Extracting Distinguishing Formulae from Bisimulation Tests for Finite Process Calculi

Authors BLINDED

Institutes BLINDED

Abstract. We study a problem of providing a certificate for two non-bisimilar processes from a failed bisimulation test. Such a certificate is helpful for detecting false-negative bugs and also for enhancing user experience of the tools based on process equivalence checking. Our approach is to have the bisimulation test provide a graph that logs the visited pairs of process states during the test, regardless of whether the test succeeds or fails. This graph is a generalization of a bisimulation relation, which is only defined for bisimilar processes, that is when the test succeeds. From a graph with a disproved root node, one can extract a modal logic formula, known as the distinguishing formula, that satisfied by one process but not by the other. Our approach requires little modification to existing bisimulation testing algorithms (or procedures) and is also flexible to be extended for systems with infinite states.

Keywords: bisimulation, process calculus, process algebra, process equivalence, labeled transition systems, modal logic, distinguishing formula, logic programming, tabling, model checking

1 Introduction

We study a problem of providing a certificate for two non-bisimilar processes, assuming a procedure for bisimulation test has already been implemented. The problem with such an assumption is realistic within various tools [TODO cite] equipped with equivalence checking features based on bisimulation. Providing a certificate for failure has two benefits. First, false negatives can be detected by a checker independent from the bisimulation test procedure. Second, users are provided with useful information to analyze why the processes are not equivalent when the test fails, which occurs very frequently in practice. A typical usage scenario would involve (a) defining a *specification* process preferably with a simple structure that clearly describes its desired properties and (b) testing for equivalence of an *instance* process that is closer to an actual implementation against the specification. In practice, it takes many iterations of (a) and (b) to develop a correct specification that abstracts the essence of the system and to refine the instance to match the specification being developed, which amounts to numerous failures before a few successful bisimulation tests.

TODO Our contribution TODO

Our approach requires the bisimulation test to produce a *bisimulation testing graph* (in abbr., *bisim-graph*), which is a generalization of the *bisimulation set* (in abbr., *bisim-set*) (a.k.a., bisimulation relation), so that one can extract a *distinguishing formula* from that graph when the test fails. By definition of the bisim-set, it is natural to assume that bisimulation testing procedures should be able produce a bisim-set ($R \subset S \times S$), which is a set of process state pairs, when the test succeeds. In principle, this set R has enough information to serve as a proof certificate for a pair of bisimilar processes. More generally, to produce a disproof certificate as well, we require that the bisimulation test to produce a bisim-graph $G = (V, E)$ whose vertexes ($V \subset \{\mathtt{p}, \mathtt{d}\} \times (S \times S)$) are process state pairs with an additional tag whether the two process states ($p, q \in S$) are proved ($\mathtt{p}(p, q)$) or disproved ($\mathtt{d}(p, q)$) to be bisimilar and edges are common transitions (or actions) over process state pairs ($E \subset (S \times S) \times A \times (S \times S)$) where $(p, q) \xrightarrow{\pi} (p', q') \in E$ means that both $p \xrightarrow{\pi} p'$ and $q \xrightarrow{\pi} q'$. Intuitively, the bisim-graph $G = (V, E)$ logs the information of process state pairs visited during the bisimulation test along with their transition steps. When the test succeeds, the subset of V consisting of only the proved pairs, ignoring their tags, ($\{(P, Q) \mid \mathtt{p}(p, q) \in V\}$) exactly corresponds to the bisim-set. When the test fails, the bisim-graph provides enough information to construct a *distinguishing formula* $\varphi_{(p,q)}$ such that $p \models \varphi_{(p,q)}$ but $q \not\models \varphi_{(p,q)}$ in a modal logic known to be adequate for characterizing bisimilar processes.

TODO Our method TODO and two processes are bisimulation testing procedure

TODO contributions and how we approach??
Hennessy–Milner Logic (HML) is known to be *adequate* labeled transition systems TODO TODO

TODO paper organization and future work???

2 Bisim-Graph for Finite CCS

Syntax and Semantics of Finite CCS

$$
\begin{array}{ll}
\text{Syntax} & p, q, r ::= 0 \mid \pi.p \mid p + q \mid p|q \mid (\nu a)p \\
& \quad \pi ::= a \mid \bar{a} \mid \tau \\
\text{Semantics} & \quad \tau.p \xrightarrow{\tau} p \\
& \quad a.p \xrightarrow{\downarrow a} p \\
& \quad \bar{a}.p \xrightarrow{\uparrow a} p \\
& \quad p + q \xrightarrow{\pi} p' \quad \text{when} \quad p \xrightarrow{\pi} p' \\
& \quad p + q \xrightarrow{\pi} q' \quad \text{when} \quad q \xrightarrow{\pi} q' \\
& \quad p|q \xrightarrow{\pi} p'|q \quad \text{when} \quad p \xrightarrow{\pi} p' \\
& \quad p|q \xrightarrow{\pi} p|q' \quad \text{when} \quad q \xrightarrow{\pi} q' \\
& \quad p|q \xrightarrow{\tau} p|q' \quad \text{when} \quad p \xrightarrow{\downarrow a} p' \text{ and } q \xrightarrow{\uparrow a} q' \\
& \quad p|q \xrightarrow{\tau} p|q' \quad \text{when} \quad p \xrightarrow{\downarrow a} p' \text{ and } q \xrightarrow{\uparrow a} q' \\
& \quad (\nu a)p \xrightarrow{\tau} (\nu a)p' \quad \text{when} \quad p \xrightarrow{\pi} p' \text{ and } \pi \notin \{\downarrow a, \uparrow a\}
\end{array}
$$

```
Define one : p -> a -> p -> prop
 by one (taup P) tau P
  ; one (inp X P) (dn X) P
  ; one (out X P) (up X) P
  ; one (plus P Q) A R := one P A R
  ; one (plus P Q) A R := one Q A R
  ; one (par P Q) A (par P1 Q) := one P A P1
  ; one (par P Q) A (par P Q1) := one Q A Q1
  ; one (par P Q) tau (par P1 Q1) :=
      exists X, one P (up X) P1 /\ one Q (dn X) Q1
  ; one (par P Q) tau (par P1 Q1) :=
      exists X, one P (dn X) P1 /\ one Q (up X) Q1
  ; one (new P) A (new Q) := nabla x, one (P x) A (Q x)
  .
```

Fig. 1. A Bedwyr definition for the labled semantics of finite CCS

3 Bisimulation Testing Graph

4 Hennessy-Milner Logic

Syntax	$\varphi ::= \texttt{tt} \mid \texttt{ff} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle\pi\rangle\varphi \mid [\pi]\varphi$
Semantics	$p \models \top$
	$p \models \varphi_1 \wedge \varphi_2$ when $p \models \varphi_1$ and $p \models \varphi_2$
	$p \models \varphi_1 \vee \varphi_2$ when $p \models \varphi_1$ or $p \models \varphi_2$
	$p \models \langle\pi\rangle\varphi$ when $\forall q,\ p \xrightarrow{\pi} q$ implies $q \models \varphi$
	$p \models [\pi]\varphi$ when $\exists q,\ p \xrightarrow{\pi} p$ and $q \models \varphi$

```
Define satisfy : p -> o' -> prop
by satisfy P tt := true
 ; satisfy P ff := false
 ; satisfy P (conj A B) := satisfy P A /\ satisfy P B
 ; satisfy P (disj A B) := satisfy P A \/ satisfy P B
 ; satisfy P (dia X A) := exists Q, one P X Q /\ satisfy Q A
 ; satisfy P (box X A) := forall Q, one P X Q -> satisfy Q A
 .
```

5 Finite π-calculus

$$p, q, r ::= 0 \mid \pi.p \mid p + q \mid q|q \mid (\nu a)q$$
$$\pi ::= a(x) \mid \bar{a}\langle b\rangle \mid \tau$$

6 Discussions

TODO

6.1 Systems with Infinite States

papers to cite

7 Related Work

TODO references here A framework for proof certificates in finite state exploration
`http://dx.doi.org/10.4204/EPTCS.186.4`
Algebraic laws for nondeterminism and concurrency
`http://dx.doi.org/10.1145/2455.2460`

Proof Search Specifications of Bisimulation and Modal Logics for the π-calculus
https://arxiv.org/abs/0805.2785

"Modal logics for mobile processes", by R. Milner, J. Parrow and D. Walker Theoretical Computer Science 114 (1993).