Refinement for open automata

Quentin Corradi

April 20, 2021

1 Introduction

Open automata are used to give an interpretation for open pNets.

2 Notations

Throughout this paper, tuples might be noted differently depending on what they represent. Quantification will be used loosely, any bound name in a quantification means that the object is constrained to have that previous value, and any new name is bound by the quantifier. For instance $\forall(x,y), \exists(x',y), P$ means $\forall(x,y), \exists(x',y'), y = y' \wedge P$.

Family of values, or equivalently maps will be noted $\{i \mapsto x_i \mid i \in I\}$, $\{i \leftarrow x_i \mid i \in I\}$ or $x_i^{i \in I}$. The latter will only be used when unambiguous; for instance $(ax)^{x \in \mathbb{R}}$ represents a scaling function, $c^{i \in I}$ is a constant function over I, but $\{\alpha \mapsto 1, \beta \mapsto 2, \gamma \mapsto 3\}$ must be represented with one of the two first notation. The disjoint union of two maps $\varphi : I \to X$ and $\psi : J \to Y$ with $I \cap J = \emptyset$ is $\varphi \uplus \psi : I \uplus J \to X \cup Y$.

For a relation $\leq$ which uses a property P to prove that two element are related $\forall x\, x', x \leq x' \iff \exists w : P(x,x')$, the fact that there is a valid witness $w : P(x,x')$ is noted $x \leq_w x'$.

3 Open Automaton

To define an open automaton we need some preliminary definitions.

Definition 1 (Expression algebra, Formulae, Terms)**.** An expression algebra E is a disjoint union $E := \mathcal{T} \uplus \mathcal{F}$ where $\mathcal{T}$ is a term algebra and $\mathcal{F}$ the formulae over $\mathcal{T}$.

The terms algebra is arbitrary. The formulae form a first order logic over $\mathcal{T}$ and some relations. An example of term algebra can be Peano integers (zero, variable and successor function), the formulae associated can use equality relation, $sum(a,b,c) := a = b + c$ relation and $prod(a,b,c) := a = b \times c$ relation.

Definition 2 (Free variables, Closed terms, Closed formulae)**.**

- $vars(e)$ is the set of unbound variables in $e \in E$.
- The expressions restricted to variables in V are $E_V ::= \{e \in E \mid vars(e) = \emptyset\}$; $E_V \subset E$.
- The closed expressions are expressions restricted to variables in $\emptyset$, $E_\emptyset$.

By restriction, terms and formulae restricted to variables and closed versions are also defined. A closed expression is an expression without unbound variable. We can use the previous example of first order formula on Peano integers to illustrate these definitions. x, $S(y)$ are valid terms, 0 is a closed term, $vars(x) = \{x\}$, $vars(S(y)) = \{y\}$, $vars(0) = \emptyset$. $x = 0$, $\forall y, \exists x, sum(y, z, x)$ are valid formulae, $\forall x, \neg S(x) = 0$ is a valid closed formula, $vars(x = 0) = \{x\}$, $vars(\forall y, \exists x, sum(y, z, x)) = \{z\}$, $vars(\forall x, \neg S(x) = 0) = \emptyset$.

Definition 3 (Interpretation, Values, Satisfiability, (Parallel) substitution)**.** We assume that the following are given:

- The values $\mathcal{P}$, which are interpretations of closed terms.
- The validity relation on closed formulae, $\vdash \subseteq \mathcal{F}_\emptyset$.
- The substitution in $e \in E$ of $x \in vars(e)$ by $t \in \mathcal{T}$, $e[t/x]$.
- The parallel substitution in $e \in E$ of variables in V by $\psi : V \to \mathcal{T}$, $e\{\!\{\psi\}\!\}$.

For the parallel substitution, the set V is not required to be a subset of $vars(e)$. In the case where it is not, the variables in $V \setminus vars(e)$ are not substituted. The substitutions might give a nonsensical expression; for instance let the terms be integers and pairs with (pointwise) addition, $(a + b)\{\!\{a \mapsto 7, b \mapsto (4, 5)\}\!\}$ is a ill-formed term. This can be guarded with the atom $e[t/x] \in E$ and $e\{\!\{\psi\}\!\} \in E$ and it will implicitely be the case, for instance when there is quantification on t and ψ, to simplify notations.

The interpretation of terms is supposed to be decidable. The validity of formulae might not be decidable nor complete nor consistent, however we will pretend like they are because these are really hard problems for logicians that we don't want to deal with.[1]

A value v may be used for keeping a variable state, and then injected in terms for substitution. While this is correct when $\mathcal{P} \subseteq \mathcal{T}_\emptyset$, in the other cases this will be used as a shorthand for substitution with any term which can be interpreted as v. We suppose that the interpretation of terms is compatible w.r.t. substitution, that is if two terms t, t' are interpreted with the same value, then replacing t by t' in a well-formed expression makes an equivalent well-formed expression. A quantifier followed by a finite set will be used as a shorthand for the quantification on every variable in the set, for instance $\forall A, \exists B, P$ with $A = \left\{a_1, \dots, a_{|A|}\right\}, B = \left\{b_1, \dots, b_{|B|}\right\}$ means $\forall a_1, \dots, \forall a_{|A|}, \exists b_1, \dots, \exists b_{|B|}, P$.

The validity relations will be noted using the classical relational notation $\vdash f ::= f \in \vdash$. The usual validity of a formula $f \in \mathcal{F}$ is noted $\vdash f ::= \vdash \forall vars(f), f$. The usual validity of a formula $f \in \mathcal{F}$ with some fixed valuation $\sigma : V \to \mathcal{P}$ is noted $\sigma \vdash f ::= \vdash \forall (vars(f) \setminus V), f\{\!\{\sigma\}\!\}$.

From there the expression algebra is fixed. With these common definitions and notations settled, the objects of interest can now be defined.

Definition 4 (Open automaton)**.** A semantic open automaton is a tuple $\langle S, i, V, \varphi, J, T\rangle$ with S the set of states, $i \in S$ the initial state, V the set of variable names unique to this automaton, $\varphi : V \to \mathcal{P}$ the initial valuation of variables, J the set of hole names and T a set of open-transitions.

S, V, J are arbitrary sets, only J is required to be finite.

The variable names may clash when considering two automata, in that case we suppose that we can still distinguish the variables in the formulas. All the possible open transitions, from which transitions in T are selected, are defined below.

[1] In practise a term algera with only equality is used, and it is has all these properties.

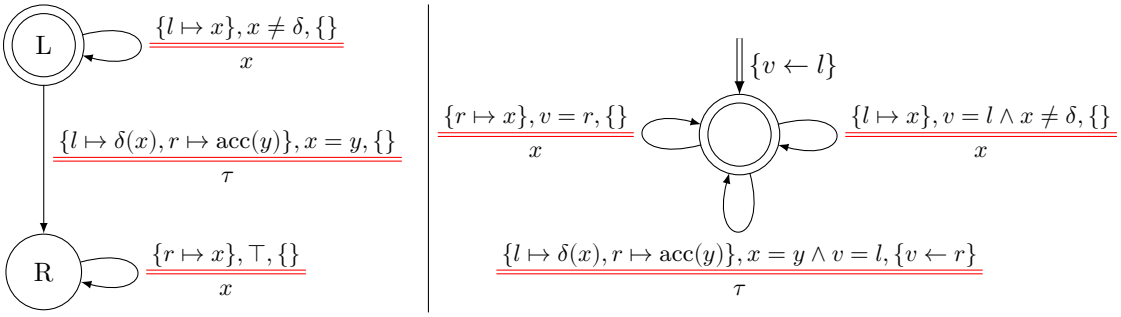

Figure 1: Enable operator, on the left state oriented, on the right variable oriented.

Definition 5 (Open transition)**.** A semantic open transition is a tuple $\frac{(\beta_j(v_{\beta j}))^{j\in J'}, g, \psi}{s \xrightarrow{\alpha(v_\alpha)} s'}$ with $s, s' \in S$ the source and target states, $\alpha \in \mathcal{A}$ an action label, $J' \subseteq J$ the holes involved, $\beta_j \in \mathcal{A}$ the action labels of the holes, $v_\alpha \in \mathcal{T}_V$ the emitted action, $v_{\beta j}^{j\in J'} \in \mathcal{T}_V$ the hole actions, $g \in \mathcal{F}_V$ the guard and $\psi : V \to \mathcal{T}_V$ the new value of the variables.

The set of action labels $\mathcal{A}$ is arbitrary and implicit. To simplify formulae, the notations $\alpha ::= \alpha(v_\alpha)$ and $\alpha = \beta ::= \alpha = \beta \wedge v_\alpha = v_\beta$ will be used.

To illustrate these definitions let's look at two implementations (figure 1) of the LOTOS operator enable in the open automata model. The enable operator runs its the left hand side agent until it chooses to finish, at which point it produces an action δ that is synchronised with the first action of the right hand side agent which must be acc, then only the latter agent runs. During the synchronised action data can be exchanged.

cite something here

I need to read a little bit more on that; is that a return code?

Example 1 (Enable, state-oriented)**.** To draw automata, the usual circle for states and simple arrows for transitions convention is used. The initial state is indicated by a double circle. States names are indicated inside the circles and transitions labels are drawn near their corresponding arrow.

The open transitions do not indicate the source and target states since that is the role of the transitions arrows; only the action is on the bottom side of the open transition. Also in the example, transitions are specified using variables in place of actions or values. This is used to represent all the transitions created by a correct substitution of these variables, as there are often infinitly many.

There is a model where open automata have finitely many transitions and variables like that are allowed in the expressions and in the action data, not as a tool to graphically represent an infinity of transitions but as a real construct. This model is the model of structural open automata as opposed to the model used here of semantic open automata. The structural version is more adapted to uses where the object manipulated must be finite, for example in algorithms, whereas the semantic version avoids the manipulation of substitutions, which simplifies proofs.

Let's get back to the example. The open automata drawn on the left is $\langle \{L, R\}, L, \emptyset, \{\}, \{l, r\}, T \rangle$

where transitions in T are all the valid transitions for any substitution of x and y of

$$\frac{\{l \mapsto x\}, x \neq \delta, \{\}}{L \xrightarrow{x} L} \qquad \frac{\{l \mapsto \delta(x), r \mapsto \mathrm{acc}(y)\}, x = y, \{\}}{L \xrightarrow{\tau} R} \qquad \frac{\{r \mapsto x\}, \top, \{\}}{R \xrightarrow{x} R}$$

Example 2 (Enable, variable-oriented)**.**

Now we can define some utilitary functions:

Definition 6 (Guard, Out-transition)**.** $\mathrm{OT}(s)$ are called the out-transitions of s.

$$guard\left(\frac{\beta_j^{j\in J'}, g, \psi}{s \xrightarrow{\alpha} s'}\right) ::= g \qquad \mathrm{OT}(s) ::= \left\{\frac{\beta_j^{j\in J'}, g, \psi}{s \xrightarrow{\alpha} s'} \in T\right\}$$

The intuition of a semantic open automaton is a partially defined LTS with variables, guards on transitions and parametrised actions. From this point semantic open automata and open transitions will be called automata and transitions for simplicity.

4 Semantic and composition of Open Automata

When the auomaton is in a state $s \in S$ with a valuation of its variables $\sigma : V \to \mathcal{P}$, it cannot perform transitions $\frac{\beta_j^{j\in J'}, g, \psi}{s \xrightarrow{\alpha} s'} \in T$ such that $\neg g\{\!\{\sigma\}\!\}$. After performing the transition and emitting action α, the automaton is in the state s' with valuation $\psi\{\!\{\sigma\}\!\}$. The holes are synchronised using a handshake like mecanism when the transition happens.

Example 3.

An open automaton can be partially specified, the partial specification comes from the holes. A hole can be filled with an open automaton, this operation is called composition.

Or substitution, or filling, what do you prefer?

Definition 7 (Composition of open automata)**.** The composition of $A_c := \langle S_c, i_c, V_c, \varphi_c, J_c, T_c\rangle$ in the hole $k \in J_p$ of $A_p := \langle S_p, i_p, V_p, \varphi_p, J_p, T_p\rangle$ is

$$A_p[A_c/k] ::= \langle S_p \times S_c, (i_p, i_c), V_p \uplus V_c, \varphi_p \uplus \varphi_c, J_c \uplus J_p \setminus \{k\}, T\rangle$$

$$\text{With } T := \left\{\frac{\beta_j^{j\in J'_c \uplus J'_p \setminus \{k\}}, g_p \wedge g_c \wedge \alpha_c = \beta_k, \psi_p \uplus \psi_c}{(s_p, s_c) \xrightarrow{\alpha_p} (s'_p, s'_c)} \;\middle|\; \frac{\beta_j^{j\in J'_p}, g_p, \psi_p}{s_p \xrightarrow{\alpha_p} s'_p} \in T_p, \frac{\beta_j^{j\in J'_c}, g_c, \psi_c}{s_c \xrightarrow{\alpha_c} s'_c} \in T_c\right\}$$

$$\cup \left\{\frac{\beta_j^{j\in J'_p}, g_p, \psi_p}{(s_p, s_c) \xrightarrow{\alpha_p} (s'_p, s_c)} \;\middle|\; \frac{\beta_j^{j\in J'_p}, g_p, \psi_p}{s_p \xrightarrow{\alpha_p} s'_p}, k \notin J', s_c \in S_c\right\}$$

Similarly to expression substitution, the parallel composition is noted $A\{\!\{A_j^{j\in J}\}\!\}$.

The actions emitted when A_c makes a transition is sychronised with the action of the hole k in transitions of A_p which have it as a hole actions (first transition set, $\alpha_c = \beta_k$). No transition of A_c is performed when a transition that do not refer to the hole k is performed in A_p (second transition set, $k \notin J'$). The composition may look like a handshake, with both automata running in parallel (product of states and joint variables) but it is actually not symmetric.

Example 4.

5 What is a refinement for Open Automata

There are several properties that we may want from a refinement relation. Depending on these properties, several kind of refinement may be used. For example if we are interested in being able to produce the same sequence of action we may want to use trace set inclusion as a refinement. Here the main properties that we want are related to composition and action refinement. A suitable kind of refinement relation for this kind of properties is simulation refinement: A bisimulation has already been proposed for open automata that is compatible with composition.

cite an article here

Other kind of refinement relation that I may explore are control refinement/hole refinement[2], (meet semi)lattice refinement[3], weak-simulation refinement[4], composition-correct refinement[5].

This is a note for possible path, some may be explored, some may not.

The important properties we will consider here are:

Definition 8. A relation $\leq$ is

- **reflexive** iff $\forall a, a \leq a$;
- **transitive** iff $\forall a\, b\, c, a \leq b \wedge b \leq c \implies a \leq c$;
- **a preorder** iff it is reflexive and transitive;
- **complete w.r.t. composition** iff $\forall a\, b, a[b] \leq a$;
- **correct w.r.t. composition** iff $\forall a\, b, a \leq b \implies \exists c, a \overset{FH}{=} b[c]$;
- **context equivalent for composition** iff $\forall a\, b\, c, a \leq b \implies a[c] \leq b[c]$;
- **congruent for composition** iff $\forall a\, b\, c, a \leq b \implies c[a] \leq c[b]$;
- **compatible with composition** iff $\forall a\, b\, c\, d, a \leq b \wedge c \leq d \implies c[a] \leq d[b]$;
- **compatibile with FH-bisimulation** iff $\forall a\, b\, c, d, a \overset{FH}{=} b \wedge c \overset{FH}{=} d \wedge a \leq c \implies b \leq d$.

Reflexivity, transitivity and preorder are classical properties on relations. Completeness w.r.t. composition is the fact that every composition is considered a refinement. Correctness w.r.t. composition is the fact that a refinement correspond to the left automaton being equivalent to some composition of the right automaton. Context equivalence is the refinement being compatible with composition with the same automaton. Congruence is the refinement being compatible with being composed in the same automaton. Compatibility with composition is the two latter at the same time. Finally compatibility with FH-bisimulation is the refinement not being able to distinguish two FH-bisimilar automata.

The relations will have to be at least preorders, simulations and forbid the introduction of deadlocks (state where no transition is possible) to be called refinement simulation. On top of that weak refinement are refinement where some actions, typically one named τ, are considered not observable and do not have to be matched. In the weak refinement the relations have to forbid the introduction of livelocks (only a sequence of τ possible) and respect τ-stuttering . While being preorder is simple to express for a relation, being a simulation, do not introduce

cite many things arround here

[2]No good formulation atm, the idea is that $a \leq b ::= sth \wedge J_a \setminus J_b \leq_{ctrl} J_b \setminus J_a$, sth is probably a clause to ensure that they behave the same without holes involved.

[3]meet = handshake, $a \leq b ::= a = a||b$ with sync holes, I think this will be equivalent to hole-identical sim, (join=non-det choice?)

[4]Weak variation for each interesting simulation refinement

[5]Basically a stricter variant of simulation-refinement where simulation has to take place the other way for some transitions (no different holes?) in order to be correct wrt composition

new deadlocks/livelocks and τ-stuttering are properties about the states of the related automata, therefore they are more complex to express. We will need a preliminary concept before defining them.

Definition 9 (Relation under predicate)**.** A relation under predicate between states of two automata $\langle S_1, i_1, J_1, V_1, \varphi_1, T_1\rangle$ and $\langle S_2, i_2, J_2, V_2, \varphi_2, T_2\rangle$ is a function in $S_1 \times S_2 \to \mathcal{F}_{V_1 \uplus V_2}$. Two states $s_1 \in S_1, s_2 \in S_2$ are related under a valuation $\sigma : V_1 \uplus V_2 \to \mathcal{P}$ iff $\sigma \vdash R(s_1, s_2)$.

Now let's adapt the properties about states of automata to open automata.

Definition 10 (Simulation on open automata)**.** A relation $\leq$ is a simulation if for any related automata $A_1 \leq_R A_2$ where $A_1 := \langle S_1, i_1, J_1, V_1, \varphi_1, T_1\rangle$, $A_2 := \langle S_2, i_2, J_2, V_2, \varphi_2, T_2\rangle$ and $R : S_1 \times S_2 \to \mathcal{F}_{V_1 \uplus V_2}$, R satisfies both

- Initial states are related under initial valuations: $\varphi_1 \uplus \varphi_2 \vdash R(i_1, i_2)$;
- From related states, all out-transitions from A_1 can be simulated in A_2 and their target states are related:

$$\forall (s_1, s_2) \in S_1 \times S_2, \sigma : V_1 \uplus V_2 \to \mathcal{P},$$
$$\forall \frac{\beta_{1j}^{j \in J_1'}, g_1, \psi_1}{s_1 \xrightarrow{\alpha_1} s_1'} \in T_1, (\sigma \vdash R(s_1, s_2) \wedge g_1) \implies \exists \frac{\beta_{2j}^{j \in J_2'}, g_2, \psi_2}{s_2 \xrightarrow{\alpha_2} s_2'} \in T_2,$$
$$\sigma \vdash \alpha_1 = \alpha_2 \wedge \bigwedge_{j \in J_1' \cap J_2'} \beta_{1j} = \beta_{2j} \wedge g_2 \wedge R(s_1', s_2') \{\!\{\psi_1 \uplus \psi_2\}\!\}$$

The last formula means that for every pair of related states under valuation of the automaton variables and every possible transitions ($\sigma \vdash g_1$) from the first automaton, there is a possible transition ($\sigma \vdash g_2$) such that the produced action matches ($\alpha_1 = \alpha_2$), the holes actions from same hole names ($J_1' \cap J_2'$) match ($\beta_{1j} = \beta_{2j}$) and the target states are related after variable update. This is a natural extenion of the notion of simulation on LTS, which is the same definition without variables and holes and guards. It will serve as a watchdog for any relation that will be defined with "simulation" in its name, except weak-simulation where the transition from the first automaton that produce τ actions are excluded.

Definition 11 (Deadlock reduction, intuitive definition)**.** A simulation $\leq$ is deadlock reducing if in all the $R : S_1 \times S_2 \to \mathcal{F}_{V_1 \uplus V_2}$ such that $A_1 \leq_R A_2$, there is one that also satisfies

$$\forall (s_1, s_2) \in S_1 \times S_2, \sigma : V_1 \uplus V_2 \to \mathcal{P},$$
$$(\exists t_2 \in \mathrm{OT}(s_2), \sigma \vdash R(s_1, s_2) \wedge \mathit{guard}(t_2)) \implies \exists \frac{\beta_{2j}^{j \in J_2'}, g_2, \psi_2}{s_2 \xrightarrow{\alpha_2} s_2'} \in T_2, \frac{\beta_{1j}^{j \in J_1'}, g_1, \psi_1}{s_1 \xrightarrow{\alpha_1} s_1'} \in T_1,$$
$$\sigma \vdash g_2 \wedge \alpha_1 = \alpha_2 \wedge \bigwedge_{j \in J_1' \cap J_2'} \beta_{1j} = \beta_{2j} \wedge g_1 \wedge R(s_1', s_2') \{\!\{\psi_1 \uplus \psi_2\}\!\}$$

This definition extends the requirements of a simulation by requiring from the relation that on related states, if there is at least one possible transition in the second automata then there must be a matching possible transition in the first. This requirement effectively prevents the presence of deadlock state in the first automaton if there is a related non-deadlock state in the second.

On the other hand there is a property that we considered for some time, that may arise when designing a refinement relation and that is interesting but actually unwanted:

Daisy equivalence: $a[D] \leq a \wedge a \leq a[D]$, where D is the single state automaton that can produce any action.

This property state that composing a hole with an automaton which can do anything is not a strict refinement. While it might seem to be a natural property because this automaton is the closest to the meaning of a hole in term of automaton, there is actually a difference between the two. Daisy equivalence allows to refine by filling a hole but also by going the other way, that is creating a hole name and putting its hole action to any transition (and possibly keeping any of the unmodified transitions). The unwanted behaviour arise for instance when two independant holes are filled, then a hole is created and its actions are put where the two other actions were. This effectively merges independant holes, or equivalently allows to consider that an automaton can be plug simultaneously into several holes which is not part of the composition semantics.

Example 5.

"Deadlock reduction" conflicts with "composition completeness" by disallowing some unwanted cases where an automaton in a hole cannot produce any action that any out-transitions expects. In particular filling a hole with the deadlock automaton (only one state without out-transitions) is not considered a refinement for a no deadlock reducing simulation refinement. A way to solve this issue is to caracterise a composition that do not introduce deadlocks, which will be used instead of the composition introduced earlier.

Definition 12 (Reachability)**.** For any open automata $A \coloneqq \langle S, i, V, \varphi, J, T\rangle$, we define the caracterisation of reachable valuations $\checkmark_A : S \to \mathcal{F}_V$ as the strongest predicate on states such that

- $\varphi \vdash \checkmark_A(i)$;
- $\forall \frac{\beta_j^{j\in J'}, g, \psi}{s \xrightarrow{\alpha} s'} \in T, \vdash \checkmark_A(s) \implies \checkmark_A(s')\{\!\{\psi\}\!\}$.

The caracterisation of reachable valuations is used to caracterise the states and valuations that are reachable in a run of an automaton. For a valuation $\sigma : V \to \mathcal{P}$ and a state

Definition 13 (Non-locking composition)**.** The composition $A_p[A_c/k] \coloneqq \langle S, i, V, \varphi, J, T\rangle$ where $A_c \coloneqq \langle S_c, i_c, V_c, \varphi_c, J_c, T_c\rangle$, $k \in J_p$ and $A_p \coloneqq \langle S_p, i_p, V_p, \varphi_p, J_p, T_p\rangle$, is a correct composition if there is a predicate on its states $p : S \to \mathcal{F}_V$ that satisfies

- Being valid on initial states under initial valuation: $\varphi \vdash p(i)$;
- Transfering validity across transitions: $\forall \frac{\beta_j^{j\in J'}, g, \psi}{s \xrightarrow{\alpha} s'} \in T, \vdash p(s) \implies p(s')\{\!\{\psi\}\!\}$;
- From states where the predicate is valid, if a transition was possible in A_p then a transition is possible in A and the predicate holds in the target state:

$$\forall s \in S, \sigma : V \to \mathcal{P}, \left(\exists t_p \in \mathrm{OT}\left(\pi_{S_p}(s)\right), \sigma \vdash p(s) \wedge \mathit{guard}(t_p)\right) \implies \exists \frac{\beta_j^{j\in J'}, g, \psi}{s \xrightarrow{\alpha} s'} \in T, \sigma \vdash g$$

For the reader not used to simulations and bisimulations, this definition with a witness can seem strange. In fact the role of the witness is to caracterise reachable configurations without

having to caracterise the fact that the state and valuation are the result of a valid path in an automaton.

When expanding the aliases S, i, V in the definition, and remarking that the last existential quantifier can be unfolded into a transition in A and one in A_p with the added condition that they match, the definition become mildly similar with the deadlock reduction one. It is not a coincidence because their goal is to caracterise the same kind of compatibility between automata. Actually it is possible to simplify both definition and make them shockingly similar.

Lemma 1. *The formula in the definition of deadlock reduction is equivalent to the following:*

$$\forall (s_1, s_2) \in S_1 \times S_2, \vdash R(s_1, s_2) \wedge \bigvee_{t_2 \in \mathrm{OT}(s_2)} \mathit{guard}(t_2) \implies \bigvee_{t_1 \in \mathrm{OT}(s_1)} \mathit{guard}(t_1)$$

Proof. Let R be a witness of the simulation $A_1 \leq A_2$.

$\implies$: If R satisfies the deadlock reduction definition, let $(s_1, s_2) \in S_1 \times S_2$. The free variables of the equivalent formula are all in $V_1 \uplus V_2$ so let $\sigma : V_1 \uplus V_2 \to \mathcal{P}$ be such that

$$\sigma \vdash R(s_1, s_2) \wedge \bigvee_{t_2 \in \mathrm{OT}(s_2)} \mathit{guard}(t_2) \implies \bigvee_{t_1 \in \mathrm{OT}(s_1)} \mathit{guard}(t_1)$$

We can decompose it by admitting the left part of the implication then decomposing the big disjunction to get $\sigma \vdash R(s_1, s_2)$ as ($\mathrm{H_p}$) and $t_2 \in \mathrm{OT}(s_2)$ such that ($\mathrm{H_{g2}}$): $\sigma \vdash \mathit{guard}(t_2)$. We are left to prove

$$\bigvee_{t_1 \in \mathrm{OT}(s_1)} \mathit{guard}(t_1) \tag{G1}$$

To do that we will use the fact that R satisfies the deadlock reduction definition with the current value of s_1, s_2, σ. Now we additionally need to prove (G2): $\exists t_2 \in \mathrm{OT}(s_2), \sigma \vdash R(s_1, s_2) \wedge \mathit{guard}(t_2)$ before getting hypothesis ($\mathrm{H_E}$):

$$\exists \frac{\beta_{2j}^{j \in J_2'}, g_2, \psi_2}{s_2 \xrightarrow{\alpha_2} s_2'} \in T_2, \frac{\beta_{1j}^{j \in J_1'}, g_1, \psi_1}{s_1 \xrightarrow{\alpha_1} s_1'} \in T_1, \sigma \vdash g_2 \wedge \alpha_1 = \alpha_2 \wedge \bigwedge_{j \in J_1' \cap J_2'} \beta_{1j} = \beta_{2j} \wedge g_1 \wedge R(s_1', s_2') \{\!\{ \psi_1 \uplus \psi_2 \}\!\}$$

The witness of (G2) is t_2 obtained previously in the big disjunction, and $\sigma \vdash R(s_1, s_2) \wedge \mathit{guard}(t_2)$ is proved by combining ($\mathrm{H_p}$) and ($\mathrm{H_{g2}}$). Now we can decompose ($\mathrm{H_E}$) to get $t_1 \in T_1$, which we know is also in $\mathrm{OT}(s_1)$, a transition that we won't use and $\sigma \vdash \mathit{guard}(t_1)$ which is what we needed to prove the branch t_1 of (G1).

$\impliedby$: If R satisfies the equivalent formula, let $(s_1, s_2) \in S_1 \times S_2$ and $\sigma : V_1 \uplus V_2 \to \mathcal{P}$. Let admit that there is a transition $t_2 \in T_2$ such that ($\mathrm{H_{g2}}$): $\sigma \vdash R(s_1, s_2) \wedge \mathit{guard}(t_2)$. We need to prove (G1):

$$\exists \frac{\beta_{2j}^{j \in J_2'}, g_2, \psi_2}{s_2 \xrightarrow{\alpha_2} s_2'} \in T_2, \frac{\beta_{1j}^{j \in J_1'}, g_1, \psi_1}{s_1 \xrightarrow{\alpha_1} s_1'} \in T_1, \sigma \vdash g_2 \wedge \alpha_1 = \alpha_2 \wedge \bigwedge_{j \in J_1' \cap J_2'} \beta_{1j} = \beta_{2j} \wedge g_1 \wedge R(s_1', s_2') \{\!\{ \psi_1 \uplus \psi_2 \}\!\}$$

We use the fact that R satisfies the equivalent formula with the current value of s_1, s_2 and σ to get

$$\vdash R(s_1, s_2) \wedge \bigvee_{t_2 \in \mathrm{OT}(s_2)} \mathit{guard}(t_2) \implies \bigvee_{t_1 \in \mathrm{OT}(s_1)} \mathit{guard}(t_1)$$

As this formula is true for all values of each free variable, it is especially true when valuated with σ. We know that $t_2 \in \mathrm{OT}(s_2)$ so ($\mathrm{H_{g2}}$) proves the left part of that implication and we immediately decompose the right side to get $t_1 \in \mathrm{OT}(s_1)$ and ($\mathrm{H_{g1}}$): $\mathit{guard}(t_1)$.

We cannot yet prove (G1) because at this point we don't know whether t_2 and t_1 do match. In order to get a matching t_2 we use the fact that R is also witness of the simulation $A_1 \leq A_2$ with the current value of s_1, s_2, σ and t_1 as ($\mathrm{H_{sim}}$):

$$(\sigma \vdash R(s_1, s_2) \wedge g_1) \implies \exists \frac{\beta_{2j}^{j \in J_2'}, g_2, \psi_2}{s_2 \xrightarrow{\alpha_2} s_2'} \in T_2, \sigma \vdash \alpha_1 = \alpha_2 \wedge \bigwedge_{j \in J_1' \cap J_2'} \beta_{1j} = \beta_{2j} \wedge g_2 \wedge R(s_1', s_2') \{\!\{\psi_1 \uplus \psi_2\}\!\}$$

The left part of the implication is the left side of ($\mathrm{H_{g2}}$) so we get $t_2' \in T_2$ and what's after as the hypothesis ($\mathrm{H_{target}}$). The witnesses for (G1) are t_2' and t_1, and the rest is proved with a combination of ($\mathrm{H_{target}}$) and ($\mathrm{H_{g1}}$). □

Lemma 2. *The second part of the formula in the definition of correct composition is equivalent to the following:*

$$\forall s \in S, \vdash p(s) \wedge \bigvee_{t_p \in \mathrm{OT}\left(\pi_{S_p}(s)\right)} \mathit{guard}(t_p) \implies \bigvee_{t \in \mathrm{OT}(s)} \mathit{guard}(t)$$

Proof.

$\implies$:

$\impliedby$: □

From this point and in the previous definitions, composition will only refer to correct composition.

6 Preliminary refinement relations

The main objective of the refinement relation in this section is to be able to say that a composition of two automata is a refinement of the base automaton ($a[b] \leq a$). However composing an open automaton can give an automaton where not much can be said in terms of hole indicies. So looking at restricted cases where holes are related in a specific manner can help understanding the general case. For two open automata $A_1 := \langle S_1, i_1, J_1, V_1, \varphi_1, T_1 \rangle$ and $A_2 := \langle S_2, i_2, J_2, V_2, \varphi_2, T_2 \rangle$, the following refinement relations are defined.

This relation is the basis for all the other ones. It applies on automaton with same holes. The goal is to caracterise open automata that have a more determined (more deterministic) behaviour without adding deadlocks.

Definition 14 (Hole-identical refinement)**.** If $J_1 = J_2$ then "A_1 is a hole-identical refinement of A_2", noted $A_1 \leq_= A_2$, is defined as:

$$\exists R : (S_1 \times S_2) \to \mathcal{F}_{V_1 \uplus V_2}, \varphi_1 \uplus \varphi_2 \vdash R(i_1, i_2) \quad \wedge \quad R \text{ hole-identical simulation of } A_1 \text{ in } A_2$$

As in other simulation-like relations, R is a witness of $A_1 \leq_= A_2$. The standard requirements for (bi)simulations are that R relates initial states, transitions can be matched, and target of matched transitions with related sources are also related. For a bisimulation transitions are

matched one-to-many from each automaton to the other, but for a simulation only one way is used. The definition above requires explicitely that initial states are related, the other requirement are encapsulated in the definition (below) of hole-identical simulation.

The specificities of open automata comes in when relating the (initial) states. The relation R relates i_1 to i_2 under the condition that the curent value of variables satisfies $R(i_1, i_2)$. If two states $s_1 \in S_1, s_2 \in S_2$ are not(/never) related then $\nvdash R(s_1, s_2)$. The predicate $R(i_1, i_2)$ is used to take into account the fact that a state is also constituted of the value of the variables. This is already used in FH-bisimulation introduced in previous articles about open automata. So the meaning of $\varphi_1 \uplus \varphi_2 \vdash R(i_1, i_2)$ is that the initial states with the initial valuation are effectively related.

Definition 15 (Hole-identical simulation)**.** "R is a hole-identical simulation of A_1 in A_2" is defined as:

Any idea on how to note that? $R \vDash A_1 \leq_= A_2$ for instance.

$$\forall (s_1, s_2) \in S_1 \times S_2,$$

$$\left(\begin{array}{l} \forall t_1 := \dfrac{\beta_{1j}^{j\in J'}, g_1, \psi_1}{s_1 \xrightarrow{\alpha_1} s_1'} \in T_1, \exists \left(\dfrac{\beta_{2xj}^{j\in J'}, g_{2x}, \psi_{2x}}{s_2 \xrightarrow{\alpha_{2x}} s_{2x}'} \in T_2 \right)^{x\in X}, \\ \forall \sigma : (V_1 \uplus vars(t_1)) \uplus V_2) \to \mathcal{P}, \\ \quad \sigma \vdash R(s_1, s_2) \wedge g_1 \implies \bigvee\limits_{x\in X} \left(\begin{array}{c} \alpha_1 = \alpha_{2x} \wedge \bigwedge\limits_{j\in J'} \beta_{1j} = \beta_{2xj} \\ \wedge\, g_{2x} \wedge R(s_1', s_{2x}')\{\!\{\psi_1 \uplus \psi_{2x}\}\!\} \end{array} \right) \end{array} \right)$$

$$\wedge\, \forall \sigma : (V_1 \uplus V_2) \to \mathcal{P}, \sigma \vdash R(s_1, s_2) \wedge \bigvee_{t_2 \in \mathrm{OT}(s_2)} guard(t_2) \implies \bigvee_{t_1 \in \mathrm{OT}(s_1)} guard(t_1)$$

The first two lines quantify on source states, transitions (and target states) and source states valuations. The 3rd line is the condition to ensure that transitions are not matched to incompatible ones: For all variable assignement of the two automata and all hole actions (quantification of σ), assuming that the states were related and the transition in A_1 is possible ($R(s_1, s_2) \wedge g_1$ part) then there is always a transition, not necessarily only one, which can be performed (g_{2x}), produce the same action ($\alpha_1 = \alpha_{2x}$), accept the same action from the holes ($\beta_{1j} = \beta_{2xj}$) and satisfy the predicate for relating the target states after variable update ($R(s_1', s_{2x}')\{\!\{\psi_1 \uplus \psi_{2x}\}\!\}$).

The notion of refinement here is stating that A_1 is a refinement of A_2 if A_1 can be simulated in A_2. The hole-identical part is referring to the fact that holes indicies are the same and identical holes indicies have to perform the same actions. On top of that the last line ensures no deadlock introduction. It can be interpreted "assuming the predicate holds and there is any possible transition in A_2 (= no deadlock), then there must be a possible transition in A_1 (= no deadlock)". It does not have to ensure that this transition isn't garbage or that the target states are related because the first part of the simulation already does it. Also if there was a deadlock then there is no transition because no trasition can be simulated in a deadlock, that's why it is more a deadlock equivalence than a deadlock reduction.

Example 6.

Lemma 3 (Equivalent definition)**.**

$$\forall (s_1, s_2) \in S_1 \times S_2,$$

$$\left(\begin{array}{l} \forall t_1 := \dfrac{\beta_{1j}^{j\in J'}, g_1, \psi_1}{s_1 \xrightarrow{\alpha_1} s_1'} \in T_1, \exists \left(\dfrac{\beta_{2xj}^{j\in J'}, g_{2x}, \psi_{2x}}{s_2 \xrightarrow{\alpha_{2x}} s_{2x}'} \in T_2 \right)^{x\in X}, \\ \forall \sigma : (V_1 \uplus vars(t_1)) \uplus V_2) \to \mathcal{P}, \\ \quad \sigma \vdash R(s_1, s_2) \wedge g_1 \implies \bigvee\limits_{x\in X} \left(\begin{array}{c} \alpha_1 = \alpha_{2x} \wedge \bigwedge\limits_{j\in J'} \beta_{1j} = \beta_{2xj} \\ \wedge\, g_{2x} \wedge R(s_1', s_{2x}')\{\!\{\psi_1 \uplus \psi_{2x}\}\!\} \end{array} \right) \end{array} \right)$$

$$\iff$$

$$\left(\begin{array}{l} \forall t_1 := \dfrac{\beta_{1j}^{j\in J_1'}, g_1, \psi_1}{s_1 \xrightarrow{\alpha_1} s_1'} \in T_1, \sigma : (V_1 \uplus vars(t_1)) \uplus V_2) \to \mathcal{P}, \\ \quad (\sigma \vdash R(s_1, s_2) \wedge g_1) \implies \exists \dfrac{\beta_{2j}^{j\in J_2'}, g_2, \psi_2}{s_2 \xrightarrow{\alpha_2} s_2'} \in T_2, \\ \qquad \sigma \vdash \alpha_1 = \alpha_2 \wedge \bigwedge\limits_{j\in J_1' \cap J_2'} \beta_{1j} = \beta_{2j} \wedge g_2 \wedge R(s_1', s_2')\{\!\{\psi_1 \uplus \psi_2\}\!\} \end{array} \right)$$

The first property is one part of the hole-identical simulation, the second property is the central part of the refinement simulation requirement. If we prove the direct way of this equivalence and $\leq_=$ is a preorder then $\leq_=$ is a refinement simulation. The other way is there because I hope that it will be easier to use this formulation to prove that $\leq_=$ is a preorder. One may wonder why it was not used in the definition if it is equivalent. The reason is that the formulation used is compatible with SMT solver, the place where it will most probably be used in practise.

Proof. Let $(s_1, s_2) \in S_1 \times S_2$,

$\Rightarrow$: We admit the up formula as H_{def}, let $t_1 := \dfrac{\beta_{1j}^{j\in J_1'}, g_1, \psi_1}{s_1 \xrightarrow{\alpha_1} s_1'} \in T_1$ and $\sigma : (V_1 \uplus vars(t_1)) \uplus V_2) \to \mathcal{P}$ be such that $H_{source} := \sigma \vdash R(s_1, s_2) \wedge g_1$. H_{def} applied to t_1 gives $\left(t_{2x} := \dfrac{\beta_{2xj}^{j\in J_1'}, g_{2x}, \psi_{2x}}{s_2 \xrightarrow{\alpha_{2x}} s_{2x}'} \in T_2 \right)^{x\in X}$ and the property H_{tmp} to which we give σ to get

$$H_{smt} := \sigma \vdash R(s_1, s_2) \wedge g_1 \implies \bigvee_{x\in X} \left(\begin{array}{c} \alpha_1 = \alpha_{2x} \wedge \bigwedge_{j\in J_1'} \beta_{1j} = \beta_{2xj} \\ \wedge\, g_{2x} \wedge R(s_1', s_{2x}')\{\!\{\psi_1 \uplus \psi_{2x}\}\!\} \end{array} \right)$$

Unfolding the definition of $\sigma \vdash f$ where f is not a closed formula in H_{smt} gives an exists that we use to get $\nu : \biguplus\limits_{x\in X} vars(t_{2x}) \to \mathcal{P}$ and

$$H_{valid} := \sigma \uplus \nu \vdash R(s_1, s_2) \wedge g_1 \implies \bigvee_{x\in X} \left(\begin{array}{c} \alpha_1 = \alpha_{2x} \wedge \bigwedge_{j\in J_1'} \beta_{1j} = \beta_{2xj} \\ \wedge\, g_{2x} \wedge R(s_1', s_{2x}')\{\!\{\psi_1 \uplus \psi_{2x}\}\!\} \end{array} \right)$$

$R(s_1, s_2) \wedge g_1$ in H_{valid} does not depends on ν so we can use H_{source} to prove it and get H_{cover}. H_{cover} is a disjunction that we can eliminate, action that gives a value x and

$$H_{target}(x) \coloneqq \sigma \uplus \nu \vdash \alpha_1 = \alpha_{2x} \wedge \bigwedge_{j \in J'_1} \beta_{1j} = \beta_{2xj} \wedge g_{2x} \wedge R(s'_1, s'_{2x}) \{\!\{ \psi_1 \uplus \psi_{2x} \}\!\}$$

Now that we have a specific x we can say that in the bottom formula t_{2x} is a witness in T_2. By doing that, what we are left to prove is $\sigma \vdash \alpha_1 = \alpha_{2x} \wedge \bigwedge_{j \in J'_1 \cap J'_1} \beta_{1j} = \beta_{2xj} \wedge g_2 \wedge R(s'_1, s'_2) \{\!\{ \psi_1 \uplus \psi_2 \}\!\}$.
As a valuation for variables in $vars(t_{2x})$ we have ν, so $H_{target}(x)$ concludes the proof.

$\Leftarrow$: We admit the bottom formula as H_{ref}, let $t_1 \coloneqq \dfrac{\beta_{1j}^{j \in J'}, g_1, \psi_1}{s_1 \xrightarrow{\alpha_1} s'_1} \in T_1$ □

Theorem 1. *The hole-identical refinement relation is a preorder.*

Proof.

Reflexivity: Let A be an open automaton with variables in V. The automaton variables on the right hand side of the relation will be noted v' for the equivalent variable $v \in V$ in the automaton on the left hand side. The simulation $R = \left\{ (s, s) \mapsto \bigwedge_{v \in V} v = v' \,\middle|\, s \in S_1 \right\}$ is a witness of $A \leq A$. Checking it is a simple exercise left to the reader in order to understand how the definition works.

Transitivity: Let A_1, A_2, A_3 be open automata with respectively variables in V_1, V_2, V_3 and states S_1, S_2, S_3. And let R_{12} be a witness of $A_1 \leq_= A_2$ and R_{23} be a witness of $A_2 \leq_= A_3$.

$$R_{13}(s_1, s_3) \coloneqq \exists V_2, \bigvee_{s_2 \in S_2} R_{12}(s_1, s_2) \wedge R_{23}(s_2, s_3)$$

Now let's prove that R_{13} is a witness of $A_1 \leq_= A_3$:

-

□

Theorem 2 (Hole-identical refinement correction)**.**

Proof. □

The goal of the following relation is to capture the case where holes are filled with automata that do not have any hole. It is supposed to be a relation for which filling holes with fully specified automata is a considered refinement.

Definition 16 (Hole-subset refinement)**.** If $J_1 \subseteq J_2$ then "A_1 is a hole-subset refinement of A_2", noted $A_1 \leq_\subseteq A_2$ is defined as:

$$\exists R : (S_1 \times S_2) \to \mathcal{F}_{V_1 \uplus V_2}, \varphi_1 \uplus \varphi_2 \vdash R(i_1, i_2) \wedge R \text{ hole-subset simulation of } A_1 \text{ in } A_2$$

This definitions is essentially the same as the hole-identical one excepted the constraint on holes which has been softened.

Definition 17 (Hole-subset simulation)**.** "R is a hole-subset simulation of A_1 in A_2" is defined as:

$$\forall s_1 \in S_1, s_2 \in S_2,$$

$$\left(\begin{array}{l} \forall \underline{\underline{\beta_{1j}^{j\in J_1'}, g_1, \psi_1}} \in T_1, \exists \left(\underline{\underline{\beta_{2xj}^{j\in J_{2x}'}, g_{2x}, \psi_{2x}}} \in T_2 \right)^{x\in X}, \\ \quad s_1 \xrightarrow{\alpha_1} s_1' \qquad\qquad s_2 \xrightarrow{\alpha_{2x}} s_{2x}' \\ (\forall x \in X, J_1' = J_{2x}' \cap J_1) \wedge \forall \sigma : (V_1 \uplus vars(t_1) \uplus V_2) \to \mathcal{P}, \\ \quad \sigma \vdash R(s_1, s_2) \wedge g_1 \implies \bigvee_{x\in X} \left(\begin{array}{c} \alpha_1 = \alpha_{2x} \wedge \bigwedge_{j\in J_1'} \beta_{1j} = \beta_{2xj} \\ \wedge\ g_{2x} \wedge R(s_1', s_{2x}')\{\!\{\psi_1 \uplus \psi_{2x}\}\!\} \end{array} \right) \end{array} \right)$$

$$\wedge \forall \sigma : (V_1 \uplus V_2) \to \mathcal{P}, \sigma \vdash R(s_1, s_2) \wedge \bigvee_{t_2 \in \mathrm{OT}(s_2)} guard(t_2) \implies \bigvee_{t_1 \in \mathrm{OT}(s_1)} guard(t_2)$$

The difference with the hole-identical simulation is that the holes involved in the matched transitions don't have to be exactly the same anymore. The new constraint is that the holes in common (that is J_1') should be involved at the same time and their action have to match. What happens on the other holes is unspecified except for the fact that, by being free variables they still need to be valued such that the transition is possible.

Example 7.

An alternative version of this definition could enforce $\exists J_2' \subseteq J_2, \forall x \in X, J_{2x}' = J_2'$. Let's call it the alternative hole-subset simulation. This version would mean that the holes involved in every matched transitions must be the same. While this may seem more natural this leads to the unwanted behaviour that an automaton FH-bisimilar to a refinement of some specification might not be a refinement of that specification:

Proposition 1 (Alternative hole-subset simulation is not compatible with FH-bisimulation)**.** *TODO: collapse a transition that was differenciated by having one filled hole involved and it should be simple*

This is a sufficient reason to discard this version although it seems more natural.

Proposition 2 (Hole-subset refinement is compatible with FH-bisimulation)**.**

Example 8.

Proposition 3 (Hole-subset refinement is an extension of hole-identical refinement)**.** *Hole-subset refinement and hole-identical match when holes are identical.*

Proof. When $J_1 = J_2$, $J_1' = J_{2x}' \cap J_1 \iff J_1' = J_{2x}' \cap J_2 \iff J_1' = J_{2x}'$, which is the implicit constraint on hole actions in the hole-identical simulation. By that rewriting their definition match. □

Let's assume that this relation is also correct, the proof of correctness is given later for a more general refinement relation.

Theorem 3 (Composition is a refinement)**.**

Proof. □

Theorem 4 (Context refinement)**.**

Proof. □

Theorem 5 (Congruence with composition)**.**

Proof. □

The goal of the following relation is to capture the case where holes are filled with one hole automata. It is supposed to be a relation for which filling holes with one hole automata is a considered refinement.

Definition 18 (Hole-matching refinement)**.** If $|J_1| = |J_2|$ then "A_1 is a hole-matching refinement of A_2", noted $A_1 \leq_{\#} A_2$ is defined as:

$$\begin{aligned}\exists R : (S_1 \times S_2) \to \mathcal{F}, \quad & \exists f : J_1 \setminus J_2 \to J_2 \setminus J_1, \\ & \varphi_1 \uplus \varphi_2 \vdash R(i_1, i_2) \quad \wedge \quad f(J_1 \setminus J_2) = J_2 \setminus J_1 \\ & \qquad\qquad \wedge R \text{ hole-f-matching simulation of } A_1 \text{ in } A_2\end{aligned}$$

This definitions is essentially the same as the hole-identical except that the constraint on holes has been softened and it is compensated with a invertible map between non-shared holes.

Definition 19 (Hole-f-matching simulation)**.** R is a hole-f-matching simulation of A_1 in A_2 is defined as:

$$\begin{aligned}&\forall s_1 \in S_1, s_2 \in S_2, \\ &\left(\begin{array}{l} \forall \frac{\beta_{1j}^{j \in J_1'}, g_1, \psi_1}{s_1 \xrightarrow{\alpha_1} s_1'} \in T_1, \exists \left(\frac{\beta_{2xj}^{j \in J_{2x}'}, g_{2x}, \psi_{2x}}{s_2 \xrightarrow{\alpha_{2x}} s_{2x}'} \in T_2 \right)^{x \in X}, \\ (\forall x \in X, J_1' \cap J_2 = J_{2x}' \cap J_1 \wedge f(J_1' \setminus J_2) \subseteq J_{2x}') \wedge \forall \sigma : (V_1 \uplus vars(t_1) \uplus V_2) \to \mathcal{P}, \\ \quad \sigma \vdash R(s_1, s_2) \wedge g_1 \implies \bigvee_{x \in X} \left(\begin{array}{c} \alpha_1 = \alpha_{2x} \wedge \bigwedge_{j \in J_1' \cap J_2} \beta_{1j} = \beta_{2xj} \\ \wedge\, g_{2x} \wedge R(s_1', s_{2x}') \{\!\{ \psi_1 \uplus \psi_{2x} \}\!\} \end{array} \right) \end{array} \right) \\ &\qquad \wedge \forall \sigma : (V_1 \uplus V_2) \to \mathcal{P}, \sigma \vdash R(s_1, s_2) \wedge \bigvee_{t_2 \in \mathrm{OT}(s_2)} guard(t_2) \implies \bigvee_{t_1 \in \mathrm{OT}(s_1)} guard(t_1)\end{aligned}$$

7 Refinement relation for open automata

Definition 20 (Open automata refinement)**.** For any two open automata $A_1 \coloneqq \langle S_1, i_1, J_1, V_1, \varphi_1, T_1 \rangle$ and $A_2 \coloneqq \langle S_2, i_2, J_2, V_2, \varphi_2, T_2 \rangle$, "$A_1$ is a refinement of A_2", noted $A_1 \leq A_2$, is defined as:

$$\begin{aligned}\exists R : (S_1 \times S_2) \to \mathcal{F}_{V_1 \uplus V_2}, \exists f : J_1 \setminus J_2 \to J_2 \setminus J_1, & \\ \varphi_1 \uplus \varphi_2 \vdash R(i_1, i_2) \wedge R \text{ f simulation of } A_1 \text{ in } A_2 & \end{aligned}$$

Definition 21 (f simulation)**.** R is a f simulation of A_1 in A_2 is defined as:

$$\forall s_1 \in S_1, s_2 \in S_2,$$

$$\left(\begin{array}{l} \forall \underline{\underline{\beta_{1j}^{j\in J_1'}, g_1, \psi_1}} \in T_1, \exists \left(\underline{\underline{\beta_{2xj}^{j\in J_{2x}'}, g_{2x}, \psi_{2x}}} \in T_1 \right)^{x\in X}, \\ \quad s_1 \xrightarrow{\alpha_1} s_1' \qquad\qquad s_2 \xrightarrow{\alpha_{2x}} s_{2x}' \\ (\forall x \in X, J_1' \cap J_2 = J_{2x}' \cap J_1 \wedge f(J_1' \setminus J_2) \subseteq J_{2x}') \wedge \forall \sigma : (V_1 \uplus vars(t_1) \uplus V_2) \to \mathcal{P}, \\ \quad \sigma \vdash R(s_1, s_2) \wedge g_1 \implies \bigvee_{x\in X} \left(\begin{array}{c} \alpha_1 = \alpha_{2x} \wedge \bigwedge_{j\in J_1'\cap J_2} \beta_{1j} = \beta_{2xj} \\ \wedge\, g_{2x} \wedge R(s_1', s_{2x}') \{\!\{ \psi_1 \uplus \psi_{2x} \}\!\} \end{array} \right) \end{array} \right)$$

$$\wedge \forall \sigma : (V_1 \uplus V_2) \to \mathcal{P}, R(s_1, s_2) \wedge \bigvee_{t_2 \in \mathrm{OT}(s_2)} guard(t_2) \implies \bigvee_{t_1 \in \mathrm{OT}(s_1)} guard(t_1)$$

The objective of the first part of this definition is to be able to simulate A_1 in A_2 when holes of the same name receive the same closed automaton while other holes receive "compatible" closed automata. The objective of the other part of this definition is to prevent the appearance of new deadlocks, which should mean with the first property that the compared automaton are deadlock equivalent.

8 New equivalence relation induced by pre-order

9 Conclusion