Minimal Depth Distinguishing Formulas without Until for Branching Bisimulation

Jan Martens[0000−0003−4797−7735] and Jan Friso Groote[0000−0003−2196−6587]

Eindhoven University of Technology, Eindhoven, The Netherlands
{j.j.m.martens,j.f.groote}@tue.nl

Abstract. In [14] an algorithm for distinguishing formulas for branching bisimulation is proposed. This algorithm has two shortcomings. First, it uses a dedicated until operator, and second, the generated formulas are in no sense minimal. Here we propose a method that generates formulas fitting in the modal mu-calculus, or more precisely, in Hennessy-Milner logic with one regular modality. We provide an algorithm that generates a distinguishing formula that is guaranteed to have minimal depth. Our technical exposition heavily relies on branching apartness, the dual of branching bisimulation.

1 Introduction

Behavioural equivalences are useful to establish whether two systems behave the same. For instance an implementation of a system must behave the same as its specification. Well known behavioural equivalences are strong bisimulation [18] and branching bisimulation [9], but there are many others [8].

When the behaviour of two systems are not equivalent, i.e., when they are apart, it can be very difficult to figure out what the reason is, as in realistic systems it is not uncommon that the difference is only exposed after performing hundreds of equal steps. Fortunately, modal formulas come to the rescue. Following [12] if two systems are not strongly bisimilar, there is a modal formula that is valid in one and not in the other, exemplifying what the reason for inequality is. This formula is called a *distinguishing formula*, or in case of apartness, a *witness*. In [4] an algorithm is given to calculate distinguishing formulas for strong bisimulation.

In order to make a distinguishing formula extra useful it should be minimal in size. Unfortunately, generating a minimal distinguishing formula for strong bisimulation is NP-hard [15], and as a consequence it is NP-hard for many other equivalences as well. But fortunately, it turns out that generating distinguishing formulas that have a minimal depth or have a minimal number of nested negations is computationally tractable [15]. This also holds for the rather weak notion of minimality proposed in [4]. If modal formulas are generated that are not at least minimal depth, they tend to become much larger than needed, and this often renders them effectively unusable

Algorithms or generation schemes for distinguishing formulas for branching bisimulation are proposed in [6,14]. These formulas contain the until operator to explicitly deal with the internal action τ [5]. The until operator is not part of the common mu-calculus [2,11,16]. We want to generate distinguishing formulas that fit the modal mu-calculus, and only require a minimal extension of Hennessy-Milner logic. As the modal mu-calculus is very expressive, the proposed until operators can of course be expressed in the modal mu-calculus, but this makes the formulas unduly complex. Our extension only consists of the regular modality $\langle\tau^*\rangle\phi$ with its normal classical semantic interpretation. Besides that we use the abbreviation $\langle\hat{\tau}\rangle\phi$ standing for $\langle\tau\rangle\phi \vee \phi$, but this is not a real extension. This results in a class of formulas preserved under branching bisimulation with fewer extra modalities than in [17].

In this article we propose an algorithm to generate minimal depth distinguishing formulas for branching bisimulation, or minimal witnesses for branching apartness. The syntax of the formulas fits Hennessy-Milner formulas with the additional use of the regular modality $\langle\tau^*\rangle\phi$, which, unlike the until operator, is part of the standard syntax of the modal mu-calculus as supported by standard behavioural analysis tools [3,16].

2 Preliminaries

Branching bisimilarity is a behavioural equivalence on labelled transition systems that accounts for internal actions. Its dual, branching apartness, formalises that two processes are not branching bisimilar. We use the set of action labels $Act_\tau = Act \cup \{\tau\}$ where Act is an alphabet and $\tau \notin Act$ is the label for internal steps.

Definition 1 (Labelled transition systems). *A labelled transition system (LTS) is given by a 3-tuple $L = (S, Act_\tau, \rightarrow)$ consisting of:*

- *a finite set of states S,*
- *a finite set of action labels Act_τ containing a silent action $\tau \in Act_\tau$, and*
- *a transition relation $\rightarrow \subseteq S \times Act_\tau \times S$.*

Given a transition relation $\rightarrow$, we write $s \xrightarrow{a} s'$ iff $(s, a, s') \in \rightarrow$. We write $s_0 \twoheadrightarrow s_n$ iff there is a silent path from s_0 to s_n, i.e., there is a sequence of states $s_0, \ldots, s_n \in S$, such that $s_0 \xrightarrow{\tau} s_1 \cdots s_{n-1} \xrightarrow{\tau} s_n$. For states $s, s' \in S$ and an action label $a \in Act_\tau$, we write $s \xrightarrow{(a)} s'$ iff either $s \xrightarrow{a} s'$ or $a = \tau$ and $s = s'$, to indicate silent or non-behaviour.

We define branching bisimilarity using branching bisimulation relations in the standard way.

Definition 2. *(Cf. [11, Def. 2.4.2]) Given an LTS $L = (S, Act_\tau, \rightarrow)$. A symmetric relation $R \subseteq S \times S$ is called a branching bisimulation, iff for all sRt and $s \xrightarrow{a} s'$, then either*

1. *$a = \tau$ and $s'Rt$, or*
2. *there are $t', t'' \in S$ such that $t \twoheadrightarrow t' \xrightarrow{a} t''$, sRt', and $s'Rt''$.*

Two states $s, t \in S$ are said to be branching bisimilar, written as $s \leftrightarrow_b t$, iff there is a branching bisimulation R such that sRt.

Next, we define branching apartness $\#$, the dual of branching bisimilarity. Branching apartness is defined as the union of all i-apartness relations.

Definition 3. *Given an LTS $L = (S, Act_\tau, \to)$. We characterise branching apartness $\#_i$ for all $i \in \mathbb{N}$ inductively. We define $\#_0 = \emptyset$, and for each $s, t \in S$ we let $s \#_{i+1} t$ iff $s \#_i t$ or:*

1. *there is a path $s \twoheadrightarrow s' \xrightarrow{a} s''$ such that for all paths $t \twoheadrightarrow t' \xrightarrow{(a)} t''$, either $s' \#_i t'$ ór $s'' \#_i t''$, or*
2. *symmetrically, there is a path $t \twoheadrightarrow t' \xrightarrow{a} t''$ such that for all paths $s \twoheadrightarrow s' \xrightarrow{(a)} s''$, either $t' \#_i s'$ ór $t'' \#_i s''$.*

The relation $\# \subseteq S \times S$, called the branching apartness relation, is defined by $\# = \bigcup_{i \in \mathbb{N}} \#_i$.

As our LTSs are finite, $\#$ is also the fixed point of the chain $\#_0, \#_1, \ldots$. In other words, there is an $i \in \mathbb{N}$ for which $\#_{i+1} = \#_i$ and $\# = \#_i$ for any such i. We first prove that branching apartness is a *proper apartness* relation conform [7].

Theorem 4. *Given an LTS $L = (S, Act, \to)$. For all $k \in \mathbb{N}$ the relation $\#_k$ is a proper apartness relation, i.e. the relation $\#_k$ is:*

- *irreflexive: $\neg(s \#_k s)$ for all $s \in S$,*
- *symmetric: $s \#_k t \implies t \#_k s$ for all $s, t \in S$, and*
- *co-transitive: $s \#_k t \implies (t \#_k u \vee s \#_k u)$ for all $s, t, u \in S$.*

Proof. The only property which is non-trivial is co-transitivity. We prove the property with induction on k. Assume $\#_i$ is a proper apartness relation for some $i \in \mathbb{N}$, and let $s, t \in S$ be two states such that $s \#_{i+1} t$. In order to prove co-transitivity we assume a state $u \in S$ and show $\neg(s \#_{i+1} u) \implies t \#_{i+1} u$.

By definition, and without loss of generality we can assume that there is a path $s \twoheadrightarrow s' \xrightarrow{a} s''$ such that for all $t \twoheadrightarrow t' \xrightarrow{(a)} t''$ it holds that either $s' \#_i t'$ or $s'' \#_i t''$. Assuming $\neg(s \#_{i+1} u)$, we know there is a path $u \twoheadrightarrow u' \xrightarrow{(a)} u''$ such that $\neg(s' \#_i u')$ and $\neg(s'' \#_i u'')$.

From this path we reason towards the conclusion that $t \#_{i+1} u$. We distinguish the following two cases.

- If $a \neq \tau$ or $u' \neq u''$ then $u' \xrightarrow{a} u''$. For any path $t \twoheadrightarrow t' \xrightarrow{(a)} t''$ it holds that $t' \#_i s'$ or $t'' \#_i s''$. Since we assumed that $\neg(s' \#_i u')$ and $\neg(s'' \#_i u'')$ we know by applying the induction hypothesis that $t' \#_i u'$ or $t'' \#_i u''$. This witnesses $t \#_{i+1} u$.
- If $a = \tau$ and $u' = u''$, then $\neg(s' \#_i u')$ and $\neg(s'' \#_i u')$. Now distinguish:
  - If $u \neq u'$ then $u \twoheadrightarrow u_p \xrightarrow{\tau} u'$ for some $u_p \in S$. For any $t \twoheadrightarrow t'$ it holds that $t' \#_i s'$ or $t' \#_i s''$ since $t' \xrightarrow{(\tau)} t'$. By applying the induction hypothesis and since $\neg(s' \#_i u')$ and $\neg(s'' \#_i u')$ it holds that $t' \#_i u'$. This witnesses $t \#_{i+1} u$.
  - If $u = u'$ then also $\neg(s' \#_i u)$. Since $t \twoheadrightarrow t \xrightarrow{(a)} t$ by applying the induction hypothesis $t \#_i u$. By definition this also means $t \#_{i+1} u$.

This covers all cases and establishes that $t \mathrel{\#_{i+1}} u$, which means that $\#_{i+1}$ is co-transitive. Since the base case $\#_0 = \emptyset$ is trivially co-transitive the induction holds. □

Next, we prove that branching apartness $\#$ is the dual of branching bisimilarity.

Theorem 5. *Given an LTS $L = (S, Act_\tau, \rightarrow)$ and states $s, t \in S$. Then*

$$s \mathrel{\#} t \iff s \not\leftrightarrow_b t.$$

The proof of this theorem employs the co-transitivity property of apartness relations, and the following lemma, cf. [1, Lemma 6].

Lemma 6. *Given an LTS $L = (S, Act_\tau, \rightarrow)$, states $s, t \in S$, and a branching bisimulation relation R such that sRt. For all states $s' \in S$ if $s \twoheadrightarrow s'$ then there is a state $t \in S$ such that $t \twoheadrightarrow t'$ and $s'Rt'$.*

Proof. Assume $s \twoheadrightarrow s'$, then there is a path $s_0 \xrightarrow{\tau} \cdots \xrightarrow{\tau} s_n$ such that $s_0 = s$ and $s_n = s'$. We prove by induction that for every $i \in [0, n]$ there is a state $t_i \in S$ such that $t \twoheadrightarrow t_i$ and $s_i R t_i$.

For the base case, $i = 0$, the property trivially holds since $t \twoheadrightarrow t$ and sRt. For the induction step $i+1$ we distinguish two cases conform Definition 2. Since $s_i \xrightarrow{\tau} s_{i+1}$, either

1. $s_{i+1} R t_i$, and since $t_i \twoheadrightarrow t_i$ the property holds, or
2. there are $t', t'' \in S$ such that $t_i \twoheadrightarrow t' \xrightarrow{\tau} t''$, $s_i R t'$ and $s_{i+1} R t''$. So, $t \twoheadrightarrow t''$ and $s_{i+1} R t''$ as had to be shown.

This finishes the induction step and proves the lemma. □

This lemma allows us to prove Theorem 5.

Proof of Theorem 5. We show this in both directions separately.

- $\Rightarrow$ We prove by induction on $i \in \mathbb{N}$ that if $s \mathrel{\#_i} t$ then $s \not\leftrightarrow_b t$. This property trivially holds for $i = 0$. For the inductive case, we assume $s \mathrel{\#_{i+1}} t$. This means there is a path $s \twoheadrightarrow s' \xrightarrow{a} s''$ that witnesses for any $t \twoheadrightarrow t' \xrightarrow{(a)} t''$ either $s' \mathrel{\#_i} t'$ or $s'' \mathrel{\#_i} t''$.
  To arrive at a contradiction we assume that $R \subseteq S \times S$ is a branching bisimulation relation such that sRt. By Lemma 6 there is a $t \twoheadrightarrow t'$ such that $s'Rt'$. We distinguish both cases of Definition 2 with respect to the transition $s' \xrightarrow{a} s''$ and show that both cases lead to a contradiction.
  1. In the first case $a = \tau$ and $s''Rt'$. In this case both $s'Rt'$ and $s''Rt'$. By applying the induction hypothesis, this means $\neg(s' \mathrel{\#_i} t')$, and $\neg(s'' \mathrel{\#_i} t')$ contradicting our assumption since neither $t' \mathrel{\#_i} s'$ or $t'' \mathrel{\#_i} s''$.
  2. In the second case there is a path $t' \twoheadrightarrow \tilde{t}' \xrightarrow{a} t''$ such that $s'R\tilde{t}'$, and $s''Rt''$. This means by the induction hypothesis that $\neg s' \mathrel{\#_i} \tilde{t}'$ and $\neg s'' \mathrel{\#_i} t''$. This also contradicts the assumption.

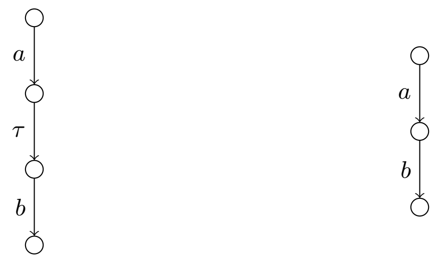

Fig. 1. Two simple branching bisimilar transition systems

We showed that both cases lead to a contradiction which means there is no branching bisimulation relation R such that $(s,t) \in R$. This completes the induction, and hence if $s \# t$ then $s \not\leftrightarrow_b t$.

$\Leftarrow$ We prove this by showing that if $\neg(s \# t)$ then $s \leftrightarrow_b t$. For this purpose assume $\neg(s \# t)$, and consider the relation

$$R = \{\langle u, v\rangle \mid \neg(u \#_{i+1} v)\}$$

for some $i \in \mathbb{N}$ such that $\#_{i+1} = \#_i$.

We show that R is a branching bisimulation relation. Note that this is enough to prove this part of this Theorem, as if $\neg(s \# t)$ then sRt, and as R is a branching bisimulation relation $s \leftrightarrow_b t$.

So, consider a pair of states $u, v \in S$ such that uRv. Given a transition $u \xrightarrow{a} u'$, since $\neg(u \#_{i+1} v)$ and $u \twoheadrightarrow u \xrightarrow{a} u'$, by contraposition of Definition 3, there is a $v \twoheadrightarrow v' \xrightarrow{(a)} v''$ such that both $\neg(u \#_i v')$ and $\neg(u' \#_i v'')$. Since $\#_i = \#_{i+1}$ this implies $\langle u, v'\rangle \in R$ and $\langle u', v''\rangle \in R$. Either $a \neq \tau$ and this is exactly conform item 2 in Definition 2. Otherwise, if $a = \tau$, then either

1. $v' = v''$, then since uRv' and $u'Rv''$, by the fact that R is transitive (Theorem 4) we conclude that $u'Ru$, and since uRv this means $u'Rv$ conform case 1 of Definition 2. Or
2. $v' \neq v''$, in which case $v' \xrightarrow{\tau} v''$, conform case 2 of Definition 2.

The relation R is clearly symmetric and hence, this completes the proof that R is a branching bisimulation relation. □

3 A Hennessy-Milner Theorem

We want to exemplify states that are not branching bisimilar by modal formulas. For strong bisimulation this can be performed using formulas in Hennessy-Milner logic [12]. A nice and natural property is that the distinguishing formulas that we generate stem from a class of formulas that is respected by branching bisimulation. The modality $\langle\tau\rangle\phi$ can therefore not be used, as $\langle a\rangle\langle\tau\rangle true$ is valid in the transition system at the left in Figure 1 but not in the branching bisimilar transition system at the right. So, some adaptation of Hennessy-Milner logic is required.

A possible solution is provided by allowing dedicated until operators [5,14]. But we do not like this, because until operators are not common within the modal mu-calculus. However, regular formulas are commonly used within the modal mu-calculus [3,16] and therefore we decide to use $\langle\tau^*\rangle\phi$, which is equivalent to $\mu X.\langle\tau\rangle X \vee \phi$. This formula expresses that zero or more internal actions τ can be performed after which ϕ holds.

This leads to the following logic $\mathtt{HML}_\tau$.

Definition 7. *($\mathtt{HML}_\tau$) Hennessy-Milner Logic with silent closure is defined over the following syntax:*

$$\phi ::= \mathit{true} \mid \phi \wedge \phi \mid \neg\phi \mid \langle\tau^*\rangle\phi \mid \langle a\rangle\phi,$$

where $a \in Act_\tau$. We write $\mathtt{HML}_\tau$ for the set of all formulas over this syntax.

Given an LTS $L = (S, Act_\tau, \rightarrow)$, the semantics $[\![\phi]\!] \subseteq S$ of a formula $\phi \in \mathtt{HML}_\tau$ is the set of states where ϕ holds, defined by:

$$\begin{aligned}
[\![\mathit{true}]\!] &= S,\\
[\![\phi_1 \wedge \phi_2]\!] &= [\![\phi_1]\!] \cap [\![\phi_2]\!],\\
[\![\neg\phi]\!] &= S \setminus [\![\phi]\!],\\
[\![\langle a\rangle\phi]\!] &= \{s \mid s \xrightarrow{a} s' \text{ such that } s' \in [\![\phi]\!]\}, \text{ and}\\
[\![\langle\tau^*\rangle\phi]\!] &= \{s \mid s \twoheadrightarrow s' \text{ such that } s' \in [\![\phi]\!]\}.
\end{aligned}$$

Other modalities can be expressed in this syntax, e.g., $[a]\phi := \neg\langle a\rangle\neg\phi$ and $\phi_1 \vee \phi_2 := \neg(\neg\phi_1 \wedge \neg\phi_2)$.

Given a formula $\phi \in \mathtt{HML}_\tau$, and two states $s, t \in S$, we write $s \sim_\phi t$, iff $s \in [\![\phi]\!] \iff t \in [\![\phi]\!]$. For a set of formulas $\Phi \subseteq \mathtt{HML}_\tau$ we use the notation $s \sim_\Phi t$ iff $s \sim_\phi t$ for all $\phi \in \Phi$.

If formulas of the shape of $\langle\tau\rangle\phi$ are allowed, branching bisimilar states can be distinguished. Therefore, we instead use the modality $\langle\hat{\tau}\rangle$ which is defined as $\langle\hat{\tau}\rangle\phi := \langle\tau\rangle\phi \vee \phi$. Using this, we define a syntactic restriction in order to characterize exactly branching bisimilar states in a Hennessy-Milner-like theorem.

Definition 8. *For every $i \in \mathbb{N}$, we define the set $\mathcal{F}^i_{br} \subseteq \mathtt{HML}_\tau$ inductively. As base case we take $\mathcal{F}^0_{br} = \{\mathit{true}\}$, and we define $\mathcal{F}^{i+1}_{br}$ to be the smallest set containing:*

- *$\phi \in \mathcal{F}^{i+1}_{br}$, if $\phi \in \mathcal{F}^i_{br}$,*
- *$\neg\phi \in \mathcal{F}^{i+1}_{br}$, if $\phi \in \mathcal{F}^{i+1}_{br}$,*
- *$\phi_1 \wedge \phi_2 \in \mathcal{F}^{i+1}_{br}$, if $\phi_1, \phi_2 \in \mathcal{F}^{i+1}_{br}$, and*
- *$\langle\tau^*\rangle(\langle a\rangle\phi_1 \wedge \phi_2) \in \mathcal{F}^{i+1}_{br}$, where $a \in Act \cup \{\hat{\tau}\}$, and $\phi_1, \phi_2 \in \mathcal{F}^i_{br}$.*

Example 9. In Figure 2 two pairs of labelled transition systems are depicted that are not branching bisimilar, but the two LTSs at the left are weakly bisimilar [18]. The formula

$$\langle\tau^*\rangle\langle a\rangle\neg\langle\tau^*\rangle\langle b\rangle\mathit{true}$$

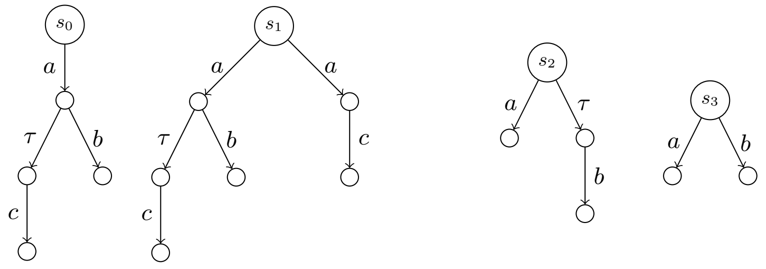

Fig. 2. Two pairs of non branching bisimilar transition systems

is valid in s_0 but not in s_1, whereas the formula

$$\langle\tau^*\rangle(\langle b\rangle \mathit{true} \wedge \neg\langle\tau^*\rangle\langle a\rangle \mathit{true})$$

is valid in s_2 but not in s_3. We simplified the distinguishing formulas using $\phi \wedge \mathit{true} = \phi$. Before this simplification they both fit in $\mathcal{F}^2_{br}$.

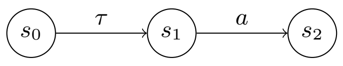

Fig. 3. The formula $\langle\tau^*\rangle(\langle\tau\rangle \mathit{true} \wedge \mathit{true})$ holds in s_0 but not in s_1.

Example 10. As already indicated above, we explicitly disallow the $\langle\tau\rangle$ modality and instead use the modality $\langle\hat{\tau}\rangle$. If formulas of the shape of $\langle\tau^*\rangle(\langle\tau\rangle\phi_1 \wedge \phi_2)$ are allowed, branching bisimilar states can be distinguished.

Consider the LTS $\mathcal{A}$ shown in Figure 3. In this example we see that $s_0 \leftrightarrows_b s_1$. The formula $\phi = \langle\tau^*\rangle(\langle\tau\rangle \mathit{true} \wedge \mathit{true})$ is a distinguishing formula for s_0 and s_1, since $s_0 \in [\![\phi]\!]$ and $s_1 \notin [\![\phi]\!]$. After restricting to formulas in $\mathcal{F}^i_{br}$ only allowing $\langle\hat{\tau}\rangle$ modalities, there is no distinguishing formula. The modified formula $\phi' = \langle\tau^*\rangle(\langle\hat{\tau}\rangle \mathit{true} \wedge \mathit{true})$ does not distinguish s_0 and s_1.

Lemma 11. *Let $L = (S, Act_\tau, \rightarrow)$ be an LTS, $s, t \in S$ two states, and $i \in \mathbb{N}$. If $s \not\sim_{\mathcal{F}^i_{br}} t$, then there is a formula $\phi = \langle\tau^*\rangle(\langle a\rangle\phi_1 \wedge \phi_2) \in \mathcal{F}^i_{br}$ such that $s \not\sim_\phi t$.*

Proof. As $s \not\sim_{\mathcal{F}^i_{br}} t$, there is a formula $\phi \in \mathcal{F}^i_{br}$ such that $s \not\sim_\phi t$. Assume there is no formula smaller than ϕ with the same property.

– It is not possible that $\phi = \mathit{true}$ as it cannot distinguish s and t.
– If $\phi = \neg\phi'$ for some $\phi' \in \mathcal{F}^i_{br}$, then also $s \not\sim_{\phi'} t$, and so, ϕ' distinguishes s and t meaning that ϕ was not smallest.
– If $\phi = \phi_1 \wedge \phi_2$, then at least $s \not\sim_{\phi_1} t$ or $s \not\sim_{\phi_2} t$. In either case ϕ was not the smallest formula to distinguish s and t.

– The only option that remains is that $\phi = \langle\tau^*\rangle(\langle a\rangle\phi_1 \wedge \phi_2)$.

$\square$

Theorem 12. *Given an LTS $L = (S, Act_\tau, \rightarrow)$, and states $s, t \in S$. For every $i \in \mathbb{N}$:*

$$s \mathrel{\#_i} t \iff s \not\sim_{\mathcal{F}^i_{br}} t.$$

Proof. We prove this property with induction. The property holds trivially for $i = 0$. Assuming the property holds for some $i \in \mathbb{N}$, we show both directions for $i + 1$.

($\Rightarrow$) Under the assumption $s \mathrel{\#_{i+1}} t$, we build a formula $\phi_{dist} \in \mathcal{F}^{i+1}_{br}$ such that $s \not\sim_{\phi_{dist}} t$. By definition of $s \mathrel{\#_{i+1}} t$, we can without loss of generality assume that there is a path $s \twoheadrightarrow s' \xrightarrow{a} s''$ such that for all $t \twoheadrightarrow t' \xrightarrow{(a)} t''$ it holds that $s' \mathrel{\#_i} t'$ or $s'' \mathrel{\#_i} t''$. Consider the sets of states:

$$T = \{t'' \mid t \twoheadrightarrow t' \xrightarrow{(a)} t'' \text{ where } t'' \mathrel{\#_i} s''\} \text{ and}$$
$$T_\tau = \{t' \mid t \twoheadrightarrow t' \xrightarrow{(a)} t'' \text{ where } t' \mathrel{\#_i} s'\}.$$

By the induction hypothesis there is for each $u \in T$ a formula $\phi_u \in \mathcal{F}^i_{br}$ such that $s'' \not\sim_{\phi_u} t''$. Define $\phi'_u = \phi_u$ if $s'' \in [\![\phi_u]\!]$, and $\phi'_u = \neg\phi_u$ otherwise. Now by construction, $\phi'_u \in \mathcal{F}^i_{br}$ and also $s'' \in [\![\phi'_u]\!]$ and $t'' \notin [\![\phi'_u]\!]$. We define $\Phi_T = \bigwedge_{u \in T} \phi'_u$. Note that $\Phi_T \in \mathcal{F}^i_{br}$, $s'' \in [\![\Phi_T]\!]$ and $[\![\Phi_T]\!] \cap T = \emptyset$.
Similarly, for the set T_τ we construct the formula

$$\Phi_{T_\tau} = \bigwedge_{u \in T_\tau, s' \in [\![\phi_u]\!], u \notin [\![\phi_u]\!]} \phi_u,$$

which is the conjunction containing a formula $\phi_u \in \mathcal{F}^i_{br}$ for each $u \in T_\tau$ such that $s' \in [\![\phi_u]\!]$ and $u \notin [\![\phi_u]\!]$. By construction, $\phi_{T_\tau} \in \mathcal{F}^i_{br}$, while $s' \in [\![\Phi_{T_\tau}]\!]$ and $[\![\Phi_{T_\tau}]\!] \cap T_\tau = \emptyset$.
Using the constructed conjuncts Φ_{T_τ} and Φ_T we construct $\phi_{dist} = \langle\tau^*\rangle\Phi$ as follows:

$$\phi_{dist} = \begin{cases} \langle\tau^*\rangle(\langle\hat{\tau}\rangle\Phi_T \wedge \Phi_{T_\tau}) & \text{if } a = \tau, \\ \langle\tau^*\rangle(\langle a\rangle\Phi_T \wedge \Phi_{T_\tau}) & \text{otherwise.} \end{cases}$$

Now we confirm that $s \not\sim_{\phi_{dist}} t$. The path $s \twoheadrightarrow s' \xrightarrow{a} s''$ witnesses that $s \in [\![\phi_{dist}]\!]$, since $s' \in [\![\Phi_{T_\tau}]\!]$ and $s'' \in [\![\Phi_T]\!]$. To see that $t \notin [\![\phi_{dist}]\!]$ we must understand, given a path $t \twoheadrightarrow t'$, that $t' \notin [\![\Phi]\!]$. To do so we case distinguish on whether $t' \mathrel{\#_i} s'$:

1. In the first case if $t' \mathrel{\#_i} s'$, then $t' \in T_\tau$ and $t' \notin \Phi$.
2. In the second case, necessarily $t'' \mathrel{\#_i} s''$ for all $t' \xrightarrow{(a)} t''$, and thus $t'' \notin [\![\Phi_T]\!]$. If $a \neq \tau$ this means that $t' \notin [\![\langle a\rangle\Phi_T]\!]$. If $a = \tau$ then by definition $t' \xrightarrow{(\tau)} t'$ and thus also $t' \notin [\![\Phi_T]\!]$. Hence, also in this case $t' \notin [\![\langle\hat{\tau}\rangle\Phi_T]\!]$.

This covers that in both cases $t' \notin \Phi$, and we conclude that $t \notin [\![\phi_{dist}]\!]$.

($\Leftarrow$) Assume $s \not\sim_{\mathcal{F}_{br}^{i+1}} t$ for some $s, t \in S$. By Lemma 11 there is a formula $\phi = \langle \tau^* \rangle (\langle a \rangle \phi_1 \wedge \phi_2)$ for some $\phi_1, \phi_2 \in \mathcal{F}_{br}^i$ and $a \in Act \cup \{\hat{\tau}\}$, such that $s \not\sim_\phi t$. Since s and t are chosen arbitrarily, we can safely assume that $s \in [\![\phi]\!]$ and $t \notin [\![\phi]\!]$. Since $s \in [\![\phi]\!]$, there is a path $s \twoheadrightarrow s'$ such that $s' \in [\![\langle a \rangle \phi_1 \wedge \phi_2]\!]$. We distinguish on whether $a \in Act$ or $a = \hat{\tau}$.

- In the first case if $a \in Act$, there is a transition $s' \xrightarrow{a} s''$ such that $s'' \in [\![\phi_1]\!]$. We show that the path $s \twoheadrightarrow s' \xrightarrow{a} s''$ witnesses $s \,\#_{i+1}\, t$. Since $t \notin [\![\phi]\!]$, we know that for all $t \twoheadrightarrow t'$ it holds that $t' \notin [\![\langle a \rangle \phi_1 \wedge \phi_2]\!]$. Hence, if $t \twoheadrightarrow t' \xrightarrow{(a)} t''$ either $t'' \notin [\![\phi_1]\!]$ or $t' \notin [\![\phi_2]\!]$. We apply our induction hypothesis on $\phi_1, \phi_2 \in \mathcal{F}_{br}^i$ and derive that either $t' \,\#_i\, s'$ or $t'' \,\#_i\, s''$.
- In the other case if $a = \hat{\tau}$ either $s' \xrightarrow{\tau} s''$ such that $s'' \in [\![\phi_1]\!]$ or $s' \in [\![\phi_1]\!]$. In the first case the path $s \twoheadrightarrow s' \xrightarrow{\tau} s''$ witnesses $s \,\#_{i+1}\, t$ as in the case above. In the latter case, when $s' \in [\![\phi_1]\!]$, then since $t' \notin [\![\phi_1]\!]$ or $t' \notin [\![\phi_2]\!]$, by induction $s' \,\#_i\, t'$. If $s = s'$ this means $s \,\#_i\, t$ and if $s \neq s'$, then there is a path $s \twoheadrightarrow s_p \xrightarrow{\tau} s'$ which witnesses $s \,\#_{i+1}\, t$.

This completes the inductive case, and proves the property. □

From Theorem 12 and 5 it follows that if two states are not branching bisimilar there is a distinguishing formula in Hennessy-Milner logic with silent closure.

Corollary 13. *Let $L = (S, Act_\tau, S)$ be a labelled transition system. Then for any two states $s, t \in S$:*

$$s \not\leftrightarrow_b t \iff s \not\sim_\phi t \text{ for some } \phi \in \mathcal{F}_{br}^i \text{ and } i \in \mathbb{N}.$$

4 Computing minimal depth distinguishing formulas.

The proof of Theorem 12 explicitly constructs a distinguishing formula in $\texttt{HML}_\tau$. Based on this construction we introduce an algorithm that computes a minimal depth distinguishing $\texttt{HML}_\tau$ formula, listed as Algorithm 1.

The algorithm uses two auxiliary functions. First, it requires the function $\Delta : S \times S \to \mathbb{N} \cup \{\infty\}$, defined as:

$$\Delta(s,t) = \begin{cases} i & \text{if } s \,\#_i\, t \text{ and } \neg(s \,\#_{i-1}\, t), \\ \infty & \text{otherwise.} \end{cases}$$

Second, it uses the function δ_i to select suitable distinguishing observations, defined as:

$$\delta_i(s,t) = \{(a, s', s'') \mid s \twoheadrightarrow s' \xrightarrow{a} s'' \text{and for all } t \twoheadrightarrow t' \xrightarrow{(a)} t'' \\ \text{it holds that } t' \,\#_{i-1}\, s' \text{ or } t'' \,\#_{i-1}\, s''\}.$$

Given an input $s, t \in S$ such that $s \,\#_i\, t$, the algorithm constructs a distinguishing formula as follows. First a pair $(a, s', s'') \in \delta_i(s,t)$ is selected. This pair

```
input  : A state $s \in S$, and a set $T \subseteq S$ such that $s \# t$ for all $t \in T$.
output: A formula $\phi \in \mathcal{F}_{br}$ such that $s \in [\![\phi]\!]$ and $T \cap [\![\phi]\!] = \emptyset$.
Function Dist($s, T$) is
    /* Return a formula $\Phi$ such that $s \in [\![\Phi]\!]$ and $T \cap [\![\Phi]\!] = \emptyset$.  */
    while $T \neq \emptyset$ do
        Select $t_{max} \in T$ such that $\Delta(s, t_{max}) \geq \Delta(s, t')$ for all $t' \in T$;
        $\phi_{t_{max}} := \phi(s', t_{max})$;
        $\Phi := \Phi \wedge \phi_{t_{max}}$;
        $T := T \cap [\![\phi_{t_{max}}]\!]$;
    end
    return $\Phi$;
end
input  : Two states $s, t \in S$ such that $s \# t$.
output: A formula $\phi \in \mathcal{F}_{br}$ such that $s \in [\![\phi]\!]$ and $t \notin [\![\phi]\!]$.
Function $\phi(s, t)$ is
    $i := \Delta(s, t)$;
    if $\delta_i(s, t) = \emptyset$ then
        return $\neg\, \phi(t, s)$
    Select $(a, s', s'') \in \delta_i(s, t)$;
    $\hat{a} = a$;
    if $a = \tau$ then
        $\hat{a} := \hat{\tau}$;
    $T_\tau := \{t' \mid t \twoheadrightarrow t'\}$;
    $T := \{t'' \mid t' \in T_\tau,\ t' \xrightarrow{(a)} t''$ and $t'' \#_{i-1} s''\}$;
    $\Phi_T :=$ Dist($s'', T$);
    $T_\tau := T_\tau \cap [\![\langle \hat{a} \rangle \Phi_T]\!]$;
    $\Phi_{T_\tau} :=$ Dist($s', T_\tau$);
    return $\langle \tau^* \rangle\, (\langle \hat{a} \rangle \Phi_T \wedge \Phi_{T_\tau})$;
end
```

Algorithm 1: Minimal-depth distinguishing witnesses.

encodes a path $s \twoheadrightarrow s' \xrightarrow{a} s''$ that witnesses the properties in Definition 3. If this path does not exist then for t such a path necessarily exists, and the negation is returned.

For all states $t', t'' \in S$ such that $t \twoheadrightarrow t' \xrightarrow{a} t''$, the algorithm recursively computes a distinguishing formula that distinguishes s'' from t'', or if that is not possible s' from t'. The conjunction of all these formulas is returned and guarantees by construction that $s \in [\![\phi(s,t)]\!]$ and $t \notin [\![\phi(s,t)]\!]$.

By using techniques from dynamic programming we know that this algorithm has polynomial runtime. The function is called at most once for every combination of states.

4.1 Implementation details & example

We implemented this method. We first compute the LTS modulo branching bisimulation with the more efficient algorithm introduced in [10,13]. On this reduced state space we compute the relations $\#_i$ for every $i \in \mathbb{N}$ by a naive partition refinement algorithm. In this section we introduce this simple partition refinement algorithm.

A partition $\pi \subseteq 2^S$ is a disjoint cover of S, i.e. every element $B \in \pi$ is non-empty, i.e., $B \neq \emptyset$, and for all $B' \in \pi$ it holds that $B' \cap B = \emptyset$ or $B' = B$. A partition π naturally describes an apartness relation $\sharp_\pi \subseteq S \times S$:

$$\sharp_\pi = \{(s,t) \mid \forall B \in \pi . s \notin B \vee t \notin B\}.$$

Given an LTS $L = (S, Act_\tau, \rightarrow)$, an action $a \in Act_\tau$ and two subsets $B, B' \subseteq S$ we define the set $split(B, a, B')$ as all states of S that can follow a silent path to a state in B which reaches B' with an a action:

$$split(B, a, B') = \{s \mid \exists s' \in B, s'' \in B' . s \twoheadrightarrow s' \xrightarrow{a} s'' \text{ such that } a \neq \tau \text{ or } B \neq B'\}.$$

A partition π' is a stable refinement of the partition π iff for every block $C \in \pi'$ it holds that

$$C \subseteq split(B, a, B') \text{ or } C \cap split(B, a, B') = \emptyset$$

for all actions $a \in Act_\tau$ and blocks $B, B' \in \pi$.

We establish the correctness of our partition refinement with the following theorem.

Theorem 14. *Given an LTS $L = (S, Act_\tau, \rightarrow)$, a partition π_i such that $\sharp_{\pi_i} = \#_i$ for some $i \in \mathbb{N}$, and states $s, t \in S$ such that $\neg(s \#_i t)$. It holds that $s \#_{i+1} t$ if and only if there are blocks $B, B' \in \pi$ and an action $a \in Act_\tau$ such that $s \in split(B, a, B') \iff t \notin split(B, a, B')$.*

Proof. We prove this in both directions separately.

⇒ Assume $s \#_{i+1} t$. Additionally, without loss of generality assume there is a path $s \twoheadrightarrow s' \xrightarrow{a} s''$ such that for all $t \twoheadrightarrow t' \xrightarrow{(a)} t''$ either $s' \#_i t'$ or $s'' \#_i t''$. We distinguish two cases:
  - In the first case assume $a \neq \tau$ or $s' \#_i s''$. Then by definition $s \in split(B, a, B')$ where $B, B' \in \pi$ are the blocks such that $s' \in B$ and $s'' \in B'$.
    Since we know $t' \notin B$ or $t'' \notin B'$, it follows that $t \notin split(B, a, B')$.
  - For the remaining case assume $a = \tau$ and $\neg(s' \#_i s'')$. In this case the path $s \twoheadrightarrow s' \xrightarrow{\tau} s''$ does not directly give a pair of blocks B, B' that witnesses the split. In order to obtain this pair we first show that $s \#_i s'$. The path $t \twoheadrightarrow t \xrightarrow{(a)} t$ witnesses $t \#_i s'$, and since we assumed $\neg(s \#_i t)$ by co-transitivity $s' \#_i s$.
    Now, there is a path $s \twoheadrightarrow u \xrightarrow{\tau} u' \twoheadrightarrow s' \xrightarrow{\tau} s''$ such that $\neg(u' \#_i s')$ and $u \#_i u'$. For the blocks $B, B' \in \pi_i$ such that $u \in B$ and $u' \in B'$ it holds that $s \in split(B, \tau, B')$ and $t \notin split(B, \tau, B')$ similar to the first case of this case distinction.

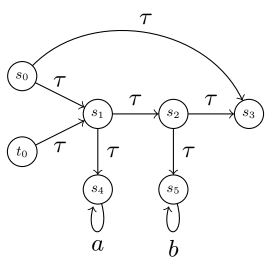

Fig. 4. The LTS $L = (\{s_0, s_1, s_2, s_3, s_4, s_5, t_0\}, \{a, b, \tau\}, \to)$ in which s_0 and t_0 are branching apart.

$\Leftarrow$ Assume $s \in \mathit{split}(B, a, B')$ and $t \notin \mathit{split}(B, a, B')$ for some $B, B' \in \pi_i$ and $a \in \mathit{Act}_\tau$. Let $s \twoheadrightarrow s' \xrightarrow{a} s''$ be the path that witnesses $s \in \mathit{split}(B, a, B')$. Let $t \twoheadrightarrow t' \xrightarrow{(a)} t''$ be a path. Since $t \notin \mathit{split}(B, a, B')$, either $t' \notin B$ or $t'' \notin B'$. Distinguish the following cases,

- $a = \tau$ and $\neg(t' \,\#_i\, t'')$. Then since $B \neq B'$, either $t' \,\#_i\, s'$ or $t'' \,\#_i\, s''$.
- otherwise if $a \neq \tau$ or $t' \,\#_i\, t''$ then since $t \notin \mathit{split}(B, a, B')$ either $t' \notin B$ or $t'' \notin B'$. In other words $t' \,\#_i\, s'$ or $t'' \,\#_i\, s''$.

So, in both cases we can conclude $s \,\#_{i+1}\, t$, which had to be shown. □

Example 15. In this example we apply our algorithm to the LTS in Figure 4. In this LTS the distinguishing behaviour of the states s_0 and t_0 is a τ-action. This results in a bit more complicated distinguishing formula. First we compute the partitions π_0, π_1, π_2 such that $\sharp_{\pi_i} = \#_i$.

$$\pi_0 = \{\{t_0, s_0, s_1, s_2, s_3, s_4, s_5\}\}$$
$$\pi_1 = \{\{t_0, s_0, s_1\}, \{s_2, s_5\}, \{s_4\}, \{s_3\}\}$$
$$\pi_2 = \{\{t_0, s_1\}, \{s_0\}, \{s_2\}, \{s_5\}, \{s_4\}, \{s_3\}\}$$

When computing $\phi(s_0, t_0)$, a path in $\delta_2(s_0, t_0)$ is computed. The path from s_0 that witnesses $s_0 \,\#_2\, t_0$ is $s_0 \twoheadrightarrow s_0 \xrightarrow{\tau} s_3$. Now indeed, for all paths $t_0 \twoheadrightarrow t' \xrightarrow{(\tau)} t''$ it holds that either $s_0 \,\#_1\, t'$ or $s_3 \,\#_1\, t''$. The algorithm computes the sets of states reachable by t_0 that can be distinguished from s_3, which is $T = \{t_0, s_1, s_2, s_4, s_5\}$, which is captured by the formulas $\phi_1 = \phi(s_3, s_4) \wedge \phi(s_3, s_5)$. Now compute the set $T_\tau \cap [\![\langle\hat{\tau}\rangle\phi_1]\!] = \{s_2, s_3\}$, which is distinguished by the formula $\phi(s_0, s_2) = \phi(s_0, s_3) = \langle\tau^*\rangle\langle b\rangle \mathit{true}$. Hence, the formula computed results in:

$$\phi(s_0, t_0) = \langle\tau^*\rangle(\langle\hat{\tau}\rangle\phi_1 \wedge \phi_2)$$
$$\phi_1 = \phi(s_3, s_4) \wedge \phi(s_3, s_5) = \neg\langle\tau^*\rangle\langle a\rangle \mathit{true} \wedge \neg\langle\tau^*\rangle\langle b\rangle \mathit{true}$$
$$\phi_2 = \phi(s_0, s_3) = \langle\tau^*\rangle\langle b\rangle \mathit{true}.$$

A natural question that could arise is whether $\langle\hat{\tau}\rangle$ is required. In other words, whether there is a logic without $\langle\hat{\tau}\rangle\phi$ and with $\langle\tau^*\rangle\phi$ that characterises branching bisimilarity.

To make this statement formal we consider $\texttt{HML}_{\tau^*}$ containing the formulas over the syntax:

$$\psi ::= \mathit{true} \mid \psi \wedge \psi \mid \neg\psi \mid \langle\tau^*\rangle\psi \mid \langle a\rangle\psi,$$

where $a \in \mathit{Act}$ and $a \neq \tau$. We prove that formulas over this logic, or any subset thereof, does not characterise branching bisimilarity. This is witnessed by the fact that it cannot distiqnguish the non branching bisimilar states s_0 and t_0 in Figure 4.

Theorem 16. *Given the LTS L shown in figure 4, for all formulas $\psi \in \texttt{HML}_{\tau^*}$,*

$$s_0 \in [\![\psi]\!] \iff t_0 \in [\![\psi]\!].$$

Proof. We prove this with structural induction on the formula. For the base case if $\psi = \mathit{true}$ this is trivial. For the induction assume $\psi \in \texttt{HML}_{\tau^*}$ and the property holds for all smaller formulas $\psi_1 \in \texttt{HML}_{\tau^*}$. We show that for ψ it holds that $s_0 \in [\![\psi]\!] \iff t_0 \in [\![\psi]\!]$ by distinguishing on the shape of ψ. The cases $\psi = \psi_1 \wedge \psi_2$ and $\psi = \neg\psi_1$ follow directly from the induction hypothesis.

If $\psi = \langle a\rangle\psi_1$ then since $a \neq \tau$ it follows that $s_0 \notin [\![\psi]\!]$ and $t_0 \notin [\![\psi]\!]$. Which leaves the only interesting case when $\psi = \langle\tau^*\rangle\psi_1$. To show $s_0 \in [\![\psi]\!] \implies t_0 \in [\![\psi]\!]$ assume $s_0 \in [\![\psi]\!]$. This means there is a path $s \twoheadrightarrow s'$ such that $s' \in \psi_1$. We distinguish on whether $s' = s$.

- In the case that $s' \neq s_0$ there is also a path $t \twoheadrightarrow s'$ and thus $t \in [\![\psi]\!]$.
- If it is the case that $s' = s_0$ then by induction $t_0 \in [\![\psi_1]\!]$, which means since $t_0 \twoheadrightarrow t_0$ also $t_0 \in [\![\psi]\!]$.

This covers all the cases and proves $s_0 \in [\![\psi]\!] \implies t_0 \in [\![\psi]\!]$. The proof for the other direction is identical, and completes the proof of the theorem. □

References

1. A.A. Basten. Branching bisimilarity is an equivalence indeed! *Information Processing Letters*, 58(3):141–147, 1996.
2. J.C. Bradfield and C. Stirling. Modal mu-calculi. In P. Blackburn, J.F.A.K. van Benthem, and F. Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in logic and practical reasoning*, pages 721–756. North-Holland, 2007. `doi:10.1016/s1570-2464(07)80015-2`.
3. O. Bunte, J.F. Groote, J.J.A. Keiren, M. Laveaux, T. Neele, E.P. de Vink, W. Wesselink, A.J. Wijs, and T.A.C. Willemse. The mCRL2 toolset for analysing concurrent systems - improvements in expressivity and usability. In *Tools and Algorithms for the Construction and Analysis of Systems - 25th International Conference (TACAS 2019), Proceedings, Part II*, pages 21–39, 2019. `doi:10.1007/978-3-030-17465-1\_2`.
4. R. Cleaveland. On automatically explaining bisimulation inequivalence. In E.M. Clarke and R.P. Kurshan, editors, *Proc. Computer-Aided Verification (CAV 1990)*, pages 364–372, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg. `doi:10.1007/BFb0023750`.

5. R. De Nicola and F.W. Vaandrager. Three logics for branching bisimulation. *J. ACM*, 42(2):458–487, mar 1995. `doi:10.1145/201019.201032`.
6. J.H. Geuvers. Apartness and distinguishing formulas in Hennessy-Milner logic. In *A Journey from Process Algebra via Timed Automata to Model Learning: Essays Dedicated to Frits Vaandrager on the Occasion of His 60th Birthday*, pages 266–282. Springer, 2022.
7. J.H. Geuvers and B. Jacobs. Relating apartness and bisimulation. *Log. Methods Comput. Sci.*, 17(3), 2021. `doi:10.46298/lmcs-17(3:15)2021`.
8. R.J. van Glabbeek. The linear time - branching time spectrum I. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, *Handbook of Process Algebra*, pages 3–99. North-Holland / Elsevier, 2001. `doi:10.1016/b978-044482830-9/50019-9`.
9. R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics. *J. ACM*, 43(3):555–600, 1996. `doi:10.1145/233551.233556`.
10. J.F. Groote, D.N. Jansen, J.J.A. Keiren, and A.J. Wijs. An $O(m\mathrm{log}n)$ algorithm for computing stuttering equivalence and branching bisimulation. *ACM Trans. Comput. Log.*, 18(2):13:1–13:34, 2017. `doi:10.1145/3060140`.
11. J.F. Groote and M.R. Mousavi. *Modeling and Analysis of Communicating Systems*. MIT Press, 2014. URL: `https://mitpress.mit.edu/books/modeling-and-analysis-communicating-systems`.
12. M. Hennessy and R. Milner. On observing nondeterminism and concurrency. In Jaco de Bakker and Jan van Leeuwen, editors, *Automata, Languages and Programming (ICALP '1980)*, page 299–309, Berlin, Heidelberg, 1980. Springer-Verlag. `doi:10.5555/646234.758793`.
13. D.N. Jansen, J.F. Groote, J.J.A. Keiren, and A.J. Wijs. An $O(m \log n)$ algorithm for branching bisimilarity on labelled transition systems. In Armin Biere and David Parker, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 26th International Conference, TACAS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings, Part II*, volume 12079 of *Lecture Notes in Computer Science*, pages 3–20. Springer, 2020. `doi:10.1007/978-3-030-45237-7\_1`.
14. H.P. Korver. Computing distinguishing formulas for branching bisimulation. In *Computer Aided Verification: 3rd International Workshop, CAV'91 Aalborg, Denmark, July 1–4, 1991 Proceedings 3*, pages 13–23. Springer, 1992.
15. J.J.M. Martens and J.F. Groote. Computing minimal distinguishing Hennessy-Milner formulas is NP-hard, but variants are tractable. In G.A. Pérez and J.-F. Raskin, editors, *34th International Conference on Concurrency Theory, CONCUR 2023, September 18-23, 2023, Antwerp, Belgium*, volume 279 of *LIPIcs*, pages 32:1–32:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. `doi:10.4230/LIPIcs.CONCUR.2023.32`.
16. R. Mateescu and M. Sighireanu. Efficient on-the-fly model-checking for regular alternation-free mu-calculus. *Sci. Comput. Program.*, 46(3):255–281, 2003. `doi:10.1016/S0167-6423(02)00094-1`.
17. R. Mateescu and A.J. Wijs. Property-dependent reductions adequate with divergence-sensitive branching bisimilarity. *Sci. Comput. Program.*, 96:354–376, 2014. `doi:10.1016/j.scico.2014.04.004`.
18. R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, Cham, 1980. `doi:10.1007/3-540-10235-3`.