge \text{Pred}_{Q', Q'}, \text{Post}_P \cup \text{Post}_{Q'} \rangle \rangle \quad \Rightarrow \quad \langle S'_P \cup S'_{Q'} \rangle \xrightarrow{\alpha} \langle S'_P \cup S'_{Q'} \rangle$$

with

$$\left( \bigvee_{x \in X} (\forall j \in J_Q. (\beta_j)^X = \gamma_{jx} \Rightarrow \alpha_Q = \alpha_x) \right)$$

that ensures that the open transitions can be recomposed when the OT fires.

Side conditions necessary to prove weak-FH bisimulations are:

$$\forall x. (S'_P \cup S'_{Q}, S'_P \cup S'_{Q'} \cup \text{Pred}_{Q, Q'}) \in \mathcal{R}$$

which is true, and

$$\text{Pred}_{Q, Q'} \wedge \text{Pred}_P \wedge \text{Pred}_Q \Rightarrow \left( \bigvee_{x \in X} (\forall j \in J_Q. (\beta_j)^X = \gamma_{jx} \wedge \forall j \in J_P. (\beta_j)^X = (\beta_j)^X) \Rightarrow (\text{Pred}_P \wedge \text{Pred}_{Q, Q'} \langle \langle \text{Post}_P \cup \text{Post}_{Q'} \cup \text{Post}_Q \rangle \rangle) \right)$$

We conclude by observing that $\text{Post}_P$ has no effect on variables of $Q$ and $Q'$, and thus on $\text{Pred}_{Q, Q'}$.

**Theorem 8.** Context equivalence. Consider two FH-bisimilar open pNets:

$P = \langle \langle P_i^{x \in I}, \text{Sort}_i^{x \in I}, \mathcal{S} \rangle \rangle$ and $P' = \langle \langle P'_i^{x \in I}, \text{Sort}_i^{x \in I}, \mathcal{S}' \rangle \rangle$ (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 $\text{Sort}(Q) = \text{Sort}_{j_0}$. 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 $P'[Q]_{j_0}$:

$$\mathcal{R} = \{ (\alpha S_P \cup S_{Q'}, \langle S_{P'} \cup S_{Q} \rangle, \text{Pred}_{P, P'}) | (S_P, S_{P'}, \text{Pred}_{P, P'}) \in \mathcal{R}_{P, P'} \}$$

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):

$$P'[Q]_{j_0} \models \langle \langle \beta_j^{x \in J_P \cup J_Q}, \text{Pred}_P \wedge \text{Pred}_Q \wedge \text{Pred}_P \cup \text{Post}_Q \rangle \rangle \quad \Rightarrow \quad \langle S'_{P} \cup S'_{Q} \rangle \xrightarrow{\alpha} \langle S'_{P} \cup S'_{Q} \rangle$$
By the decomposition lemma for OTs (Lemma 1), we obtain the 2 following OTs (equality side-conditions have been unlined for clarity):

\[
P \models \begin{array}{c}
\beta_j \in J_P \cup (j_0 \rightarrow \alpha_Q), \text{Pred}_P, \text{Post}_Q
\end{array} \xrightarrow{\text{S}_P} \begin{array}{c}
\text{S}_{P'}
\end{array} \quad \text{and} \quad Q \models \begin{array}{c}
\beta_j \in J_Q \cup \text{Pred}_Q, \text{Post}_Q
\end{array} \xrightarrow{\text{S}_Q} \begin{array}{c}
\text{S}_{Q'}
\end{array}
\]

With \( \text{Pred} \iff \alpha_Q = \beta_{j_0} \)

By definition of \( R \) we have \((\text{S}_P, \text{S}_{P'}, \text{Pred}_{P, P'}) \in R_{P, P'} \). And thus, by definition of weak FH-bisimulation, there exists a family of weak open transitions \( \text{WOT}_x \):

\[
\gamma_j \in J_P \cup \{j_0\}, \text{Pred}_{P', P_x}, \text{Post}_{P_x}
\xrightarrow{\text{S}_{P'}} \begin{array}{c}
\text{S}'_{P', P_x}
\end{array}
\]

where

\[
\forall x. (\text{S}'_P, \text{S}'_{P', x}, \text{Pred}_{P, P_x}) \in R_{P, P'}
\]

and

\[
\text{Pred}_{P, P'} \land \text{Pred}_P \iff \left( \bigvee_{x \in X} (\forall j \in J_P. (\beta_j)^\forall = \gamma_{jx} \land (\alpha_Q)^\forall = \gamma_{j_0}) \Rightarrow (\text{Pred}_{P'} \land \alpha = \alpha_x \land \text{Pred}_{P, P_x} \{\text{Post}_{P', x} \cup \text{Post}_P\}) \right)
\]

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 \( \gamma_{j_0} = [\beta] \):

\[
P'[Q]_{j_0} \models \begin{array}{c}
(\beta_j \in J_Q \cup \{j_0\}, \text{Pred}_{P', P_x} \land \text{Pred}_Q \land \alpha_Q = \beta, \text{Post}_{P', x} \cup \text{Post}_Q
\end{array} \xrightarrow{\text{S}'_{P'} \cup \text{S}_Q} \begin{array}{c}
\text{S}'_{P', Q'}
\end{array}
\]

Side conditions necessary to prove weak FH-bisimulations are:

\[
\forall x. (\text{S}'_P \cup \text{S}'_Q, \text{S}'_{P', x} \cup \text{S}'_Q, \text{Pred}_{P, P_x}) \in R
\]

which is true, and

\[
\text{Pred}_{P, P'} \land \text{Pred}_P \land \text{Pred}_Q \land \text{Pred} \Rightarrow
\left( \bigvee_{x \in X} (\forall j \in J_P. (\beta_j)^\forall = \gamma_{jx} \land \forall j \in J_Q. (\beta_j)^\forall = (\beta_j)^\forall) \Rightarrow (\text{Pred}_{P', x} \land \text{Pred}_Q \land \alpha_Q = \beta \land \alpha_x = \alpha \land \text{Pred}_{P, P_x} \{\text{Post}_{P', x} \cup \text{Post}_P \cup \text{Post}_Q\}) \right)
\]

We conclude by observing that \( \text{Post}_Q \) has no effect on variables of \( P \) and \( P' \), and thus on \( \text{Pred}_{P, P'} \) and \( \text{Pred} \) 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. 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 \texttt{in(msg)} or constants without argument, typically as \texttt{tau().}

- Variables can be declared as global variables of a pLTS (e.g. \texttt{m_msg} in \texttt{PerfectBuffer}), or a pNet Node in the case of synchronisation vector variables (e.g. \texttt{p_a}), or as input variables in a pLTS, as \texttt{?msg} in \texttt{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 \texttt{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:

```plaintext
SimpleProtocolImpl:
  import "Data_Alg.Funcp"
  root SimpleProtocol
  const in, out: Action
  const tau, p_send, q_recv, n_send, n_recv, m_send, m_recv, m_error: 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 := m, 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(), _, _>-> 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 n: Data c: Nat
  vector SV0 <p_send(n), in(m)>-> synchro(in(m))
  vector SV1 <p_a, _>-> p_a [p_a != p_send(x)]
  vector SV2 <_, out(n, ec), qrecv(m, ec)>-> synchro(out(m, ec))
  vector SV3 <_, _, q_b>-> q_b [q_b != q_recv(x, y)]
```

90
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 $\tau$" 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.

$W_r = \{\}, True, ()$

$S \xrightarrow{\ast} S$

$W_1 = \{P\rightarrow p-a], [\forall x.p-a \neq p-send(x)], ()$

$S \xrightarrow{\{p-a\}} S$

$W_2 = \{Q\rightarrow q-b], [\forall x,y.q-b \neq q-recev(x,y)], ()$

$S \xrightarrow{\{q-b\}} S$

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.

$WI_3(n) = \{P\rightarrow p-send(m)], True, (s\_msg \leftarrow m, s\_ec \leftarrow n)$

$\{000, 202\} \xrightarrow{\{\texttt{in}(m)\}} 100$

$WI_3a(n) = \{P\rightarrow p-send(m)], True, (m\_msg \leftarrow m, m\_ec \leftarrow n, s\_ec \leftarrow n)$

$\{000, 202\} \xrightarrow{\{\texttt{in}(m)\}} 210$

$WI_3b(n) = \{P\rightarrow p-send(m)], True, (s\_ec \leftarrow n)$

$\{000, 202\} \xrightarrow{\{\texttt{in}(m)\}} 220$
\[ WI_{3e}(n) = \{ P \rightarrow p\text{-send}(m) \}, \text{True}, (r_{msg} \leftarrow m, r_{ec} \leftarrow n) \]
\[ \rightarrow \{000, 202\} \xrightarrow{in(m)} 201 \]
\[ WI_{4}(n) = \{ \}, \text{True}, (m_{msg} \leftarrow s_{msg}, m_{ec} \leftarrow s_{ec} + n, s_{ec} \leftarrow s_{ec} + n) \]
\[ \rightarrow 100 \Rightarrow 210 \]
\[ WI_{4a}(n) = \{ \}, \text{True}, (s_{ec} \leftarrow s_{ec} + n) \]
\[ \rightarrow 100 \Rightarrow 220 \]
\[ WI_{5}(n) = \{ \}, \text{True}, (s_{ec} \leftarrow s_{ec} + n) \]
\[ \rightarrow 210 \Rightarrow 220 \]
\[ WI_{5a}(n) = \{ \}, \text{True}, (s_{ec} \leftarrow s_{ec} + 1 + n) \]
\[ \rightarrow 210 \Rightarrow 100 \]
\[ WI_{6}(n) = \{ \}, \text{True}, (s_{ec} \leftarrow s_{ec} + 1 + n) \]
\[ \rightarrow 220 \Rightarrow 100 \]
\[ WI_{6a}(n) = \{ \}, \text{True}, (m_{msg} \leftarrow s_{msg}, m_{ec} \leftarrow s_{ec} + 1 + n, s_{ec} \leftarrow s_{ec} + 1 + n) \]
\[ \rightarrow 220 \Rightarrow 210 \]

Because

\[ Post_{6a} = post_{4} \circ post_{456} \circ post_{6} \]
\[ = ((m_{msg} \leftarrow s_{msg}, m_{ec} \leftarrow s_{ec}) \circ (s_{ec} \leftarrow s_{ec} + n)) \circ (s_{ec} \leftarrow s_{ec} + 1) \]
\[ = (m_{msg} \leftarrow s_{msg}, m_{ec} \leftarrow (s_{ec} + 1) + n, s_{ec} \leftarrow (s_{ec} + 1) + n) \]

\[ WI_{456e}(n) = \{ \}, \text{True}, (s_{ec} \leftarrow s_{ec} + n) \]
\[ \rightarrow 100 \Rightarrow 100 \]
\[ WI_{564e}(n) = \{ \}, \text{True}, (m_{msg} \leftarrow s_{msg}, s_{ec} \leftarrow s_{ec} + 1 + n, m_{ec} \leftarrow s_{ec} + 1 + n) \]
\[ \rightarrow 210 \Rightarrow 210 \]
\[ WI_{645e}(n) = \{ \}, \text{True}, (s_{ec} \leftarrow s_{ec} + 1 + n) \]
\[ \rightarrow 220 \Rightarrow 220 \]
\[ WI_{7}(n) = \{ \}, \text{True}, (r_{msg} \leftarrow s_{msg}, r_{ec} \leftarrow s_{ec} + n) \]
\[ \rightarrow 210 \Rightarrow 201 \]
\[ WI_{7a}(n) = \{ \}, \text{True}, (r_{msg} \leftarrow s_{msg}, r_{ec} \leftarrow m_{ec} + n) \]
\[ \rightarrow 220 \Rightarrow 201 \]
\[ \forall n \geq 1. WI_{7b}(n) = \{ \}, \text{True}, (r_{msg} \leftarrow m_{msg}, r_{ec} \leftarrow s_{ec} + n) \]
\[ \rightarrow 100 \Rightarrow 201 \]
\[ WI_{8} = \{ Q \rightarrow q\text{-recv}(r_{1}\_msg, r_{1}\_ec) \}, \text{True}, () \]
\[ \xrightarrow{\text{out}(r_{1}\_msg, r_{1}\_ec)} \{202, 000\} \]
\[ \forall n \geq 1. WI_{8a}(n) = \{ Q \rightarrow q\text{-recv}(m_{msg}, m_{ec} + n) \}, \text{True}, () \]
\[ \xrightarrow{\text{out}(m_{msg}, m_{ec} + n)} \{202, 000\} \]
\[ \forall n \geq 1. WI_{8b}(n) = \{ Q \rightarrow q\text{-recv}(s_{msg}, s_{ec} + n) \}, \text{True}, () \]
\[ \xrightarrow{\text{out}(s_{msg}, s_{ec} + n)} \{202, 000\} \]
\[ \forall n \geq 1. WI_{S_e}(n) = \{ (s,s) | s \in \text{States of WOA} \} \cup \{(202,000)\} \]

Then for all \( \tau \) transitions above we have a similar WOT that include a non-\( \tau \) move from an external action of \( P \) or \( Q \), like for example:

\[ WI_4 P(n) = \{ m_{\text{msg}} \leftarrow s_{\text{msg}}, m_{\text{ec}} \leftarrow s_{\text{ec}} + n, s_{\text{ec}} \leftarrow s_{\text{ec}} + n \}
\]

\[ 100 \xrightarrow{\text{out}(s_{\text{msg}}, s_{\text{ec}} + n)} 210 \]

and \( WI_4 Q(n) = \{ m_{\text{msg}} \leftarrow s_{\text{msg}}, m_{\text{ec}} \leftarrow s_{\text{ec}} + n, s_{\text{ec}} \leftarrow s_{\text{ec}} + n \}
\]

\[ 100 \xrightarrow{\text{out}(s_{\text{msg}}, s_{\text{ec}} + n)} 210 \]

but also e.g.:

\[ WI_{4564} P(n) = \{ P \rightarrow p-a \}, \forall x.p-a \neq p\text{-send}(x) \}
\]

\[ (s_{\text{msg}} \leftarrow s_{\text{msg}}, s_{\text{ec}} \leftarrow s_{\text{ec}} + n) \]

\[ 100 \xrightarrow{\text{out}(s_{\text{msg}}, s_{\text{ec}} + n)} 100 \]

The following table give a summary of WOTs, when sharing their names as much as possible.

<table>
<thead>
<tr>
<th>WOT name</th>
<th>Pairs of source states and target states</th>
<th># WOTs</th>
</tr>
</thead>
<tbody>
<tr>
<td>( WI_1 ) ( WI_2 ) ( WI_\tau )</td>
<td>{(s,s)</td>
<td>s \in \text{States of WOA} } \cup {(202,000)}</td>
</tr>
<tr>
<td>( WI_{3a} ) ( n )</td>
<td>{(202,100),(000,100)}</td>
<td>2</td>
</tr>
<tr>
<td>( WI_{3b} ) ( n )</td>
<td>{(202,210),(000,210)}</td>
<td>2</td>
</tr>
<tr>
<td>( WI_{3c} ) ( n )</td>
<td>{(202,220),(000,220)}</td>
<td>2</td>
</tr>
<tr>
<td>( WI_{3a} ) ( n ) ( WI_4 P(n) )</td>
<td>{(202,201),(000,201)}</td>
<td>2</td>
</tr>
<tr>
<td>( WI_{4} ) ( n )</td>
<td>{(100,210)}</td>
<td>3</td>
</tr>
<tr>
<td>( WI_{4a} ) ( n ) ( WI_{4a} P(n) ) ( WI_{4a} Q(n) )</td>
<td>{(100,220)}</td>
<td>3</td>
</tr>
<tr>
<td>( WI_{4564} ) ( n )</td>
<td>{(100,100)}</td>
<td>3</td>
</tr>
<tr>
<td>( WI_{3} ) ( n ) ( WI_{3} P(n) ) ( WI_{3} Q(n) )</td>
<td>{(100,200)}</td>
<td>3</td>
</tr>
<tr>
<td>( WI_{3a} ) ( n ) ( WI_{5a} P(n) ) ( WI_{5a} Q(n) )</td>
<td>{(100,200)}</td>
<td>3</td>
</tr>
<tr>
<td>( WI_{644} ) ( n ) ( WI_{644} P(n) ) ( WI_{644} Q(n) )</td>
<td>{(100,200)}</td>
<td>3</td>
</tr>
<tr>
<td>( WI_{6} ) ( n )</td>
<td>{(202,100)}</td>
<td>3</td>
</tr>
<tr>
<td>( WI_{6a} ) ( n )</td>
<td>{(202,210)}</td>
<td>3</td>
</tr>
<tr>
<td>( WI_{454} ) ( n ) ( WI_{454} P(n) ) ( WI_{454} Q(n) )</td>
<td>{(202,220)}</td>
<td>3</td>
</tr>
<tr>
<td>( WI_{7} ) ( n ) ( WI_{7} P(n) )</td>
<td>{(202,200)}</td>
<td>3</td>
</tr>
<tr>
<td>( WI_{7a} ) ( n ) ( WI_{7a} P(n) ) ( WI_{7a} Q(n) )</td>
<td>{(202,200)}</td>
<td>3</td>
</tr>
<tr>
<td>( WI_{7b} ) ( n ) ( WI_{7b} P(n) )</td>
<td>{(202,200)}</td>
<td>3</td>
</tr>
<tr>
<td>( WI_{8} ) ( n )</td>
<td>{(202,200),(201,000)}</td>
<td>2</td>
</tr>
<tr>
<td>( WI_{8a} ) ( n )</td>
<td>{(210,202),(210,000)}</td>
<td>2</td>
</tr>
<tr>
<td>( WI_{8b} ) ( n )</td>
<td>{(220,202),(220,000)}</td>
<td>2</td>
</tr>
<tr>
<td>( WI_{8c} ) ( n )</td>
<td>{(100,200),(100,000)}</td>
<td>2</td>
</tr>
</tbody>
</table>

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 $R$ that is the candidate for our weak FH-bisimulation relation:

<table>
<thead>
<tr>
<th>$SimpleProtocolSpec$ states</th>
<th>$SimpleProtocolImpl$ states</th>
<th>Predicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_0$</td>
<td>000</td>
<td>True</td>
</tr>
<tr>
<td>$b_0$</td>
<td>202</td>
<td>True</td>
</tr>
<tr>
<td>$b_1$</td>
<td>100</td>
<td>$b_{msg} = s_{msg} \land b_{ec} = s_{ec}$</td>
</tr>
<tr>
<td>$b_1$</td>
<td>210</td>
<td>$b_{msg} = m_{msg} \land b_{ec} = m_{ec}$</td>
</tr>
<tr>
<td>$b_1$</td>
<td>220</td>
<td>$b_{msg} = s_{msg} \land b_{ec} = s_{ec}$</td>
</tr>
<tr>
<td>$b_1$</td>
<td>201</td>
<td>$b_{msg} = r_{msg} \land b_{ec} = r_{ec}$</td>
</tr>
</tbody>
</table>

Consider the first triple $<b_0, 000, True>$, 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):

- $SS_1 << WI_1$
- $SS_2 << WI_2$
- $SS_3 << WI_3$
- $SI_1 << WS_1$
- $SI_2 << WS_2$
- $SI_3 << WS_3$

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 $b_0$ 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 $\tau$ as an action of a sub-net are of the form “$<-, \tau, ->$”.

- 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 $m_1$ for $SimpleProtocolSpec$ and $m_2$ for $SimpleProtocolImpl$.

In our running example in page 34 we have shown the proof for one of the transitions of ($b_0, 202, True$), namely that $SS_3$ is covered by $WI_3(0)$. We give here another example with $SS_1 << WI_1$, from the first triple ($b_0, 000, True$). It includes less trivial predicates in the OTs:

$$SS_1 = \begin{cases} \{p \rightarrow p-a_1\}, \forall m_1.p-a_1 \neq p-send(m_1),() \\ b_0 \xrightarrow{p-s_1} b_0 \end{cases}$$
Let us check formally the conditions:

- Their sets of active (non-silent) holes is the same: $J' = J_x = \{P\}$.
- Triple $(b0, 000, True)$ is in $\mathcal{R}$.
- The verification condition

\[
\forall f \psi_{OT}. \{Pred \land Pred_{OT} \implies \bigvee_{x \in X} [\exists f \psi_{OT_x}. (\forall j \in J_x. (\beta_j) \land \gamma_j \land Pred_{OT_x} \land \alpha = \alpha_x \land \text{Pred}^x \text{ Post}_{OT} \sqcup \text{Post}_{OT_x}] \}
\]

Gives us:

\[
\forall p-a_1. \{True \land \forall m. p-a_1 \neq p\text{-send}(m) \implies \exists p-a_2. (p-a_1 = p-a_2 \land \forall m. p-a_2 \neq p\text{-send}(m) \land p-a_1 = p-a_2 \land True) \}
\]

That is trivially true, choosing $p-a_2 = p-a_1$ for each given $p-a_1$.

All others pairs from this set are just as easily proven true.