Compositional analysis of modular Petri nets using hierarchical state space abstraction.

Yves-Stan Le Cornec

Compositional analysis of modular Petri nets using hierarchical state space abstraction

Yves-Stan Le Cornec

IBISC, University of Évry, 23 bd de France, 91037 Évry, France
yves-stan.lecornec@ibisc.univ-evry.fr

Abstract. We propose an approach to perform efficient model-checking of μ-calculus formulae on modular Petri nets. Given a formula φ, each module can be analysed separately, possibly yielding a conclusion about the truth value of φ on the global system. When no conclusion can be drawn locally, a minimal state space preserving φ is computed for the module and can be incrementally composed with others, thus enabling for hierarchical analysis of a modular Petri net in a bottom-up fashion.

1 Introduction

State space explosion is a well known problem when dealing with model-checking of large systems. One way to address this problem in the context of Petri nets is *modularity*: a large Petri net is decomposed into subsystems which are then synchronised on shared places or transitions [2]. When only transitions are shared across sub-systems, like in [6], one way to alleviate state space explosion is to build a *modular state space* that consists of the state space of its subsystems (*i.e.*, its modules) and a synchronisation graph of them. This is usually a much smaller object than the state space of the full Petri net (*i.e.*, with all modules combined).

In this paper, we propose another way to analyse modular Petri nets when the goal is to model-check properties expressed as modal μ-calculus formulae, *i.e.*, to verify whether the state space of a modular Petri net is a model for a given formula φ. Our approach allows to analyse modules independently of each other. When the formula depends only on one module, only this particular module needs to be analysed. When the formula depends on several modules, each can be processed separately and its state space minimised before being combined with the others, *i.e.*, we compute a smaller transition system that is equivalent to the initial one with respect to the formula φ. This minimisation is an extension to the modal μ-calculus of the approach defined in [1] for CTL. Furthermore, our approach is fully compositional: on the one hand, the semantics of any composition of modules is equivalent to the synchronised product of the individual semantics of each module; on the other hand, minimisation of the semantics preserves the truth value of φ. So, a system composed of several modules can be decomposed into an arbitrary hierarchy forming a tree in which each leaf is a module and each internal node corresponds to the composition of the modules below it. Analysis can be performed by traversing this tree bottom-up in such a way that, at each node, we consider a particular subsystem whose

semantics can be computed and analysed so that, either we can raise some global conclusion about the truth of φ, or we can minimise the semantics (which will be reused at the upper level) while preserving the truth of φ. This approach leaves a lot of room to define strategies to choose an optimal order of analysis of modules in order to minimise the amount of work necessary to bring the conclusion. The current paper concentrates on defining the analysis method, this kind of optimisations being left to future work.

The rest of the paper is organised as follows. In the next section, we recall main definitions about modular Petri nets and define the semantics in terms of labelled transition systems. Next, we define the modal μ-calculus logic and its semantics. Section 4 forms the core of our contribution, defining the formula-dependent abstraction and giving the main results that enable hierarchical analysis and abstraction. For readability, proofs are moved in the appendix, after a conclusion section with perspectives.

2 Modular Petri nets

In the following, we consider place/transitions nets for simplicity, but a generalisation to high-level Petri nets (in particular to coloured Petri nets) is straightforward because our work is based on the labelled transitions systems used for the Petri nets semantics. To start with, let us recall the definition of Petri nets to fix the notations.

Definition 1. *A* Petri net $N \stackrel{\text{df}}{=} (P, T, W)$ *is a tuple such that:*

- P *is the finite set of* places*;*
- T *is the finite set of* transitions*, such that* $P \cap T = \emptyset$*;*
- W *is a multiset over* $(P \times T) \cup (T \times P)$ *defining the* arcs weights*;*

For $t \in T$*, we denote by* ${}^\bullet t$ *(resp.,* $t^\bullet$*) the multiset over* P *such that for all* $s \in P$*,* ${}^\bullet t(s) \stackrel{\text{df}}{=} W(s,t)$ *(resp.,* $t^\bullet(s) \stackrel{\text{df}}{=} W(t,s)$*).*

A marking M *of* N *is a multiset over* P *indicating how many tokens each place holds. A transition* t *is* enabled *at marking* M *iff* ${}^\bullet t \leq M$*, in which case the* firing *of* t *yields a new marking* $M' \stackrel{\text{df}}{=} M - {}^\bullet t + t^\bullet$*. This is denoted by* $M\ [t\rangle\ M'$*, moreover, we denote by* $[M\rangle$ *the smallest set containing* M *such that if* $M' \in [M\rangle$ *and* $M'\ [t\rangle\ M''$ *then* $M'' \in [M\rangle$*. We assume that* $[M\rangle$ *is finite.*

Our definition of modular Petri nets is adapted from [6]. We use here non-disjoint sets of transitions instead of explicit *transitions fusion sets* to define the transitions shared across modules. This especially means that we would have to make copies of a transition in order to model a choice between different synchronizations.

Definition 2. *A* modular Petri net *is a collection of* modules $(N_1, \ldots N_n)$ *where each* N_i *is a Petri net* (P_i, T_i, W_i)*, and such that the* P_i*'s are pairwise disjoint. Transitions that belong to only one* T_i *are called* local *while those shared among at least two* T_i*'s are called* fused*. Such a modular net is equivalent to a* flat Petri

net *obtained as the component-wise union of its modules. Because this union is commutative and associative, we shall use a binary notation for it:* $N_1 \oplus \cdots \oplus N_n$.

Example 1. Figure 1 shows two modules which are part of a modular Petri net. Transition f_3 is assumed to be fused with another module not shown here. $\diamond$

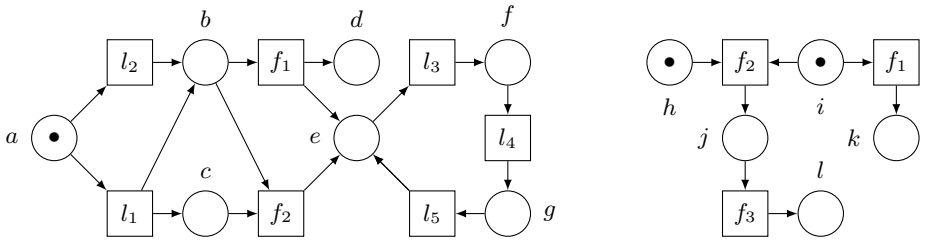

Fig. 1. Two modules part of a modular Petri net.

The semantics of Petri nets and modular Petri nets can be defined in terms of *labelled transitions systems* (*LTS*).

2.1 LTS semantics of Petri nets and modular Petri nets

A LTS is a tuple $S \stackrel{\text{df}}{=} (Q, q_0, A, R, L)$ where Q is a set of *states*, $q_0 \in Q$ is the *initial state*, A is the set of *actions* used as transition labels, $R \subseteq Q \times A \times Q$ is the set of *transitions* and L is a labelling of states with Boolean formulae on propositional variables from a set $\mathbb{V}$. A transition $(q, a, q') \in R$ is usually denoted by $q \xrightarrow{a} q'$.

Definition 3. *The* LTS semantics of a Petri net $N \stackrel{\text{df}}{=} (P, T, W)$ *initially marked by* M_0 *is the LTS* $[\![N]\!] \stackrel{\text{df}}{=} (Q, q_0, A, R, L)$ *such that:* $Q \stackrel{\text{df}}{=} [M_0\rangle$*;* $q_0 \stackrel{\text{df}}{=} M_0$*;* $A \stackrel{\text{df}}{=} T$*;* $R \stackrel{\text{df}}{=} \{M \xrightarrow{t} M' \mid M\ [t\rangle\ M'\}$*; and* $L(M) \stackrel{\text{df}}{=} \bigwedge_{M(p)>0} \texttt{p=M(p)}$ *with* $\mathbb{V} \stackrel{\text{df}}{=} \{\texttt{p=k} | p \in P, k \in \mathbb{N}^+\}$.

In this definition, states are labelled by a conjunction of propositional variables of the form `s=k` denoting the number of tokens in each non void place. This choice is arbitrary and can be changed in many ways, this will not affect the current work as long as we are able to evaluate atomic formulae. In order to bring modularity at the semantics level, we shall partition A as $A^{\mathsf{loc}} \uplus A^{\mathsf{fus}}$ corresponding respectively to the local and fused transitions of a modular Petri net.

Example 2. Figure 2 shows the LTS semantics of the modules from example 1.$\diamond$

Definition 4. *The* modular LTS semantics of a modular Petri net $(N_i)_{1 \le i \le n}$*, where* $N_i \stackrel{\text{df}}{=} (P_i, T_i, W_i)$ *for all* i*, is a collection of LTS* $(Q_i, q_{0i}, A_i, R_i, L_i)_{1 \le i \le n}$ *where each* A_i *is partitioned as* $A_i^{\mathsf{loc}} \uplus A_i^{\mathsf{fus}}$ *such that, for* $1 \le i \le n$*:*

- $(Q_i, q_{0i}, A_i, R_i, L_i) = [\![N_i]\!]$ *is the LTS semantics of* N_i *considered alone;*
- $A_i^{\mathsf{loc}} \stackrel{\text{df}}{=} R_i \setminus \bigcup_{j \neq i} R_j$ *and* $A_i^{\mathsf{fus}} \stackrel{\text{df}}{=} A_i \setminus A_i^{\mathsf{loc}}$.

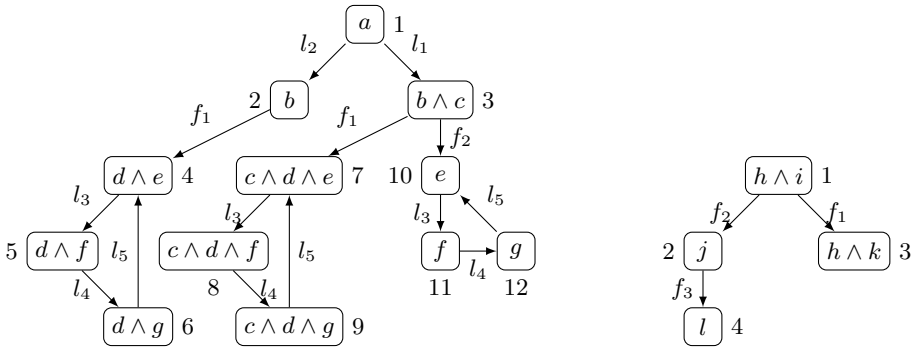

Fig. 2. LTS semantics of the modules from figure 1. For the left LTS, we have $A^{\mathsf{loc}} \stackrel{\text{df}}{=} \{l_1, l_2, l_3, l_4\}$ and $A^{\mathsf{fus}} \stackrel{\text{df}}{=} \{f_1, f_2\}$; for the right LTS we have $A^{\mathsf{loc}} \stackrel{\text{df}}{=} \emptyset$ and $A^{\mathsf{fus}} \stackrel{\text{df}}{=} \{f_1, f_2, f_3\}$. As there is at most one token per place, we write a instead of $a = 1$.

The collection of LTS obtained from a modular Petri net can be transformed into a single LTS by taking the *synchronised product* of its components, where synchronisation takes place on the fused transitions, which is the usual definition of a n-ary synchronised product. We denote by $x[i]$ the i-th component of a tuple x and by $x[i \leftarrow y_i]$ the tuple x in which the i-th component has been replaced by y_i, this latter notation is naturally extended to the replacement of several components.

Definition 5. *Let $(S_i)_{1 \leq i \leq n}$ be the LTS semantics of a modular Petri net with $S_i \stackrel{\text{df}}{=} (Q_i, q_{0_i}, A_i, R_i, L_i)$ and $A_i \stackrel{\text{df}}{=} A_i^{\mathsf{loc}} \uplus A_i^{\mathsf{fus}}$, the* synchronised product *of the S_i's, is the LTS (Q, q_0, A, R, L) defined by:*

- $Q \stackrel{\text{df}}{=} \prod_{1 \leq i \leq n} Q_i$;
- $q_0 \stackrel{\text{df}}{=} (q_{0_1}, \ldots, q_{0_n})$;
- $A \stackrel{\text{df}}{=} \bigcup_{1 \leq i \leq n} A_i$;
- *R is the smallest subset of $Q \times A \times Q$ such that $x \xrightarrow{a} y \in R$ iff either*
  - *it exists i such that $a \in A_i^{\mathsf{loc}}$, $x[i] \xrightarrow{a} y_i \in R_i$ and $y = x[i \leftarrow y_i]$,*
  - *or, for all i such that $a \in A_i^{\mathsf{fus}}$, we have $x[i] \xrightarrow{a} y_i \in R_i$ and $y = x[i \leftarrow y_i]$.*
- *for all $x \in Q$, $L(x) \stackrel{\text{df}}{=} \bigwedge_{1 \leq i \leq n} L_i(x[i])$.*

Because this product is associative and commutative, we shall also use a binary notation for it: $S_1 \otimes \cdots \otimes S_n$.

In the definition of R above, the first point corresponds to the cases where a module evolves on a local transition. So only one component of the compound state evolves. The second point corresponds to the firing of a fused transition, in which case all the modules sharing this transition must simultaneously fire and the corresponding components of the compound state will simultaneously evolve. Notice that, by definition of a fused transition, if $a \in A_i^{\mathsf{fus}}$ for some i,

then there exists at least one $j \neq i$ such that $a \in A_j^{\mathsf{fus}}$ also (otherwise, we would have $a \in A_i^{\mathsf{loc}}$).

From the definitions above, it immediately follows that the synchronised product of the LTS semantics of a modular Petri net is equivalent to the LTS semantics of the flat Petri net.

Theorem 1. *Let $(N_i)_{1 \leq i \leq n}$ be a modular Petri net. We have*

$$[\![N_1 \oplus \cdots \oplus N_n]\!] \quad \sim \quad [\![N_1]\!] \otimes \cdots \otimes [\![N_n]\!]$$

where $\sim$ denotes the isomorphism of LTS. □

Because of this, we can define the notation $[\![N_1, \ldots, N_n]\!] \stackrel{\text{df}}{=} [\![N_1]\!] \otimes \cdots \otimes [\![N_n]\!]$. These notations are intended to put into light a first level of compositionality. For example, consider a modular Petri net $(N_1, \ldots, N_5)$. It is possible to see it as, *e.g.*, three subsystems (N_1, N_2), (N_3, N_4) and N_5 and to compute $[\![N_1, N_2]\!] \otimes [\![N_3, N_4]\!] \otimes [\![N_5]\!]$, just like if we would have considered $(N_1 \oplus N_2, N_3 \oplus N_4, N_5)$ as the initial system, which is also equivalent to $(N_1 \oplus N_2) \oplus (N_3 \oplus N_4) \oplus (N_5)$. So we can decompose a modular Petri net into a hierarchy and compute the semantics at any level of this hierarchy. This is the first step towards full compositionality; the next step will be to introduce LTS minimisation with respect to a μ-calculus formula in order to be able to apply minimisation hierarchically.

3 The modal μ-calculus

The modal μ-calculus (or simply μ-calculus) is a temporal logic that encompasses widely used logics such as, in particular, CTL* (and thus also LTL and CTL) [5]. A μ-formula is derived from the following grammar, where B is a Boolean formula, X is a propositional variable and α is a set of actions:

$$\varphi \quad ::= \quad B \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\alpha\rangle\varphi \mid \mu X.\varphi \mid X$$

Moreover, in a formula $\mu X.\varphi$, φ must be positive in the variable X, *i.e.*, every free occurrence of X must be in the scope of an even number of negations $\neg$.

A formula φ is evaluated over a LTS $S \stackrel{\text{df}}{=} (Q, A, R, L)$ and can be seen as a function of its free variables to 2^Q. In particular, if φ is a closed formula then it is a function with no arguments that returns the subset of Q where φ holds. In formula $\mu X.\langle a\rangle X \vee B$, the sub-formula $\langle a\rangle X \vee B$ defines a function that, given $X \subseteq Q$, returns the states y such that either $y \xrightarrow{a} x$ for some $x \in X$, or B holds on y. From this point of view, $\mu X.\varphi$ is the least fixed point of function φ. More generally, the semantics φ^N of φ over S is defined as follows:

- B holds in every state whose label implies B: $B^S \stackrel{\text{df}}{=} \{x \in Q \mid L(x) \Rightarrow B\}$;
- $\varphi_1 \vee \varphi_2$ holds in every state where φ_1 or φ_2 holds: $(\varphi_1 \vee \varphi_2)^S \stackrel{\text{df}}{=} \varphi_1^S \cup \varphi_2^S$;
- $\neg\varphi$ holds in every state where φ does not: $(\neg\varphi)^S \stackrel{\text{df}}{=} Q \setminus \varphi^S$;
- $\langle\alpha\rangle\varphi$ holds in every state where a transition labelled by $a \in \alpha$ leads to a state where φ holds: $(\langle\alpha\rangle\varphi)^S \stackrel{\text{df}}{=} \{x \in Q \mid \exists y \in \varphi^S, \exists a \in \alpha, x \xrightarrow{a} y \in R\}$;

– $\mu X.\varphi$ is the least fixed point of function φ: $(\mu X.\varphi)^S \stackrel{\text{df}}{=} \bigcap_{\rho \in 2^Q \wedge \varphi(\rho) \subseteq \rho} \rho$;
– $X^S \stackrel{\text{df}}{=} X$, can be seen as the identity function.

For closed formulae, φ^S is a subset of Q, which can be equivalently seen as the image of a function with no arguments. But for formulae with free variables, φ^S is a function exactly like φ is and the above definition can be read as transformation of functions. For instance, for a φ with a single free variable X, the definition of $(\neg\varphi)^S$ can be reformulated as $(\neg\varphi)^S(X) \stackrel{\text{df}}{=} Q \setminus \varphi^S(X)$. Note that, to simplify the definitions in the sequel, we have considered α as a set of actions in $\langle\alpha\rangle$ instead of as a single action as more usual. Moreover, because the semantics of a formula is a set of states, we may use one or the other form interchangeably.

The effective construction of $\mu X.\varphi$ is made inductively by defining $0X.\varphi \stackrel{\text{df}}{=} \emptyset$, and $nX.\varphi \stackrel{\text{df}}{=} \varphi[X \leftarrow (n-1)X.\varphi]$ where $\varphi[X \leftarrow Y]$ denotes φ in which variable X is substituted by Y everywhere. Knaster-Tarski's theorem ensures that there exists $k \in \mathbb{N}$ such that $kX.\varphi = \mu X.\varphi$. Indeed, φ is a monotonous function over a complete lattice, and thus the set of its fixed-points is also a complete lattice [8].

Example 3. Let us consider the LTS from the left of figure 2 and the formula $\mu X.(\langle A\rangle X \vee d)$ (which means that the module can reach a state where place d is marked). We can compute the least fixed-point of $\varphi \stackrel{\text{df}}{=} \langle A\rangle X \vee d$ as follows:

– $0X.\varphi \stackrel{\text{df}}{=} \emptyset$
– $1X.\varphi \stackrel{\text{df}}{=} \varphi(0X.\varphi) = (\langle A\rangle X)(\emptyset) \cup d() = \emptyset \cup \{4,5,6,7,8,9\}$
– $2X.\varphi \stackrel{\text{df}}{=} (\langle A\rangle X)(1X.\varphi) \cup \{4,5,6,7,8,9\} = \{2,3,4,5,6,7,8,9\}$
– $3X.\varphi \stackrel{\text{df}}{=} (\langle A\rangle X)(2X.\varphi) \cup \{4,5,6,7,8,9\} = \{1,2,3,4,5,6,7,8,9\}$
– $4X.\varphi \stackrel{\text{df}}{=} (\langle A\rangle X)(3X.\varphi) \cup \{4,5,6,7,8,9\} = 3X.\varphi$ ◇

To write formulae more comfortably, we can use the following operators:

– $\varphi_1 \wedge \varphi_2 \stackrel{\text{df}}{=} \neg(\neg\varphi_1 \vee \neg\varphi_2)$;
– $[\alpha]\varphi \stackrel{\text{df}}{=} \neg\langle\alpha\rangle\neg\varphi$, which yields $([\alpha]\varphi)^S \stackrel{\text{df}}{=} \{x \in Q \mid \forall y \in \varphi^S, x \xrightarrow{a} y \in T\}$;
– $\nu X.\varphi \stackrel{\text{df}}{=} \neg\mu X.\neg\varphi[X \leftarrow \neg X]$ is the greatest fixed-point of φ.

Finally, for $q \in Q$, we write $S, q \models \varphi$ iff $q \in \varphi^S$, and $S \models \varphi$ iff $S, q_0 \models \varphi$.

4 Formula-dependent abstraction

In this section, we introduce an operation $\lfloor S \rfloor_\varphi$ that, given the LTS S of a module and a formula φ, returns a LTS, (usually) smaller that S without changing the truth of φ, neither over S, nor over the global system (*i.e.*, S synchronised with the LTS of the other modules). To do so, we define an equivalence relation (initially between states of a same LTS, then extended to LTS), denoted by $\sim^\varphi$, that preserves the truth value of φ on the global system and is a congruence with respect to synchronised product $\otimes$.

Let $S = (q_0, Q, A, R, L)$ with $A \stackrel{\text{df}}{=} A^{\mathsf{loc}} \uplus A^{\mathsf{fus}}$ be the LTS of a module, $q \in Q$ a state, and φ a formula. Let also Ex be the actions appearing in φ but not in A, which corresponds to the context of S, *i.e.*, the other modules. To start with,

we define the set Pass^{φ} such that if $q \in \mathsf{Pass}^{\varphi}$ then φ is necessarily true on any bigger system in which S is a module in state q. Similarly, we define Fail^{φ} such that if $q \in \mathsf{Fail}^{\varphi}$ then φ is necessarily false on any bigger system encompassing S in state q. We shall write $\mathsf{Pass}^{\varphi}(S)$ or $\mathsf{Fail}^{\varphi}(S)$ when the LTS of interest needs to be precised. We say that formula φ can be evaluated on a state q belonging to S if $q \in \mathsf{Pass}^{\varphi} \cup \mathsf{Fail}^{\varphi}$, which is denoted by $\varphi \stackrel{?}{=}\!| S, q$. This means that we can conclude about the truth of φ in state q independently of the context in which S may be embedded as a module. Similarly, φ can be evaluated on S if it can be evaluated it on its initial state, which is denoted by $\varphi \stackrel{?}{=}\!| S$.

The definition below is made with respect to a context Σ that is a map from the free variables of a formula to sets of states (when needed, Σ may be seen as a set of pairs). For a formula φ that contains free variables $X_1, \ldots, X_n$ that do not appear in the environment Σ, $\mathsf{Pass}^{\Sigma,\varphi}$ is the function $(x_1, \ldots, x_n) \mapsto \mathsf{Pass}^{\Sigma \cup \{(X_i, x_i) | 1 \leq i \leq n\}, \varphi}$.

Definition 6. *Let φ be a formula and $S \stackrel{\text{df}}{=} (q_0, Q, A, R, L)$ with $A \stackrel{\text{df}}{=} A^{\mathsf{loc}} \uplus A^{\mathsf{fus}}$ be a LTS. We set $\mathsf{Pass}^{\varphi} \stackrel{\text{df}}{=} \mathsf{Pass}^{\emptyset,\varphi}$ and $\mathsf{Fail}^{\varphi} \stackrel{\text{df}}{=} \mathsf{Fail}^{\emptyset,\varphi}$, where $\mathsf{Pass}^{\emptyset,\varphi}$ and $\mathsf{Fail}^{\emptyset,\varphi}$ are functions defined inductively on the syntax of φ:*

- $\mathsf{Pass}^{\Sigma,X} \stackrel{\text{df}}{=} \Sigma(X)$;
- $\mathsf{Pass}^{\Sigma,B} \stackrel{\text{df}}{=} \{x \in Q \mid L(x) \Rightarrow B\}$;
- $\mathsf{Pass}^{\Sigma,\neg\varphi} \stackrel{\text{df}}{=} \mathsf{Fail}^{\Sigma,\varphi}$;
- $\mathsf{Pass}^{\Sigma,\varphi_1 \vee \varphi_2} \stackrel{\text{df}}{=} \mathsf{Pass}^{\Sigma,\varphi_1} \cup \mathsf{Pass}^{\Sigma,\varphi_2}$;
- $\mathsf{Pass}^{\Sigma,\langle\alpha\rangle\varphi} \stackrel{\text{df}}{=} (\langle \alpha \cap A^{\mathsf{loc}} \rangle X)(\mathsf{Pass}^{\Sigma,\varphi})$;
- $\mathsf{Pass}^{\Sigma,\mu X.\varphi} \stackrel{\text{df}}{=} \mu X.\mathsf{Pass}^{\varphi}$.
- $\mathsf{Fail}^{\Sigma,X} \stackrel{\text{df}}{=} \Sigma(X)$;
- $\mathsf{Fail}^{\Sigma,B} \stackrel{\text{df}}{=} \mathsf{Pass}^{\Sigma,\neg B}$;
- $\mathsf{Fail}^{\Sigma,\neg\varphi} \stackrel{\text{df}}{=} \mathsf{Pass}^{\Sigma,\varphi}$;
- $\mathsf{Fail}^{\Sigma,\varphi_1 \vee \varphi_2} \stackrel{\text{df}}{=} \mathsf{Fail}^{\Sigma,\varphi_1} \cap \mathsf{Fail}^{\Sigma,\varphi_2}$;
- $\mathsf{Fail}^{\Sigma,\langle\alpha\rangle\varphi} \stackrel{\text{df}}{=} ([\alpha \setminus Ex]X)(\mathsf{Fail}^{\Sigma,\varphi}) \cap F$, *where* $F \stackrel{\text{df}}{=} Q$ *if* $A \cap Ex = \emptyset$ *and* $F \stackrel{\text{df}}{=} \mathsf{Fail}^{\Sigma,\varphi}$ *otherwise;*
- $\mathsf{Fail}^{\Sigma,\mu X.\varphi} \stackrel{\text{df}}{=} \nu X.\mathsf{Fail}^{\Sigma,\varphi}$.

Together with Pass^{φ} and Fail^{φ}, we aim to define $\sim^{\varphi}$ as a relation between the states such that if $x \sim^{\varphi} y$, then, in a larger system embedding S, formula φ does not allow to distinguish the states embedding x or y. (Thus it will be possible to reduce S by merging these two states.) In order to compute relation $\sim^{\varphi}$ we define $\mathfrak{F}^{\Sigma,\varphi}$ and build $\mathfrak{F}^{\varphi} \stackrel{\text{df}}{=} \mathfrak{F}^{\emptyset,\varphi}$.

The computation of $\mathfrak{F}^{\emptyset,\varphi}$ described in definition 7, yields a triple (p, f, r) such that at the end of computation, $p = \mathsf{Pass}^{\varphi}$,$f = \mathsf{Fail}^{\varphi}$ and $r \stackrel{\text{df}}{=} \sim^{\varphi}$ (this is not necessarily the case at every step). The rules used to build $\mathfrak{F}^{\emptyset,\varphi}$, as shown in exemple 4, operate on elements from $Q \times Q \times Q^2$. As in the definitions of Pass and Fail, if φ contains free variables $X_1, \ldots, X_n$ which do not appear in the environment Σ, then $\mathfrak{F}^{\Sigma,\varphi}$ is the function $(x_1, \ldots, x_n) \mapsto \mathfrak{F}^{\Sigma \cup \{(X_i, x_i) | 1 \leq i \leq n\}, \varphi}$. However unlike in the previous definition, environment Σ now takes values from $Q \times Q \times Q^2$. We also define Σ^p and Σ^f as the projections of Σ on its first and second components, *i.e.*, the smallest environments such that if $(X, (p, f, r)) \in \Sigma$, then $(X, p) \in \Sigma^p$ and $(X, f) \in \Sigma^f$.

Definition 7. *Let φ be a formula and $S \stackrel{\text{df}}{=} (Q, A, R, L)$ with $A \stackrel{\text{df}}{=} A^{\mathsf{loc}} \uplus A^{\mathsf{fus}}$ be a LTS. $\mathfrak{F}^{\Sigma,\varphi}$ is defined recursively on the syntax of φ as follows:*

1. $\mathfrak{F}^{\Sigma,B} \stackrel{\text{df}}{=} (\mathsf{Pass}^{\Sigma,B}, \mathsf{Fail}^{\Sigma,B}, \{(x,y) \mid (L(x) \Rightarrow B) \Leftrightarrow (L(y) \Rightarrow B)\})$.
2. $\mathfrak{F}^{\Sigma,\neg\varphi_1} \stackrel{\text{df}}{=} \mathfrak{F}^{\Sigma,\neg}(\mathfrak{F}^{\Sigma,\varphi_1})$
   with $\mathfrak{F}^{\Sigma,\neg}(p,f,r) \stackrel{\text{df}}{=} (f,p,r)$.
3. $\mathfrak{F}^{\Sigma,\varphi_1\vee\varphi_2} \stackrel{\text{df}}{=} \mathfrak{F}^{\Sigma,\vee}(\mathfrak{F}^{\Sigma,\varphi_1}, \mathfrak{F}^{\Sigma,\varphi_2})$
   with $\mathfrak{F}^{\Sigma,\vee}((p_1,f_1,r_1),(f_2,p_2,r_2)) \stackrel{\text{df}}{=} (p_1 \cup p_2, f_1 \cap f_2, r_1 \cap r_2)$.
4. $\mathfrak{F}^{\Sigma,\langle\alpha\rangle\varphi_1} \stackrel{\text{df}}{=} \mathfrak{F}^{\Sigma,\langle\alpha\rangle}(\mathfrak{F}^{\Sigma,\varphi_1})$
   with $\mathfrak{F}^{\Sigma,\langle\alpha\rangle}(p,f,r) \stackrel{\text{df}}{=} (\mathsf{Pass}^{\Sigma^p,\langle\alpha\rangle X}(p), \mathsf{Fail}^{\Sigma^f,\langle\alpha\rangle X}(f), r')$ *where* $(x,y) \in r'$ *iff* $(x,y) \in r$ *and either*
   (a) $x \in \mathsf{Pass}^{\Sigma^p,\langle\alpha\rangle X}(p)$ *and* $y \in \mathsf{Pass}^{\Sigma^p,\langle\alpha\rangle X}(p)$,
   (b) *or,* $x \in \mathsf{Fail}^{\Sigma^f,\langle\alpha\rangle X}(f)$ *and* $y \in \mathsf{Fail}^{\Sigma^f,\langle\alpha\rangle X}(f)$,
   (c) *or, we have*
      i. *for every* $a \in \alpha \cap A^{\mathsf{fus}}$, *if* $x \xrightarrow{a} x' \in R$ *and* $x' \notin f$ *then it exists* $y \xrightarrow{a} y' \in R$ *such that* $(x',y') \in r$,
      ii. *and, for every* $a \in \alpha \cap A^{\mathsf{fus}}$, *if* $y \xrightarrow{a} y' \in R$ *and* $y' \notin f$ *then it exists* $x \xrightarrow{a} x' \in R$ *such that* $(x',y') \in r$,
      iii. *and, for every* $a \in \alpha \cap A^{\mathsf{loc}}$, *if* $x \xrightarrow{a} x' \in R$ *and* $x' \notin f$ *then it exists* $y \xrightarrow{a'} y' \in R$ *such that* $a' \in \alpha \cap A^{\mathsf{loc}}$ *and* $(x',y') \in r$,
      iv. *and, for every* $a \in \alpha \cap A^{\mathsf{loc}}$, *if* $y \xrightarrow{a} y' \in R$ *and* $x' \notin f$ *then it exists* $x \xrightarrow{a'} x' \in R$ *such that* $a' \in \alpha \cap A^{\mathsf{loc}}$ *and* $(x',y') \in r$.
5. $\mathfrak{F}^{\Sigma,X} \stackrel{\text{df}}{=} \Sigma(X)$.
6. $\mathfrak{F}^{\Sigma,\mu X.\varphi_1}$ *is the fixed-point reached by iterating function* $\mathfrak{F}^{\Sigma,\varphi_1}$ *starting from* $(Pass^{\Sigma^p,\mu X.\varphi_1}, Fail^{\Sigma^f,\mu X.\varphi_1}, r_0)$ *with* $(x,y) \in r_0$ *iff either*
   (a) $x \in Pass^{\Sigma^p,\mu X.\varphi_1}$ *and* $y \in Pass^{\Sigma^p,\mu X.\varphi_1}$,
   (b) *or* $x \in Fail^{\Sigma^f,\mu X.\varphi_1}$ *and* $y \in Fail^{\Sigma^f,\mu X.\varphi_1}$,
   (c) *or* $x, y \in Q \setminus (Fail^{\Sigma^f,\mu X.\varphi_1} \cup Pass^{\Sigma^p,\mu X.\varphi_1})$.

With regard to $\sim^{\varphi}$, this definition can be intuitively understood as follows:

1. The labels of two equivalent states must be identical with respect to the atomic formulae which appear in B.
2. The relations corresponding to a formula and to its negation are the same.
3. Two equivalent states must be equivalent on both sub-formulae.
4. When φ involves the next states through $\langle\alpha\rangle$, two equivalent states x and y must be equivalent w.r.t. the sub-formula and either
   (a) both x and y ensure that φ globally holds,
   (b) or, both x and y ensure that φ does not hold globally,
   (c) or,
      i. if from x we can reach x' through a fused transition, then this must be possible from y through the same action, reaching a state y' equivalent to x'.Note that we only have to consider the x' which are not in the Fail set of the sub-formula,
      ii. and the same thing symmetrically for y.

iii. moreover, if from x we can reach x' (which does not belong to the Fail set of the sub-formula) through a local action, then this must be possible from y through the same or another local action from the set α, reaching a state y' equivalent to x',
iv. and the same thing symmetrically for y.

5. The value of X is fetched from the environment (by construction, we are always have every free variable in the environment).
6. Function $\mathfrak{F}^{\Sigma,\varphi_1}$ is repeatedly applied starting from the greatest relation for which $\mathsf{Pass}^{\Sigma,\mu X.\varphi_1}$ and $\mathsf{Fail}^{\Sigma,\mu X.\varphi_1}$ are equivalence classes. Each iteration (which is a composition of the previous rules) will then differentiate some states until we reach a fixed-point. This necessarily occurs because, for any relation r, we have $\mathfrak{F}^{\Sigma,\varphi_1}(\mathsf{Pass}^{\Sigma^p,\mu X.\varphi_1}, \mathsf{Fail}^{\Sigma^p,\mu X.\varphi_1}, r) = (\mathsf{Pass}^{\Sigma^p,\mu X.\varphi_1}, \mathsf{Fail}^{\Sigma^p,\mu X.\varphi_1}, r')$ with $r' \subseteq r$. So we always eventually reach a fixed-point when applying $\mathfrak{F}^{\Sigma,\varphi_1}$ repeatedly while computing $\mathfrak{F}^{\Sigma,\mu X.\varphi_1}$.

Definition 8. *Let φ be a closed formula, $\sim^{\varphi}$ is defined as the third component returned by $\mathfrak{F}^{\varphi}$.*

Example 4. Let $\varphi \stackrel{\text{df}}{=} B_1 \vee \langle\alpha\rangle B_2$ for some Boolean formulae B_1 and B_2. As defined above, we can compute

$$\mathfrak{F}^{B_1 \vee \langle\alpha\rangle B_2} = \mathfrak{F}^{\emptyset,\vee}(\mathfrak{F}^{\emptyset,B_1}, \mathfrak{F}^{\emptyset,\langle\alpha\rangle}(\mathfrak{F}^{\emptyset,B_2})) = (\mathsf{Pass}^{\varphi}, \mathsf{Fail}^{\varphi}, \sim^{\varphi}) \qquad \diamond$$

The next lemma states that we have indeed defined an equivalence relation. This equivalence is defined within the context of a single LTS, this can be generalised to compare two LTS.

Lemma 1. *Relation $\sim^{\varphi}$ as defined above is indeed an equivalence relation.*

Definition 9. *Let φ be a formula, and S_1 and S_2 be two LTS whose initial states are $q_{0,1}$ and $q_{0,2}$ respectively. S_1 is equivalent to S_2 w.r.t. φ, which is denoted by $S_1 \sim^{\varphi} S_2$, iff $q_{0,1} \sim^{\varphi} q_{0,2}$ in S defined as the component-wise disjoint union of S_1 and S_2.*

Using relation $\sim^{\varphi}$ we define the reduction of a LTS by merging its equivalent states, which is done by considering the quotient set of Q by $\sim^{\varphi}$.

Definition 10. *Let $S \stackrel{\text{df}}{=} (Q, q_0, A, R, L)$ be a LTS with $A \stackrel{\text{df}}{=} A^{\mathsf{loc}} \uplus A^{\mathsf{fus}}$, and φ be a formula. The* reduction of S w.r.t. φ, *denoted by $\lfloor S \rfloor_{\varphi}$, is the LTS (Q', q'_0, A', R', L') such that:*

- *$Q' \stackrel{\text{df}}{=} Q/\!\sim^{\varphi}$ is the quotient set of Q by $\sim^{\varphi}$;*
- *$q'_0 \stackrel{\text{df}}{=} [q_0]_{\varphi}$ is the equivalence class containing q_0;*
- *A' is exactly A and is partitioned the same way;*
- *$R' \stackrel{\text{df}}{=} \{(c_1, a, c_2) \in Q' \times A' \times Q' \mid \exists q_1 \in c_1, \exists q_2 \in c_2, (q_1, a, q_2) \in R\}$;*
- *$L'(c) \stackrel{\text{df}}{=} \bigvee_{q \in [c]_{\varphi}} L(q)$.*

We now introduce several properties of the definitions above, progressively leading to our main compositionality result. First, we state that Pass and Fail effectively allow to locally conclude about the global truth value of a formula.

Lemma 2. *Let $S \stackrel{\text{df}}{=} S_1 \otimes \cdots \otimes S_n$ be a product LTS and $x \stackrel{\text{df}}{=} (x_1, \ldots, x_n)$ one of its states.*

1. *If $x_i \in \mathsf{Pass}^{\varphi}(S_i)$ for some $1 \leq i \leq n$, then $x \in \mathsf{Pass}^{\varphi}(S)$.*
2. *If $x_i \in \mathsf{Fail}^{\varphi}(S_i)$ for some $1 \leq i \leq n$, then $x \in \mathsf{Fail}^{\varphi}(S)$.*

Then, we state that $\sim^{\varphi}$ correctly captures the states that are equivalent w.r.t. the capability to evaluate locally the global truth of φ. Moreover, it also preserves the truth value of φ.

Theorem 2. *Let S be a LTS, φ a formula, and x and y two states of S such that $x \sim^{\varphi} y$. If $\varphi \stackrel{?}{=\!\models} S, x$, then $\varphi \stackrel{?}{=\!\models} S, y$ and $S, x \models \varphi \Leftrightarrow S, y \models \varphi$.*

Corollary 1. *Let φ be a formula, and S_1 and S_2 two LTS such that $S_1 \sim^{\varphi} S_2$. If $\varphi \stackrel{?}{=\!\models} S_1$ then $\varphi \stackrel{?}{=\!\models} S_2$ and $S_1 \models \varphi \Leftrightarrow S_2 \models \varphi$.*

Moreover, the reduction w.r.t φ yields a LTS that is equivalent to the original one. Then, we state the consistency of reduction $\lfloor S \rfloor_{\varphi}$ w.r.t. the synchronised product, which allows to extend the previous lemma to compound LTS. Finally, this reduction is a congruence for the product of LTS, which means that we can replace any LTS of a product by the corresponding reduced LTS while preserving the equivalence relation.

Lemma 3. *Let S be a LTS and φ a formula, we have: $S \sim^{\varphi} \lfloor S \rfloor_{\varphi}$.*

Theorem 3. *Let $S \stackrel{\text{df}}{=} S_1 \otimes \cdots \otimes S_n$ be a product LTS, $x \stackrel{\text{df}}{=} (x_1, \ldots, x_n)$ and $y \stackrel{\text{df}}{=} (y_1, \ldots, y_n)$ two of its states, and φ a formula. If $x_i \sim^{\varphi} y_i$ for all $1 \leq i \leq n$ then $x \sim^{\varphi} y$.*

Corollary 2. $S_1 \otimes \cdots \otimes S_n \sim^{\varphi} \lfloor S_1 \rfloor_{\varphi} \otimes \cdots \otimes \lfloor S_n \rfloor_{\varphi}$

Combining these results with theorem 1, we can perform modular analysis with hierarchical reductions. Indeed, given a formula φ and a modular Petri net $(N_1, \ldots, N_n)$, let us define $\llbracket N_i \rrbracket_{\varphi} \stackrel{\text{df}}{=} \lfloor \llbracket N_i \rrbracket \rfloor_{\varphi}$, we have:

$$\llbracket N_1 \oplus \cdots \oplus N_n \rrbracket \quad \sim \quad \llbracket N_1 \rrbracket \otimes \cdots \otimes \llbracket N_n \rrbracket \quad \sim^{\varphi} \quad \llbracket N_1 \rrbracket_{\varphi} \otimes \cdots \otimes \llbracket N_n \rrbracket_{\varphi}$$

Furthermore, lemma 2 tells us that we can stop building the system as soon as we find a sub-system on which the formula can be evaluated, *i.e.*, as soon as $\varphi \stackrel{?}{=\!\models} \llbracket N_i \rrbracket_{\varphi}$ for some i, because the truth value of that formula over the global system will be the same as over this sub-system. Finally, because modules can be freely associated and commuted, we can conduct the analysis hierarchically, reducing at each level and possibly stopping before the whole system semantics is constructed.

Example 5. Let us consider again the example 2 (and figure 2) and check that we can reach a state where the property $h \wedge d$ is true, that is expressed in μ-calculus as $\mu X.(\langle A \rangle X \vee (h \wedge d))$. Remember that we assumed there exists a third module synchronised over f_3. We will see that in this case, analysing the first two modules is sufficient to prove the property.

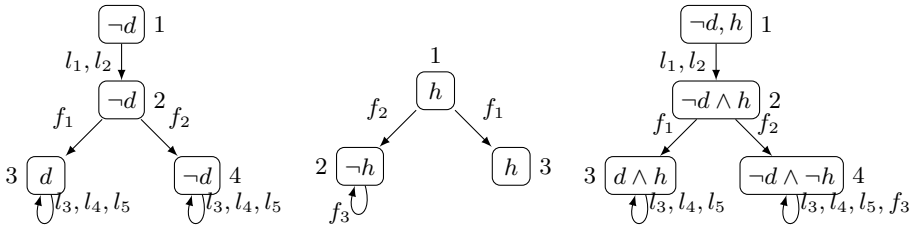

Fig. 3. Left and middle: the semantics of modules from Example 1 reduced w.r.t. $\mu X.(\langle A\rangle X \vee (h \wedge d))$. For clarity, only labels involving d and h have been displayed, a label such as $\neg x$ denotes a real label where x is not involved and thus implies $\neg x$. Right: the synchronised product of the two LTS on the left.

We begin by reducing the first LTS w.r.t. the formula. We are in case 6 of definition 7, $Pass^{\mu X.(\langle A\rangle X \vee (h \wedge d))} = \emptyset$ (we can never conclude without knowing the value of h which is and external variable) and $Fail^{\mu X.(\langle A\rangle X \vee (h \wedge d))} = \{10, 11, 12\}$ (we know that from these states we can only access states where d is false). We now apply function $\mathfrak{F}^{\langle A\rangle X \vee (h \wedge d)}$ repeatedly starting from:

- $(p_0, f_0, r_0) \stackrel{\text{df}}{=} (\emptyset, \{10, 11, 12\}, \{\{10, 11, 12\}, \{1, 2, 3, 4, 5, 6, 7, 8, 9\}\})$ where r_0 is given as the set of its equivalence classes instead of as a set of pairs, which is more compact;
- $\mathfrak{F}^{\langle A\rangle X}(p_0, f_0, r_0) = (\emptyset, \{10, 11, 12\}, \{\{10, 11, 12\}, \{1\}, \{2, 3\}, \{4, 5, 6, 7, 8, 9\}\})$ and $\mathfrak{F}^{h \wedge d} \stackrel{\text{df}}{=} (\emptyset, \{10, 11, 12\}, \{\{10, 11, 12\}, \{1, 2, 3, 4, 5, 6, 7, 8, 9\}\})$ so we have $(p_1, f_1, r_1) \stackrel{\text{df}}{=} \mathfrak{F}^{\langle A\rangle X \vee (h \wedge d)}(p_0, f_0, r_0) = (\emptyset, \{10, 11, 12\}, \{\{10, 11, 12\}, \{1\}, \{2, 3\}, \{4, 5, 6, 7, 8, 9\}\})$
- $(p_2, f_2, r_2) \stackrel{\text{df}}{=} \mathfrak{F}^{\langle A\rangle X \vee (h \wedge d)}(p_1, f_1, r_1) = (p_1, f_1, r_1)$ (We have reached the fixed-point so we can use r_1 to build the reduced LTS from figure 3)

Doing the same with the second LTS, we get the reduced LTS depicted in figure 3. Their product is depicted on the right of the same figure. To compose this product with the rest of the system, we shall first try to minimise it. Doing so, we also compute $\mathsf{Pass}^{\mu X.\langle A\rangle X \vee (h \wedge d)}$, obtaining set $\{1, 2, 3\}$ that contains the initial state of the graph. Therefore we know that the formula is true over the global system and we can stop the analysis. ◇

5 Conclusion

We have shown that it is possible to define the semantics of a modular Petri net as a hierarchical composition of the semantics of its modules taken in any order. At each step, a subset of modules is considered, and its semantics can be computed and analysed with respect to a modal μ-calculus formula φ. Possibly, this allows to draw a conclusion about the truth value of φ on the whole system without the need to consider the rest of the system. If no conclusion can be

drawn at this step, a minimised semantics can be computed for the subset of modules at hand, and reused for the sequel of the hierarchical analysis.

In [4], the authors define the decomposition of a Petri net according to a formula and the verification of this formula in a compositional way. Moreover, [3] makes use of the modular description of a system to reduce it hierarchically. The main difference with our work is that they both consider abstractions that preserve every formula from LTL\X in which chosen actions appear. Our approach only preserves one formula from the $\mu - calculus$ so, on the one hand, we can express more properties, and on the other hand, targeting a particular formula let us expects better reductions. But, as a consequence, we have to recompute the abstraction for each new formula. However more thoroughgoing comparisons remain to be done. In [7], the author considers the incremental construction of Petri nets through refinements (of transitions, places and place types), also allowing for incremental state space construction. Properties may be verified at an intermediary step avoiding to construct the fully refined state space. However, these properties are not expressed as logic formulae but are classical Petri net properties (deadlock, home state, etc.).

Future work will address the question of finding a good order for conducting such a hierarchical analysis, in order to minimise the computational effort needed to obtain a result. In particular, it is not clear if we should start by combining strongly connected modules with the aim of obtaining good reductions at the beginning, or if we should instead prefer the modules the most involved in the formula of interest. Another prospect is to find the best form for the formula we want to verify, in order too increase the efficiency of the reduction. Since we require equivalent states to be equivalent on every sub-formula, if we can minimize the number of sub-formulae then we are likely to compute a better reduction. For instance formula $\langle a \rangle true \vee \langle b \rangle true$ is equivalent to $\langle a, b \rangle true$. However if one state only has one outgoing transition labelled by a, and another state only has one outgoing transition labelled by b, they will be equivalent w.r.t. the second formula but not w.r.t. the first one.

So we believe that the current paper defines a framework that is suitable to perform hierarchical analysis, but also that it is the starting point of a lot more work to find suitable strategies for efficient analysis.

References

1. A. Aziz, T. Shiple, V. Singhal, R. Brayton, and A. Sangiovanni-Vincentelli. Formula-dependent equivalence for compositional ctl model checking. *Formal Methods in System Design*, 21(2):193–224, 2002.
2. S. Christensen and L. Petrucci. Modular analysis of Petri nets. *The Computer Journal*, 43(3):224–242, 2000.
3. K. Klai and L. Petrucci. Modular construction of the symbolic observation graph. In *Application of Concurrency to System Design, 2008. ACSD 2008. 8th International Conference on*, pages 88–97. IEEE, 2008.
4. K. Klai, L. Petrucci, and M. Reniers. An incremental and modular technique for checking ltl\ x properties of petri nets. *Formal Techniques for Networked and Distributed Systems–FORTE 2007*, pages 280–295, 2007.

5. D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27(3):333–354, 1983.
6. C. Lakos and L. Petrucci. Modular analysis of systems composed of semiautonomous subsystems. In *proc. of ACSD'04*. IEEE Computer Society Press, 2004.
7. G. Lewis. *Incremental specification and analysis in the context of coloured Petri nets*. PhD thesis, University of Tasmania, 2002.
8. A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific journal of Mathematics*, 5(2):285–309, 1955.

A Proof of theorem 2

Theorem 2 can be rewritten using the property $\mathcal{P}$ defined, for any $(p, f, r) \in 2^Q \times 2^Q \times 2^{Q\times Q}$, by $\mathcal{P}(p, f, r)$ holds iff $(r \setminus (Q \setminus p)^2 \subseteq p^2$ and $r \setminus (Q \setminus f)^2 \subseteq f^2)$.

We want to prove that for any formula φ, we have $\mathcal{P}(\mathfrak{F}^\varphi)$. Let us show that this property is true for every base case and that it is preserved by the rules used for building $\mathfrak{F}^\varphi$.

Case $\mathfrak{F}^{\Sigma,B}$. Take (x, y) in $\sim^B$. We know that $L(x) \Rightarrow B \Leftrightarrow L(y) \Rightarrow B$. Then,

- if x (wlog) belongs to $\mathsf{Pass}^{\Sigma^p,B}$ then $L(x) \Rightarrow B$ is true, and so is $L(y) \Rightarrow B$ which means that y is in $\mathsf{Pass}^{\Sigma^p,B}$.
- if x belongs to $\mathsf{Fail}^{\Sigma^f,B}$ then $L(x) \Rightarrow B$ is false, and so is $L(y) \Rightarrow B$ which means that y is in $\mathsf{Fail}^{\Sigma^f,B}$.

Case $\mathfrak{F}^{\Sigma,\neg}$. Take (p, f, r) such that $\mathcal{P}(f, p, r)$. We then have $\mathcal{P}(\mathfrak{F}^{\Sigma,\neg}(p, f, r))$ because $\mathfrak{F}^{\Sigma,\neg}(p, f, r) = (f, p, r)$.

Case $\mathfrak{F}^{\Sigma,\vee}$. Take (p_1, f_1, r_1) and (p_2, f_2, r_2) such that $\mathcal{P}(p_1, f_1, r_1)$ and $\mathcal{P}(p_2, f_2, r_2)$. We have $\mathfrak{F}^{\Sigma,\vee}((p_1, f_1, r_1), (p_2, f_2, r_2)) = (p_1 \cup p_2, f_1 \cap f_2, r_1 \cap r_2)$.

- for the first case we have $(r_1 \cap r_2) \setminus (Q \setminus (p_1 \cup p_2))^2 = (r_1 \cap r_2) \setminus ((Q \setminus p_1)^2 \cap (Q \setminus p_2)^2) = (r_1 \cap r_2) \setminus (Q \setminus p_1)^2 \cup (r_1 \cap r_2) \setminus (Q \setminus p_2)^2 \subseteq r_1 \setminus (Q \setminus (p_1))^2 \cup r_2 \setminus (Q \setminus (p_2))^2 \subseteq p_1^2 \cup p_2^2 \subseteq (p_1 \cup p_2)^2$
- and for the second case $(r_1 \cap r_2) \setminus (Q \setminus (f_1 \cap f_2))^2 = (r_1 \cap r_2) \setminus (Q \setminus f_1 \cup Q \setminus f_2)^2 \subseteq (r_1 \cap r_2) \setminus ((Q \setminus f_1)^2 \cup (Q \setminus f_2)^2) \subseteq (r_1 \cap r_2) \setminus (Q \setminus f_1)^2 \cap (r_1 \cap r_2) \setminus (Q \setminus f_2)^2 \subseteq r_1 \setminus (Q \setminus f_1)^2 \cap r_2 \setminus (Q \setminus f_2)^2 \subseteq f_1^2 \cap f_2^2 \subseteq (f_1 \cap f_2)^2$

Case $\mathfrak{F}^{\Sigma,\langle\alpha\rangle}$. Take (p, f, r) verifying $\mathcal{P}$ and note $(p', f', r') \stackrel{\text{df}}{=} \mathfrak{F}^{\Sigma,\langle\alpha\rangle}(p, f, r)$. Then take $(x, y) \in r'$.

- If $x \in \mathsf{Pass}^{\Sigma^p,\langle\alpha\rangle}(p)$.
  Exists $x' \in p$ and $a \in \alpha \cap A^{loc}$ such that $x \xrightarrow{a} x'$.
  Because (x, y) belongs to r', we have $y \xrightarrow{a'} y'$ and $(x', y') \in r$. Since we know that $\mathcal{P}(p, f, r)$, y' belongs to p too and y to $\mathsf{Pass}^{\Sigma^p,\langle A\rangle}(p)$.
- If $x \in \mathsf{Fail}^{\Sigma^f,\langle\alpha\rangle}(f)$.
  - If x is in f then so is y because $\mathcal{P}(p, f, r)$ and $r' \subseteq r$. This is needed when $\alpha \cap Ex \neq \emptyset$.

- For all $y \xrightarrow{a} y'$ with a in α we have $x \xrightarrow{a'} x'$ with a' in α and (x', y') in r. Because $x \in \mathsf{Fail}^{\Sigma^f,\langle\alpha\rangle}(f)$ this means that every x' is in f. Then so is every y', and finally $y \in \mathsf{Fail}^{\Sigma^f,\langle A\rangle}(f)$.

Case $\mathfrak{F}^{\Sigma,X}$. We only put in the environment values verifying $\mathcal{P}$ (see next case).

Case $\mathfrak{F}^{\Sigma,\mu X.\varphi_1}$.

- $(\mathsf{Pass}^{\Sigma,\mu X.\varphi_1}, \mathsf{Fail}^{\Sigma,\mu X.\varphi_1}, r_0)$ with $(x, y) \in r_0$ iff
  1. $x \in Pass^{\Sigma^p,\mu X.\varphi_1}$ and $y \in Pass^{\Sigma^p,\mu X.\varphi_1}$ or
  2. $x \in Fail^{\Sigma^f,\mu X.\varphi_1}$ and $y \in Fail^{\Sigma^f,\mu X.\varphi_1}$ or
  3. both x and y belong to $Q \setminus (Fail^{\Sigma^f,\mu X.\varphi_1} \cup Pass^{\Sigma^p,\mu X.\varphi_1})$

  verify $\mathcal{P}$.
- $\mathfrak{F}^{\Sigma,\varphi_1}$ is a composition of the above functions, so it preserves proposition $\mathcal{P}$.

B Proof of theorem 3

Let $S \stackrel{\text{df}}{=} \bigotimes_{i\in I} S_i$ be a LTS , φ a formula and Σ_i environments we denote by π^φ the product relation of the $\sim_{S_i}^{\Sigma_i,\varphi}$, *i.e.*, (x, y) is in π^φ iff (x_i, y_i) is in $\sim_{S_i}^{\Sigma_i,\varphi}$ for all $i \in I$. Let us define the property $\mathcal{P}_\pi((p, f, r), \Sigma, \{\Sigma_i | i \in I\}, \psi)$ which means: $p = \mathsf{Pass}^{\Sigma^p,\psi}$, $f = \mathsf{Fail}^{\Sigma^f,\psi}$ and $\pi^\psi \subseteq r$. We show that every base case verify this property and that the various rules preserve it. The part of the property about Pass and Fail can almost be obtained by construction so we do not explicitly mention it in this proof.

Case $\mathfrak{F}^{\Sigma,B}$. Take (x, y) in π^P. For all i in I we have the following property: $L_i(x_i) \Rightarrow B \equiv L_i(y_i) \Rightarrow B$. Now, we know that $\bigwedge_I L_i(x_i) \Rightarrow B \equiv \bigwedge_I L_i(y_i) \Rightarrow B$ and therefore $(x, y) \in \sim^{\Sigma,B}$. So we have $\mathcal{P}_\pi(\mathfrak{F}^{\Sigma,B}, \Sigma, \{\Sigma_i | i \in I\}, B)$.

Case $\mathfrak{F}^{\Sigma,\neg}$. Take (p, f, r) such that $\mathcal{P}_\pi((p, f, r), \Sigma, \{\Sigma_i | i \in I\}, \varphi_1)$. We have $\mathcal{P}_\pi(\mathfrak{F}^\neg(p, f, r), \Sigma, \{\Sigma_i | i \in I\}, \neg\varphi_1)$ because $\mathfrak{F}^\neg(p, f, r) = (f, p, r)$ and $(\mathsf{Pass}^{\neg\Psi}, \mathsf{Fail}^{\neg\Psi}, \pi^{\neg\Psi}) = (\mathsf{Fail}^\Psi, \mathsf{Pass}^\Psi, \pi^\Psi)$

Case $\mathfrak{F}^{\Sigma,\vee}$. For any formulae φ_1 and φ_2, for any (p_1, f_1, r_1) and (p_2, f_2, r_2) such that $\mathcal{P}_\pi((p_1, f_1, r_1), \Sigma, \{\Sigma_i | i \in I\}, \varphi_1)$ and $\mathcal{P}_\pi((p_2, f_2, r_2), \Sigma, \{\Sigma_i | i \in I\}, \varphi_2)$, we have $\mathcal{P}_\pi(\mathfrak{F}^{\Sigma,\vee}((p_1, f_1, r_1), (p_2, f_2, r_2)), \Sigma, \{\Sigma_i | i \in I\}, \varphi_1 \vee \varphi_2)$ because $\mathfrak{F}^{\Sigma,\vee}((p_1, f_1, r_1), (p_2, f_2, r_2)) = (p_1 \cup p_2, f_1 \cap f_2, r_1 \cap r_2)$ and $(\mathsf{Pass}^{\varphi_1\vee\varphi_2}, \mathsf{Fail}^{\varphi_1\vee\varphi_2}, \pi^{\varphi_1\vee\varphi_2}) = (\mathsf{Pass}^{\varphi_1} \cup \mathsf{Pass}^{\varphi_2}, \mathsf{Fail}^{\varphi_1} \cap \mathsf{Fail}^{\varphi_2}, \pi^{\varphi_1} \cap \pi^{\varphi_2})$

Case $\mathfrak{F}^{\Sigma,\langle\alpha\rangle}$. Take (p, f, r) such that $\mathcal{P}_\pi((p, f, r), \Sigma, \{\Sigma_i | i \in I\}, \varphi_1)$.

Let us consider (x, y) in $\pi^{\langle A\rangle\varphi_1}$, it is indeed true that $(x, y) \in \pi^{\varphi_1}$. We now have take into account three different possibilities.

- If exists i such that x_i and y_i are in $\mathsf{Pass}_i^{\Sigma_i^p,\langle A\rangle\varphi_1}$ then x and y are in $\mathsf{Pass}^{\Sigma^p,\langle A\rangle\varphi_1}$.
- The same goes for Fail.
- In the third case:

- Let us consider any $x \xrightarrow{a} x'$ such that $x' \notin \mathsf{Fail}^{\Sigma^f,\varphi_1}$ and exists $i \in I$ such that $a \in \alpha \cap A_i^{fus}$, and let's show that exists y' such that $y \xrightarrow{a} y'$ and $(x', y') \in \pi^{\varphi_1}$.
  For every i in I:
  * If a is in A_i^{fus}: We have $x[i] \xrightarrow{a} x'[i]$. Because $(x[i], y[i])$ is in $\sim_i^{\Sigma,\langle A\rangle\varphi_1}$ and $x'[i] \notin \mathsf{Fail}_i^{\Sigma_i^f,\varphi_1}$, there exists y'_i such that $y[i] \xrightarrow{a} y'_i$ and $(x'[i], y'_i) \in \sim_i^{\Sigma_i,\varphi_1}$
  * If a is not in A_i^{fus}: We have $x'[i] = x[i]$. Let's define $y'_i \stackrel{\text{df}}{=} y[i]$; we then have $(x'[i], y'_i) \in \sim_i^{\Sigma_i,\varphi_1}$ because $(x'[i], y'_i) \in \sim_i^{\Sigma_i,\langle A\rangle\varphi_1}$

  Now if $y'[i] \stackrel{\text{df}}{=} y'_i$ for all i, then $y \xrightarrow{a} y'$ and $(x', y') \in \pi^{\varphi_1}$.
- Let us consider any $x \xrightarrow{a} x'$ such that $x' \notin \mathsf{Fail}^{\Sigma^f,\varphi_1}$ and exists $i \in I$ such that $a \in \alpha \cap A_i^{loc}$. Let's show that exists $a' \in \alpha \cap A^{in}$ and y' such that $y \xrightarrow{a'} y'$ and $(x', y') \in \pi^{\varphi_1}$.
  * We know that we have $x[i] \xrightarrow{a} x'[i]$, and $a' \in (A_i^{loc} \cap \alpha)$ such that $y[i] \xrightarrow{a'} y'_i$ and $(x'[i], y'_i) \in \sim_i^{\Sigma,\varphi_1}$
  * For $j \neq i$, we define $y'_j \stackrel{\text{df}}{=} y[j]$

  Now if $y'[i] \stackrel{\text{df}}{=} y'_i$ for all i, we have $y \xrightarrow{a'} y'$ with $a' \in (A^{loc} \cap \alpha)$ and $(x', y') \in \pi^{\varphi_1}$.

For any of these cases, (x, y) belongs to the third component of $\mathfrak{F}^{\Sigma,\langle\alpha\rangle}(p, f, r)$ so we have $\mathcal{P}_\pi(\mathfrak{F}^{\Sigma,\langle\alpha\rangle}(p, f, r), \Sigma, \{\Sigma_i | i \in I\}, \langle\alpha\rangle\varphi_1)$

Case $\mathfrak{F}^{\Sigma,X}$. We only put in the environment values which verify $\mathcal{P}_\pi(\Sigma, \{\Sigma_i | i \in I\}, X)$.

Case $\mathfrak{F}^{\Sigma,\mu X.\varphi_1}$.

- Let us show that if (x, y) is in $\pi^{\mu X.\varphi_1}$ and one of them is in $\mathsf{Pass}^{\Sigma^p,\mu X.\varphi_1}$ (resp $\mathsf{Fail}^{\Sigma^f,\mu X.\varphi_1}$) then so is the other. This means that we have the property $\mathcal{P}_\pi((p_0, f_0, r_0), \Sigma, \{\Sigma_i | i \in I\}, \mu X.\varphi_1)$, where (p_0, f_0, r_0) is the the starting point of the iteration. In order to do this we build the tuple (p, f, r) by iterating $\mathfrak{F}^{\Sigma,\varphi_1}$ starting from $(\emptyset, Q, Q^2)$. We then have $p = \mathsf{Pass}^{\Sigma^p,\mu X.\varphi_1}$, $f = \mathsf{Fail}^{\Sigma^f,\mu X.\varphi_1}$ and $r \supseteq \pi^{\mu X.\varphi_1}$ (This is the same proof we are currently doing, but with an easier base case). We can reuse the proof of theorem 2 (similarly, only the base case is different) to show that if $(x, y) \in r$ and one of them is in $\mathsf{Pass}^{\Sigma^p,\mu X.\varphi_1}$ (resp $\mathsf{Fail}^{\Sigma^f,\mu X.\varphi_1}$) then so is the other. So the property $\mathcal{P}_\pi(\Sigma^0(X), \Sigma^0, \{\Sigma_i^0 | i \in I\}, X)$ is true with:

  - $\Sigma^0 \stackrel{\text{df}}{=} \Sigma[X \leftarrow (p_0, f_0, r_0)]$ and
  - $\Sigma_i^0 \stackrel{\text{df}}{=} \Sigma_i[X \leftarrow (\mathsf{Pass}^{\Sigma_i,\mu X.\varphi_1}, \mathsf{Fail}^{\Sigma_i,\mu X.\varphi_1}, \sim_i^{\Sigma_i,\mu X.\varphi_1})]$ for all i.

- $\mathfrak{F}^{\varphi_1}$ is a composition of the previous functions. Therefore if the property $\mathcal{P}_\pi(\Sigma^n(X), \Sigma^n, \{\Sigma_i | i \in I\}, X)$ is true, then we have $\mathcal{P}_\pi(\mathfrak{F}^{\Sigma^n,\varphi_1}, \Sigma^n, \{\Sigma_i | i \in I\}, \varphi_1)$, which can be rewritten as $\mathcal{P}_\pi(\Sigma^{n+1}(X), \Sigma^{n+1}, \{\Sigma_i | i \in I\}, X)$ where $\Sigma^{n+1} = \Sigma^n[X \leftarrow \mathfrak{F}^{\Sigma^n,\varphi_1}]$. By recurrence, the property is true for any n.