A Characterisation of Open Bisimilarity using an Intuitionistic Modal Logic

Ki Yung Ahn[1], **Ross Horne**[1], **and Alwen Tiu**[1]

1 **School of Computer Science and Engineering, Nanyang Technological University, Singapore**
`{yaki,rhorne,atiu}@ntu.edu.sg`

Abstract

Open bisimilarity is a strong bisimulation congruence for the π-calculus. In open bisimilarity, free names in processes are treated as variables that may be instantiated; in contrast to late bisimilarity where free names are constants. An established modal logic due to Milner, Parrow, and Walker characterises late bisimilarity, that is, two processes satisfy the same set of formulae if and only if they are bisimilar. We propose an intuitionistic variation of this modal logic and prove that it characterises open bisimilarity. The soundness proof is mechanised in Abella. The completeness proof provides an algorithm for generating distinguishing formulae, useful for explaining and certifying whenever processes are non-bisimilar.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases bisimulation, modal logic, intuitionistic logic

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2017.7

1 Introduction

In this work, we consider open bisimilarity [13] which ensures processes equivalence under any context at any point in their execution. Open bisimulation is an appealing choice of equivalence for state-space reduction due to its lazy *call-by-need* approach to inputs, which makes it easier to automate [15]. In such a call-by-need approach, a value received is only observed when it needs to be used. Furthermore, some process calculi have been shown to enjoy sound and complete algebraic characterisations with respect to open bisimilarity.

The fine algebraic properties of open bisimilarity may be desirable for some applications. For many applications, it is desirable to avoid a situation where an equivalence technique proves that two components are equivalent in a sandbox test environment, when, in fact, they are distinguishable when plugged into a larger system. More subtly, processes may change context during execution [10]; for example, virtual machines migrate between devices, and replicas replace components at runtime to keep a system live in the face of unavoidable node failures. For some notions of observational equivalence two processes may be indistinguishable when executed in any context prescribed; however, if the same two processes execute a few steps and then are migrated to another context, then it is possible that, from that point, the processes can exhibit observably distinct behaviours.

Process equivalences for the π-calculus coarser than open bisimilarity are prone to limitations described above. For instance, late bisimilarity [8] is not a congruence, since it is not preserved by input prefixes. Furthermore, even if we take the greatest congruence relation contained in late bisimilarity, called *late congruence*, late congruence is no longer a bisimulation hence is not necessarily preserved during execution. These issues are remedied by open bisimilarity [13].

The problem we address is the nature of a modal logic characterising open bisimilarity, in the tradition pioneered by Hennessy and Milner [6]. A modal logic characterising a bisimulation should have the property that whenever two processes are not bisimilar there should exist a distinguishing formula in the modal logic that holds for one process, but not for the other process. Such distinguishing formulae are useful for explaining why two processes are not bisimilar. Modal logics characterising late bisimilarity and coarser bisimulations were developed early in the literature on the π-calculus, by Milner, Parrow and Walker [9].

A novelty of our modal logic characterising open bisimilarity, which we name $\mathcal{OM}$, is that it is intuitionistic rather than classical. A non-classical feature of $\mathcal{OM}$ is that box and diamond modalities have independent interpretations, except in special cases such as $[\tau]\mathtt{ff}$ which is equivalent to $\neg\langle\tau\rangle\mathtt{tt}$. In general, in $\mathcal{OM}$ it is rarely the case that box can be defined in terms of diamond and negation. This contrasts to a classical modal logic we would expect that $[\pi]\phi$ and $\neg\langle\pi\rangle\neg\phi$ define equivalent formulae, however such de Morgan dualities do not hold for most $\mathcal{OM}$ formulae.

More profoundly, the law of excluded middle does not hold in $\mathcal{OM}$. For example, the process $\overline{a}b \parallel c(x)$ does not satisfy the formula $\langle\tau\rangle\mathtt{tt} \vee \neg\langle\tau\rangle\mathtt{tt}$, that is, $\overline{a}b \parallel c(x) \not\models \langle\tau\rangle\mathtt{tt} \vee \neg\langle\tau\rangle\mathtt{tt}$. The failure of the formula above relies on the fact that we have not yet fixed whether $a = c$ or $a \neq c$, which amounts to the absence of the law of excluded middle for name equality, as observed in related work on logical encodings of open bisimilarity [16]. In open bisimulation, both a and c are variables that may or may not be instantiated with the same value.

As a further example, consider the following two processes.

$$R \triangleq \tau.(\overline{a}b.a(x) + a(x).\overline{a}b + \tau) + \tau.(\overline{a}b.c(x) + c(x).\overline{a}b) \qquad\qquad S \triangleq R + \tau.(\overline{a}b \parallel c(x))$$

The above processes are not open bisimilar. Process R satisfies $[\tau](\langle\tau\rangle\mathtt{tt} \vee \neg\langle\tau\rangle\mathtt{tt})$ but process S does not, since there is a τ-transition to process $\overline{a}b \parallel c(x)$ that we just agreed above does not satisfy $\langle\tau\rangle\mathtt{tt} \vee \neg\langle\tau\rangle\mathtt{tt}$. In this example, the absence of the law of excluded middle is necessary for the existence of a formula distinguishing these processes in $\mathcal{OM}$.

The absence of de Morgan dualities discussed above complicates the construction of distinguishing formulae for processes that are not open bisimilar. For example, τ and $[a = c]\tau$ are not open bisimilar, so there should be a formula distinguishing these processes. Such a formula is $\langle\tau\rangle\mathtt{tt}$, for which $\tau \models \langle\tau\rangle\mathtt{tt}$ and $[a = c]\tau \not\models \langle\tau\rangle\mathtt{tt}$. This particular construction has a bias towards τ. In the classical setting of modal logic for late bisimilarity, given such a distinguishing formula, we can dualise it to obtain another distinguishing formula $\neg\langle\tau\rangle\mathtt{tt}$ that has a bias towards $[a = c]\tau$, i.e., $[a = c]\tau \models \neg\langle\tau\rangle\mathtt{tt}$ but $\tau \not\models \neg\langle\tau\rangle\mathtt{tt}$. This dual construction **fails** in the case of our intuitionistic modal logic characterising open bisimilarity. In the intuitionistic setting, we have both $\tau \not\models \neg\langle\tau\rangle\mathtt{tt}$ and $[a = c]\tau \not\models \neg\langle\tau\rangle\mathtt{tt}$. To address this problem, our algorithm (c.f. Section 3) simultaneously constructs two distinguishing formulae, that are not necessarily dual to each other.

The precise semantics is presented in the body of this paper. The techniques are clean and modular, so results extend to open bisimilarity for more expressive process calculi.

Outline

Section 2 introduces the semantics of Open Milner–Parrow–Walker logic ($\mathcal{OM}$) and states the soundness and completeness results. Section 3 presents the proof of the correctness of an algorithm for generating distinguishing formulae, which is used to establish completeness of the logic with respect to open bisimilarity.

2 Open Milner–Parrow–Walker logic ($\mathcal{OM}$)

We recall the syntax and labelled transition semantics for the finite π-calculus (Fig. 1). All features are standard: the deadlocked process that can do nothing, the ν quantifier that binds private names, the output prefix that outputs a name on a channel, the input prefix that binds the name received

$$
\begin{array}{lll}
\pi & ::= \tau & \text{(progress)} \\
 & \quad \overline{x}z & \text{(free out)} \\
 & \quad \overline{x}(z) & \text{(bound out)} \\
 & \quad x(z) & \text{(input)} \\
 & & \\
P & ::= 0 & \text{(deadlock)} \\
 & \quad \nu x.P & \text{(nu)} \\
 & \quad \pi.P & \text{(action)} \\
 & \quad [x = y]P & \text{(match)} \\
 & \quad P \parallel P & \text{(par)} \\
 & \quad P + P & \text{(choice)}
\end{array}
$$

$$\frac{}{\pi.P \xrightarrow{\pi} P} \qquad \frac{P \xrightarrow{\overline{x}z} Q}{\nu z.P \xrightarrow{\overline{x}(z)} Q}\ x \neq z$$

$$\frac{P \xrightarrow{\pi} R}{P + Q \xrightarrow{\pi} R} \qquad \frac{P \xrightarrow{\pi} R}{[x = x]P \xrightarrow{\pi} R}$$

$$\frac{P \xrightarrow{\pi} Q}{\nu x.P \xrightarrow{\pi} \nu x.Q}\ x \notin \mathrm{n}(\pi) \qquad \frac{P \xrightarrow{\pi} Q}{P \parallel R \xrightarrow{\pi} Q \parallel R}\ \begin{array}{c}\text{if } x \in \mathrm{bn}(\pi)\\ \text{then } x \text{ fresh for } R\end{array}$$

$$\frac{P \xrightarrow{\overline{x}(z)} P' \quad Q \xrightarrow{x(z)} Q'}{P \parallel Q \xrightarrow{\tau} \nu z.(P' \parallel Q')} \qquad \frac{P \xrightarrow{\overline{x}y} P' \quad Q \xrightarrow{x(z)} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'\{^{y}\!/\!_{z}\}}$$

Figure 1 Syntax and semantics of the π-calculus, plus symmetric rules for choice and parallel composition, where $\mathrm{n}(x(y)) = \mathrm{n}(\overline{x}(y)) = \mathrm{n}(\overline{x}y) = \{x, y\}$, $\mathrm{bn}(x(y)) = \mathrm{bn}(\overline{x}(y)) = \{y\}$ and $\mathrm{n}(\tau) = \mathrm{bn}(\tau) = \mathrm{bn}(\overline{x}y) = \emptyset$; and α-conversion is such that $\nu x.P$, $z(x).P$ and $\overline{z}(x).P$ bind x in P.

on a channel, the silent progress action τ, the name match guard, parallel composition and non-deterministic choice. There are four types of action ranged over by π, where a *free output* sends a free name, whereas a *bound output* extrudes a ν-bounded private name. Stylistically, the semantics is the late labelled transition system for the π-calculus, where the name on the input channel is a symbolic place holder for a name that is not chosen until after an input transition.

Histories are used to define both the intuitionistic modal logic and open bisimilarity. Histories represent what is known about free variables due to how they have been communicated previously to the environment. There are two types of event to record in a history: The output of a fresh private name, using action $\overline{a}(x)$, which is denoted x^o; and a (symbolic) input, using action $a(z)$, which is denoted z^i. The only thing that matters about the order of events in the history is the alternation between the bound outputs and symbolic inputs, since an input variable can only be instantiated with private names that were output earlier in the history. E.g., for history $x^o \cdot z^i$, input variable z may be instantiated with private name x; in contrast, for history $z^i \cdot x^o$, input variable z may not be instantiated with private name x. This is reflected by the constraints on substitutions in the following inductive definition.

▶ **Definition 1** (σ respecting h)**.** A substitution σ invariant on names not in $\mathrm{fv}(h)$ is respecting a history h according to the following inductive definition.

$$\frac{}{\sigma \text{ respecting } \epsilon} \qquad \frac{\sigma \text{ respecting } h}{\sigma \text{ respecting } h \cdot x^i} \qquad \frac{x \notin \mathrm{dom}(\sigma) \cup \mathrm{fv}(h\sigma) \quad \sigma \text{ respecting } h}{\sigma \text{ respecting } h \cdot x^o}$$

Note that the above inductive definition fulfils the role of sets of inequality constraints called *distinctions* in the original work on open bisimilarity [13]. The definition above also captures the alternations between nominal and universal quantifiers in embeddings of open bisimilarity in the intuitionistic logic LINC [16, 3]. Although distinctions are more general than histories, it is shown in [16] that given a history h and its corresponding distinction D, the corresponding definitions of open bisimilarity coincide.

2.1 The semantics of the intuitionistic modal logic $\mathcal{OM}$

The semantics of the modal logic $\mathcal{OM}$ is defined in terms of the late labelled transitions system (Fig. 2) and history respecting substitutions (Definition 1). Intuitively, each judgement must hold for all possible respectful substitutions, which explains the asymmetry between the box and diamond

$$\begin{array}{lll}
P \models^h \mathtt{tt} & & \text{always holds.} \\
P \models^h \phi_1 \wedge \phi_2 & \text{iff} & P \models^h \phi_1 \text{ and } P \models^h \phi_2. \\
P \models^h \phi_1 \vee \phi_1 & \text{iff} & P \models^h \phi_1 \text{ or } P \models^h \phi_2. \\
P \models^h \langle x = x \rangle \phi & \text{iff} & P \models^h \phi. \\
P \models^h \langle \alpha \rangle \phi & \text{iff} & \exists\, Q,\ P \xrightarrow{\alpha} Q \text{ and } Q \models^h \phi. \\
P \models^h \langle \overline{a}(z) \rangle \phi & \text{iff} & \exists\, Q,\ P \xrightarrow{\overline{a}(z)} Q \text{ and } Q \models^{h \cdot z^o} \phi. \\
P \models^h \langle a(z) \rangle \phi & \text{iff} & \exists\, Q,\ P \xrightarrow{a(z)} Q \text{ and } Q \models^{h \cdot z^i} \phi. \\
P \models^h [x = y] \phi & \text{iff} & \forall \sigma \text{ respecting } h,\ x\sigma = y\sigma \implies P\sigma \models^{h\sigma} \phi\sigma. \\
P \models^h [\alpha] \phi & \text{iff} & \forall \sigma \text{ respecting } h, \forall Q,\ P\sigma \xrightarrow{\alpha\sigma} Q \implies Q \models^{h\sigma} \phi\sigma. \\
P \models^h [\overline{a}(z)] \phi & \text{iff} & \forall \sigma \text{ respecting } h,\ \forall Q, P\sigma \xrightarrow{\overline{a\sigma}(z)} Q \implies Q \models^{h\sigma \cdot z^o} \phi\sigma. \\
P \models^h [a(z)] \phi & \text{iff} & \forall \sigma \text{ respecting } h, \forall Q, P\sigma \xrightarrow{a\sigma(z)} Q \implies Q \models^{h\sigma \cdot z^i} \phi\sigma.
\end{array}$$

$$\begin{array}{rcll}
h & ::= & \epsilon & \text{(empty)} \\
 & & h \cdot x^o & \text{(name)} \\
 & & h \cdot x^i & \text{(variable)} \\
\phi & ::= & \mathtt{tt} & \text{(true)} \\
 & & \mathtt{ff} & \text{(false)} \\
 & & \phi \wedge \phi & \text{(and)} \\
 & & \phi \vee \phi & \text{(or)} \\
 & & \langle x = y \rangle \phi & \text{(dia-match)} \\
 & & \langle \pi \rangle \phi & \text{(dia-action)} \\
 & & [x = y] \phi & \text{(box-match)} \\
 & & [\pi] \phi & \text{(box-action)}
\end{array}$$

Figure 2 Syntax and semantics of the modal logic $\mathcal{OM}$, where α is τ or $\overline{a}b$; and z is fresh for P, h, and σ.

modalities. For the diamond modality $\langle \pi \rangle$, a π transition must be possible regardless of the substitution. It is sufficient to consider the identity substitution because applying a respectful substitution cannot prevent a transition. For the box modality $[\pi]$ there may exist substitutions σ other than the identity substitution enabling a $\pi\sigma$ transition, hence we should consider all respectful substitutions.

► **Definition 2** (satisfaction)**.** Process P satisfies formula ϕ with history h, written $P \models^h \phi$, according to the inductive definition in Fig. 2. Satisfaction, written $P \models \phi$, is satisfaction with a history of inputs $x_0^i \cdot \ldots \cdot x_n^i$, where $\mathrm{fv}(P) \subseteq \{x_0, \ldots, x_n\}$.

2.1.1 Why an intuitionistic modal logic?

In the open bisimulation game, every transition step is closed under respectful substitutions. Modal logic $\mathcal{OM}$ reflects in its semantics the substitutions that can be applied to a process. A natural semantics would be a Kripke-like semantics, where *worlds* are process-history pairs and the accessibility relation relates instances of such world. More precisely, consider a relation on worlds as follows: $(P, h) \leq (Q, h')$ iff there exists a substitution σ respecting h such that $P\sigma = Q$ and $h\sigma = h'$. The pair $(\mathcal{P}, \leq)$, where $\mathcal{P}$ is the set of worlds, forms a Kripke frame that is reflexive and transitive. Consequently, we obtain a semantics for an intuitionistic logic, where implication is closed under respectful substitutions as follows.

$$P \models^h \phi_1 \supset \phi_2 \qquad \text{iff} \qquad \forall \sigma \text{ respecting } h, P\sigma \models^{h\sigma} \phi_1\sigma \implies P\sigma \models^{h\sigma} \phi_2\sigma$$

Intuitionistic negation $\neg\phi$ can then be defined as $\phi \supset \mathtt{ff}$.

Recall the example from the introduction $\overline{a}b \parallel c(x) \not\models \langle\tau\rangle\mathtt{tt} \vee \neg\langle\tau\rangle\mathtt{tt}$, demonstrating that the law of excluded middle is invalid. Neither $\overline{a}b \parallel c(x) \models \langle\tau\rangle\mathtt{tt}$ nor $\overline{a}b \parallel c(x) \models \neg\langle\tau\rangle\mathtt{tt}$ hold. The former holds only if $\overline{a}b \parallel c(x)$ is guaranteed to make a τ transition; but such a transition is only possible assuming $a = c$, hence $\overline{a}b \parallel c(x) \not\models \langle\tau\rangle\mathtt{tt}$. For the latter, we should consider all substitutions which enable a τ transition; and, since such a substitution $\{^c/_a\}$ exists, $\overline{a}b \parallel c(x) \not\models \neg\langle\tau\rangle\mathtt{tt}$. Notice that the satisfaction would hold by forcing the assumption $a \neq c$. Of course, for open bisimilarity we make no a priori assumption about whether $a = c$ or $a \neq c$, since both are variables that may, or may not, be instantiated with the same value.

The intuitionistic implication and negation above are used only to explain the origin of $\mathcal{OM}$ and for contrast with properties expected of classical modal logics. The distinguishing formula algorithm, considered in subsequent sections, does not depend on these connectives.

2.2 Open bisimilarity, soundness and completeness

We recall the definition of open bisimilarity. Open bisimilarity is a greatest fixed point of symmetric relations closed under all respectful substitutions and labelled transitions actions at every step. Notice that a symbolic input or output of a fresh private name updates the history.

▶ **Definition 3** (open bisimilarity)**.** Open bisimilarity with history h is the greatest symmetric relation such that: if $P \sim^{h} Q$ then, for all substitutions σ respecting h, the following hold, where α is a τ or $\overline{a}b$ action and x is fresh for $P\sigma$, $Q\sigma$ and $h\sigma$:

- $P\sigma \xrightarrow{\alpha\sigma} P' \implies \exists Q', Q\sigma \xrightarrow{\alpha\sigma} Q'$ and $P' \sim^{h\sigma} Q'$.
- $P\sigma \xrightarrow{\overline{a\sigma}(x)} P' \implies \exists Q', Q\sigma \xrightarrow{\overline{a\sigma}(x)} Q'$ and $P' \sim^{h\sigma \cdot x^{o}} Q'$.
- $P\sigma \xrightarrow{a\sigma(x)} P' \implies \exists Q', Q\sigma \xrightarrow{a\sigma(x)} Q'$ and $P' \sim^{h\sigma \cdot x^{i}} Q'$.

Open bisimilarity, written $P \sim Q$, is defined to be open bisimilarity with a history $x_0^i \cdot \ldots \cdot x_n^i$ such that $\mathrm{fv}(P) \cup \mathrm{fv}(Q) \subseteq \{x_0, \ldots, x_n\}$.

2.2.1 Soundness and completeness results

The main result of this paper is that, for finite π-calculus processes open bisimilarity ($\sim$) coincides with the relation between processes with no distinguishing formula ($\overset{\models}{\sim}$).

▶ **Definition 4** (logical equivalence)**.** $P \overset{\models}{\sim} Q$ is defined whenever, for all ϕ, $P \models \phi$ iff $Q \models \phi$.

▶ **Theorem 5** (soundness)**.** *For π-calculus processes (including replication), $P \sim Q$ implies $P \overset{\models}{\sim} Q$.*

▶ **Theorem 6** (completeness)**.** *For finite π-calculus processes, $P \overset{\models}{\sim} Q$ implies $P \sim Q$.*

The proof of soundness has been mechanically checked in the proof assistant Abella [2] using the two-level logic approach [4] to reason about the π-calculus semantics specified in λProlog [11]. The proof of soundness proceeds by induction on the structure of the logical formulae in the definition of logical equivalence. The proof of completeness is explained in detail in Section 3. Soundness extends to infinite π-calculus processes with replication, but completeness holds for decidable fragments such as Fig. 1.

Firstly, we provide examples demonstrating the implications of Theorems 5 and 6. Due to soundness, if two processes are bisimilar, we cannot find a distinguishing formula that holds for one process but does not hold for the other process. Due to completeness, if it is impossible to prove that two process are open bisimilar, then we can construct a distinguishing formula that holds for one process but does not hold for the other process. Thus Theorems 5 and 6 guarantee that an $\mathcal{OM}$ formulae can be used to characterise non-bisimilarity.

2.2.2 Example processes distinguishable by postconditions

All modalities are essential for the soundness and completeness of $\mathcal{OM}$. Perhaps the least obvious modality is $\langle x = y \rangle$. When prefixed with a box modality it indicates a postcondition that always holds after an action. To see, this consider the process $[x = y]\tau$. The judgement $[x = y]\tau \models [\tau]\langle x = y \rangle \mathtt{tt}$ holds since for any θ such that $([x = y]\tau)\theta \xrightarrow{\tau} 0$ it must be the case that $x\theta = y\theta$. Hence, by definition of diamond, $0 \models \langle x\theta = y\theta \rangle \mathtt{tt}$ iff $0 \models \mathtt{tt}$. In contrast, $\tau \not\models [\tau]\langle x = y \rangle \mathtt{tt}$ since, taking the identity substitution in the definition of box, $\tau \xrightarrow{\tau} 0$, but $0 \models \langle x = y \rangle \mathtt{tt}$ cannot be proven in general. The formula $[\tau]\langle x = y \rangle \mathtt{tt}$ is therefore a distinguishing formula satisfied by $[x = y]\tau$ but not τ.

The use of $\langle x = y \rangle$ as a postcondition contrasts to the use of $[x = y]$ as a precondition. Consider the same process as above with the formula $[x = y]\langle \tau \rangle \mathtt{tt}$. Observe that, for substitutions θ such that

$x\theta = y\theta$, $([x = y]\tau)\theta \xrightarrow{\tau} 0$ and $0 \models \mathtt{tt}$ holds, hence $([x = y]\tau)\theta \models \langle\tau\rangle\mathtt{tt}$; and thereby the judgement $[x = y]\tau \models [x = y]\langle\tau\rangle\mathtt{tt}$ holds. In contrast, $0 \not\models [x = y]\langle\tau\rangle\mathtt{tt}$.

We will return to the above two formulae shortly, as they are critical for the algorithm for generating distinguishing formulae.

2.3 Sketch of algorithm for generating distinguishing formulae

The completeness proof in Section 3 relies on an algorithm for generating distinguishing formulae for non-bisimilar processes. Here, we provide a sketch of the algorithm executed on key examples.

2.3.1 Example requiring intuitionistic assumptions

The algorithm proceeds over the structure of a tree of moves that show two processes are non-bisimilar. In base cases, we have a pair of processes where, under a substitution, one process can make a transition, but the other process cannot match the transition. We revisit two examples of base cases, discussed previously:

$[x = y]\tau \not\sim \mathbf{0}$ **:** The left process leads by $([x = y]\tau)\{^y/_x\} \xrightarrow{\tau} 0$, but 0 cannot make a τ transition, under any substitution; hence $[x = y]\tau \models [x = y]\langle\tau\rangle\mathtt{tt}$ and $0 \models [\tau]\mathtt{ff}$ are distinguishing formulae.

$[x = y]\tau \not\sim \tau$ **:** The right process leads by $\tau \xrightarrow{\tau} 0$, but $[x = y]\tau\theta$ can make a τ transition only when $x\theta = y\theta$; hence $[x = y]\tau \models [\tau]\langle x = y\rangle\mathtt{tt}$ and $\tau \models \langle\tau\rangle\mathtt{tt}$ are distinguishing formulae.

In an inductive case, the two processes cannot be distinguished by an immediate transition. However, under some substitutions, one process can make a π transition to a state, say P', that, under the same substitution the other process can only make a corresponding π transition to reach states Q'_i that are non-bisimilar to P'. This allows a distinguishing formula to be inductively constructed from the distinguishing formulae for P' paired with each Q'_i.

For example, consider how the algorithm would find distinguishing formulae for P and Q below.

$$\begin{tikzcd} P \triangleq \tau.[x = y]\tau + \tau + \tau.\tau \arrow[d, "\tau"] & \not\sim & \tau + \tau.\tau \triangleq Q \arrow[d, "\tau"'] \arrow[dr, bend left, "\tau"] & \\ P' \triangleq [x = y]\tau & & 0 \triangleq Q'_1 & \tau \triangleq Q'_2 \end{tikzcd}$$

The first step in the strategy for non-bisimilarity is to show that P can make a τ transition to a state that is not bisimilar to any state reachable by a τ transition from the other process. One possibility is the transition to P' as illustrated above. In reply, Q may attempt a corresponding τ transition either to Q'_1 or Q'_2. Inductively, we require that $P' \not\sim Q'_1$ and $P' \not\sim Q'_2$. Both are instances of the base case discussed above where we discovered distinguishing formulae for each of them.

This enables us to construct distinguishing formulae for the inductive case. The distinguishing formula satisfied by P is a diamond followed by the conjunction of the left distinguishing formulae that is satisfied by P' in the base cases: $\tau.[x = y]\tau + \tau + \tau.\tau \models \langle\tau\rangle([\tau]\langle x = y\rangle\mathtt{tt} \wedge [x = y]\langle\tau\rangle\mathtt{tt})$. The distinguishing formula satisfied by Q is a box followed by the disjunction of the right distinguishing formulae from the base cases: $\tau + \tau.\tau \models [\tau](\langle\tau\rangle\mathtt{tt} \vee [\tau]\mathtt{ff})$.

To confirm that they are indeed distinguishing formulae for P and Q, swap the processes and formulae above to observe that each process fails to satisfy the other formula. To be precise, assume for contradiction that $\tau{+}\tau.\tau \models \langle\tau\rangle([\tau]\langle x = y\rangle\mathtt{tt} \wedge [x = y]\langle\tau\rangle\mathtt{tt})$ holds. By definition of $\langle\tau\rangle$, this holds iff either $0 \models [\tau]\langle x = y\rangle\mathtt{tt} \wedge [x = y]\langle\tau\rangle\mathtt{tt}$ or $\tau \models [\tau]\langle x = y\rangle\mathtt{tt} \wedge [x = y]\langle\tau\rangle\mathtt{tt}$ holds. Now observe that $0 \models [x = y]\langle\tau\rangle\mathtt{tt}$ holds iff we make the additional assumption in the meta framework that x and y are persistently distinct, i.e., for all σ, $x\sigma \neq y\sigma$. In addition, observe that $\tau \models [\tau]\langle x = y\rangle\mathtt{tt}$ holds iff we make the additional assumption in the meta framework that x and y are persistently equal, i.e., for all σ, $x\sigma = y\sigma$. In fact, by these observations we are able to mechanically prove the following in intuitionistic framework Abella: $\tau + \tau.\tau \models \langle\tau\rangle([\tau]\langle x = y\rangle\mathtt{tt} \wedge [x = y]\langle\tau\rangle\mathtt{tt})$ iff $\forall x, y.\,(x = y \vee x \neq y)$.

Notice that $\forall x, y. (x = y \vee x \neq y)$ is an instance of the law of excluded middle; hence, assuming the law of excluded middle, the formula for Q also holds for P; and vice versa. Indeed there would be no distinguishing formulae for these processes; and hence in a classical framework the modal logic would be incomplete. Fortunately, since intuitionistic logics do not assume the law of excluded middle, as long as we evaluate the semantics in an intuitionistic framework, we are able to establish that $Q \not\models \langle\tau\rangle([\tau]\langle x = y\rangle\mathtt{tt} \wedge [x = y]\langle\tau\rangle\mathtt{tt})$, as required.

2.3.2 Example involving private names that are distinguishable

The alternation between inputs and outputs in the history affects what counts as a respectful substitution. Intuitively, respectful substitutions ensure that a private name can never be input earlier than it was output. Consider the following processes: $P \triangleq \nu x.\overline{a}x.a(y).\tau \nsim \nu x.\overline{a}x.a(y).[x = y]\tau \triangleq Q$.

These processes are not open bisimilar because P can make the following three transition steps: $\nu x.\overline{a}x.a(y).\tau \xrightarrow{\overline{a}(x)} a(y).\tau \xrightarrow{a(y)} \tau \xrightarrow{\tau} 0$. However, Q can only match the first two steps. At the third step, a base case of the algorithm for $\tau \nsim^{a^i x^o y^i} [x = y]\tau$ applies. In this case, any substitution θ respecting $a^i x^o y^i$ where $[x = y]\tau\theta \xrightarrow{\tau} 0$ is such that $y\theta = x$, $x \notin \mathrm{dom}(\theta)$ and $a\theta \neq x$, which is satisfiable. Thus $[x = y]\tau \models^{a^i x^o y^i} [\tau]\langle x = y\rangle\mathtt{tt}$ and $\tau \models^{a^i x^o y^i} \langle\tau\rangle\mathtt{tt}$. By applying inductive cases, we obtain $\nu x.\overline{a}x.a(y).\tau \models \langle\overline{a}(x)\rangle\langle a(y)\rangle\langle\tau\rangle\mathtt{tt}$ and $\nu x.\overline{a}x.a(y).[x = y]\tau \models [\overline{a}(x)][a(y)][\tau]\langle x = y\rangle\mathtt{tt}$.

2.3.3 Example involving private names that are indistinguishable

In contrast to the previous example, consider the following processes where a fresh name is output and compared to a name already known: $\nu x.\overline{a}x \sim \nu x.\overline{a}x.[x = a]\tau$.
These processes are open bisimilar, hence by Theorem 5 there is no distinguishing formula. The existence of a distinguishing formula of the form $\langle\overline{a}(x)\rangle[x = a]\langle\tau\rangle\mathtt{tt}$ is *prevented* by the history. Both $\nu x.\overline{a}x.[x = a]\tau \models \langle\overline{a}(x)\rangle[x = a]\langle\tau\rangle\mathtt{tt}$ and $\nu x.\overline{a}x \models \langle\overline{a}(x)\rangle[x = a]\langle\tau\rangle\mathtt{tt}$ hold. The latter holds since $\nu x.\overline{a}x \models^{a^i} \langle\overline{a}(x)\rangle[x = a]\langle\tau\rangle\mathtt{tt}$ holds if and only if $\nu x.\overline{a}x \xrightarrow{\overline{a}(x)} 0$ and $0 \models^{a^i x^o} [x = a]\langle\tau\rangle\mathtt{tt}$. By definition of $[x = a]$, this holds if only if for all θ respecting $a^i x^o$ and such that $x\theta = a\theta$, $0 \models^{a^i x^o} \langle\tau\rangle\mathtt{tt}$. Clearly 0 cannot make a τ transition, hence $0 \models^{a^i x^o} \langle\tau\rangle\mathtt{tt}$ does not hold. However, fortunately, there is no substitution θ respecting $a^i x^o$ such that $x\theta = a\theta$. By the definition of respecting substitution, θ must satisfy $x \notin \mathrm{dom}(\theta)$ and $x \neq a\theta$, contradicting constraint $x\theta = a\theta$. Thereby $0 \models^{a^i x^o} [x = a]\langle\tau\rangle\mathtt{tt}$ holds vacuously; hence $\nu x.\overline{a}x \models^{a^i} \langle\overline{a}(x)\rangle[x = a]\langle\tau\rangle\mathtt{tt}$ holds as required.

3 Completeness of open bisimilarity with respect to $\mathcal{OM}$

There is a constructive definition of non-bisimilarity. Since bisimilarity is defined in terms of a greatest fixed point of relations satisfying a certain closure property, non-bisimilarity is defined in terms of a least fixed point satisfying the dual property. This leads to the following constructive definition of non-bisimilarity from which a non-bisimilarity algorithm can be extracted. Since non-bisimilarity is defined in terms of a least fixed point, there is a finite winning strategy, consisting of a finite tree of moves such that in each branch eventually a pair of processes and a history is reached such that one process can make a move that the other cannot always match.

▶ **Definition 7** (non-bisimilarity)**.** Firstly, we inductively define the family of relation $\nsim^h_n$, for $n \in \mathbb{N}$. The base case is when, for some respectful substitution one player can make a move, that cannot be matched by the other player without assuming a stronger substitution. The class of all such pairs of processes form the base case for the construction of the non-bisimilarity relation, say $P \nsim^h_0 Q$. More precisely, the relation $\nsim^h_0$ is the least symmetric relation such that for any P and Q, $P \nsim^h_0 Q$ whenever there exist process P', action π and substitution σ respecting h, such that the following holds.

- $P\sigma \xrightarrow{\pi\sigma} P'$, for $x \in \mathrm{bn}(\pi)$, x is fresh for $P\sigma$, $Q\sigma$ and $h\sigma$, and there is no Q' such that $Q\sigma \xrightarrow{\pi\sigma} Q'$.

Inductively, $\nsim^h_{n+1}$ is the least symmetric relation extending $\nsim^h_n$ such that $P \nsim^h_{n+1} Q$ whenever for some substitution σ respecting h, one of the following holds, where α is τ or $\overline{a}b$:

- $P\sigma \xrightarrow{\alpha\sigma} P'$ and for all Q_i such that $Q\sigma \xrightarrow{\alpha} Q_i$, $P' \nsim^{h\sigma}_n Q_i$.
- $P\sigma \xrightarrow{\overline{a\sigma}(x)} P'$, and for all Q_i and x fresh for $P\sigma$, $Q\sigma$ and $h\sigma$, such that $Q\sigma \xrightarrow{\overline{a\sigma}(x)} Q_i$, $P' \nsim^{h\sigma \cdot x^o}_n Q_i$.
- $P\sigma \xrightarrow{a\sigma(x)} P'$, and for all Q_i and x fresh for $P\sigma$, $Q\sigma$ and $h\sigma$, such that $Q\sigma \xrightarrow{a\sigma(x)} Q_i$, $P' \nsim^{h\sigma \cdot x^i}_n Q_i$.

Thereby, the relation $P \nsim^h_n Q$ contains all processes that can be distinguished by a strategy with depth at most n, i.e., at most n moves are required to reach a pair of processes in $\nsim^h_0$, at which point there is an accessible world in which a process can make a move that the other process cannot match.

The relation $\nsim^h$, pronounced non-bisimilarity with history h, is defined to be the least relation containing $\nsim^h_n$ for all $n \in \mathbb{N}$, i.e. $\bigcup_{n\in\mathbb{N}} \nsim^h_n$. Similarly to open bisimulation, $P \nsim Q$ is defined as $P \nsim^{x_1^i \cdots x_m^i} Q$ where $\mathrm{fv}(P) \cup \mathrm{fv}(Q) \subseteq \{x_1, \ldots x_m\}$.

3.1 Preliminaries

We require the following terminology for substitutions, and abbreviations for formulae.

▶ **Definition 8.** Composition of substitutions σ and θ is defined such that $P(\sigma \cdot \theta) = (P\sigma)\theta$, for all processes P. For substitutions σ and θ, $\sigma \leq \theta$ whenever there exists σ' such that $\sigma \cdot \sigma' = \theta$. For a finite substitution $\sigma = \{^{z_1}/_{x_1}\} \cdots \{^{z_n}/_{x_n}\}$ the formula $[\sigma]\phi$ abbreviates the formula $[x_n = z_n] \ldots [x_1 = z_1]\phi$. Similarly, $\langle\sigma\rangle\phi$ abbreviates $\langle x_n = z_n\rangle \ldots \langle x_1 = z_1\rangle\phi$. For finite set of formulae ϕ_i, formula $\bigvee_i \phi_i$ abbreviates $\phi_1 \vee \ldots \vee \phi_n$, where the empty disjunction is $\mathtt{ff}$. Similarly $\bigwedge_i \phi_i$ abbreviates $\phi_1 \wedge \ldots \wedge \phi_n$, where the empty conjunction is $\mathtt{tt}$.

We require the following technical lemmas. The first (image finiteness, as used in [5]) ensures that there are finitely many reachable states in one step, up to renaming. The second extends the definition of the box-match modality to finite substitutions. The third is required in inductive cases involving bound output and input. The fourth is a monotonicity property for satisfaction. The fifth is a monotonicity property for transitions ensuring names bound by label are not changed by a substitution.

▶ **Lemma 9.** *For process P and action π there are finitely many P_i such that $P \xrightarrow{\pi} P_i$.*

▶ **Lemma 10.** *If for all θ respecting h and $\sigma \leq \theta$, it holds that $P\theta \models^{h\theta} \phi\theta$, then $P \models^h [\sigma]\phi$ holds.*

▶ **Lemma 11.** *If $\sigma \cdot \theta$ respects h, then θ respects $h\sigma$.*

▶ **Lemma 12.** *If $P \models^h \phi$ holds then $P\theta \models^{h\theta} \phi\theta$ holds for any θ respecting h.*

▶ **Lemma 13.** *If $P \xrightarrow{\pi} Q$ then $P\theta \xrightarrow{\pi\theta} Q\theta$, for all θ such that if $x \in \mathrm{bn}(\pi)$ and $y\theta = x$ then $x = y$.*

3.2 Algorithm for distinguishing formulae

The constructive definition of non-bisimilarity gives a tree of substitutions and actions forming a strategy showing that two processes are not open bisimilar. The following proposition shows that $\mathcal{OM}$ formulae are sufficient to capture such strategies. For any strategy that distinguishes two processes, we can construct *distinguishing $\mathcal{OM}$* formulae. A distinguishing formula holds for one process but not for the other process. Furthermore, there are always at least two distinguishing formulae, one biased to the left and another biased to the right, as in the construction of the proof for the following proposition. As discussed in the introduction, the left bias cannot be simply obtained by negating the right bias and vice versa; both must be constructed simultaneously and may be unrelated by negation.

▸ Proposition 14. If $P \nsim Q$ then there exists ϕ_L such that $P \models \phi_L$ and $Q \not\models \phi_L$, and also there exists ϕ_R such that $Q \models \phi_R$ and $P \not\models \phi_R$.

Proof. Since $\nsim^h$ is defined by a least fixed point over a family of relations $\nsim^h_n$, if $P \nsim^h Q$, there exists n such that $P \nsim^h_n Q$, so we can proceed by induction on the depth of a winning strategy.

In the base case, assume that $P \nsim^h_0 Q$, hence by definition, for substitution σ respecting h, $P\sigma \xrightarrow{\pi\sigma} P'$, for $x \in \mathrm{bn}(\pi)$, x is fresh for $P\sigma$, $Q\sigma$ and $h\sigma$, such that there is no Q' such that $Q\sigma \xrightarrow{\pi\sigma} Q'$, up to symmetry of $\nsim^h_n$. There exist finitely many pairs of variables x_j and y_j, selected from $\mathrm{fv}(P) \cup \mathrm{fv}(Q) \cup \mathrm{fv}(\pi)$ such that $x_j\sigma \neq y_j\sigma$, and, for any R and substitution θ respecting h, if $Q\theta \xrightarrow{\pi\theta} R$ there exists j such that $x_j\theta = y_j\theta$. To see why, assume for contradiction that there is some θ respecting h such that $Q\theta \xrightarrow{\pi\theta} R$ but there is no x and y in $\mathrm{fv}(P) \cup \mathrm{fv}(Q) \cup \mathrm{fv}(\pi)$ such that $x\sigma \neq y\sigma$ and $x\theta = y\theta$. Stated otherwise, for all x and y in $\mathrm{fv}(P) \cup \mathrm{fv}(Q) \cup \mathrm{fv}(\pi)$ if $x\theta = y\theta$ then $x\sigma = y\sigma$, which is precisely the definition of a function, i.e. substitution, say θ', defined on $\mathrm{fv}(P\theta) \cup \mathrm{fv}(Q\theta) \cup \mathrm{fv}(\pi\theta)$ such that θ' maps $z\theta$ to $z\sigma$. In that case, $\theta \cdot \theta' = \sigma$ on $\mathrm{fv}(P) \cup \mathrm{fv}(Q) \cup \mathrm{fv}(\pi)$; and hence, by Lemma 13, since for $x \in \mathrm{bn}(\pi)$, x is fresh, $Q\theta\theta' \xrightarrow{\pi\theta\theta'} R\theta'$ contradicting the initial assumption for the base case that no transition $Q\sigma \xrightarrow{\pi\sigma} Q'$ exists for any Q'.

In this case, there are two distinguishing formulae $[\sigma]\langle\pi\rangle\mathtt{tt}$ and $[\pi]\bigvee_j\langle x_j = y_j\rangle\mathtt{tt}$ biased to P and Q respectively. There are four cases to check to confirm that these are distinguishing formulae.

Case $P \models^h [\sigma]\langle\pi\rangle\mathtt{tt}$: Consider all θ respecting h such that $\sigma \leq \theta$. By definition there exists θ' such that $\sigma \cdot \theta' = \theta$, so since $P\sigma \xrightarrow{\pi\sigma} P'$, by Lemma 13, $P\theta \xrightarrow{\pi\theta} P'\theta'$. Thereby, since $P'\theta' \models^{h'} \mathtt{tt}$ holds, $P\theta \models^{h\theta} \langle\pi\theta\rangle\mathtt{tt}$. Hence, by Lemma 10, $P \models^h [\sigma]\langle\pi\rangle\mathtt{tt}$.

Case $Q \not\models^h [\sigma]\langle\pi\rangle\mathtt{tt}$: Assume $Q \models^h [\sigma]\langle\pi\rangle\mathtt{tt}$ for contradiction. Now, since σ respects h and $\sigma \leq \sigma$, by Lemma 10, $Q \models^h [\sigma]\langle\pi\rangle\mathtt{tt}$ holds only if $Q\sigma \models^{h\sigma} \langle\pi\sigma\rangle\mathtt{tt}$ holds; which holds only if there exists Q' such that $Q\sigma \xrightarrow{\pi\sigma} Q'$, contradicting the assumption that no such Q' exists. Thereby $Q \not\models^h [\sigma]\langle\pi\rangle\mathtt{tt}$.

Case $Q \models^h [\pi]\bigvee_j\langle x_j = y_j\rangle\mathtt{tt}$: Consider substitutions θ respecting h and Q' such that $Q\theta \xrightarrow{\pi\theta} Q'$. It must be the case that there exists j such that $x_j\theta = y_j\theta$, thereby $Q' \models^{h\theta} \langle x_j\theta = y_j\theta\rangle\mathtt{tt}$ holds; hence clearly $Q' \models^{h\theta} \left(\bigvee_j\langle x_j = y_j\rangle\mathtt{tt}\right)\theta$ holds. Hence $Q \models^h [\pi]\bigvee_j\langle x_j = y_j\rangle\mathtt{tt}$.

Case $P \not\models^h [\pi]\bigvee_j\langle x_j = y_j\rangle\mathtt{tt}$: Assume for contradiction $P \models^h [\pi]\bigvee_j\langle x_j = y_j\rangle\mathtt{tt}$. This holds iff for all processes S and substitutions θ respecting h, $P\theta \xrightarrow{\pi\theta} S$ implies $S \models^{h'} \left(\bigvee_j\langle x_j = y_j\rangle\mathtt{tt}\right)\theta$. Since we know that σ respects h and $P\sigma \xrightarrow{\pi\sigma} P'$, we have $P' \models^{h''} \left(\bigvee_j\langle x_j = y_j\rangle\mathtt{tt}\right)\sigma$. This holds only if for some j, $P' \models^{h''} \langle x_j\sigma = y_j\sigma\rangle\mathtt{tt}$; hence, $x_j\sigma = y_j\sigma$ for some j, which contradicts the assumption that $x_j\sigma \neq y_j\sigma$. Thereby $P \not\models^h [\pi]\bigvee_j\langle x_j = y_j\rangle\mathtt{tt}$.

Now consider the inductive cases. Given P, Q, if $P \nsim^h_{n+1} Q$, up to symmetry of $\nsim^h_{n+1}$, there are three cases to consider, for some substitution σ respecting h, where α is either τ or $\overline{a}b$:

- $P\sigma \xrightarrow{\alpha\sigma} P'$ and for all Q_i such that $Q\sigma \xrightarrow{\alpha\sigma} Q_i$, $P' \nsim^{h\sigma}_n Q_i$.
- $P\sigma \xrightarrow{\overline{a\sigma}(x)} P'$, and for all Q_i and x fresh for $P\sigma$, $Q\sigma$ and $h\sigma$, such that $Q\sigma \xrightarrow{\overline{a\sigma}(x)} Q_i$, $P' \nsim^{h\sigma\cdot x^o}_n Q_i$.
- $P\sigma \xrightarrow{a\sigma(x)} P'$, and for all Q_i and x fresh for $P\sigma$, $Q\sigma$ and $h\sigma$, such that $Q\sigma \xrightarrow{a\sigma(x)} Q_i$, $P' \nsim^{h\sigma\cdot x^i}_n Q_i$.

We consider the second case above involving bound output only, the other two cases are similar — differing only in the accounting for respectful substitutions according to Def. 1.

For $P\sigma \xrightarrow{\overline{a\sigma}(x)} P'$, by Lemma 9, there exist finitely many Q_i such that $Q\sigma \xrightarrow{\overline{a\sigma}(x)} Q_i$. For each i, since $P' \nsim^{h\sigma\cdot x^o}_n Q_i$, by the induction hypothesis, there exist ϕ^L_i and ϕ^R_i such that $P' \models^{h\sigma\cdot x^o} \phi^L_i\sigma$ and $Q_i \not\models^{h\sigma\cdot x^o} \phi^L_i\sigma$ and $P' \not\models^{h\sigma\cdot x^o} \phi^R_i\sigma$ and $Q_i \models^{h\sigma\cdot x^o} \phi^R_i\sigma$. Furthermore, assume that σ is minimal with respect to the order over substitutions in the sense that, if $\theta \leq \sigma$ is such that $P\theta \xrightarrow{\overline{a\theta}(x)} P''$, where x is fresh for $P\theta$, $Q\theta$ and $h\theta$, and for all Q'' such that $Q\theta \xrightarrow{\overline{a\theta}(x)} Q''$, $P'' \nsim^{h\theta\cdot x^o}_n Q''$, then $\theta = \sigma$.

By a similar argument to the base case, there are finitely many pairs of variables x_j and y_j selected from $\mathrm{fv}(P) \cup \mathrm{fv}(Q) \cup \{a\}$ such that $x_j\sigma \neq y_j\sigma$ and, for any substitution θ respecting h, if, for some

S, $Q\theta \xrightarrow{\overline{a\theta}(x)} S$ then either: there exists some j such that $x_j\theta = y_j\theta$; or both $\sigma \leq \theta$ and $\theta \leq \sigma$ hold and hence there exist i and θ' such that $\sigma \cdot \theta' = \theta$ and $S_i\theta' = S$. Notice cases where $\theta < \sigma$ are eliminated by minimality of σ.

From the above, distinguishing formulae $[\sigma]\langle\overline{a}(x)\rangle\bigwedge_i \phi_i^L$ and $[\overline{a}(x)](\bigvee_i \phi_i^R \vee \bigvee_j\langle x_j = y_j\rangle)$ can be constructed. There are four cases to consider to verify these are indeed distinguishing formulae.

Case $P \vDash^h [\sigma]\langle\overline{a}(x)\rangle\bigwedge_i \phi_i^L$: Consider all θ such that $\sigma \leq \theta$, θ respects h, and without loss of generality x is fresh such that $x \notin \mathrm{dom}(\theta)$ and $x \notin \mathrm{fv}(h\theta)$. By definition, there exists θ' such that $\sigma \cdot \theta' = \theta$. Now since $\sigma \cdot \theta'$ respects h, by Lemma 11, θ' respects $h\sigma$ hence since $x \notin \mathrm{dom}(\theta')$ and $x \notin \mathrm{fv}(h\sigma\theta')$, θ' respects $h\sigma \cdot x^o$. Thereby since θ' respects $h\sigma \cdot x^o$ and also $P' \vDash^{h\sigma\cdot x^o} \phi_i^L\sigma$ holds, by Lemma 12, it holds that $P'\theta' \vDash^{h\theta\cdot x^o} \phi_i^L\theta$. The above holds for all i, hence it holds that $P'\theta' \vDash^{h\theta\cdot x^o} \bigwedge_i \phi_i^L\theta$. Now, since $P\sigma \xrightarrow{\overline{a\sigma}(x)} P'$, by Lemma 13, since x is fresh, $P\theta \xrightarrow{\overline{a\theta}(x)} P'\theta'$ holds; and hence $P\theta \vDash^{h\theta} \left(\langle\overline{a}(x)\rangle\bigwedge_i \phi_i^L\right)\theta$ holds. Thereby, by Lemma 10, $P \vDash^h [\sigma]\langle\overline{a}(x)\rangle\bigwedge_i \phi_i^L$ holds.

Case $Q \nvDash^h [\sigma]\langle\overline{a}(x)\rangle\bigwedge_i \phi_i^L$: Assume for contradiction that $Q \vDash^h [\sigma]\langle\overline{a}(x)\rangle\bigwedge_i \phi_i^L$. Since σ respects h and $\sigma \leq \sigma$, by Lemma 10, the above assumption holds only if $Q\sigma \vDash^{h\sigma} \left(\langle\overline{a}(x)\rangle\bigwedge_i \phi_i^L\right)\sigma$ holds. Now $Q\sigma \vDash^{h\sigma} \langle\overline{a\sigma}(x)\rangle\bigwedge_i \phi_i^L\sigma$ holds only if there exists Q' such that $Q\sigma \xrightarrow{\overline{a\sigma}(x)} Q'$ and $Q' \vDash^{h\sigma\cdot x^o} \bigwedge_i \phi_i^L\sigma$, which holds only if $Q' \vDash^{h\sigma\cdot x^o} \phi_i^L\sigma$ for all i. Notice that $Q' = Q_k$ for some k, and therefore $Q_k \vDash^{h\sigma\cdot x^o} \phi_k^L\sigma$; but it was assumed that $Q_k \nvDash^{h\sigma\cdot x^o} \phi_k^L\sigma$ leading to a contradiction. Therefore $Q \nvDash^h [\sigma]\langle\overline{a}(x)\rangle\bigwedge_i \phi_i^L$.

Case $Q \vDash^h [\overline{a}(x)](\bigvee_i \phi_i^R \vee \bigvee_j\langle x_j = y_j\rangle\mathtt{tt})$: Fix Q' and θ respecting h such that $Q\theta \xrightarrow{\overline{a\theta}(x)} Q'$ and without loss of generality assume x is fresh such that $x \notin \mathrm{dom}(\theta)$ and $x \notin \mathrm{fv}(h\theta)$. There are two sub-cases to consider. Firstly consider where for some k, $x_k\theta = y_k\theta$, in which case it holds that $Q' \vDash^{h\theta\cdot x^o} \langle x_k\theta = y_k\theta\rangle\mathtt{tt}$, and hence $Q' \vDash^{h\theta\cdot x^o} (\bigvee_i \phi_i^R \vee \bigvee_j\langle x_j = y_j\rangle\mathtt{tt})\theta$, by definition of disjunction. Secondly consider where there exists θ' such that $\sigma \cdot \theta' = \theta$ and for some ℓ, $Q\sigma \xrightarrow{\overline{a\sigma}(x)} Q_\ell$ such that $Q_\ell\theta' = Q'$. Now since $\sigma \cdot \theta'$ respects h, by Lemma 11, θ' respects $h\sigma$, hence by definition of respectful substitutions, since $x \notin \mathrm{dom}(\theta)$ and $x \notin \mathrm{fv}(h\theta)$, θ' respects $h\sigma \cdot x^o$. Thereby, by Lemma 12, since $Q_\ell \vDash^{h\sigma\cdot x^o} \phi_\ell^R\sigma$ and θ' respects $h\sigma \cdot x^o$, $Q_\ell\theta' \vDash^{h\theta\cdot x^o} \phi_\ell^R\theta$ holds. Hence $Q' \vDash^{h\theta\cdot x^o} (\bigvee_i \phi_i^R \vee \bigvee_j\langle x_j = y_j\rangle\mathtt{tt})\theta$, by definition of disjunction. Thus by definition of $[\overline{a}(x)]$, we can conclude that $Q \vDash^h [\overline{a}(x)](\bigvee_i \phi_i^R \vee \bigvee_j\langle x_j = y_j\rangle\mathtt{tt})$ holds.

Case $P \nvDash^h [\overline{a}(x)](\bigvee_i \phi_i^R \vee \bigvee_j\langle x_j = y_j\rangle\mathtt{tt})$: Assume $P \vDash^h [\overline{a}(x)](\bigvee_i \phi_i^R \vee \bigvee_j\langle x_j = y_j\rangle\mathtt{tt})$ for contradition. Since σ respects h and $P\sigma \xrightarrow{\overline{a\sigma}(x)} P'$, the previous assumption can hold only if $P' \vDash^{h\sigma\cdot x^o} (\bigvee_i \phi_i^R \vee \bigvee_j\langle x_j = y_j\rangle\mathtt{tt})\sigma$. This holds only if, for some i, $P' \vDash^{h\sigma\cdot x^o} \phi_i^R\sigma$, or, for some j, $P' \vDash^{h\sigma\cdot x^o} \langle x_j\sigma = y_j\sigma\rangle\mathtt{tt}$. However, for all i, $P' \nvDash^{h\sigma\cdot x^o} \phi_i^R\sigma$; and also, for all j, we have $x_j\sigma \neq y_j\sigma$ and $P' \nvDash^{h\sigma\cdot x^o} \langle x_j\sigma = y_j\sigma\rangle\mathtt{tt}$, leading to a contradiction in either case. Thereby $P \nvDash^h [\overline{a}(x)](\bigvee_i \phi_i^R \vee \bigvee_j\langle x_j = y_j\rangle\mathtt{tt})$.

By induction we have established that, for any history h, processes P and Q, and any n, if $P \nsim_n^h Q$ then there exists ϕ_L such that $P \vDash^h \phi_L$ and $Q \nvDash^h \phi_L$, and also there exists ϕ_R such that $Q \vDash^h \phi_R$ and $P \nvDash^h \phi_R$. The result then follows by observing that $\nsim^h$ is the least relation containing all $\nsim_n^h$; and, furthermore, $P \nsim Q$ holds simply when $P \nsim^{x_1^i\cdots x_n^i} Q$ holds, where $\mathrm{fv}(P) \cup \mathrm{fv}(Q) \subseteq \{x_1^i, \ldots, x_n^i\}$. ◂

Since open bisimilarity is decidable for finite π-calculus processes, the constructive non-bisimilarity in Definition 7 coincides with the negation of open bisimilarity.

▸ **Lemma 15.** *For finite processes, $P \nsim Q$ holds, according to constructive non-bisimilarity in Definition 7, if and only if $P \sim Q$ does not hold.*

Combining Proposition 14 with Lemma 15 yields immediately the completeness of $\mathcal{OM}$ with respect to open bisimilarity. Completeness (Theorem 6) establishes that the set of all pairs of processes that have the same set of distinguishing formulae is an open bisimilarity. The proof can now be stated as follows.

Proof of Theorem 6: Assume that for finite processes P and Q, for all formulae ϕ, $P \models \phi$ iff $Q \models \phi$. Now for contradiction suppose that $P \sim Q$ does not hold. By Lemma 15, $P \nsim Q$ must hold. Hence by Proposition 14 there exists ϕ_L such that $P \models \phi_L$ but $Q \not\models \phi_L$, but by the assumption above $Q \models \phi_L$, leading to a contradiction. Thereby $P \sim Q$. ◀

Notice that soundness (Theorem 5) and the non-bisimilarity algorithm (Proposition 14) also hold for infinite π-calculus processes (using replication for instance). However, for infinite π-calculus processes, open bisimilarity is undecidable; hence additional insight may be needed to justify whether Lemma 15 holds for infinite processes. Thereby in the infinite case, while it is impossible that $P \sim Q$ and $P \nsim Q$ holds, it may be the case that neither holds. A possibility is that a more expressive logic is required to completely characterise open bisimilarity for infinite processes.

3.3 Example runs of distinguishing formulae algorithm

We provide further examples of non-bisimilar processes that illustrate subtle aspects of the algorithm. In particular, these examples illustrate the need for disjunctions of postconditions in both the base case and inductive steps.

3.3.1 Multiple postconditions and postconditions in an inductive step

The following example leads to multiple postconditions. Consider the following non-bisimilar processes: $[x = y]\tau + [w = z]\tau \nsim \tau$. Observe that clearly $\tau \xrightarrow{\tau} 0$ but $([x = y]\tau + [w = z]\tau)\theta \xrightarrow{\tau}$ only if $x\theta = y\theta$ or $w\theta = z\theta$. Thus, $[x = y]\tau + [w = z]\tau \models [\tau](\langle x = y\rangle\mathtt{tt} \vee \langle w = z\rangle\mathtt{tt})$ is a distinguishing formula biased to the left process, while $\tau \models \langle\tau\rangle\mathtt{tt}$ is biased to the right.

Now consider an example where postconditions are required in the inductive case. Firstly observe that $\overline{a}a + \overline{b}b \nsim \overline{a}a$ are distinguished since $\overline{a}a + \overline{b}b \xrightarrow{\overline{b}b} 0$, but process $\overline{a}a$ can only make a $\overline{b}b$ transition under a substitution such that $a = b$. Hence we have the distinguishing formulae $\overline{a}a + \overline{b}b \models \langle\overline{b}b\rangle\mathtt{tt}$ and $\overline{a}a \models [\overline{b}b]\langle a = b\rangle\mathtt{tt}$.

For the inductive case, consider $P \triangleq \tau.(\overline{a}a + \overline{b}b) + [x = y]\tau.\overline{a}a \nsim \tau.(\overline{a}a + \overline{b}b) + \tau.\overline{a}a \triangleq Q$. Let us lead by $Q \xrightarrow{\tau} \overline{a}a$, which can only be matched by $P \xrightarrow{\tau} \overline{a}a + \overline{b}b$. By the above observation, we have distinguishing formulae for $\overline{a}a + \overline{b}b \nsim \overline{a}a$. Furthermore, $P\theta \xrightarrow{\tau}$ for substitutions θ such that $x\theta = y\theta$.

This leads to the following distinguishing formula for the left side, consisting of a box τ followed by a disjunction of the left distinguishing formula for $\overline{a}a + \overline{b}b \nsim \overline{a}a$, and the postcondition for any additional τ transitions. $\tau.(\overline{a}a + \overline{b}b) + [x = y]\tau.\overline{a}a \models [\tau](\langle\overline{b}b\rangle\mathtt{tt} \vee \langle x = y\rangle\mathtt{tt})$.

The distinguishing formula for the right process is diamond τ followed by the right distinguishing formula for $\overline{a}a + \overline{b}b \nsim \overline{a}a$, as follows: $\tau.(\overline{a}a + \overline{b}b) + \tau.\overline{a}a \models \langle\tau\rangle[\overline{b}b]\langle a = b\rangle\mathtt{tt}$.

As a further example involving post conditions generated by the inductive case, consider the following two non-bisimilar processes: $[x = y]\tau.\tau + \tau \nsim \tau.\tau + \tau$. We can devise the following distinguishing formula biased to the left process: $[x = y]\tau.\tau + \tau \models [\tau][\tau]\langle x = y\rangle\mathtt{tt}$. However, this is different from the left-biased formula generated by the algorithm: $[x = y]\tau.\tau + \tau \models [\tau]([\tau]\mathtt{ff} \vee \langle x = y\rangle\mathtt{tt})$. Thus, there may exist alternative distinguishing formulae other than those generated by the algorithm in the completeness proof.

3.3.2 Formulae generated by substitutions applied to labels

In some cases substitutions applied to labels play a role when generating distinguishing formulae. For a minimal example consider the following non-bisimilar processes: $\overline{a}a \nsim \overline{a}b$. A distinguishing strategy is where process $\overline{a}b$ makes a $\overline{a}b$ transition, which cannot be matched by $\overline{a}a$. However, $(\overline{a}a)\sigma \xrightarrow{(\overline{a}b)\sigma} 0$ for any substitution such that $a\sigma = b\sigma$, leading to distinguishing formula $[\overline{a}b]\langle a = b\rangle\mathtt{tt}$ biased to $\overline{a}a$. Notice substitution σ is applied to both the process and the label.

For a trickier example consider the following: $\nu b.\overline{a}b.a(x).[x = b]\overline{x}x \nsim \nu b.\overline{a}b.a(x).\overline{x}x$. After two actions, the problem reduces to base case $[x = b]\overline{x}x \nsim^{a^i \cdot b^o \cdot x^i} \overline{x}x$, where $\overline{x}x$ can perform a $\overline{x}x$ action, but $[x = b]\overline{x}x$ cannot. However, $([x = b]\overline{x}x)\{^b\!/_x\} \xrightarrow{\overline{x}x\{^b\!/_x\}} 0$ does hold, and furthermore $\{^b\!/_x\}$ respects $a^i \cdot b^o \cdot x^i$. From these observations we can construct a distinguishing formula biased to the left as follows: $\nu b.\overline{a}b.a(x).[x = b]\overline{x}x \vDash [a(b)][a(x)][\overline{x}x]\langle x = b\rangle\mathtt{tt}$.

3.3.3 An elaborate example demanding intuitionistic assumptions

For a more elaborate example consider the following.

$$\tau.(\underbrace{\tau + \tau.\tau + \tau.[x = y][w = z]\tau}_{P'}) \triangleq P \quad \nsim \quad Q \triangleq \tau.(\underbrace{\tau + \tau.\tau + \tau.[x = y]\tau}_{Q'}) + P$$

A non-bisimilarity strategy is as follows: firstly, lead by transition $Q \xrightarrow{\tau} Q'$ on the right, matched by transition $P \xrightarrow{\tau} P'$; secondly, lead by $P' \xrightarrow{\tau} [x = y][w = z]\tau$ on the left, matched in three possible ways by $Q' \xrightarrow{\tau} 0$, $Q' \xrightarrow{\tau} \tau$ and $Q' \xrightarrow{\tau} [x = y]\tau$. To distinguish 0 from $[x = y][w = z]\tau$ observe that $([x = y][w = z]\tau)\{^y\!/_x\}\{^z\!/_w\} \xrightarrow{\tau} 0$ but 0 can make no τ transition; hence distinguishing formulae for 0 and $[x = y][w = z]\tau$ are $0 \vDash [\tau]\mathtt{ff}$ and $[x = y][w = z]\tau \vDash [x = y][w = z]\langle\tau\rangle\mathtt{tt}$. To distinguish $[x = y]\tau$ from $[x = y][w = z]\tau$, observe that $([x = y]\tau)\{^y\!/_x\} \xrightarrow{\tau} 0$, but $([x = y][w = z]\tau)\{^y\!/_x\}$ can only make a τ transition under a substitution such that also $w = z$; hence $[x = y]\tau \vDash [x = y]\langle\tau\rangle\mathtt{tt}$ and $[x = y][w = z]\tau \vDash [\tau]\langle w = z\rangle\mathtt{tt}$ are distinguishing formulae. The same formulae also distinguish τ from $[x = y][w = z]\tau$. Thereby the algorithm in the completeness proof generates the following:

$$P \vDash [\tau]\langle\tau\rangle([\tau]\langle w = z\rangle\mathtt{tt} \wedge [x = y][w = z]\langle\tau\rangle\mathtt{tt}) \qquad Q \vDash \langle\tau\rangle[\tau]([\tau]\mathtt{ff} \vee [x = y]\langle\tau\rangle\mathtt{tt})$$

The strategy explained above is not unique. An alternative strategy can generate different distinguishing formulae: $P \vDash [\tau][\tau](\langle\tau\rangle\mathtt{tt} \vee [\tau]\langle w = z\rangle\mathtt{tt})$ and $Q \vDash \langle\tau\rangle\langle\tau\rangle([x = y]\langle\tau\rangle\mathtt{tt} \wedge [\tau]\langle x = y\rangle\mathtt{tt})$. Note if we assume the law of excluded middle, both processes above become equivalent to $\tau.(\tau+\tau.\tau)$. Fortunately, we do not assume the law of excluded middle.

4 Related work

We consider the relationship between the intuitionistic modal logic for open bisimilarity presented in this work and established classical logics. We also compare this work to existing work claiming to characterise open bisimilarity for the π-calculus.

4.1 Comparison to classical logics for late bisimilarity

The late Milner-Parrow-Walker logic, called $\mathcal{LM}$ [9] for "($\mathcal{L}$) late modality with ($\mathcal{M}$) match" differs from the logic presented in this paper in three significant ways: firstly, free names are a priori assumed to be distinct; secondly, $\mathcal{LM}$ is classical, that is, the law of excluded middle for name equalities is assumed; and thirdly the late input box modality is defined differently as follows — involving an existential quantification over substitutions:

- $P \vDash^L [a(x)]^L\phi$ iff for all Q such that $P \xrightarrow{a(x)} Q$ there exists name z such that $Q\{^z\!/_x\} \vDash^L \phi\{^z\!/_x\}$.

To see that logical equivalence for $\mathcal{LM}$ does not define a congruence, consider the processes $[x = y]\overline{x}x$ and 0. These processes satisfy the same set of late formulae (any formula equivalent to $\mathtt{tt}$), since, for $\mathcal{LM}$, x and y are a priori assumed to be distinct names. However, $a(y).[x = y]\overline{x}x$ and

$a(y).0$ have distinguishing formulae $a(y).[x = y]\overline{x}x \models^L [a(y)]^L\langle\overline{x}x\rangle\mathtt{tt}$ biased to the left and its de Morgan complement $a(y).0 \models^L \langle a(y)\rangle[\overline{x}x]\mathtt{ff}$ biased to the right.

Between open bisimilarity and late bisimilarity there is late congruence, which is the greatest congruence relation contained in late bisimilarity. Late congruence must contain open bisimilarity, since open bisimilarity is contained in late bisimilarity and open bisimilarity is a congruence. Late congruence also has a simpler characterisation: P and Q are late congruent whenever for all substitutions σ, and $P\sigma$ is late bisimilar to $Q\sigma$. The quantification over all substitutions, combined with the law of excluded middle, has the effect that we check late bisimilarity with respect to all combinations of equalities and inequalities between free names.

As for open bisimilarity, $[x = y]\overline{x}x$ and 0 are not late congruent. This is because for substitution $\{^x\!/\!_y\}$, $([x = y]\overline{x}x)\{^x\!/\!_y\}$ and $0\{^x\!/\!_y\}$ are clearly not late bisimilar. This illustrates that late congruence is strictly finer than late bisimilarity. However, open bisimilarity is still strictly finer than late congruence, since $\tau + \tau.\tau + \tau.[x = y]\tau$ and $\tau + \tau.\tau$ are late congruent. Late congruence holds since, for any substitution θ, $(\tau + \tau.\tau)\theta$ and $(\tau + \tau.\tau + \tau.[x = y]\tau)\theta$ are late bisimilar. In contrast, we know these processes are not open bisimilar; and furthermore, have distinguishing formula that rely on the absence of the law of excluded middle.

4.2 Other embeddings into intuitionistic nominal logic

Tiu and Miller [16] studied embeddings of the π-calculus into the logic LINC, as well as late and open bisimilarity and their respectful modal logics. This is the most closely related work since our encodings in Abella were adapted from their work. In their encoding, both late and open bisimilarity are encoded by essentially the same modalities, differing only in the the law of the excluded middle for names and the quantification of free variables. However, no examples of distinguishing formulae for open bisimilarity were provided; and, critically, the proof made flawed assumptions about the existence of a syntactic negation of a formula, which we observe in this work is not permitted.

A problem with the approach of Tiu and Miller is the reuse of the input box modality from $\mathcal{LM}$, which involves an existential quantification over substitutions. In contrast, our input box modality in $\mathcal{OM}$ involves universal quantification over all respectful substitutions. Our choice in $\mathcal{OM}$ is critical for generating distinguishing formulae. For example, the following processes are not open bisimilar: $a(x).\tau + a(x) + a(x).[x = a]\tau \nsim a(x).\tau + a(x)$.

For the above processes, the algorithm for distinguishing formulae in Proposition 14, correctly generates the following $\mathcal{OM}$ formula biased to the right: $a(x).\tau + a(x) \models [a(x)](\langle\tau\rangle\mathtt{tt} \vee [\tau]\mathtt{ff})$.

However, using only late modalities, as in Tiu and Miller, there is no distinguishing formula for these processes biased to the right: e.g., the formula with a late modality $[a(x)]^L(\langle\tau\rangle\mathtt{tt} \vee [\tau]\mathtt{ff})$ succeeds for both processes, even when rejecting the law of excluded middle; also the formula $[a(x)]^L(\langle x = a\rangle[\tau]\mathtt{ff} \vee \langle\tau\rangle[x = a]\mathtt{ff})$ fails for both processes, despite being distinguishing in classical $\mathcal{LM}$. The choice of modalities we make in $\mathcal{OM}$ make sense, since in open bisimilarity the choice of substitution is deferred as late as possible — possibly several transitions later.

4.3 A generic formalisation using nominal logic

Recently, Parrow et al. [12] provided a general proof of the soundness and completeness of logical equivalence for various modal logics with respect to corresponding bisimulations. The proof is parametric on properties of substitutions, which can be instantiated for a range of bisimulations. Moreover, their proof is mechanised using Nominal Isabelle. The conference version [12], sketches how to instantiate the abstract framework for open bisimilarity in the π-calculus without input prefixes only. However, we understand from communication with the authors that open bisimilarity for the π-calculus with input prefixes will be covered in a forthcoming extended version.

Stylistically, our intuitionistic modal logic is quite different from an instantiation of the abstract framework of Parrow et al. for open bisimilarity. Their framework, is classical and works by syntactically restricting "effect" modalities in formulae, depending on the type of bisimulation. Their effects represent substitutions that reach worlds permitted by the type of bisimulation. In contrast, the modalities of the intuitionistic modal logic $\mathcal{OM}$ in this paper are syntactically closer to long established modalities for the π-calculus [9]; differing instead in their semantic interpretation and in the absence of classical negation. An explanation for the stylistic differences is that for every intuitionistic logic, such as the intuitionistic modal logic in this work, there should be a corresponding classical modal logic based on an underlying Kripke semantics. Such a Kripke semantics would reflect the accessible worlds, as achieved by the syntactically restricted effect modalities in the abstract classical framework instantiated for open bisimilarity.

5 Conclusion

The main result of this paper is a sound and complete logical characterisation of open bisimilarity for the π-calculus. To achieve this result, we introduce modal logic $\mathcal{OM}$, defined in Fig. 2. The soundness of $\mathcal{OM}$ with respect to open bisimilarity, Theorem 5, is mechanically proven in Abella. The details of the completeness, Theorem 6, are provided in Section 3.

There are several novel features of $\mathcal{OM}$ compared to established modal logics for π-calculus, such as $\mathcal{LM}$ characterising late bisimilarity. Firstly, as demonstrated in Examples 2.3.1 and 3.3.3, the absence of the law of excluded middle is essential for the existence of distinguishing formulae in $\mathcal{OM}$ for certain processes that are not open bisimilar (but are late congruent). The absence of the law of excluded middle is an intuitionistic assumption; and, as explained in the introduction, $\mathcal{OM}$ can indeed be considered to be a conservative extension of intuitionistic logic. Furthermore, in contrast to classical modal logics such as $\mathcal{LM}$, diamond and box modalities have independent interpretations, not dual to each other. These properties are expected under criterion set out for intuitionistic modal logics [14]. The absence of de Morgan dualities over modalities complicates the construction of distinguishing formulae.

The completeness proof involves an algorithm, Proposition 14, that constructs distinguishing formulae for non-bisimilar processes. To use this algorithm, firstly attempt to prove that two processes are open bisimilar. If they are non-bisimilar, after a finite number of steps, a distinguishing strategy, according to Def. 7, will be discovered. The strategy can then be used to inductively construct two distinguishing formulae, biased to each process. A key feature of the construction is that there are restricted versions of absolute truth by preconditions ($[\sigma]\langle\pi\rangle\mathtt{tt}$ restricted from $\langle\pi\rangle\mathtt{tt}$) and, dually, there are relaxed versions of absolute falsity by postconditions ($[\pi]\langle\sigma\rangle\mathtt{tt}$ relaxed from $[\pi]\mathtt{ff}$), as demonstrated in Examples 2.2.2 and 3.3.1.

Our logic $\mathcal{OM}$ is suitable for formal and automated reasoning; in particular, it has natural encodings in Abella for mechanised reasoning, used to establish Theorem 5, and Bedwyr [3] for automatic proof search. All bisimulations and satisfactions in examples have been automatically checked in Bedwyr and are available online: https://github.com/kyagrd/NonBisim2DF. In addition, our distinguishing formulae generation algorithm is implemented in Haskell [1].

Future work includes justifying whether or not $\mathcal{OM}$ is complete for infinite processes with replication or recursion, as discussed around Lemma 15. A related problem is to extend $\mathcal{OM}$ with fixed points, as in the μ-calculus [7]. Such an extension could lead to intuitionistic model checkers invariant under open bisimulation, where the call-by-need approach to inputs is related to symbolic execution. We are also interested in extensions of $\mathcal{OM}$ for open bisimulation in the spi-calculus [15].

Acknowledgments

We are grateful to Sam Staton for providing an example that helped us in the completeness proof. The authors receive support from MOE Tier 2 grant MOE2014-T2-2-076. The third author receives support from NTU Start Up grant M4081190.020.

References

1 Ki Yung Ahn, Ross Horne, and Alwen Tiu. Generating witness of non-bisimilarity for the pi-calculus. *CoRR*, abs/1705.10908, 2017. URL: http://arxiv.org/abs/1705.10908.

2 David Baelde, Kaustuv Chaudhuri, Andrew Gacek, Dale Miller, Gopalan Nadathur, Alwen Tiu, and Yuting Wang. Abella: A system for reasoning about relational specifications. *Journal of Formalized Reasoning*, 7(2):1–89, 2014. doi:10.6092/issn.1972-5787/4650.

3 David Baelde, Andrew Gacek, Dale Miller, Gopalan Nadathur, and Alwen Tiu. *The Bedwyr System for Model Checking over Syntactic Expressions*, pages 391–397. Springer Berlin Heidelberg, 2007. doi:10.1007/978-3-540-73595-3_28.

4 Andrew Gacek, Dale Miller, and Gopalan Nadathur. A two-level logic approach to reasoning about computations. *Journal of Automated Reasoning*, 49(2):241–273, 2012. doi:10.1007/s10817-011-9218-1.

5 Maciej Gazda and Wan Fokkink. Modal logic and the approximation induction principle. *Mathematical Structures in Computer Science*, 22(2):175–201, 2012. doi:10.1017/S0960129511000387.

6 Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985. doi:10.1145/2455.2460.

7 Dexter Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27(3):333–354, 1983. doi:10.1016/0304-3975(82)90125-6.

8 Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, Parts I and II. *Information and Computation*, 100(1):1–77, 1992. doi:10.1016/0890-5401(92)90008-4.

9 Robin Milner, Joachim Parrow, and David Walker. Modal logics for mobile processes. *Theoretical Computer Science*, 114(1):149–171, 1993. doi:10.1016/0304-3975(93)90156-N.

10 Ugo Montanari and Vladimiro Sassone. Dynamic congruence vs. progressing bisimulation for CCS. *Fundamenta informaticae*, 16(2), 1992.

11 Gopalan Nadathur and Dale Miller. An Overview of λProlog. In *Fifth International Logic Programming Conference*. MIT Press, 1988.

12 Joachim Parrow, Johannes Borgström, Lars-Henrik Eriksson, Ramunas Gutkovas, and Tjark Weber. Modal logics for nominal transition systems. In *CONCUR 2015*, volume 42 of *LIPIcs*, pages 198–211, 2015. doi:10.4230/LIPIcs.CONCUR.2015.198.

13 Davide Sangiorgi. A theory of bisimulation for the π-calculus. *Acta Informatica*, 33(1):69–97, 1996. doi:10.1007/s002360050036.

14 Alex K. Simpson. *The proof theory and semantics of intuitionistic modal logic*. PhD thesis, University of Edinburgh, UK, 1994.

15 Alwen Tiu and Jeremy Dawson. Automating open bisimulation checking for the spi calculus. In *Computer Security Foundations Symposium (CSF), 2010 23rd IEEE*, pages 307–321. IEEE, 2010. doi:10.1109/CSF.2010.28.

16 Alwen Tiu and Dale Miller. Proof search specifications of bisimulation and modal logics for the π-calculus. *ACM Transactions on Computational Logic*, 11(2):13:1–13:35, 2010. doi:10.1145/1656242.1656248.

APPENDIX

A Mechanized Proofs in Abella

We developed two versions of the proofs: first with no free variables and second with two free variables. The first helps to grasp the overall structure of our proofs and the second demonstrates that our proofs will generalize for any number of free variables. In general, there are countably many instances of the following theorems to consider, where $\bar{x}$ is the list of n free variables in processes P and Q:

$$\texttt{Theorem bisim\_sat}_n\texttt{:}\forall P\ Q\texttt{,}[\![P(\bar{x}) \sim Q(\bar{x})]\!] \rightarrow [\![P(\bar{x}) \stackrel{\models}{\sim} Q(\bar{x})]\!]\texttt{.}$$
$$\texttt{Theorem sat\_bisim}_n\texttt{:}\forall P\ Q\texttt{,}[\![P(\bar{x}) \stackrel{\models}{\sim} Q(\bar{x})]\!] \rightarrow [\![P(\bar{x}) \sim Q(\bar{x})]\!]\texttt{.}$$

These are the soundness and completeness theorems (Theorems 5 and 6) in Section 2 transcribed in Abella. We can develop fully mechanized proofs for $\texttt{bisim\_sat}_n$ (Section A.1) and partially mechanized proofs for $\texttt{sat\_bisim}_n$ (Section A.2). The missing part of the proof for $\texttt{sat\_bisim}_n$ corresponds to Proposition 14 in Section 3.

We use the notation $[\![\cdot]\!]$ for the encoding of the concepts defined in Sect. 2 into Abella's logic. More specifically,

$$\begin{aligned}
[\![P(\bar{x}) \sim Q(\bar{x})]\!] &\triangleq \forall \bar{x}\texttt{, bisim } ([\![P]\!]\bar{x})\ ([\![Q]\!]\bar{x}) \\
[\![P(\bar{x}) \stackrel{\models}{\sim} Q(\bar{x})]\!] &\triangleq \forall F\texttt{, } [\![P(\bar{x}) \models F(\bar{x}) \iff Q(\bar{x}) \models F(\bar{x})]\!] \\
&\triangleq \forall F\texttt{, } [\![P(\bar{x}) \models F(\bar{x})]\!] \rightarrow [\![Q(\bar{x}) \models F(\bar{x})]\!] \\
&\qquad \wedge\ [\![Q(\bar{x}) \models F(\bar{x})]\!] \rightarrow [\![P(\bar{x}) \models F(\bar{x})]\!] \\
[\![P(\bar{x}) \models F(\bar{x})]\!] &\triangleq \forall \bar{x}\texttt{, sat } ([\![P]\!]\bar{x})\ ([\![F]\!]\bar{x})
\end{aligned}$$

where $\bar{x}$ are the free variables of P, Q, and F and the definitions for `bisim` and `sat` are provided in Fig. 4. The free variables $\bar{x}$ are encoded as universally quantified variables and applied to as arguments for higher-order syntax encodings of P, Q, and F. Both the bisimulation and satisfaction defined in Aballa's reasoning logic refer to the λProlog [11] specification of the finite π-calculus (Fig. 3), which are surrounded by curly braces (e.g., $\texttt{\{one P A P}_1\texttt{\}}$). Abella supports two-level logic approach [4] of having a computational specification in λProlog as an object logic and using Abella's reasoning logic to describe and reason about the properties of the specification.

A.1 Bisimulation implies Logical Equivalence

We prove `bisim_sat0` by splitting it into two lemmas:

$$\texttt{bisim\_sat0\_L :} \forall P\,Q\,F\texttt{,} [\![P \sim Q]\!] \rightarrow [\![P \models F]\!] \rightarrow [\![Q \models F]\!]$$

$$\texttt{bisim\_sat0\_R :} \forall P\,Q\,F\texttt{,} [\![P \sim Q]\!] \rightarrow [\![Q \models F]\!] \rightarrow [\![P \models F]\!]$$

Once we prove one of them, the other is proved simply by instantiating P with Q and Q with P. We prove `bisim_sat0_L` by induction on the second argument $[\![P \models F]\!]$.

Then, it is easy to prove $\texttt{bisim\_sat}_n$ for any n because the proofs for the lemmas $\texttt{bisim\_sat}_n\texttt{\_L}$ and `_R` are straightforward corollaries of the lemmas `bisim_sat0_L` and `_R`. For instance, when $n = 2$, the proof script for `bisim_sat2_L` is as simple as follows:

```
Theorem bisim_sat2_L : ∀ P Q (F : n → n → o′),
    (∀ X Y, bisim (P X Y) (Q X Y)) % H1
  → (∀ X Y, sat (P X Y) (F X Y))   % H2
  → (∀ X Y, sat (Q X Y) (F X Y)).
```

```
sig finite-pic. % file: finite-pic.sig

kind n  type. % names

kind p  type. % processes
type null        p.
type taup        p → p.
type plus, par   p → p → p.
type match, out  n → n → p → p.
type in          n → (n → p) → p.
type nu          (n → p) → p.

kind a  type. % actions (transition labels)
type tau         a.
type up, dn      n → n → a.

% one step for free transitions
type one      p →      a  →      p  → o.
% one step for binding transitions
type oneb     p → (n → a) → (n → p) → o.


module finite-pic. % file: finite-pic.mod

oneb (in X M) (dn X) M.      % bound input
one (out X Y P) (up X Y) P.  % free output
one  (taup P) tau P.         % tau
% match prefix
one  (match X X P) A Q :- one  P A Q.
oneb (match X X P) A M :- oneb P A M.
% sum
one  (plus P Q) A R :- one  P A R.
one  (plus P Q) A R :- one  Q A R.
oneb (plus P Q) A M :- oneb P A M.
oneb (plus P Q) A M :- oneb Q A M.
% par
one  (par P Q) A (par P1 Q) :- one P A P1.
one  (par P Q) A (par P Q1) :- one Q A Q1.
oneb (par P Q) A (x\ par (M x) Q) :- oneb P A M.
oneb (par P Q) A (x\ par P (N x)) :- oneb Q A N.
% restriction
one  (nu x\ P x) A (nu x\ Q x) :- pi x\ one  (P x) A (Q x).
oneb (nu x\ P x) A (y\ nu x\ Q x y) :- pi x\ oneb (P x) A (y\ Q x y).
% open
oneb (nu x\M x) (up X) N :- pi y\ one (M y) (up X y) (N y).
% close
one (par P Q) tau (nu y\ par (M y) (N y))
    :- oneb P (dn X) M , oneb Q (up X) N.
one (par P Q) tau (nu y\ par (M y) (N y))
    :- oneb P (up X) M , oneb Q (dn X) N.
% comm (interaction)
one (par P Q) tau (par (M Y) T) :- oneb P (dn X) M, one Q (up X Y) T.
one (par P Q) tau (par R (M Y)) :- oneb Q (dn X) M, one P (up X Y) R.
```

Figure 3 λProlog specification of the finite π-calculus operational semantics. (Adopted from one of the examples distributed with Abella.)

```
Specification "finite-pic". % load the finite pi-calc. spec. in Fig.3

CoDefine bisim : p → p → prop
by bisim P Q := (∀ A P1, {one  P  A     P1} →
                   ∃ Q1, {one  Q  A     Q1} ∧      bisim P1 Q1)
             ∧ (∀ X M, {oneb P (dn X) M} →
                   ∃ N, {oneb Q (dn X) N} ∧ ∀ w, bisim (M w) (N w))
             ∧ (∀ X M, {oneb P (up X) M} →
                   ∃ N, {oneb Q (up X) N} ∧ ∇ w, bisim (M w) (N w))
             ∧ (∀ A Q1, {one  Q  A     Q1} →
                   ∃ P1, {one  P  A     P1} ∧      bisim Q1 P1)
             ∧ (∀ X N, {oneb Q (dn X) N} →
                   ∃ M, {oneb P (dn X) M} ∧ ∀ w, bisim (N w) (M w))
             ∧ (∀ X N, {oneb Q (up X) N} →
                   ∃ M, {oneb P (up X) M} ∧ ∇ w, bisim (N w) (M w)).

Kind o'  type.               % syntax of the modal logic
Type tt, ff           o'.
Type ⋎, ⋏             o' → o' → o'.
Type □=, ◇=           n → n → o' → o'.
Type □, ◇             a → o' → o'.
Type □↑, ◇↑, □↓, ◇↓   n → (n → o') → o'.

Define sat : p → o' → prop  % semantics of the modal logic
by sat P tt
 ; sat P (⋎ A B) := sat P A ∧ sat P B
 ; sat P (⋏ A B) := sat P A ∨ sat P B
 ; sat P (□= X Y A) := X = Y → sat P A
 ; sat P (◇= X Y A) := X = Y ∧ sat P A
 ; sat P (□ X A) := ∀ P1, {one P X P1} → sat P1 A
 ; sat P (◇ X A) := ∃ P1, {one P X P1} ∧ sat P1 A
 ; sat P (□↑X A) := ∀ Q, {oneb P (up X) Q} →  ∇ w, sat (Q w) (A w)
 ; sat P (◇↑X A) := ∃ Q, {oneb P (up X) Q} ∧  ∇ w, sat (Q w) (A w)
 % basic input modality
 ; sat P (□↓X A) := ∀ Q, {oneb P (dn X) Q} →  ∀ w, sat (Q w) (A w)
 % late input modality
 ; sat P (◇↓X A) := ∃ Q, {oneb P (dn X) Q} ∧  ∀ w, sat (Q w) (A w).
```

Figure 4 A coinductive definition of the bisimulation and an inductive definition of a modal logic for the finite calculus in Abella. (Adopted from one of the examples distributed with Bedwyr [3], which implements [16], and modified for two-level logic style in Abella.)

```
Theorem bisim_sat0 : ∀ P Q F,
   bisim P Q → ((sat P F → sat Q F) ∧ (sat Q F → sat P F)).
Theorem sat_bisim0 : ∀ P Q,
   (∀ F, (sat P F → sat Q F) ∧ (sat Q F → sat P F)) → bisim P Q.

Theorem sateq_one_exists_L0 : ∀ P Q A P1,
  (∀ F, (sat P F → sat Q F) ∧ (sat Q F → sat P F)) →
  {one P A P1} →
  ∃ Q1, {one Q A Q1}.
Theorem sateq_one_L0 : ∀ P Q A P1,
  (∀ F, (sat P F → sat Q F) ∧ (sat Q F → sat P F)) →
  {one P A P1} →
  ∃ Q1, {one Q A Q1} ∧ (∀ F, (sat P1 F → sat Q1 F) ∧ (sat Q1 F → sat P1 F)).

Theorem bisim_sat2 : ∀ P Q (F : n → n → o'),
   (∀ X Y, bisim (P X Y) (Q X Y)) →
   ( (∀ X Y, sat (P X Y) (F X Y) →  ∀ X Y, sat (Q X Y) (F X Y))
   ∧ (∀ X Y, sat (Q X Y) (F X Y) →  ∀ X Y, sat (P X Y) (F X Y)) ).
Theorem sat_bisim2 : ∀ (X : n) (Y : n) P Q,
   (∀ F, (∀ X Y, sat (P X Y) (F X Y) →  ∀ X Y, sat (Q X Y) (F X Y))
       ∧ (∀ X Y, sat (Q X Y) (F X Y) →  ∀ X Y, sat (P X Y) (F X Y)) ) →
   bisim (P X Y) (Q X Y).

Theorem sateq_one_exists_L2 : ∀ (X : n) (Y : n) P Q A P1,
     ( ∀ F, (∀ X Y, sat (P X Y) (F X Y) →  ∀ X Y, sat (Q X Y) (F X Y))
          ∧ (∀ X Y, sat (Q X Y) (F X Y) →  ∀ X Y, sat (P X Y) (F X Y)) )
  → {one (P X Y) (A X Y) (P1 X Y)} → ∃ Q1, {one (Q X Y) (A X Y) (Q1 X Y)}.
Theorem sateq_one_L2 : ∀ (X : n) (Y : n) P Q A P1,
     ( ∀ F, (∀ X Y, sat (P X Y) (F X Y) →  ∀ X Y, sat (Q X Y) (F X Y))
          ∧ (∀ X Y, sat (Q X Y) (F X Y) →  ∀ X Y, sat (P X Y) (F X Y)) )
  → {one (P X Y) (A X Y) (P1 X Y)} →
     ( ∃ Q1, {one (Q X Y) (A X Y) (Q1 X Y)} ∧
       ∀ F, (∀ X Y, sat (P1 X Y) (F X Y) →  ∀ X Y, sat (Q1 X Y) (F X Y))
          ∧ (∀ X Y, sat (Q1 X Y) (F X Y) →  ∀ X Y, sat (P1 X Y) (F X Y)) ).
```

Figure 5 Two versions of main theorems and lemmas stated in Abella: one for closed processes and the other for processes with two free-variables.

```
% Proof.
  intros.
  assert bisim (P X Y) (Q X Y). backchain H1.
  assert sat (P X Y) (F X Y).  backchain H2.
  backchain bisim_sat0_L.  % Q.E.D.
```

The proof script for bisim_sat$_n$_L would be structurally identical to above, only differing in the number of free variables (i.e., $\mathtt{X}_1 \cdots \mathtt{X}_n$ instead of just two variables X Y). Therefore, the proof of $(\sim\, \subseteq\, \overset{\vDash}{\sim})$ can be fully mechanized in Abella for any given instance of n.

A.2 Logical Equivalence implies Bisimulation

We can prove sat_bisim0 assuming six lemmas, which state a closedness property of logical equivalence with regards to a commonly labeled transition step between two processes. Each of these six closedness lemmas discharges one of the six cases in the coinductive definition of bisim (see lines 3-15 in Fig. 4). Here is one of the six lemmas:

$$\texttt{Theorem sateq\_one\_L0 :}\ \forall P\, Q\, A\, P_1,\ [\![P \overset{\vDash}{\sim} Q]\!] \\ \to [\![P \xrightarrow{A} P_1]\!] \to \exists Q_1,\ [\![Q \xrightarrow{A} Q_1]\!] \land [\![P_1 \overset{\vDash}{\sim} Q_1]\!].$$

Another lemma sateq_one_R0 is a symmetric counterpart of above, where Q leads the transition rather than P. This pair of lemmas, sateq_one_L0 and _R0, states that for two logically equivalent processes, if one of them has a transition step, then the other also has a transition step with the same label such that the resulting pair of processes at next step are logically equivalent as well. Additionally, there are two more pairs of closedness lemmas: sateq_oneb_dn_L0 and _R0 for bounded inputs and sateq_oneb_up_L0 and _R0 for bounded outputs.

It is difficult to directly prove the six closedness lemmas. So, we first prove six existence lemmas, which have weaker conclusions than the closedness lemmas. Here is an existence lemma weakened from sateq_one_L0:

$$\texttt{Theorem sateq\_one\_exists\_L0 :}\ \forall P\, Q\, A\, P_1, \\ [\![P \overset{\vDash}{\sim} Q]\!] \to [\![P \xrightarrow{A} P_1]\!] \to \exists Q_1,\ [\![Q \xrightarrow{A} Q_1]\!].$$

We prove this by case analysis over $[\![P \xrightarrow{A} P_1]\!]$. For each case, we show $[\![Q \vDash \langle A\rangle \mathtt{tt}]\!]$ from $[\![P \vDash \langle A\rangle \mathtt{tt}]\!]$, which is obvious from $[\![P \xrightarrow{A} P_1]\!]$. From $[\![Q \vDash \langle A\rangle \mathtt{tt}]\!]$, $\exists Q_1, [\![Q \xrightarrow{A} Q_1]\!]$ is immediate. The proofs for the other five lemmas are similar.

The only gap we need to fill is from the existence lemmas to their corresponding closedness lemmas. This gap can be filled by two conditions: (1) the image finiteness of transition steps and (2) the existence of *a pair of distinguishing formulae* for logically inequivalent processes. The former is usually assumed by definition of the labeled transition system and the latter would usually be obliged to be derived using the former.

The first condition, image finiteness, means that a process can only have finitely many next step process configurations for each label. Using image finiteness for labeled transition systems are fairly standard. For instance, it is also assumed in proofs for the adequacy of Hennessy–Milner logic (HML) for CCS [5]. In our context of proving the closedness lemma sateq_one_L, image finiteness amounts to the existence of finitely many Q_is such that $[\![Q \xrightarrow{\alpha} Q_i]\!]$. Assuming this, we can exhaustively enumerate all such instances as $Q_1, Q_2, \cdots, Q_m$.

The remaining proof for sateq_one_L is to show that at least one of $P_1 \overset{\vDash}{\sim} Q_1, \cdots, P_1 \overset{\vDash}{\sim} Q_m$ holds, that is, it is impossible to fail all of them. Let us assume that it all fail, that is, $P_1 \overset{\vDash}{\nsim} Q_1, \cdots, P_1 \overset{\vDash}{\nsim} Q_m$. If we can find a formula ϕ_i for each $P_1 \overset{\vDash}{\nsim} Q_i$ such that $P_1 \vDash \phi_i$ holds but $Q_i \vDash \phi_i$ does not hold, then it contradicts the assumption $P \overset{\vDash}{\sim} Q$ because we can build a formula $\phi = \langle\alpha\rangle \bigwedge_{i=1\ldots m} \phi_i$, which makes

$P \models \phi$ hold but not $Q \models \phi$. Thus, the essence of the closedness lemma proof is whether we can always find a formula ϕ_i such that $P_1 \models \phi_i$ holds but $Q_i \models \phi_i$ does not hold, given that $P_1 \not\overset{\models}{\sim} Q_i$.

Having the second condition below guarantees that we can always find such ϕ_i. There are two reasons for $P_1 \not\overset{\models}{\sim} Q_i$:

- $\exists\,\phi_i$ such that $P_1 \models \phi_i$ holds but $Q_i \models \phi_i$ does not.
- $\exists\,\phi'_i$ such that $Q_i \models \phi'_i$ holds but $P_1 \models \phi'_i$ does not.

Thus, $P_1 \not\overset{\models}{\sim} Q_i$ guarantees the existence of either ϕ_i or ϕ'_i. However, the existence of one of them may not necessarily guarantee the other. In cases of HML for CCS and MPW for late bisimulation in the π-calculus we know that both ϕ_i and ϕ'_i exists because the negation of formulae can be defined syntactically. However, the encodings of the input box and diamond modalities for $\mathcal{OM}$ ($\Box^{\downarrow}$and $\Diamond^{\downarrow}$ in Fig. 3) are not logically dual; only if one of the ∀ w in lines 35 and 37 were ∃ w could the modalities be classically dual to each other. Therefore, syntactic negation for $\mathcal{OM}$ formulas cannot be defined in general, even before taking free variables and the intuitionistic framework into consideration.

We would need even more careful argument to establish the existence of the pair of formulae ϕ_i and ϕ'_i in the presence of free variables. That is, we need to establish the following: for any processes P and Q such that $P \overset{\models}{\sim} Q$ fails, that is $P \not\overset{\models}{\sim} Q$, we can find a pair of formulae that is satisfied by P or Q respectively but fails for the other process, despite the absence of syntactic negation. Our key observation in establishing this is to use open bisimulation instead of logical equivalence. From $(\sim\ \subseteq\ \overset{\models}{\sim})$, which is established in Section A.1, we get $(\not\overset{\models}{\sim}\ \subseteq\ \not\sim)$ by de Morgan's law. Therefore, it suffices to find a pair of formulae for $P \not\sim Q$, instead of $P \not\overset{\models}{\sim} Q$. In Section 3, we show how to find this pair of distinguishing formulae (Proposition 14).

A.3 Encoding of the history in direct semantics

So far we have only considered universal quantification, which encodes free variables and also inputs. Free variables are modeled as inputs from the top-level environment both in direct semantics and in Abella encoding. It is sufficient to consider universal quantifications only while formalizing the soundness and completeness theorems. More generally, during the internal steps of bisimulation and satisfaction evaluation, fresh names from bounded outputs may be generated. These names are encoded as ∇-quantified variables in Abella.

For instance, $[\![F(x,y,z) \models^{x^i\,y^o\,z^i} P(x,y,z)]\!] \triangleq \forall x, \nabla y, \forall z,$ `sat` $([\![F]\!]\,x\,y\,z)\,([\![P]\!]\,x\,y\,z)$. The ∇-quantified variables are distinct from the other variables already in scope, hence, $x = y$ is false. However, $y = z$ is not necessarily false because z is introduced afterwards so that there exists a substitution $\theta = \{^{y}/_{z}\}$ such that $y\theta = z\theta$. This exactly coincides with the restriction on respectful substitutions due to bound outputs in histories according to Def. 1.

In Section 4.2, we discuss related work on this style of encoding the π-calculus and its properties within a logic that supports higher-order abstract syntax and ∇-quantifiers.

A.4 Formalized example on excluded middle

```
Theorem excl_middle1 :
 (∀ (x : n) (y : n), x = y ∨ (x = y → false)) →
 (∀ x y, sat ⟦τ + τ.τ⟧ (◇ tau (⋎ (□ tau (◇= x y tt)) (□= x y (◇ tau tt))))).

Theorem excl_middle2 :
 (∀ x y, sat ⟦τ + τ.τ⟧ (◇ tau (⋎ (□ tau (◇= x y tt)) (□= x y (◇ tau tt))))) →
 (∀ (x : n) (y : n), x = y ∨ (x = y → false)).
```

The judgement $\tau + \tau.\tau \models \langle\tau\rangle([\tau]\langle x = y\rangle\mathtt{tt} \wedge [x = y]\langle\tau\rangle\mathtt{tt})$ is not satisfiyable in $\mathcal{OM}$. This judgement is in fact necessary and sufficient to the exluded middle over names, which is formally justified by proving the two theorems above in Abella.

B Automatic testing of the examples in Bedwyr

Manually verifying examples involving open bisimilarity can be additionally tedious and error prone than prior notions of bisimulation because we must consider all possible substitutions in each transition step. To avoid such mistakes we tested every example in this paper using Bedwyr [3], which is a system that automatically searches for proofs in a logic that is closely related to the reasoning logic of Abella. That is, Bedwyr can automatically decided a certain subset of the judgements in Abella. Since Bedwyr does not support two-level logic approach of reasoning about λProlog specifications, both the semantics of the π-calculus and the modal logic must be defined by the same logic in Bedwyr, which corresponds to the reasoning logic level in Abella.

Bedwyr definitions for testing examples are very similar to the Abella definitions in the previous section, except for few definitions with different names (e.g., the `null` process in Abella corresponds to `z` in Bedwyr). Here, we list the output of test runs of the examples. Further details of the Bedwyr source code can be found ounline at https://github.com/kyagrd/NonBisim2DF.

First example in Section 1

$\boxed{\overline{a}b \parallel c(x) \not\models \langle\tau\rangle\mathtt{tt} \vee [\tau]\mathtt{ff}}$

```
?= forall a b c,
   sat (par (out a b z) (in c x\ z)) (disj (diaAct tau tt) (boxAct tau ff)).
No solution.
```

$\boxed{\overline{a}b \parallel c(x) \not\models \langle\tau\rangle\mathtt{tt}}$

```
?= forall a b c, sat (par (out a b z) (in c x\ z)) (diaAct tau tt).
No solution.
```

$\boxed{\overline{a}b \parallel c(x) \not\models [\tau]\mathtt{ff}}$

```
?= forall a b c, sat (par (out a b z) (in c x\ z)) (boxAct tau ff).
No solution.
```

Further example in Section 1

$$Q \triangleq \tau.(\overline{a}b.a(x) + a(x).\overline{a}b + \tau) + \tau.(\overline{a}b.c(x) + c(x).\overline{a}b) \qquad\qquad P \triangleq Q + \tau.(\overline{a}b \parallel c(x))$$

$\boxed{P \not\sim Q}$ (Q and P corresponds to R and S in the further example in Section 1.)

```
?= ABCx = a\b\c\ out a b (in c x\z) /\ CxAB = a\b\c\ in c x\out a b z /\
   ABAx = a\b\ out a b (in a x\z)   /\ AxAB = a\b\ in a x\out a b z   /\
   Q = a\b\c\ plus (taup (plus (ABCx a b c) (CxAB a b c)))
                   (taup (plus (plus (ABAx a b) (AxAB a b)) (taup z))) /\
   P = a\b\c\ plus (taup (par (out a b z) (in c x\z))) (Q a b c) /\
   forall a b c, bisim (P a b c) (Q a b c).
No solution.
```

$\boxed{Q \models [\tau](\langle\tau\rangle\mathtt{tt} \vee [\tau]\mathtt{ff})}$

```
?= ABCx = a\b\c\ out a b (in c x\z) /\ CxAB = a\b\c\ in c x\out a b z /\
   ABAx = a\b\ out a b (in a x\z)   /\ AxAB = a\b\ in a x\out a b z   /\
   Q = a\b\c\ plus (taup (plus (ABCx a b c) (CxAB a b c)))
                   (taup (plus (plus (ABAx a b) (AxAB a b)) (taup z))) /\
   P = a\b\c\ plus (taup (par (out a b z) (in c x\z))) (Q a b c) /\
   forall a b c, sat (Q a b c) (boxAct tau (disj (diaAct tau tt) (boxAct tau ff))).
Found a solution:
 P = x1\x2\x3\
     plus (taup (par (out x1 x2 z) (in x3 (x4\ z))))
          (plus (taup (plus (out x1 x2 (in x3 (x4\ z))) (in x3 (x4\ out x1 x2 z))))
                (taup (plus (plus (out x1 x2 (in x1 (x4\ z))) (in x1 (x4\ out x1 x2 z))) (taup z))))
 Q = x1\x2\x3\
     plus (taup (plus (out x1 x2 (in x3 (x4\ z))) (in x3 (x4\ out x1 x2 z))))
          (taup (plus (plus (out x1 x2 (in x1 (x4\ z))) (in x1 (x4\ out x1 x2 z))) (taup z)))
 AxAB = x1\x2\ in x1 (x3\ out x1 x2 z)
 ABAx = x1\x2\ out x1 x2 (in x1 (x3\ z))
 CxAB = x1\x2\x3\ in x3 (x4\ out x1 x2 z)
 ABCx = x1\x2\x3\ out x1 x2 (in x3 (x4\ z))
More [y] ?
No more solutions (found 1).
```

$\boxed{P \not\models [\tau](\langle\tau\rangle\mathtt{tt} \vee [\tau]\mathtt{ff})}$

```
?= ABCx = a\b\c\ out a b (in c x\z) /\ CxAB = a\b\c\ in c x\out a b z /\
   ABAx = a\b\ out a b (in a x\z)   /\ AxAB = a\b\ in a x\out a b z   /\
   Q = a\b\c\ plus (taup (plus (ABCx a b c) (CxAB a b c)))
                   (taup (plus (plus (ABAx a b) (AxAB a b)) (taup z))) /\
   P = a\b\c\ plus (taup (par (out a b z) (in c x\z))) (Q a b c) /\
   forall a b c, sat (P a b c) (boxAct tau (disj (diaAct tau tt) (boxAct tau ff))).
No solution.
```

Last example in Section 1

$\boxed{\tau \nsim [a = c]\tau}$

```
?= forall a c, bisim (taup z) (match a c (taup z)).
No solution.
```

$\boxed{\tau \models \langle\tau\rangle\mathtt{tt}}$

```
?= forall a c, sat (taup z) (diaAct tau tt).
Found a solution.
More [y] ?
No more solutions (found 1).
```

$\boxed{[a = c]\tau \not\models \langle\tau\rangle\mathtt{tt}}$

```
?= forall a c, sat (match a c (taup z)) (diaAct tau tt).
No solution.
```

$\boxed{\tau \not\models [\tau]\mathtt{ff}}$

```
?= forall a c, sat (taup z) (boxAct tau ff).
No solution.
```

$\boxed{[a = c]\tau \not\models [\tau]\mathtt{ff}}$

```
?= forall a c, sat (match a c (taup z)) (boxAct tau ff).
No solution.
```

Examples in Section 2.2.2 and the base cases of Section 2.3.1

$\boxed{[x = y]\tau \nsim \tau}$

```
?= forall x y, bisim (match x y (taup z)) (taup z).
No solution.
```

$\boxed{[x = y]\tau \models [\tau]\langle x = y\rangle\mathtt{tt}}$

```
?= forall x y, sat (match x y (taup z)) (boxAct tau (diaMatch x y tt)).
Found a solution.
More [y] ?
No more solutions (found 1).
```

$\boxed{\tau \not\models [\tau]\langle x = y\rangle\mathtt{tt}}$

```
?= forall x y, sat (taup z) (boxAct tau (diaMatch x y tt)).
No solution.
```

$\boxed{\tau \models [x = y]\langle\tau\rangle\mathtt{tt}}$

```
?= forall x y, sat (taup z) (boxMatch x y (diaAct tau tt)).
Found a solution.
More [y] ?
No more solutions (found 1).
```

$\boxed{[x = y]\tau \models [x = y]\langle\tau\rangle\mathtt{tt}}$

```
?= forall x y, sat (match x y (taup z)) (boxMatch x y (diaAct tau tt)).
Found a solution.
More [y] ?
No more solutions (found 1).
```

$\boxed{[x = y]\tau \not\models \langle\tau\rangle\mathtt{tt}}$

```
?= forall x y, sat (match x y (taup z)) (diaAct tau tt).
No solution.
```

$\boxed{\tau \models \langle\tau\rangle\mathtt{tt}}$

```
?= forall x y, sat (taup z) (diaAct tau tt).
Found a solution.
More [y] ?
No more solutions (found 1).
```

$\boxed{[x = y]\tau \nsim 0}$

```
?= forall x y, bisim (match x y (taup z)) z.
No solution.
```

$\boxed{0 \not\models [x = y]\langle\tau\rangle\mathtt{tt}}$

```
?= forall x y, sat z (boxMatch x y (diaAct tau tt)).
No solution.
```

$\boxed{0 \models [\tau]\langle x = y\rangle\mathtt{tt}}$

```
?= forall x y, sat z (boxAct tau (diaMatch x y tt)).
Found a solution.
More [y] ?
No more solutions (found 1).
```

$\boxed{0 \models [\tau]\mathtt{ff}}$

```
?= forall x y, sat z (boxAct tau ff).
Found a solution.
More [y] ?
No more solutions (found 1).
```

$\boxed{0 \not\models \langle\tau\rangle\mathtt{tt}}$

```
?= forall x y, sat z (diaAct tau tt).
No solution.
```

The inductive case example of Section 2.3.1

$\boxed{\tau.[x = y]\tau + \tau + \tau.\tau \not\sim \tau + \tau.\tau}$

```
?= T = taup z /\ TT = taup T /\
   P = (x\ y\ plus (taup (match x y T)) Q) /\ Q = plus TT T /\
   forall x y, bisim (P x y) Q. % (t.[x=y].t + t.t + t) /~ (t.t + t)
No solution.
```

$\boxed{\tau.[x = y]\tau + \tau + \tau.\tau \models \langle\tau\rangle([\tau]\langle x = y\rangle\mathtt{tt} \wedge [x = y]\langle\tau\rangle\mathtt{tt})}$

```
?= T = taup z /\ TT = taup T /\
   P = (x\ y\ plus (taup (match x y T)) Q) /\ Q = plus TT T /\
   forall x y, sat (P x y) (diaAct tau (conj (boxAct tau (diaMatch x y tt))
                                             (boxMatch x y (diaAct tau tt)))).
Found a solution:
 Q = plus (taup (taup z)) (taup z)
 P = x1\x2\
     plus (taup (match x1 x2 (taup z))) (plus (taup (taup z)) (taup z))
 TT = taup (taup z)
 T = taup z
More [y] ?
No more solutions (found 1).
```

$\boxed{\tau + \tau.\tau \not\models \langle\tau\rangle([\tau]\langle x = y\rangle\mathtt{tt} \wedge [x = y]\langle\tau\rangle\mathtt{tt})}$

```
?= T = taup z /\ TT = taup T /\
   P = (x\ y\ plus (taup (match x y T)) Q) /\ Q = plus TT T /\
   sat Q (diaAct tau (conj (boxAct tau (diaMatch x y tt))
                           (boxMatch x y (diaAct tau tt)))).
No solution.
```

$\boxed{\tau + \tau.\tau \models \langle\tau\rangle([\tau]\langle x = y\rangle\mathtt{tt} \wedge [x = y]\langle\tau\rangle\mathtt{tt})}$

```
?= T = taup z /\ TT = taup T /\
   P = (x\ y\ plus (taup (match x y T)) Q) /\ Q = plus TT T /\
   forall x y, sat Q (boxAct tau (disj (diaAct tau tt) (boxAct tau ff))).
Found a solution:
 Q = plus (taup (taup z)) (taup z)
 P = x1\x2\
     plus (taup (match x1 x2 (taup z))) (plus (taup (taup z)) (taup z))
 TT = taup (taup z)
 T = taup z
More [y] ?
No more solutions (found 1).
```

$\boxed{\tau.[x = y]\tau + \tau + \tau.\tau \not\models \langle\tau\rangle([\tau]\langle x = y\rangle\mathtt{tt} \wedge [x = y]\langle\tau\rangle\mathtt{tt})}$

```
?= T = taup z /\ TT = taup T /\
   P = (x\ y\ plus (taup (match x y T)) Q) /\ Q = plus TT T /\
   forall x y, sat (P x y) (boxAct tau (disj (diaAct tau tt) (boxAct tau ff))).
No solution.
```

$\boxed{[x = y]\tau + \tau + \tau.\tau \sim \tau + \tau.\tau}$

```
?= T = taup z /\ TT = taup T /\
   P = (x\ y\ plus (match x y T) Q) /\ Q = plus TT T /\
   forall x y, bisim (P x y) Q. % ([x=y].t + t.t + t) ~ (t.t + t)
Found a solution:
 Q = plus (taup (taup z)) (taup z)
 P = x1\x2\ plus (match x1 x2 (taup z)) (plus (taup (taup z)) (taup z))
 TT = taup (taup z)
 T = taup z
More [y] ?
No more solutions (found 1).
```

Example involving private names in Section 2.3.2

$\boxed{P \triangleq \nu x.\overline{a}x.a(y).\tau \nsim \nu x.\overline{a}x.a(y).[x = y]\tau \triangleq Q}$

```
?= P = a\ nu x\ out a x (in a y\ taup z) /\
   Q = a\ nu x\ out a x (in a y\ match x y (taup z)) /\
   forall a, bisim (P a) (Q a).
No solution.
```

$\boxed{\nu x.\overline{a}x.a(y).\tau \models \langle\overline{a}(x)\rangle\langle a(y)\rangle\langle\tau\rangle\mathtt{tt}}$

```
?= P = a\ nu x\ out a x (in a y\ taup z) /\
   Q = a\ nu x\ out a x (in a y\ match x y (taup z)) /\
   forall a, sat (P a) (diaOut a x\ diaInL a y\ diaAct tau tt).
Found a solution:
 Q = x1\ nu (x2\ out x1 x2 (in x1 (x3\ match x2 x3 (taup z))))
 P = x1\ nu (x2\ out x1 x2 (in x1 (x3\ taup z)))
More [y] ?
No more solutions (found 1).
```

$\boxed{\nu x.\overline{a}x.a(y).[x = y]\tau \not\models \langle\overline{a}(x)\rangle\langle a(y)\rangle\langle\tau\rangle\mathtt{tt}}$

```
?= P = a\ nu x\ out a x (in a y\ taup z) /\
   Q = a\ nu x\ out a x (in a y\ match x y (taup z)) /\
   forall a, sat (Q a) (diaOut a x\ diaInL a y\ diaAct tau tt).
No solution.
```

$\boxed{\nu x.\overline{a}x.a(y).\tau \not\models [\overline{a}(x)][a(y)][\tau]\langle x = y\rangle\mathtt{tt}}$

```
?= P = a\ nu x\ out a x (in a y\ taup z) /\
   Q = a\ nu x\ out a x (in a y\ match x y (taup z)) /\
   forall a, sat (P a) (boxOut a x\ boxIn a y\ boxAct tau (diaMatch x y tt)).
No solution.
```

$\boxed{\nu x.\overline{a}x.a(y).[x = y]\tau \models [\overline{a}(x)][a(y)][\tau]\langle x = y\rangle \mathtt{tt}}$

```
?= P = a\ nu x\ out a x (in a y\ taup z) /\
   Q = a\ nu x\ out a x (in a y\ match x y (taup z)) /\
   forall a, sat (Q a) (boxOut a x\ boxIn a y\ boxAct tau (diaMatch x y tt)).
Found a solution:
 Q = x1\ nu (x2\ out x1 x2 (in x1 (x3\ match x2 x3 (taup z))))
 P = x1\ nu (x2\ out x1 x2 (in x1 (x3\ taup z)))
More [y] ?
No more solutions (found 1).
```

Example involving private names in Section 2.3.3

$\boxed{\nu x.\overline{a}x \sim \nu x.\overline{a}x.[x = a]\tau}$

```
?= P = a\ nu x\ out a x z /\ Q = a\ nu x\ out a x (match x a (taup z)) /\
   forall a, bisim (P a) (Q a).
Found a solution:
 Q = x1\ nu (x2\ out x1 x2 (match x2 x1 (taup z)))
 P = x1\ nu (x2\ out x1 x2 z)
More [y] ?
No more solutions (found 1).
```

Examples in Section 3.3.1

$\boxed{[x = y]\tau + [w = z]\tau \nsim \tau}$

```
?= T = taup z /\ forall x y u v, bisim (plus (match x y T) (match u v T)) T.
No solution.
```

$\boxed{[x = y]\tau + [w = z]\tau \models [\tau](\langle x = y\rangle \mathtt{tt} \vee \langle w = z\rangle \mathtt{tt})}$

```
?= T = taup z /\
   forall x y u v,
   sat (plus (match x y T) (match u v T)) (boxAct tau (disj (diaMatch x y tt) (diaMatch u v tt))).
Found a solution:
 T = taup z
More [y] ?
No more solutions (found 1).
```

$\boxed{\tau \not\models [\tau](\langle x = y\rangle \mathtt{tt} \vee \langle w = z\rangle \mathtt{tt})}$

```
?= T = taup z /\
   forall x y u v,
   sat T (boxAct tau (disj (diaMatch x y tt) (diaMatch u v tt))).
No solution.
```

$\boxed{\tau \models \langle\tau\rangle \mathtt{tt}}$

```
?= T = taup z /\ forall x y u v, sat T (diaAct tau tt).
Found a solution:
 T = taup z
More [y] ?
```

$\boxed{[x = y]\tau + [w = z]\tau \not\models \langle\tau\rangle \mathtt{tt}}$

```
?= T = taup z /\
   forall x y u v, sat (plus (match x y T) (match u v T)) (diaAct tau tt).
No solution.
```

$\boxed{P \triangleq \tau.(\overline{a}a + \overline{b}b) + [x = y]\tau.\overline{a}a \;\not\sim\; \tau.(\overline{a}a + \overline{b}b) + \tau.\overline{a}a \triangleq Q}$

```
?= R = a\b\ taup (plus (out a a z) (out b b z)) /\ S = a\b\ taup (out a a z) /\
   P = a\b\x\y\ plus (R a b) (match x y (S a b)) /\ Q = a\b\ plus (R a b) (S a b) /\
   forall a b x y, bisim (P a b x y) (Q a b).
No solution.
```

$\boxed{\tau.(\overline{a}a + \overline{b}b) + [x = y]\tau.\overline{a}a \models [\tau](\langle\overline{b}b\rangle\mathtt{tt} \vee \langle x = y\rangle\mathtt{tt})}$

```
?= R = a\b\ taup (plus (out a a z) (out b b z)) /\ S = a\b\ taup (out a a z) /\
   P = a\b\x\y\ plus (R a b) (match x y (S a b)) /\ Q = a\b\ plus (R a b) (S a b) /\
   forall a b x y, sat (P a b x y) (boxAct tau (disj (diaAct (up b b) tt) (diaMatch x y tt))).
Found a solution:
 Q = x1\x2\ plus (taup (plus (out x1 x1 z) (out x2 x2 z))) (taup (out x1 x1 z))
 P = x1\x2\x3\x4\ plus (taup (plus (out x1 x1 z) (out x2 x2 z))) (match x3 x4 (taup (out x1 x1 z)))
 S = x1\x2\ taup (out x1 x1 z)
 R = x1\x2\ taup (plus (out x1 x1 z) (out x2 x2 z))
More [y] ?
No more solutions (found 1).
```

$\boxed{\tau.(\overline{a}a + \overline{b}b) + \tau.\overline{a}a \not\models [\tau](\langle\overline{b}b\rangle\mathtt{tt} \vee \langle x = y\rangle\mathtt{tt})}$

```
?= R = a\b\ taup (plus (out a a z) (out b b z)) /\ S = a\b\ taup (out a a z) /\
   P = a\b\x\y\ plus (R a b) (match x y (S a b)) /\ Q = a\b\ plus (R a b) (S a b) /\
   forall a b x y, sat (Q a b) (boxAct tau (disj (diaAct (up b b) tt) (diaMatch x y tt))).
No solution.
```

$\boxed{\tau.(\overline{a}a + \overline{b}b) + [x = y]\tau.\overline{a}a \not\models \langle\tau\rangle[\overline{b}b]\langle a = b\rangle\mathtt{tt}}$

```
?= R = a\b\ taup (plus (out a a z) (out b b z)) /\ S = a\b\ taup (out a a z) /\
   P = a\b\x\y\ plus (R a b) (match x y (S a b)) /\ Q = a\b\ plus (R a b) (S a b) /\
   forall a b x y, sat (P a b x y) (diaAct tau (boxAct (up b b) (diaMatch a b tt))).
No solution.
```

$\boxed{\tau.(\overline{a}a + \overline{b}b) + \tau.\overline{a}a \models \langle\tau\rangle[\overline{b}b]\langle a = b\rangle\mathtt{tt}}$

```
?= R = a\b\ taup (plus (out a a z) (out b b z)) /\ S = a\b\ taup (out a a z) /\
   P = a\b\x\y\ plus (R a b) (match x y (S a b)) /\ Q = a\b\ plus (R a b) (S a b) /\
   forall a b x y, sat (Q a b) (diaAct tau (boxAct (up b b) (diaMatch a b tt))).
Found a solution:
 Q = x1\x2\ plus (taup (plus (out x1 x1 z) (out x2 x2 z))) (taup (out x1 x1 z))
 P = x1\x2\x3\x4\ plus (taup (plus (out x1 x1 z) (out x2 x2 z))) (match x3 x4 (taup (out x1 x1 z)))
 S = x1\x2\ taup (out x1 x1 z)
 R = x1\x2\ taup (plus (out x1 x1 z) (out x2 x2 z))
More [y] ?
No more solutions (found 1).
```

Examples in Section 3.3.2

$\boxed{\overline{a}a \nsim \overline{a}b}$

```
?= forall a b, bisim  (out a a z)  (out a b z).
No solution.
```

$\boxed{\overline{a}a \models [\overline{a}b]\langle a = b\rangle \mathtt{tt}}$

```
?= forall a b, sat (out a a z) (boxAct (up a b) (diaMatch a b tt)).
Found a solution.
More [y] ?
No more solutions (found 1).
```

$\boxed{\overline{a}b \models [\overline{a}b]\langle a = b\rangle \mathtt{tt}}$

```
?= forall a b, sat (out a b z) (boxAct (up a b) (diaMatch a b tt)).
No solution.
```

$\boxed{\nu b.\overline{a}b.a(x).[x = b]\overline{x}x \nsim \nu b.\overline{a}b.a(x).\overline{x}x}$

```
?= P = a\ nu b\ out a b (in a x\ match x b (out x x z)) /\
   Q = a\ nu b\ out a b (in a x\ out x x z) /\
   forall a, bisim (P a) (Q a).
No solution.
```

$\boxed{\nu b.\overline{a}b.a(x).[x = b]\overline{x}x \models [a(b)][a(x)][\overline{x}x]\langle x = b\rangle \mathtt{tt}}$

```
?= P = a\ nu b\ out a b (in a x\ match x b (out x x z)) /\
   Q = a\ nu b\ out a b (in a x\ out x x z) /\
   forall a, sat (P a) (diaOut a b\ diaInL a x\ boxAct (up x x) (diaMatch x b tt)).
Found a solution:
 Q = x1\ nu (x2\ out x1 x2 (in x1 (x3\ out x3 x3 z)))
 P = x1\ nu (x2\ out x1 x2 (in x1 (x3\ match x3 x2 (out x3 x3 z))))
More [y] ?
No more solutions (found 1).
```

$\boxed{\nu b.\overline{a}b.a(x).\overline{x}x \not\models [a(b)][a(x)][\overline{x}x]\langle x = b\rangle \mathtt{tt}}$

```
?= P = a\ nu b\ out a b (in a x\ match x b (out x x z)) /\
   Q = a\ nu b\ out a b (in a x\ out x x z) /\
   forall a, sat (Q a) (diaOut a b\ diaInL a x\ boxAct (up x x) (diaMatch x b tt)).
No solution.
```

Example in Section ??

$\boxed{[x = y]\tau.\tau + \tau \not\sim \tau.\tau + \tau}$

```
?= forall x y, bisim (plus (match x y (taup (taup z))) (taup z)) (plus (taup (taup z)) (taup z)).
No solution.
```

$\boxed{[x = y]\tau.\tau + \tau \models [\tau][\tau]\langle x = y\rangle \mathtt{tt}}$

```
?= forall x y,
   sat (plus (match x y (taup (taup z))) (taup z)) (boxAct tau (boxAct tau (diaMatch x y tt))).
Found a solution.
More [y] ?
No more solutions (found 1).
```

$\boxed{\tau.\tau + \tau \not\models [\tau][\tau]\langle x = y\rangle \mathtt{tt}}$

```
?= forall x y,
   sat (plus (taup (taup z)) (taup z)) (boxAct tau (boxAct tau (diaMatch x y tt))).
No solution.
```

$\boxed{[x = y]\tau.\tau + \tau \models [\tau]([\tau]\mathtt{ff} \vee \langle x = y\rangle \mathtt{tt})}$

```
?= forall x y,
   sat (plus (match x y (taup (taup z))) (taup z)) (boxAct tau (disj (boxAct tau ff)
                                                                     (diaMatch x y tt))).
Found a solution.
More [y] ?
No more solutions (found 1).
```

$\boxed{\tau.\tau + \tau \not\models [\tau]([\tau]\mathtt{ff} \vee \langle x = y\rangle \mathtt{tt})}$

```
?= forall x y,
   sat (plus (taup (taup z)) (taup z)) (boxAct tau (disj (boxAct tau ff)
                                                         (diaMatch x y tt))).
No solution.
```

Elaborate example in Section 3.3.3

$\boxed{Q \triangleq \tau.(\tau + \tau.\tau + \tau.[x = y][w = z]\tau) \not\sim \tau.(\tau + \tau.\tau + \tau.[x = y]\tau) + Q \triangleq P}$ (P and Q are swapped in Section 3.3.3.)

```
?= T = taup z /\ TT = taup T /\
   Q = x\y\u\v\ taup (plus (plus T TT) (taup (match x y (match u v T)))) /\
   P = x\y\u\v\ plus (taup (plus (plus T TT) (taup (match x y T)))) (Q x y u v) /\
   forall x y u v, bisim (P x y u v) (Q x y u v).
No solution.
```

$\boxed{P \models \langle\tau\rangle[\tau]([\tau]\mathtt{ff} \vee [x = y]\langle\tau\rangle \mathtt{tt})}$

```
?= T = taup z /\ TT = taup T /\
   Q = x\y\u\v\ taup (plus (plus T TT) (taup (match x y (match u v T)))) /\
   P = x\y\u\v\ plus (taup (plus (plus T TT) (taup (match x y T)))) (Q x y u v) /\
   forall x y u v,
   sat (P x y u v) (diaAct tau (boxAct tau (disj (boxAct tau ff)
                                                 (boxMatch x y (diaAct tau tt))))).
Found a solution:
 P = x1\x2\x3\x4\
     plus
      (taup (plus (plus (taup z) (taup (taup z))) (taup (match x1 x2 (taup z)))))
      (taup (plus (plus (taup z) (taup (taup z))) (taup (match x1 x2 (match x3 x4 (taup z))))))
 Q = x1\x2\x3\x4\
     taup (plus (plus (taup z) (taup (taup z))) (taup (match x1 x2 (match x3 x4 (taup z)))))
 TT = taup (taup z)
 T = taup z
More [y] ?
No more solutions (found 1).
```

$\boxed{Q \not\models \langle\tau\rangle[\tau]([\tau]\mathtt{ff} \vee [x = y]\langle\tau\rangle\mathtt{tt})}$

```
?= T = taup z /\ TT = taup T /\
   Q = x\y\u\v\ taup (plus (plus T TT) (taup (match x y (match u v T)))) /\
   P = x\y\u\v\ plus (taup (plus (plus T TT) (taup (match x y T)))) (Q x y u v) /\
   forall x y u v,
   sat (Q x y u v) (diaAct tau (boxAct tau (disj (boxAct tau ff)
                                   (boxMatch x y (diaAct tau tt))))).
No solution.
```

$\boxed{Q \models [\tau]\langle\tau\rangle([\tau]\langle w = z\rangle\mathtt{tt} \wedge [x = y][w = z]\langle\tau\rangle\mathtt{tt})}$

```
?= T = taup z /\ TT = taup T /\
   Q = x\y\u\v\ taup (plus (plus T TT) (taup (match x y (match u v T)))) /\
   P = x\y\u\v\ plus (taup (plus (plus T TT) (taup (match x y T)))) (Q x y u v) /\
   forall x y u v,
   sat (Q x y u v) (boxAct tau (diaAct tau (conj (boxAct tau (diaMatch u v tt))
                                                  (boxMatch x y (boxMatch u v (diaAct tau tt)))))).
Found a solution:
 P = x1\x2\x3\x4\
     plus
      (taup (plus (plus (taup z) (taup (taup z))) (taup (match x1 x2 (taup z)))))
      (taup (plus (plus (taup z) (taup (taup z))) (taup (match x1 x2 (match x3 x4 (taup z))))))
 Q = x1\x2\x3\x4\
     taup (plus (plus (taup z) (taup (taup z))) (taup (match x1 x2 (match x3 x4 (taup z)))))
 TT = taup (taup z)
 T = taup z
More [y] ?
No more solutions (found 1).
```

$\boxed{P \not\models [\tau]\langle\tau\rangle([\tau]\langle w = z\rangle\mathtt{tt} \wedge [x = y][w = z]\langle\tau\rangle\mathtt{tt})}$

```
?= T = taup z /\ TT = taup T /\
   Q = x\y\u\v\ taup (plus (plus T TT) (taup (match x y (match u v T)))) /\
   P = x\y\u\v\ plus (taup (plus (plus T TT) (taup (match x y T)))) (Q x y u v) /\
   forall x y u v,
   sat (P x y u v) (boxAct tau (diaAct tau (conj (boxAct tau (diaMatch u v tt))
                                                  (boxMatch x y (boxMatch u v (diaAct tau tt)))))).
No solution.
```

$\boxed{P \models \langle\tau\rangle\langle\tau\rangle([x = y]\langle\tau\rangle\mathtt{tt} \wedge [\tau]\langle x = y\rangle\mathtt{tt})}$

```
?= T = taup z /\ TT = taup T /\
   Q = x\y\u\v\ taup (plus (plus T TT) (taup (match x y (match u v T)))) /\
   P = x\y\u\v\ plus (taup (plus (plus T TT) (taup (match x y T)))) (Q x y u v) /\
   forall x y u v,
   sat (P x y u v) (diaAct tau (diaAct tau (conj (boxMatch x y (diaAct tau tt))
                                                  (boxAct tau (diaMatch x y tt))))).
Found a solution:
 P = x1\x2\x3\x4\
     plus
      (taup (plus (plus (taup z) (taup (taup z))) (taup (match x1 x2 (taup z)))))
      (taup (plus (plus (taup z) (taup (taup z))) (taup (match x1 x2 (match x3 x4 (taup z))))))
 Q = x1\x2\x3\x4\
     taup (plus (plus (taup z) (taup (taup z))) (taup (match x1 x2 (match x3 x4 (taup z)))))
 TT = taup (taup z)
 T = taup z
More [y] ?
No more solutions (found 1).
```

$\boxed{Q \not\models \langle\tau\rangle\langle\tau\rangle([x = y]\langle\tau\rangle\mathtt{tt} \wedge [\tau]\langle x = y\rangle\mathtt{tt})}$

```
?= T = taup z /\ TT = taup T /\
   Q = x\y\u\v\ taup (plus (plus T TT) (taup (match x y (match u v T)))) /\
   P = x\y\u\v\ plus (taup (plus (plus T TT) (taup (match x y T)))) (Q x y u v) /\
   forall x y u v,
   sat (Q x y u v) (diaAct tau (diaAct tau (conj (boxMatch x y (diaAct tau tt))
                                                 (boxAct tau (diaMatch x y tt)))))).
No solution.
```

$\boxed{Q \models [\tau][\tau](\langle\tau\rangle\mathtt{tt} \vee [\tau]\langle w = z\rangle\mathtt{tt})}$

```
?= T = taup z /\ TT = taup T /\
   Q = x\y\u\v\ taup (plus (plus T TT) (taup (match x y (match u v T)))) /\
   P = x\y\u\v\ plus (taup (plus (plus T TT) (taup (match x y T)))) (Q x y u v) /\
   forall x y u v,
   sat (Q x y u v) (boxAct tau (boxAct tau (disj (diaAct tau tt)
                                                 (boxAct tau (diaMatch u v tt)))))).
Found a solution:
 P = x1\x2\x3\x4\
     plus
      (taup (plus (plus (taup z) (taup (taup z))) (taup (match x1 x2 (taup z)))))
      (taup (plus (plus (taup z) (taup (taup z))) (taup (match x1 x2 (match x3 x4 (taup z))))))
 Q = x1\x2\x3\x4\
     taup (plus (plus (taup z) (taup (taup z))) (taup (match x1 x2 (match x3 x4 (taup z)))))
 TT = taup (taup z)
 T = taup z
More [y] ?
No more solutions (found 1).
```

$\boxed{P \models [\tau][\tau](\langle\tau\rangle\mathtt{tt} \vee [\tau]\langle w = z\rangle\mathtt{tt})}$

```
?= T = taup z /\ TT = taup T /\
   Q = x\y\u\v\ taup (plus (plus T TT) (taup (match x y (match u v T)))) /\
   P = x\y\u\v\ plus (taup (plus (plus T TT) (taup (match x y T)))) (Q x y u v) /\
   forall x y u v,
   sat (P x y u v) (boxAct tau (boxAct tau (disj (diaAct tau tt)
                                                 (boxAct tau (diaMatch u v tt)))))).
No solution.
```

Examples in Section 4.1

$\boxed{a(y).[x = y]\overline{x}x \models^L [a(y)]^L\langle\overline{x}x\rangle \mathtt{tt}}$

```
?= nabla a x, lsat (a :: x :: nil) (in a y\match x y (out x x z)) (boxInL a y\diaAct (up x x) tt).
Found a solution.
More [y] ?
No more solutions (found 1).
```

$\boxed{a(y).0 \not\models^L [a(y)]^L\langle\overline{x}x\rangle \mathtt{tt}}$

```
?= nabla a x, lsat (a :: x :: nil) (in a y\z) (boxInL a y\diaAct (up x x) tt).
No soulution.
```

$\boxed{a(y).0 \models^L \langle a(y)\rangle[\overline{x}x]\mathtt{ff}}$

```
?= nabla a x, lsat (a :: x :: nil) (in a y\z) (diaInL a y\boxAct (up x x) ff).
Found a solution.
More [y] ?
No more solutions (found 1).
```

$\boxed{a(y).[x = y]\overline{x}x \not\models^L \langle a(y)\rangle[\overline{x}x]\mathtt{ff}}$

```
?= nabla a x, lsat (a :: x :: nil) (in a y\match x y (out x x z)) (diaInL a y\boxAct (up x x) ff).
No solution.
```

Examples in Section 4.2

$\boxed{a(x).\tau + a(x) + a(x).[x = a]\tau \nsim a(x).\tau + a(x)}$

```
?= T = taup z /\
   Q = a\x\ plus (in a x\T) (in a x\z) /\
   P = a\x\ plus (Q a x) (in a x\match x a T) /\
   forall a x, bisim (P a x) (Q a x).
No solution.
```

$\boxed{a(x).\tau + a(x) + a(x).[x = a]\tau \not\models [a(x)](\langle\tau\rangle\mathtt{tt} \vee [\tau]\mathtt{ff})}$

```
?= T = taup z /\
   Q = a\x\ plus (in a x\T) (in a x\z) /\
   P = a\x\ plus (Q a x) (in a x\match x a T) /\
   forall a x, sat (P a x) (boxIn a x\disj (diaAct tau tt) (boxAct tau ff)).
No solution.
```

$\boxed{a(x).\tau + a(x) \models [a(x)](\langle\tau\rangle\mathtt{tt} \vee [\tau]\mathtt{ff})}$

```
?= T = taup z /\
   Q = a\x\ plus (in a x\T) (in a x\z) /\
   P = a\x\ plus (Q a x) (in a x\match x a T) /\
   forall a x, sat (Q a x) (boxIn a x\disj (diaAct tau tt) (boxAct tau ff)).
Found a solution:
 P = x1\x2\
     plus (plus (in x1 (x3\ taup z)) (in x1 (x3\ z)))
      (in x1 (x3\ match x3 x1 (taup z)))
 Q = x1\x2\ plus (in x1 (x3\ taup z)) (in x1 (x3\ z))
 T = taup z
More [y] ?
No more solutions (found 1).
```

$\boxed{a(x).\tau + a(x) + a(x).[x = a]\tau \models^L [a(x)]^L(\langle\tau\rangle\mathtt{tt} \vee [\tau]\mathtt{ff})}$

```
?= T = taup z /\
   Q = a\x\ plus (in a x\T) (in a x\z) /\
   P = a\x\ plus (Q a x) (in a x\match x a T) /\
   nabla a x, lsat (a :: x :: nil) (P a x) (boxInL a x\disj (diaAct tau tt) (boxAct tau ff)).
Found a solution:
 P = x1\x2\
     plus (plus (in x1 (x3\ taup z)) (in x1 (x3\ z)))
      (in x1 (x3\ match x3 x1 (taup z)))
 Q = x1\x2\ plus (in x1 (x3\ taup z)) (in x1 (x3\ z))
 T = taup z
More [y] ?
No more solutions (found 1).
```

$\boxed{a(x).\tau + a(x) \models^L [a(x)]^L(\langle\tau\rangle\mathtt{tt} \vee [\tau]\mathtt{ff})}$

```
?= T = taup z /\
   Q = a\x\ plus (in a x\T) (in a x\z) /\
   P = a\x\ plus (Q a x) (in a x\match x a T) /\
   nabla a x, lsat (a :: x :: nil) (Q a x) (boxIn a x\disj (diaAct tau tt) (boxAct tau ff)).
Found a solution:
 P = x1\x2\
     plus (plus (in x1 (x3\ taup z)) (in x1 (x3\ z)))
      (in x1 (x3\ match x3 x1 (taup z)))
 Q = x1\x2\ plus (in x1 (x3\ taup z)) (in x1 (x3\ z))
 T = taup z
More [y] ?
No more solutions (found 1).
```