Behavioural Equivalences on Finite-State Systems are PTIME-hard*

Zdeněk Sawa and Petr Jančar

Centre for Applied Cybernetics,
Dept of Computer Science, Technical University of Ostrava (FEI VŠB-TU)
17. listopadu 15, CZ-708 33 Ostrava, Czech Republic
zdenek.sawa@vsb.cz, petr.jancar@vsb.cz

Abstract

The paper shows a LOGSPACE-reduction from the boolean circuit value problem which demonstrates that any relation subsuming bisimilarity and being subsumed by trace preorder (i.e., language inclusion) is PTIME-hard, even for finite acyclic labelled transition systems. This reproves and substantially extends the result of Balcázar, Gabarró and Sántha (1992) for bisimilarity.

Keywords: verification, finite transition systems, bisimulation equivalence, trace equivalence, computational complexity, PTIME-hardness

1 Introduction

The ever-growing need of formal verification of (complex) systems is reflected by the broad and active research in this area. A particular research branch

*The authors acknowledge the support by the Grant Agency of the Czech Republic, Grant No. 201/03/1161, and in final stages also the support by the Ministry of Education of the Czech Republic, Project 1M6840770004.

is devoted to studying computational complexity of (and designing efficient algorithms for) various verification problems. Checking behavioural equivalences or preorders (between, say, a specification and an implementation) constitutes an important class of such problems.

Finite state systems (which might consist of communicating components running concurrently) are then a natural primary target for such studies. These systems are usually modelled as so called *labelled transition systems* (LTSs for short), which capture the notion of (global) states and their changes by performing transitions – these are labelled by actions (or action names). We deal here only with *finite LTSs*, which can be viewed as classical nondeterministic finite automata, NFA (without a necessity of defining accepting states).

It is useful to recall from classical language theory that checking language inclusion or equivalence for NFA is PSPACE-complete (cf., e.g., [5]). This easily applies also to the case where all states are deemed as accepting; we speak about *trace inclusion* (*trace preorder*) or *trace equivalence* then.

At studying systems from behavioural point of view, it was soon observed that trace equivalence is too coarse, and during 1980's the notion of *bisimulation equivalence* (also called *bisimilarity*) emerged as a more appropriate fundamental concept (cf. [7]). (Very roughly speaking, two bisimilar systems can simultaneously simulate each other.)

Also other notions of equivalences (or preorders) turned out to be useful for various specific aims. Van Glabbeek [11] classified such equivalences in a hierarchy called *linear time/branching time spectrum*. The diagram in Figure 1 shows the most prominent members of the hierarchy and their interrelations (the arrow from R to S means that equivalence R is finer than equivalence S). We note that bisimilarity is the finest in the spectrum while trace equivalence is the coarsest. Roughly speaking, we can also divide the equivalences into *'trace-like'* and *'simulation-like'* ones.

For the aims of automated verification, it is natural to study the computational complexity of each relevant relation (equivalence or preorder) X in the case of finite LTSs. We have already mentioned that PSPACE-completeness of trace equivalence follows from classical language theory. On the other hand, there are polynomial-time algorithms in the simulation-like part of the spectrum; we can refer, e.g., to [3] for a survey.

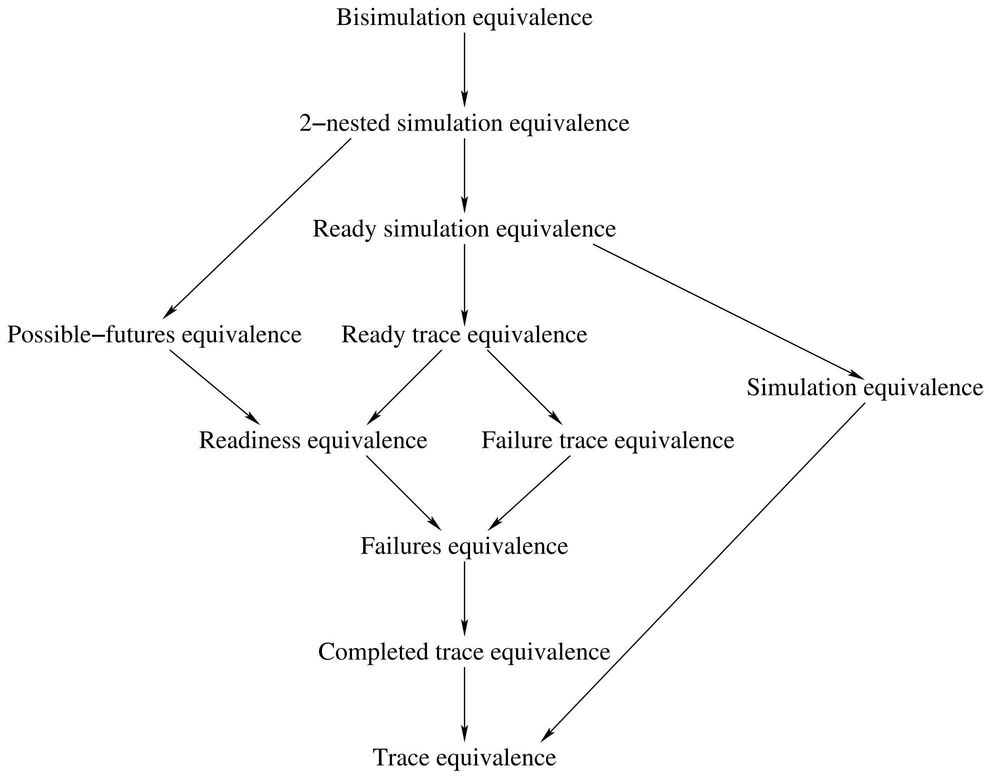

Figure 1: The linear time/branching time spectrum

Balcázar, Gabarró and Sántha [1] considered the question of an efficient parallelization of the (polynomial) algorithm for *bisimilarity*. They showed that the problem of checking bisimilarity is PTIME-hard (i.e., all problems from PTIME are reducible to this problem by a LOGSPACE-reduction). This shows that the problem seems to be 'inherently sequential', i.e., we can not hope for a real gain by parallelization (unless NC = PTIME, which is considered to be very unlikely; cf. e.g. [4]).

Paper [1] shows a LOGSPACE reduction from (a special version of) the boolean circuit value problem, which is well-known to be PTIME-complete. Their reduction handles just bisimilarity; in particular, it does not show PTIME-hardness of other 'simulation-like' equivalences (which are known to be in PTIME as well).

In this paper, we show another reduction from (a less constrained version of) the circuit value problem which we find simple and elegant and which immediately shows that deciding an *arbitrary relation* which subsumes bisimulation equivalence and is subsumed by trace preorder is PTIME-hard. Thus the result of [1] is substantially extended.

The value of this extension is primarily relevant for the problems in PTIME, i.e., in the simulation-like part of the spectrum, which can also comprise 'not yet discovered' equivalences; of course, PTIME-hardness trivially follows for problems which are known to be PSPACE-hard (in the trace-like part). It is also worth to point out that our construction provides the PTIME-hardness result even for *acyclic* finite state systems; the problems for this subclass are generally easier, though trace equivalence is still CO-NP-complete.

At the end of the paper, after Section 2 with definitions and Section 3 with the technical proof, we add some remarks on 'lifting' the result to settle a conjecture in [8]. Finally we remark that a preliminary version of this paper appeared in [10].

2 Definitions

A *labelled transition system* (an *LTS* for short) is a tuple $\langle S, Act, \longrightarrow\rangle$ where S is a set of *states*, Act is a set of *actions* (or *labels*), and $\longrightarrow \subseteq S \times Act \times S$ is a *transition relation*. A tuple $\langle p, a, q\rangle \in \longrightarrow$, written also $p \xrightarrow{a} q$, is called a *transition*. We also use notation $p \xrightarrow{w} q$ for finite sequences of actions

($w \in Act^*$), with the obvious meaning. We only consider *finite* LTSs, where both sets S and Act are finite.

Now we give precise definitions of trace and bisimulation equivalences between states in LTSs. It is sufficient to consider only pairs of states belonging to *the same* LTS, since the states of two different LTSs can be naturally viewed as the states of their disjoint union.

For a state p of an LTS $\langle S, Act, \longrightarrow \rangle$, we define the set of its *traces* as

$$tr(p) = \{ \, w \in Act^* \mid p \xrightarrow{w} q \text{ for some } q \in S \, \} \, .$$

States p and q are *trace equivalent* iff $tr(p) = tr(q)$; they are in *trace preorder* iff $tr(p) \subseteq tr(q)$.

A binary relation $\mathcal{R} \subseteq S \times S$ on the state set of an LTS $\langle S, Act, \longrightarrow \rangle$ is a *bisimulation* iff the two following conditions hold for each $(p, q) \in \mathcal{R}$ (and each $a \in Act$):

- if $p \xrightarrow{a} p'$ then there is $q' \in S$ such that $q \xrightarrow{a} q'$ and $(p', q') \in \mathcal{R}$,
- if $q \xrightarrow{a} q'$ then there is $p' \in S$ such that $p \xrightarrow{a} p'$ and $(p', q') \in \mathcal{R}$.

We will say that a transition $p \xrightarrow{a} p'$ is *matched* by $q \xrightarrow{a} q'$, resp. $q \xrightarrow{a} q'$ by $p \xrightarrow{a} p'$, if $(p', q') \in \mathcal{R}$.

States p, q are *bisimulation equivalent* (or *bisimilar*), written $p \sim q$, iff $(p, q) \in \mathcal{R}$ for some bisimulation $\mathcal{R}$. Note that the union of bisimulations is a bisimulation; thus bisimilarity $\sim$ is the maximal bisimulation (the union of all bisimulations). It is easy to check that $\sim$ is really an equivalence relation.

From complexity theory we recall that a *problem* P is PTIME-*hard* iff each problem from PTIME can be reduced to P by a LOGSPACE-reduction (i.e., the working space of the reducing algorithm is in $O(\log n)$, where n denotes the size of the input on the read-only input tape; the output is being written on the write-only output tape, and may be of polynomial length). A *problem* P is PTIME-*complete* iff P is PTIME-hard and $P \in$ PTIME.

Remark. It is well-known that if a problem P is PTIME-hard then it is unlikely that there exists an efficient parallel algorithm deciding P. Here 'efficient' means working in polylogarithmic time (with time complexity $O(\log^k n)$) and with polynomial number of processors. (See e.g. [4] for further details.)

To formulate the next theorem, i.e., our main result, we use a further technical notion: A *relation* X on the states of LTSs is said to be *between bisimilarity and trace preorder* iff $p \sim q$ implies pXq and pXq implies $tr(p) \subseteq tr(q)$.

Theorem 1 *For any relation* X *between bisimilarity and trace preorder, the following problem is* PTIME*-hard:*

> INSTANCE*: A finite acyclic labelled transition system and two of its states,* p *and* q*.*
>
> QUESTION*: Is* pXq *?*

We shall prove the theorem by a LOGSPACE reduction from the problem of monotone boolean circuit value, mCVP for short.

To define mCVP, we need some definitions. A *monotone boolean circuit* is a directed acyclic graph in which the nodes (also called *gates*) are either of in-degree zero (*input* gates) or of in-degree 2; there is exactly one node of out-degree zero (the *output* gate), otherwise the out-degree is not constrained. Each non-input gate is labelled either by conjunction $\wedge$ or by disjunction $\vee$. (No negation is used; thus we have monotonicity.) An *input of the circuit* is an assignment of boolean values (values from the set $\{0, 1\}$) to input gates. Given an input, the circuit computes as expected: the (output) value of an input gate is given by the input, the value of a gate labelled with $\wedge$ (resp. $\vee$) is computed as conjunction (resp. disjunction) of values of its predecessors. The *output value of the circuit* is the value of the output gate.

See Figure 2 for an example of a monotone boolean circuit with an input; the corresponding computed values are also depicted.

> The mCVP problem is defined as follows:
>
> INSTANCE: A monotone boolean circuit with an input.
>
> QUESTION: Is the output value 1 ?

The problem is well-known to be PTIME-complete (cf. e.g. [4]). We also recall that if P_1 is PTIME-hard and P_1 is LOGSPACE reducible to P_2 then P_2 is PTIME-hard as well.

In the next section we show a LOGSPACE-algorithm which, given an instance of mCVP, constructs an LTS with two designated states p, q so that:

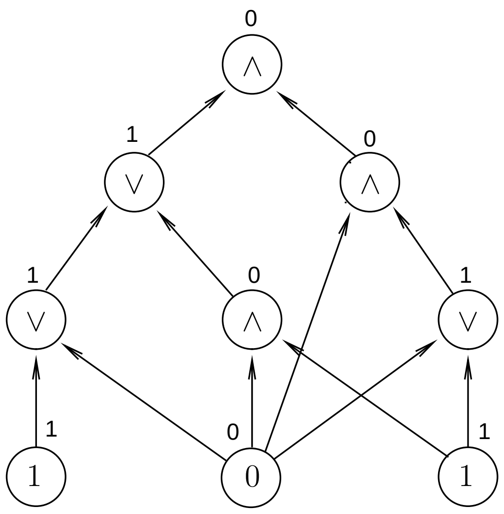

Figure 2: A monotone boolean circuit with computed values on gates

- if the output value of the circuit is 1 then $p \sim q$, and
- if the output value is 0 then $tr(p) \not\subseteq tr(q)$.

So for any relation X between bisimilarity and trace preorder we have pXq iff the output value is 1. This will immediately imply Theorem 1.

3 The Reduction

Let us fix an instance of mCVP where the set of gates is $V = \{1, 2, \ldots, n\}$. For technical reasons we assume that the gates are *topologically ordered*, i.e., for any edge (i, j) we have $i < j$; hence n is the output gate. (This assumption does not affect PTIME-completeness of mCVP, as can be seen from [4].) See Figure 10 for an example of topologically ordered gates.

We use function $t : V \longrightarrow \{0, 1, \wedge, \vee\}$ for denoting the types of gates (in the given instance of mCVP):

$$t(i) = \begin{cases} 0 & \text{if } i \text{ is an input gate with value } 0 \\ 1 & \text{if } i \text{ is an input gate with value } 1 \\ \wedge & \text{if } i \text{ is a gate labelled with } \wedge \\ \vee & \text{if } i \text{ is a gate labelled with } \vee \end{cases}$$

For every non-input gate i, we denote its (first and second) predecessors by $f(i), s(i)$ so that $f(i) < s(i)$ (and $s(i) < i$ due to the topological order). Let

$$v_i \in \{0, 1\}$$

denote the actual (computed) value on gate i, i.e.,

- if i is an input gate then $v_i = t(i)$,
- if i is a non-input gate, then v_i is computed from $v_{f(i)}$ and $v_{s(i)}$ using the operation indicated by $t(i)$.

For later use, we define inductively the words $W_i \in \{0, 1\}^*$, $i = 0, 1, \ldots, n$:

$$W_0 = \varepsilon, W_{i+1} = v_{i+1} W_i \tag{1}$$

Thus W_n is the sequence of the actual (computed) values for all gates in the reversed order wrt the given topological order; W_i is the suffix of W_n of length i.

For concreteness, we can assume that the given input instance of mCVP is of the form

$$1 : d(1);\ 2 : d(2);\ \ldots;\ n : d(n)$$

where $d(i) = t(i)$ if $t(i) \in \{0, 1\}$ and $d(i)$ stands for $\langle t(i), f(i), s(i)\rangle$ if $t(i) \in \{\wedge, \vee\}$.

Given this instance of mCVP, we show a construction of LTS $\Delta = \langle S, \mathit{Act}, \longrightarrow\rangle$, where $\mathit{Act} = \{0, 1\}$ and S is the union of the following sets:

- $\{p_i \mid 0 \leq i \leq n\}$,
- $\{q_i^j \mid 1 \leq j \leq i \leq n\}$,
- $\{r_i^{j,k} \mid 1 \leq k < j \leq i \leq n\}$.

We organize states in S into *levels*. *Level* i (where $0 \leq i \leq n$) contains all states with the same lower index i, i.e.,

$$\{p_i\} \cup Q_i$$

where

$$Q_i = \{q_i^j \mid 1 \leq j \leq i\} \cup \{r_i^{j,k} \mid 1 \leq k < j \leq i\}$$

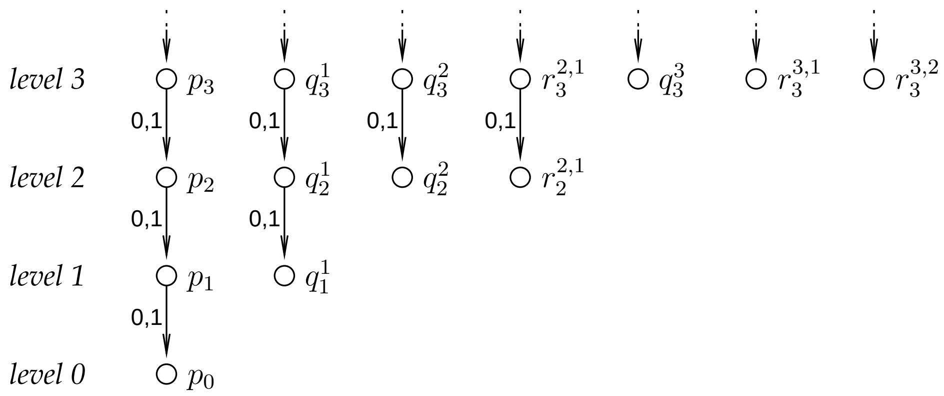

Figure 3: The states of Δ organized into levels

as depicted in Figure 3 (already with *some* transitions).

For ease of presentation, we call q_i^j *successful* if $v_j = 1$, and *unsuccessful* if $v_j = 0$; similarly, state $r_i^{j,k}$ is *successful* if *both* $v_j = 1$ and $v_k = 1$, and is *unsuccessful* otherwise. We denote the set of all successful (unsuccessful) states on level i by Succ_i (Unsucc_i). Hence

$$Q_i = \mathit{Succ}_i \cup \mathit{Unsucc}_i$$

Further, let I denote the identity relation $I = \{(s, s) \mid s \in S\}$.

The way we shall construct the transition relation of LTS Δ will guarantee the following property. (By W_i we refer to (1).)

CRUCIAL PROPERTY:

- the set
$$I \cup \{(p_i, q) \mid 0 \leq i \leq n, q \in \mathit{Succ}_i\}$$
is a bisimulation (therefore $p_i \sim q$ for each $q \in \mathit{Succ}_i$) ; and
- if $q \in \mathit{Unsucc}_i$ then
$$W_i \in tr(p_i) \setminus tr(q)$$
(therefore $tr(p_i) \not\subseteq tr(q)$) .

Hence p_n and q_n^n can serve as the two distinguished states announced at the end of the previous section, with the required property that

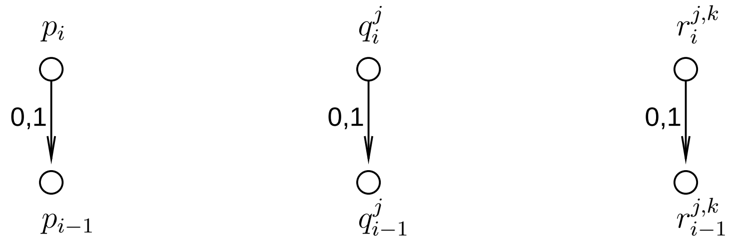

Figure 4: All the transitions added in the cases with $i > j$ and $t(i) \in \{0, 1, \wedge\}$

- $p_n \sim q_n^n$ if $v_n = 1$ (i.e., if q_n^n is successful), and
- $tr(p_n) \not\subseteq tr(q_n^n)$ otherwise.

3.1 Construction of transitions

We now construct the transition relation of Δ; we start with no transitions, and then we successively add the transitions described in what follows. There will be only transitions going from states on level i to states on level $i-1$; thus it is clear that the constructed LTS Δ will be acyclic.

Level i naturally corresponds to gate i; the actual transitions going from level i to level $i-1$ depend just on $t(i)$ and $f(i)$, $s(i)$ (when $t(i) \in \{\wedge, \vee\}$). We now describe in detail the transitions going from level i to level $i-1$ (for $i \geq 1$).

Not depending on $t(i)$, we add the following transitions

$$\begin{array}{ll} p_i \xrightarrow{0,1} p_{i-1} & \\ q_i^j \xrightarrow{0,1} q_{i-1}^j & \text{for } j < i\,, \\ r_i^{j,k} \xrightarrow{0,1} r_{i-1}^{j,k} & \text{for } j < i \text{ (recall that } 1 \leq k < j)\,. \end{array} \tag{2}$$

(We write $q \xrightarrow{0,1} q'$ instead of $q \xrightarrow{0} q'$ and $q \xrightarrow{1} q'$.)

These will be all the transitions leading from states $p_i, q_i^j, r_i^{j,k}$, with $i > j$, in the case $t(i) \in \{0, 1, \wedge\}$ (as depicted in Figure 4). We now handle the states q_i^i, $r_i^{i,k}$ in the (sub)cases $t(i) = 0$, $t(i) = 1$, $t(i) = \wedge$.

- $t(i) = 0$ (hence q_i^i and $r_i^{i,k}$ are unsuccessful):
  We do not add any other transitions (which is depicted in Figure 5).

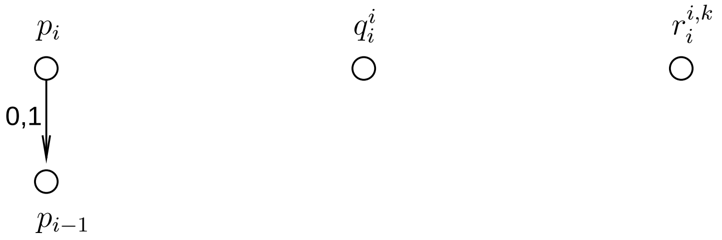

Figure 5: No further transitions in the case $t(i) = 0$

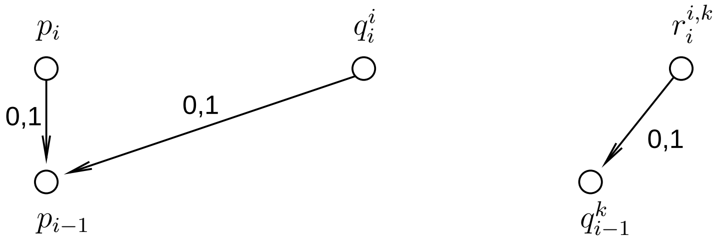

Figure 6: The transitions added when $t(i) = 1$

- $t(i) = 1$ (hence q_i^i is successful, and $r_i^{i,k}$ is successful if q_{i-1}^k is):
  We add transitions (depicted in Figure 6)

$$q_i^i \xrightarrow{0,1} p_{i-1} \qquad \text{and} \qquad r_i^{i,k} \xrightarrow{0,1} q_{i-1}^k$$

- $t(i) = \wedge$:
  We add the following transitions (depicted in Figure 7)

$$\begin{array}{lcl} q_i^i \xrightarrow{1} p_{i-1} & \text{and} & q_i^i \xrightarrow{0} r_{i-1}^{s(i),f(i)} \\ r_i^{i,k} \xrightarrow{1} q_{i-1}^k & \text{and} & r_i^{i,k} \xrightarrow{0} r_{i-1}^{s(i),f(i)} \end{array}$$

The case with $t(i) = \vee$ is the most complicated. The reason for this will appear more clearly when we shall prove correctness of our construction.

- $t(i) = \vee$:
  (See Figures 8 and 9.) Besides the transitions (2) (added independently on $t(i)$), we add the following transitions for every q on level i ($q \in \{p_i\} \cup Q_i$):

$$q \xrightarrow{0} q_{i-1}^{f(i)} \qquad \text{and} \qquad q \xrightarrow{0} q_{i-1}^{s(i)}$$

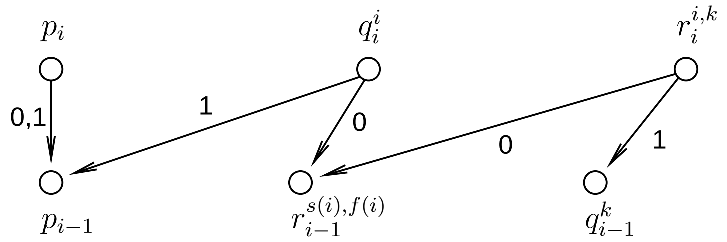

Figure 7: The transitions added when $t(i) = \wedge$

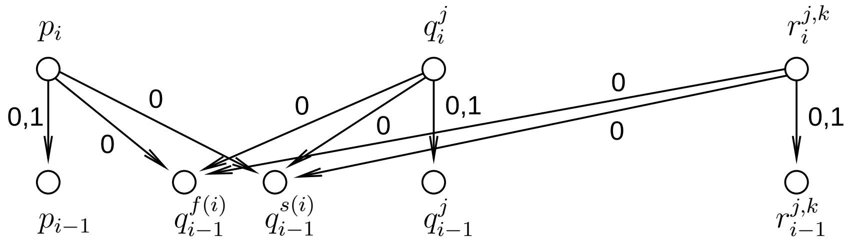

Figure 8: All the transitions added in the cases with $i > j$ and $t(i) = \vee$

and we further add transitions

$$q_i^i \xrightarrow{1} p_{i-1} \qquad \text{and} \qquad r_i^{i,k} \xrightarrow{1} q_{i-1}^k$$

To illustrate the overall construction, Figure 11 shows the LTS constructed for the boolean circuit depicted in Figure 10. (For better clarity, only states that are reachable from states p_6 and q_6^6 are depicted.)

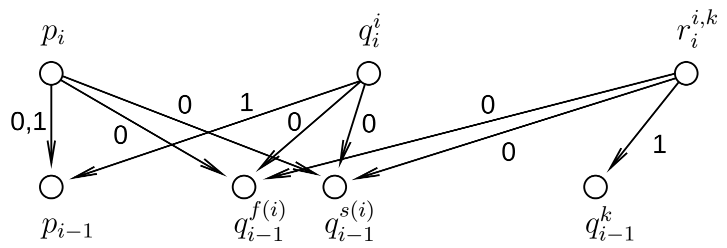

Figure 9: Further transitions in the case $t(i) = \vee$ (and $i = j$)

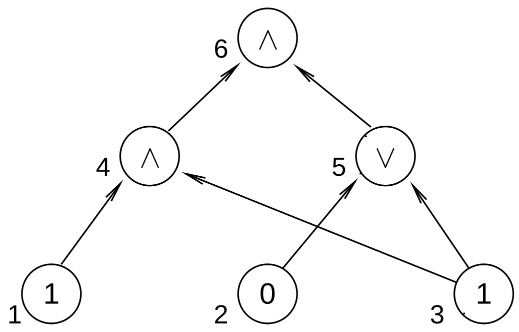

Figure 10: An example of a (topologically ordered) boolean circuit

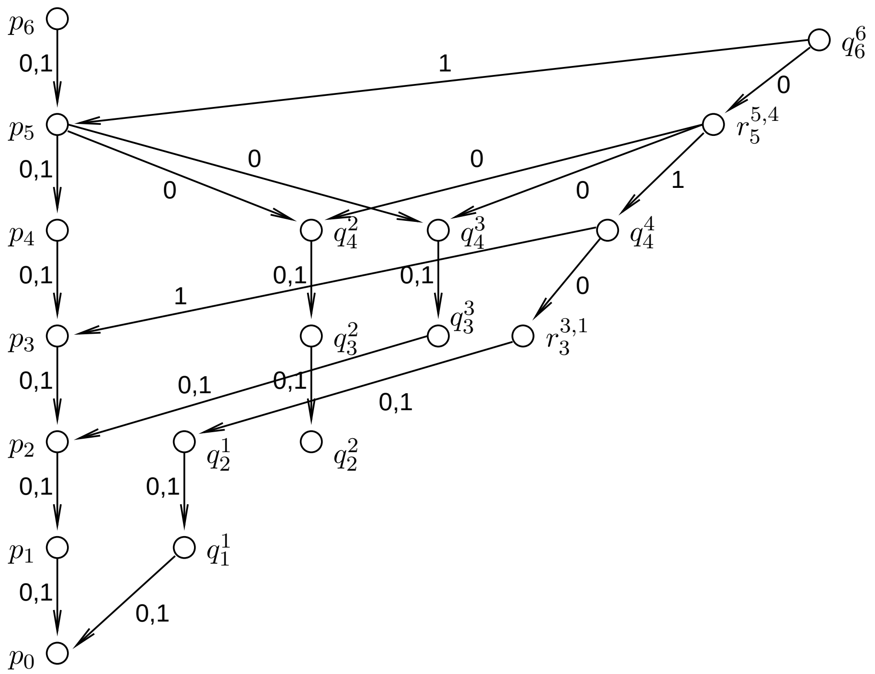

Figure 11: An example of the constructed LTS

3.2 Reduction runs in logarithmic space

We now sketch why the described reduction can be performed in logarithmic space. The algorithm performing the reduction has an instance of mCVP as its input, and outputs the states and transitions of LTS Δ in a systematic stepwise manner. It is clear that the algorithm can be bound to use only a fixed number of variables (such as i, j, k) with values no bigger than n (the number of gates); these values can be stored in $O(\log n)$ bits (and the size of the input is clearly bigger than n).

Remark. LTS Δ contains $O(n^3)$ states and also $O(n^3)$ transitions (the number of possible transitions leaving a given state is bounded, independently on n). Note that a state of the form $r_i^{j,k}$ can be reachable from p_n or q_n^n only if there is $i' \in V$, such that $t(i') = \wedge$, $s(i') = j$ and $f(i') = k$, as can be easily proved by induction. There is at most $O(n)$ pairs j, k, where such i' exists, and it is possible to test for given j, k the existence of such i' in LOGSPACE. So we could add to Δ only those states $r_i^{j,k}$, where such i' exists. In this way we can reduce the number of states (and transitions) to $O(n^2)$.

3.3 Correctness of the construction

Now we proceed to show the correctness of the described construction, i.e., that LTS Δ indeed satisfies the CRUCIAL PROPERTY mentioned before Subsection 3.1. This is proven by the following two lemmas.

Lemma 2 *$\mathcal{R} = I \cup \{(p_i, q) \mid 0 \leq i \leq n, q \in Succ_i\}$ is a bisimulation.*

Proof. Consider a pair $(p, q) \in \mathcal{R}$. We must show that every transition $p \xrightarrow{a} p'$ can be matched by some transition $q \xrightarrow{a} q'$ (so that $(p', q') \in \mathcal{R}$), and that each $q \xrightarrow{a} q'$ can be matched by some $p \xrightarrow{a} p'$.

First note that the case $(p, q) \in I$ (i.e., $p = q$) is trivial (any transition $p \xrightarrow{a} p'$ is matched by the same transition $p \xrightarrow{a} p'$).

Now consider a pair

$$(p_i, q_i^j) \in \mathcal{R}$$

(hence $q_i^j \in Succ_i$, i.e., $v_j = 1$; recall also that $j \leq i$). We handle two subcases ($j < i$ and $j = i$) separately.

- $j < i$ (Figures 4 and 8):
  Note that $q_i^j \in Succ_i$ implies $q_{i-1}^j \in Succ_{i-1}$, hence $(p_{i-1}, q_{i-1}^j) \in \mathcal{R}$. So $p_i \xrightarrow{a} p_{i-1}$ where $a \in \{0,1\}$ can be matched by $q_i^j \xrightarrow{a} q_{i-1}^j$, and $q_i^j \xrightarrow{a} q_{i-1}^j$ can be matched by $p_i \xrightarrow{a} p_{i-1}$. When $t(i)$ is one of 0, 1, $\wedge$ (Figure 4), we are done.

  When $t(i) = \vee$ (Figure 8), we must also consider transitions $p_i \xrightarrow{0} q_{i-1}^{f(i)}$, $p_i \xrightarrow{0} q_{i-1}^{s(i)}$, $q_i^j \xrightarrow{0} q_{i-1}^{f(i)}$ and $q_i^j \xrightarrow{0} q_{i-1}^{s(i)}$. The transition $p_i \xrightarrow{0} q_{i-1}^{f(i)}$ is matched by $q_i^j \xrightarrow{0} q_{i-1}^{f(i)}$ and vice versa, and $p_i \xrightarrow{0} q_{i-1}^{s(i)}$ is matched by $q_i^j \xrightarrow{0} q_{i-1}^{s(i)}$ and vice versa. (We use that $I \subseteq \mathcal{R}$.)

- $j = i$ (which means $v_i = v_j = 1$, and so $t(i) \neq 0$):
  We handle the three (sub)cases depending on $t(i)$ separately:

  - $t(i) = 1$ (Figure 6): Transition $p_i \xrightarrow{a} p_{i-1}$ where $a \in \{0,1\}$ is matched by $q_i^i \xrightarrow{a} p_{i-1}$ and vice versa.
  - $t(i) = \wedge$ (Figure 7): Transition $p_i \xrightarrow{1} p_{i-1}$ is matched by $q_i^i \xrightarrow{1} p_{i-1}$ and vice versa.

    Since $v_i = 1$, we have $v_{f(i)} = 1$ and $v_{s(i)} = 1$. So $r_{i-1}^{s(i),f(i)}$ is successful, and $(p_{i-1}, r_{i-1}^{s(i),f(i)}) \in \mathcal{R}$. Hence $p_i \xrightarrow{0} p_{i-1}$ is matched by $q_i^i \xrightarrow{0} r_{i-1}^{s(i),f(i)}$ and vice versa.
  - $t(i) = \vee$ (Figure 9): Transition $p_i \xrightarrow{1} p_{i-1}$ is matched by $q_i^i \xrightarrow{1} p_{i-1}$ and vice versa, $p_i \xrightarrow{0} q_{i-1}^{f(i)}$ by $q_i^i \xrightarrow{0} q_{i-1}^{f(i)}$ and vice versa, and $p_i \xrightarrow{0} q_{i-1}^{s(i)}$ by $q_i^i \xrightarrow{0} q_{i-1}^{s(i)}$ and vice versa.

    Since $v_i = 1$, we have $v_{f(i)} = 1$ or $v_{s(i)} = 1$, which means that at least one of $q_{i-1}^{f(i)}, q_{i-1}^{s(i)}$ is successful. From this we have that $(p_{i-1}, q_{i-1}^{f(i)}) \in \mathcal{R}$ or $(p_{i-1}, q_{i-1}^{s(i)}) \in \mathcal{R}$. So $p_i \xrightarrow{0} p_{i-1}$ can be matched by (at least) one of $q_i^i \xrightarrow{0} q_{i-1}^{f(i)}$ and $q_i^i \xrightarrow{0} q_{i-1}^{s(i)}$.

Remark. The last subcase ($t(i) = \vee$) indicates why the construction was more complicated for $t(i) = \vee$ than for $t(i) = \wedge$. We had to be careful to properly reflect the choice: for $t(i) = \vee$, the ('successful') case $v_i = 1$ means just that *some* of $v_{f(i)}$, $v_{s(i)}$ is also 'successful', i.e. 1; for $t(i) = \wedge$ it is easier – $v_i = 1$ means that *both* $v_{f(i)}$, $v_{s(i)}$ must be 1 as well.

It remains to consider a pair

$$(p_i, r_i^{j,k}) \in \mathcal{R}$$

(hence $r_i^{j,k} \in \mathit{Succ}_i$, i.e., $v_j = v_k = 1$; recall also that $k < j \leq i$). We again handle two subcases ($j < i$ and $j = i$) separately.

- $j < i$ (Figures 4 and 8):
Since $r_i^{j,k} \in \mathit{Succ}_i$, we have $r_{i-1}^{j,k} \in \mathit{Succ}_{i-1}$, hence $(p_{i-1}, r_{i-1}^{j,k}) \in \mathcal{R}$. So $p_i \xrightarrow{a} p_{i-1}$ where $a \in \{0,1\}$ can be matched by $r_i^{j,k} \xrightarrow{a} r_{i-1}^{j,k}$ and vice versa. When $t(i)$ is one of 0, 1, $\wedge$ (Figure 4), we are done.

  When $t(i) = \vee$ (Figure 8), transition $p_i \xrightarrow{0} q_{i-1}^{f(i)}$ is matched by $r_i^{j,k} \xrightarrow{0} q_{i-1}^{f(i)}$ and vice versa, and $p_i \xrightarrow{0} q_{i-1}^{s(i)}$ by $r_i^{j,k} \xrightarrow{0} q_{i-1}^{s(i)}$ and vice versa.

- $j = i$ (which means $v_i = v_k = 1$):
Note that q_{i-1}^k is successful, hence $(p_{i-1}, q_{i-1}^k) \in \mathcal{R}$. Depending on $t(i)$ (where $t(i) \neq 0$) we distinguish different (sub)cases:

  - $t(i) = 1$ (Figure 6): Transition $p_i \xrightarrow{a} p_{i-1}$ (where $a \in \{0,1\}$) is matched by $r_i^{i,k} \xrightarrow{a} q_{i-1}^k$ and vice versa.
  - $t(i) = \wedge$ (Figure 7): Transition $p_i \xrightarrow{1} p_{i-1}$ is matched by $r_i^{i,k} \xrightarrow{1} q_{i-1}^k$ and vice versa.

    Since $v_i = 1$, we have $v_{f(i)} = 1$ and $v_{s(i)} = 1$. So $r_{i-1}^{s(i),f(i)}$ is successful, hence $(p_{i-1}, r_{i-1}^{s(i),f(i)}) \in \mathcal{R}$. Therefore $p_i \xrightarrow{0} p_{i-1}$ is matched by $r_i^{i,k} \xrightarrow{0} r_{i-1}^{s(i),f(i)}$ and vice versa.
  - $t(i) = \vee$ (Figure 9): Transition $p_i \xrightarrow{1} p_{i-1}$ is matched by $r_i^{i,k} \xrightarrow{1} q_{i-1}^k$ and vice versa, $p_i \xrightarrow{0} q_{i-1}^{f(i)}$ by $r_i^{i,k} \xrightarrow{0} q_{i-1}^{f(i)}$ and vice versa, and $p_i \xrightarrow{0} q_{i-1}^{s(i)}$ by $q_i^{i,k} \xrightarrow{0} q_{i-1}^{s(i)}$ and vice versa.

    Since $v_i = 1$, we have $v_{f(i)} = 1$ or $v_{s(i)} = 1$; hence $(p_{i-1}, q_{i-1}^{f(i)}) \in \mathcal{R}$ or $(p_{i-1}, q_{i-1}^{s(i)}) \in \mathcal{R}$. Therefore transition $p_i \xrightarrow{0} p_{i-1}$ can be matched by (at least) one of transitions $r_i^{i,k} \xrightarrow{0} q_{i-1}^{f(i)}$ and $r_i^{i,k} \xrightarrow{0} q_{i-1}^{s(i)}$.

$\square$

Lemma 3 *If $q \in \mathit{Unsucc}_i$ then $W_i \in tr(p_i) \setminus tr(q)$ (for $0 \leq i \leq n$).*

Proof. Note that $tr(p_i)$ contains all sequences from $\{0, 1\}^*$ of length at most i (see Figure 3), so in particular $W_i \in tr(p_i)$.

By induction on i we now show that $W_i \notin tr(q)$ for any $q \in \mathit{Unsucc}_i$. Since $Q_0 = \emptyset$ (we have only p_0 on level 0), the claim is vacuously true for $i = 0$.

Now suppose $i > 0$. Using the induction hypothesis, it is sufficient to show that $q \xrightarrow{v_i} q'$ implies that q' is unsuccessful.

We first consider a state

$$q_i^j \in \mathit{Unsucc}_i$$

(hence $v_j = 0$; recall also $j \leq i$). We handle two subcases ($j < i$ and $j = i$) separately.

- $j < i$ (Figures 4 and 8):
  Since q_{i-1}^j is (also) unsuccessful, we are done in the case $t(i) \in \{0, 1, \wedge\}$ (Figure 4), and also in the case when $t(i) = \vee$ and $v_i = 1$ (Figure 8).

  When $t(i) = \vee$ and $v_i = 0$ (Figure 8), we must also consider transitions $q_i^j \xrightarrow{0} q_{i-1}^{f(i)}$ and $q_i^j \xrightarrow{0} q_{i-1}^{s(i)}$. But $v_i = 0$ implies $v_{f(i)} = 0$ and $v_{s(i)} = 0$, and thus both $q_{i-1}^{f(i)}$, $q_{i-1}^{s(i)}$ are unsuccessful.

- $j = i$ (hence $v_i = 0$, which also means $t(i) \neq 1$):
  We distinguish different (sub)cases depending on $t(i)$:

  - $t(i) = 0$ (Figure 5): There is no q' such that $q_i^i \xrightarrow{0} q'$, so we are done.
  - $t(i) = \wedge$ (Figure 7): The only q' such that $q_i^i \xrightarrow{0} q'$ is $r_{i-1}^{s(i),f(i)}$. From $v_i = 0$ we get $v_{f(i)} = 0$ or $v_{s(i)} = 0$, which means that $r_{i-1}^{s(i),f(i)}$ is unsuccessful.
  - $t(i) = \vee$ (Figure 9): If $q_i^i \xrightarrow{0} q'$, then q' is either $q_{i-1}^{f(i)}$ or $q_{i-1}^{s(i)}$. From $v_i = 0$ we obtain $v_{f(i)} = 0$ and $v_{s(i)} = 0$, and so both $q_{i-1}^{f(i)}$ and $q_{i-1}^{s(i)}$ are unsuccessful.

It remains to consider a state

$$r_i^{j,k} \in \mathit{Unsucc}_i$$

(hence $v_j = 0$ or $v_k = 0$; recall also that $k < j \leq i$). We handle two subcases ($j < i$ and $j = i$) separately.

- $j < i$ (Figures 4 and 8):
  Since (also) $r_{i-1}^{j,k}$ is unsuccessful, we are done in the case $t(i) \in \{0, 1, \wedge\}$ (Figure 4), and in the case when $t(i) = \vee$ and $v_i = 1$ (Figure 8).

  When $t(i) = \vee$ and $v_i = 0$ (Figure 8), we must also consider transitions $r_i^{j,k} \xrightarrow{0} q_{i-1}^{f(i)}$ and $q_i^{j,k} \xrightarrow{0} q_{i-1}^{s(i)}$. But $v_i = 0$ implies $v_{f(i)} = 0$ and $v_{s(i)} = 0$ and thus both $q_{i-1}^{f(i)}$, $q_{i-1}^{s(i)}$ are unsuccessful.

- $j = i$ (so if $v_i = 1$ then $v_k = 0$):
  We distinguish different (sub)cases depending on $t(i)$.

  - $t(i) = 0$ (Figure 5): There is no transition leaving $r_i^{i,k}$, so we are done.
  - $t(i) = 1$ (Figure 6): In this case $v_i = 1$, hence $v_k = 0$. The only state q' such that $r_i^{i,k} \xrightarrow{1} q'$ is q_{i-1}^k, and this is an unsuccessful state.
  - $t(i) = \wedge$ (Figure 7):
    The only q' such that $r_i^{i,k} \xrightarrow{0} q'$ is $r_{i-1}^{s(i),f(i)}$. In the case $v_i = 0$ we have $v_{f(i)} = 0$ or $v_{s(i)} = 0$, which means that $r_{i-1}^{s(i),f(i)}$ is unsuccessful.
    The only q' such that $r_i^{i,k} \xrightarrow{1} q'$ is q_{i-1}^k. In the case $v_i = 1$ we have $v_k = 0$, which means that q_{i-1}^k is unsuccessful.
  - $t(i) = \vee$ (Figure 9):

    The only q' such that $r_i^{i,k} \xrightarrow{1} q'$ is q_{i-1}^k. In the case $v_i = 1$ we have $v_k = 0$, which means that q_{i-1}^k is unsuccessful.

    If $r_i^{i,k} \xrightarrow{0} q'$ then q' is either $q_{i-1}^{f(i)}$ or $q_{i-1}^{s(i)}$. In the case $v_i = 0$ we have $v_{f(i)} = 0$ and $v_{s(i)} = 0$ which means that both $q_{i-1}^{f(i)}$ and $q_{i-1}^{s(i)}$ are unsuccessful.

$\square$

Additional Remarks

We have considered complexity as a function of the size of a given labelled transition systems (which describes the state space explicitly). Rabinovich [8] considered the problem for concurrent systems of communicating finite agents, measuring complexity in the size of (descriptions of) such systems; the corresponding (implicitly represented) state space can be exponential in that size. He conjectured that all relations between bisimilarity and trace equivalence are EXPTIME-hard in this setting. Laroussinie and Schnoebelen [6] have confirmed the conjecture partially. They showed EXPTIME-hardness for all relations between simulation preorder and bisimilarity. One of the authors of this paper modified and extended the construction presented here and proved Rabinovich's conjecture completely in [9].

Acknowledgement. We thank Philippe Schnoebelen for fruitful discussions.

References

[1] Balcázar, J., Gabarró, J., and Sántha, M. Deciding bisimilarity is P-complete. *Formal Aspects of Computing 4*, 6A (1992), 638–648.

[2] Bergstra, J., Ponse, A., and Smolka, S., Eds. *Handbook of Process Algebra*. Elsevier, 2001.

[3] Cleaveland, R., and Sokolsky, O. Equivalence and preorder checking for finite-state systems. In Bergstra et al. [2], pp. 391–424.

[4] Gibbons, A., and Rytter, W. *Efficient Parallel Algorithms*. Cambridge University Press, 1988.

[5] Hopcroft, J., and Ullman, J. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 1979.

[6] Laroussinie, F., and Schnoebelen, P. The state explosion problem from trace to bisimulation equivalence. In *Proc. 3rd Int. Conf. Foundations of Software Science and Computation Structures (FOSSACS'2000)* (2000), vol. 1784 of *Lecture Notes in Computer Science*, Springer, pp. 192–207.

[7] Milner, R. *Communication and Concurrency*. Prentice-Hall, 1989.

[8] Rabinovich, A. Complexity of equivalence problems for concurrent systems of finite agents. *Information and Computation 139*, 2 (1997), 111–129.

[9] Sawa, Z. Equivalence checking of non-flat systems is EXPTIME-hard. In *Proceedings of CONCUR 2003* (2003), vol. 2761 of *Lecture Notes in Computer Science*, Springer, pp. 237–250.

[10] Sawa, Z., and Jančar, P. *P*-hardness of equivalence testing on finite-state processes. In *Proceedings of SOFSEM 2001* (2001), vol. 2234 of *Lecture Notes in Computer Science*, Springer, pp. 326–335.

[11] van Glabbeek, R. The linear time – branching time spectrum. In Bergstra et al. [2], pp. 3–99.