# P-hardness of Equivalence Testing on Finite-State Processes*

Zdeněk Sawa and Petr Jančar

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

**Abstract.** The paper shows a simple LOGSPACE-reduction from the boolean circuit value problem which demonstrates that, on finite labelled transition systems, deciding an arbitrary relation which subsumes bisimulation equivalence and is subsumed by trace preorder is a polynomial-time-hard problem (and thus can not be expected to be efficiently parallelizable). By this, the result of Balcázar, Gabarró and Sántha (1992) for bisimilarity is substantially extended.

## 1 Introduction

It is not necessary to emphasize the importance of theoretical foundations for design, analysis and verification of (complex) systems, especially concurrent systems, which are composed from communicating components running in parallel. One particular research area studies computational complexity of various verification problems for finite state systems.

A general model of such systems is given by so called labelled transition systems (LTSs for short), which capture the notion of (global) states and their changes by performing transitions – which are labelled by actions (or action names). Since here we deal only with *finite LTSs*, they can be viewed as classical nondeterministic finite automata.

We consider the verification problem of testing behavioural equivalences on finite LTSs. Let us recall that classical language equivalence turned out to be mostly too coarse, and it was *bisimilarity* which was established as the most appropriate notion of general behavioural equivalence (cf. [6]). Nevertheless, other notions of equivalences (or preorders) turned out to be useful for more specific aims. Van Glabbeek [9] classified these equivalences in a hierarchy called *linear time/branching time spectrum*. The diagram in Fig. 1 shows most prominent members of the hierarchy and their interrelations (the arrow from $R$ to $S$ means that equivalence $R$ is finer than equivalence $S$). As depicted, bisimilarity (i.e., bisimulation equivalence) is the finest in the spectrum; the coarsest is *trace equivalence*, which is the classical language equivalence when we assume all states as

* Supported by the Grant Agency of the Czech Republic, Grant No. 201/00/0400

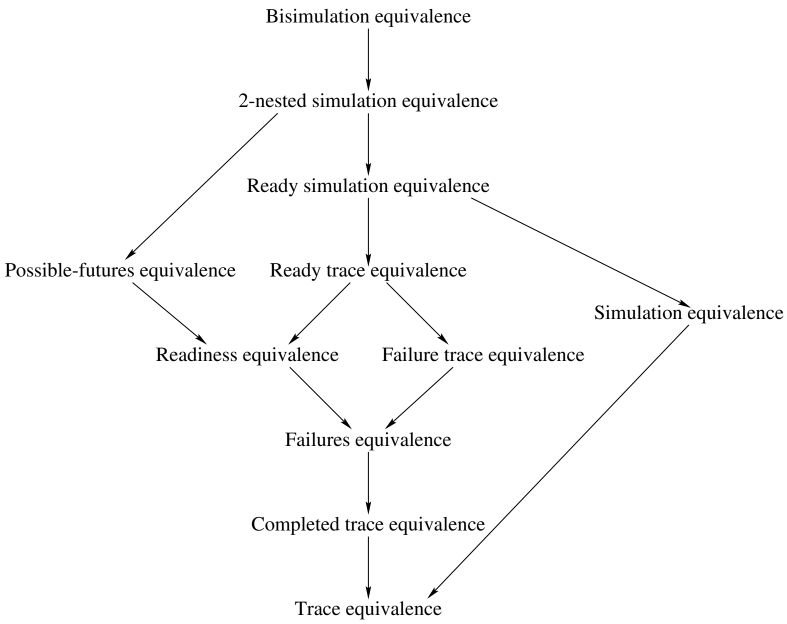


**Fig. 1.** The linear time/branching time spectrum

accepting (i.e., we are interested in the set of all sequences of actions which are performable).

For the aims of automated verification, a natural research task is to establish the complexity of the problem

INSTANCE: A finite labelled transition system and two of its states, $p$ and $q$.
QUESTION: Are $p$ and $q$ equivalent with respect to $X$ ?

for each equivalence $X$ in the spectrum. From language theory results we can easily derive PSPACE-completeness of trace equivalence. On the other hand, there is a polynomial time algorithm for bisimilarity [7, 4]. The paper [3] is a (preliminary) survey of all results in this area. Loosely speaking, 'trace-like' equivalences (on the bottom part of the spectrum) turn out to be PSPACE-complete, the 'simulation-like' equivalences (on the top of spectrum) are in PTIME. Balcázar, Gabarró and Sántha [1] have considered the question of an efficient parallelization of the algorithm for bisimilarity, and they have shown that the problem is P-complete (i.e., all polynomial-time problems are reducible to this problem by a LOGSPACE-reduction). This shows that the bisimilarity problem seems to be 'inherently sequential'; we can not get a real gain by parallelization, unless NC = PTIME, which is considered to be very unlikely (cf. e.g. [2]).

Paper [1] shows a (LOGSPACE) reduction from (a special version of) the boolean circuit value problem which is well-known to be P-complete. The reduction is aiming just at bisimilarity; in particular, it does not show P-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) 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 equivalence (more generally, by trace preorder) is P-hard. By this, the result of [1] is substantially extended; though it brings nothing new for the ('trace-like') equivalences for which PSPACE-hardness has been established, it surely is relevant for 'simulation-like' equivalences (those between bisimulation and simulation equivalences in the spectrum).

Section 2 gives necessary definitions and formulates the main result, and Section 3 contains the technical proof. We then add remarks on a possibility to 'lift' the result to settle a conjecture in [8].

## 2 Definitions

A *labelled transition system* (an *LT-system* 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*. We write $p \stackrel{a}{\longrightarrow} q$ instead of $\langle p, a, q\rangle \in \longrightarrow$; we also use $p \stackrel{w}{\longrightarrow} q$ for finite sequences of actions ($w \in Act^*$) with the natural meaning. In this paper, we only consider *finite* LT-systems, where both the sets $S$ and $Act$ are finite.

We need precise definitions of trace and bisimulation equivalences on states in LT-systems. Let us remark that it is sufficient for us only to relate states of *the same* LT-system; this could be naturally extended for states of *different* LT-systems (we can take disjoint union of these).

For a state $p$ of an LT-system $\langle S, Act, \longrightarrow \rangle$, we define the set of its *traces* as $tr(p) = \{\, w \in Act^* \mid p \stackrel{w}{\longrightarrow} q \text{ for some } q \in S \,\}$. 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 LT-system $\langle S, Act, \longrightarrow \rangle$ is a *simulation* iff for each $(p,q) \in \mathcal{R}$ and each $p \stackrel{a}{\longrightarrow} p'$ there is some $q \stackrel{a}{\longrightarrow} q'$ such that $(p', q') \in \mathcal{R}$. $\mathcal{R}$ is a *bisimulation* iff both $\mathcal{R}$ and its inverse $\mathcal{R}^{-1}$ are simulations. States $p, q$ are *bisimulation equivalent* (or *bisimilar*), written $p \sim q$, iff $(p,q) \in \mathcal{R}$ for some bisimulation $\mathcal{R}$.

We recall that a problem $P$ is P-hard if any problem in PTIME can be reduced to $P$ by a LOGSPACE reduction; recall that a Turing machine performing such a reduction uses work space of size at most $O(\log n)$, where $n$ denotes the size of the input on a read-only input tape (the output is written on a write-only output tape and its size may be polynomial). A problem $P$ is P-complete if $P$ is P-hard and $P \in$ PTIME.

*Remark.* We recall that if a problem $P$ is P-hard then it is unlikely that there exists an efficient parallel algorithm deciding $P$. 'Efficient' here means working in polylogarithmic time, i.e., with the time complexity in $O(\log^k n)$ for some constant $k$, while the number of the processors used is bounded by a polynomial in the size $n$ of the input instance. (See e.g. [2] for further details.)

We say that a *relation* $X$ (relating states in transition systems) is *between bisimilarity and trace preorder* iff $p \sim q$ implies $pXq$ and $pXq$ implies $tr(p) \subseteq tr(q)$. And we formulate the main result of our paper:

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

> Instance: *A finite labelled transition system and two of its states, $p$ and $q$.*
> Question: *Is $pXq$ ?*

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

To define mCVP we need some definitions. *Monotone boolean circuit* is a directed, acyclic graph, in which the nodes (also called *gates*) are either of indegree zero (*input* gates) or of indegree 2 (*non-input* gates). There is exactly one node of outdegree zero (the *output* gate). Non-input gates are labelled by one of $\{\wedge, \vee\}$ (notice that in monotone circuit no $\neg$-gates are used). *Input of the circuit* is an assignment of boolean values (i.e., values from the set $\{0, 1\}$) to input gates. A value on a non-input gate labelled with $\wedge$ (resp. $\vee$) is computed as the conjunction (resp. disjunction) of values on its ancestors. The output value of the circuit is the value on the output gate.

The mCVP problem is defined as follows:

> Instance: A monotone boolean circuit with its input.
> Question: Is the output value 1 ?

The problem is well-known to be P-complete (cf. e.g. [2]). We also recall that if $P_1$ is P-hard and $P_1$ is LOGSPACE reducible to $P_2$ then $P_2$ is P-hard as well.

In the next section we show how, given an instance of mCVP, to construct (in LOGSPACE) a transition system with two designated states $p, q$ so that 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, $pXq$ iff the output value is 1. This immediately implies the theorem.

## 3 The Reduction

Let us have an instance of mCVP where the set of gates is $V = \{1, 2, \ldots, n\}$. For every non-input gate $i$, we define $l(i), r(i)$ (left and right ancestor of $i$) to be the gates, such that there are edges from $l(i)$ and $r(i)$ to $i$ (and $l(i) \neq r(i)$). For technical reasons we assume the gates are topologically ordered, i.e., for every non-input gate $i$ we have $i > l(i) > r(i)$, and $n$ is the output gate (mCVP is still P-complete under this assumption). We define a function $t : V \longrightarrow \{0, 1, \wedge, \vee\}$, where $t(i)$ denotes the 'type' of gate $i$:

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

Let $v_i \in \{0, 1\}$ denote the actual value on gate $i$, i.e., if $i$ is an input gate then $v_i = t(i)$, and if $i$ is a non-input gate, then $v_i$ is computed from $v_{l(i)}, v_{r(i)}$ using operation indicated by $t(i)$.

We assume, that an input instance of mCVP consists of $n$ and of values $l(i), r(i)$ and $t(i)$ for every $1 \leq i \leq n$ (in fact, it suffices that these values can be computed from the input instance in LOGSPACE).

Given an instance of mCVP, we construct LT-system $\Delta = \langle S, Act, \longrightarrow\rangle$, where $Act = \{0, 1\}$ and $S$ is a union of the following sets:

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

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., $\{p_i\} \cup \{q_i^j \mid 1 \leq j \leq i\} \cup \{q_i^{j,k} \mid 1 \leq k < j \leq i\}$ as depicted in Fig. 2 (already with *some* transitions).

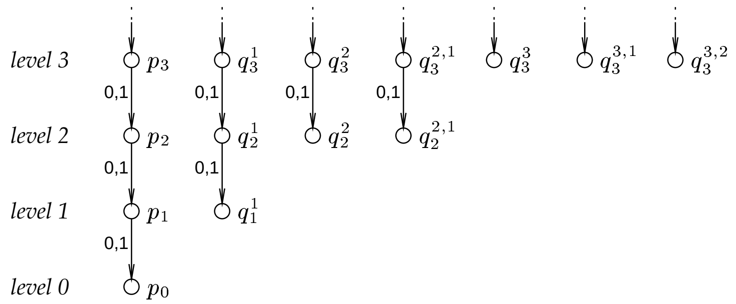


**Fig. 2.** The states of $\Delta$ organized into levels

Informally speaking, the intended purpose of states $q_i^j$ is to 'test' whether $v_j = 1$. State $q_i^j$ is viewed as 'successful' if indeed $v_j = 1$ and is 'unsuccessful' if $v_j = 0$. Similarly, state $q_i^{j,k}$ is 'successful' if $v_j = 1$ and $v_k = 1$, and 'unsuccessful' otherwise.

The way we construct transition relation $\longrightarrow$ will guarantee, that each successful state $q$ on level $i$ (i.e., of the form $q_i^j$ or $q_i^{j,k}$) is bisimilar with $p_i$, and if $q$ (on level $i$) is unsuccessful, then $tr(p_i) \not\subseteq tr(q)$. So $p_n$ and $q_n^n$ are two distinguished states announced in the previous section, with the required property, that if $v_n = 1$, then $p_n \sim q_n^n$, and otherwise $tr(p_n) \not\subseteq tr(q_n^n)$.

Transition relation $\longrightarrow$ will contain only transitions going from states on level $i$ to states on level $i-1$. We will construct transitions level by level, starting with transitions going from (states on) level 1 to level 0, then from level 2 to level 1 and so on. The actual transitions going from level $i$ to level $i-1$ will depend on $t(i)$, $l(i)$ and $r(i)$, so in this sense level $i$ corresponds to gate $i$. It is worth to emphasize here, that the added transitions does not depend on information, whether a state is successful or not.

Now we describe in detail the construction of transitions leading from states on level $i$ to states on level $i-1$.

Firstly, the following transitions are always possible from states on level $i$ (see Fig. 2):

$$\begin{array}{lll} p_i \xrightarrow{0,1} p_{i-1}, & & \\ q_i^j \xrightarrow{0,1} q_{i-1}^j & & \text{for any } j\text{, such that } 1 \leq j < i, \\ q_i^{j,k} \xrightarrow{0,1} q_{i-1}^{j,k} & & \text{for any } j,k\text{, such that } 1 \leq k < j < i. \end{array}$$

(We write $q \xrightarrow{0,1} q'$ instead of $q \xrightarrow{0} q'$ and $q \xrightarrow{1} q'$.) Depending on $t(i)$, some other transitions going from states on level $i$ may be added.

To simplify the notation, we need some further definitions. Let $Q_i$ be the set of all states of the form $q_i^j$ or $q_i^{j,k}$ on level $i$, i.e., $Q_i = \{q_i^j \mid 1 \leq j \leq i\} \cup \{q_i^{j,k} \mid 1 \leq k < j \leq i\}$, and let $Succ_i$ be the set of all successful states in $Q_i$, i.e., $Succ_i = \{q_i^j \in Q_i \mid v_j = 1\} \cup \{q_i^{j,k} \in Q_i \mid v_j = 1 \text{ and } v_k = 1\}$.

We use $w_i$ to denote the sequence of actions that correspond to actual values on gates $i, i-1, \ldots, 1$, i.e., $w_1 = v_1$ and $w_i = v_i w_{i-1}$ for $i > 1$.

We will construct $\Delta$ in such a way, that each level $i$ will satisfy the following condition:

$$\text{For each } q \in Q_i\text{:} \quad \text{if } q \in Succ_i\text{, then } p_i \sim q\text{, otherwise } w_i \notin tr(q). \tag{1}$$

Notice, that $w_i \notin tr(q)$ implies $tr(p_i) \not\subseteq tr(q)$, because $tr(p_i)$ contains all possible traces of length $i$ over $Act$, so in particular $w_i \in tr(p_i)$.

Now we describe the remaining transitions together with a proof that each level satisfies the condition (1). We proceed by induction on $i$.

*Remark.* To show for some $q \in Q_i \setminus Succ_i$, that $w_i \notin tr(q)$, it suffices to show for every $q'$, such that there is a transition $q \xrightarrow{v_i} q'$, that $w_{i-1} \notin tr(q')$, i.e., to show that $q' \notin Succ_{i-1}$ (because $q' \notin Succ_{i-1}$ implies $w_{i-1} \notin tr(q')$ by induction hypothesis).

**Base of induction ($i = 1$):** Because the circuit is topologically ordered, gate 1 must be an input gate, so $t(1)$ is either 0 or 1. If $t(1) = 0$, we do not add any transitions (see Fig. 3). Because $t(1) = 0$ implies $v_1 = 0$, we have $q_1^1 \notin Succ_1$.


**Fig. 3.** The construction for $i = 1$

Obviously $w_1 \notin tr(q_1^1)$, so the condition (1) holds.

If $t(i) = 1$, we add transitions $q_1^1 \xrightarrow{0,1} p_0$ (see Fig. 3). From $t(1) = 1$ follows $q_1^1 \in Succ_1$, and $p_1 \sim q_1^1$ is obvious, so again the condition (1) holds.

**Inductive step ($i > 1$):**

- If $t(i) = 0$: We do not add any transitions. We first consider $q_i^j$ where $i > j$ (see Fig. 4).

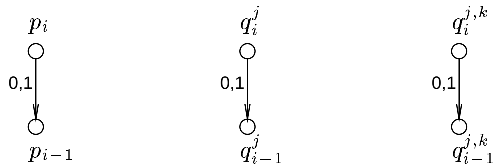


**Fig. 4.** The transitions added if $i > j$ and $t(i) \in \{0, 1, \wedge\}$

It is clear, that $q_i^j \in Succ_i$ iff $q_{i-1}^j \in Succ_{i-1}$, so $q_i^j$ satisfies the condition (1), as can be easily checked (by induction hypothesis $q_{i-1}^j \in Succ_{i-1}$ implies $p_i \sim q_{i-1}^j$, and $q_{i-1}^j \notin Succ_{i-1}$ implies $w_{i-1} \notin tr(q_{i-1}^j)$). The proof for $q_i^{j,k}$, where $i > j$, is similar.
Now, let $q$ be $q_i^j$ or $q_i^{j,k}$, where $i = j$, i.e., one of $q_i^i, q_i^{i,k}$. From $t(i) = 0$ we have $v_i = 0$, and this implies $q \notin Succ_i$. Obviously $w_i \notin tr(q)$, because no transitions are possible from $q$.

- If $t(i) = 1$: If $q$ is $q_i^j$ or $q_i^{j,k}$, where $i > j$, the situation is exactly the same as in the previous case (see Fig. 4). So let $q$ be $q_i^i$ or $q_i^{i,k}$. We add transitions from $q$ as depicted in Fig. 5.
If $q$ is $q_i^i$, then surely $q \in Succ_i$ (because $v_i = 1$), and obviously $p_i \sim q$.

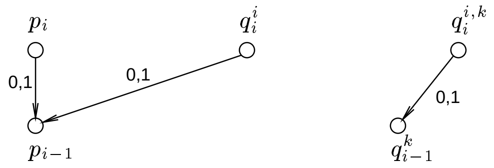


**Fig. 5.** The transitions added if $i = j$ and $t(i) = 1$

If $q$ is $q_i^{i,k}$, we can imagine this as the situation when it was tested that $v_i = 1$ and it is continued with testing that $v_k = 1$. Because $v_i = 1$, we have $q_i^{i,k} \in Succ_i$ iff $q_{i-1}^k \in Succ_{i-1}$, so the condition (1) is satisfied in $q_i^{i,k}$, as can be easily checked.

- If $t(i) = \wedge$: If $q$ is $q_i^j$ or $q_i^{j,k}$, where $i > j$, the situation is the same as in two previous cases (see Fig. 4). So let $q$ be $q_i^i$ or $q_i^{i,k}$. We add transitions from $q$ as depicted in Fig. 6.

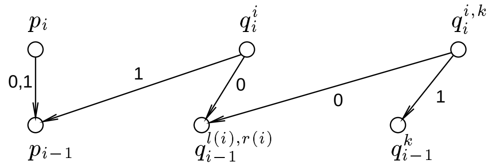


**Fig. 6.** The transitions added if $i = j$ and $t(i) = \wedge$

If $q \in Succ_i$, then $v_i = 1$. Because $t(i) = \wedge$, $v_i = 1$ implies $v_{l(i)} = 1$ and $v_{r(i)} = 1$, so $q_{i-1}^{l(i),r(i)} \in Succ_{i-1}$. From this $p_{i-1} \sim q_{i-1}^{l(i),r(i)}$ by induction hypothesis, so $p_i \xrightarrow{0} p_{i-1}$ can be matched by $q \xrightarrow{0} q_{i-1}^{l(i),r(i)}$ (and vice versa). Other transitions can be matched also ($q_i^{i,k} \in Succ_i$ implies $q_{i-1}^k \in Succ_{i-1}$), so we have $p_i \sim q$.
Now consider the case $q \notin Succ_i$. If $q$ is $q_i^i$, then $v_i = 0$, and this implies $v_{l(i)} = 0$ or $v_{r(i)} = 0$, so $q_{i-1}^{l(i),r(i)} \notin Succ_{i-1}$, and from this we have $w_i \notin tr(q_i^i)$, because $q_i^i \xrightarrow{0} q_{i-1}^{l(i),r(i)}$ is the only transition labelled with 0 possible in $q_i^i$.
If $q$ is $q_i^{i,k}$, then either $v_i = 0$ or $v_i = 1$. The case $v_i = 0$ is similar as if $q$ is $q_i^i$, so let us have $v_i = 1$. Then $v_k = 0$, and $q_{i-1}^k \notin Succ_{i-1}$, from which $w_i \notin tr(q_i^{i,k})$ follows.

- If $t(i) = \vee$: For every $q \in Q_i$ we add transitions $q \xrightarrow{0} q_{i-1}^{l(i)}$ and $q \xrightarrow{0} q_{i-1}^{r(i)}$. We also add transitions $p_i \xrightarrow{0} q_{i-1}^{l(i)}$ and $p_i \xrightarrow{0} q_{i-1}^{r(i)}$.
  Let $q$ be $q_i^j$ where $i > j$ as in Fig. 7. (The case when $q$ is $q_i^{j,k}$, where $i > j$, is almost identical.) If $q_i^j \in Succ_i$, then $q_{i-1}^j \in Succ_{i-1}$, so by induction hypothesis $p_{i-1} \sim q_{i-1}^j$. From this $p_i \sim q_i^j$ easily follows, because every possible transition can be matched (for example $p_i \xrightarrow{0} q_{i-1}^{l(i)}$ by $q_i^j \xrightarrow{0} q_{i-1}^{l(i)}$, etc.).

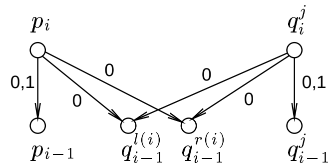


**Fig. 7.** The transitions added if $i > j$ and $t(i) = \vee$

If $q_i^j \notin Succ_i$, then $q_{i-1}^j \notin Succ_{i-1}$, so $w_{i-1} \notin tr(q_{i-1}^j)$. We need to consider two cases, $v_i$ is either 1 or 0. If $v_i = 1$, we have $w_i \notin tr(q_i^j)$, because $q_i^j \xrightarrow{1} q_{i-1}^j$ is the only transition labelled with 1 possible from $q$. If $v_i = 0$, then $v_{l(i)} = v_{r(i)} = 0$, so $q_{i-1}^{l(i)}, q_{i-1}^{r(i)} \notin Succ_{i-1}$, and from this $w_i \notin tr(q_i^j)$ easily follows.
Let us now consider the case, where $q$ is $q_i^i$ or $q_i^{i,k}$. We add transitions as depicted in Fig. 8.

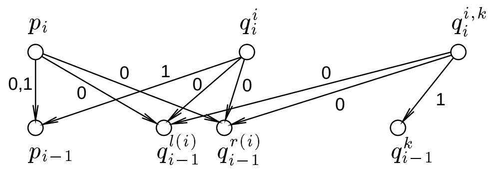


**Fig. 8.** The transitions added if $i = j$ and $t(i) = \vee$

If $q \in Succ_i$, then $p_i \xrightarrow{0} p_{i-1}$ can be matched by (at least) one of $q \xrightarrow{0} q_{i-1}^{l(i)}$, $q \xrightarrow{0} q_{i-1}^{r(i)}$, because from $v_i = 1$ we have $v_{l(i)} = 1$ or $v_{r(i)} = 1$. All other transitions are matched as in previous cases, so $p_i \sim q$.
Now suppose $q \notin Succ_i$. If $q$ is $q_i^i$, then $v_i = 0$, so $v_{l(i)} = v_{r(i)} = 0$, and $q_{i-1}^{l(i)}, q_{i-1}^{l(i)} \notin Succ_{i-1}$. From this $w_i \notin tr(q_i^i)$ easily follows. The case, when $q$ is $q_i^{i,k}$, is similar if $v_i = 0$. The only remaining possibility is that $v_i = 1$ and $v_k = 0$. Then $q_{i-1}^k \notin Succ_{i-1}$, so obviously $w_i \notin tr(q_i^{i,k})$.

To finish the proof of Theorem 1, it remains to show that the described reduction is in LOGSPACE.

The algorithm performing the reduction requires only fixed number of variables, such as $i, j, k$, and values of $t(i), l(i), r(i)$ for every $i$, that are part of the read-only input instance of mCVP. In particular, to construct transitions leading from a given state, only a fixed number of such values is needed. All such values can be represented as numbers bounded by $n$, so $O(\log n)$ bits are sufficient to store them. No other information is needed during the construction, so the algorithm uses work space of size $O(\log n)$. This finishes the proof.

*Remark.* LT-system $\Delta$ contains $O(n^3)$ states and also $O(n^3)$ transitions (because the number of possible transitions in every state is in $O(1)$ and does not depend on $n$). Notice, that a state of the form $q_i^{j,k}$ is not reachable from $p_n$ nor $q_n^n$, if there is no $i' \in V$, such that $t(i') = \wedge$, $l(i') = j$ and $r(i') = k$, 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 $i'$ in LOGSPACE, so we can add to $\Delta$ only those states $q_i^{j,k}$, where such $i'$ exists. In this way we can reduce the number of states of $\Delta$ to $O(n^2)$.

### Additional Remarks

We have considered complexity as a function of the size of given labelled transition systems (which describe the state space explicitly). Rabinovich [8] considered the problem for concurrent systems of finite agents, measuring complexity in the size of (descriptions of) such systems; the corresponding (implicitly represented) state space is exponential in that size. He conjectured that all relations between bisimilarity and trace equivalence are EXPTIME-hard in this setting. Laroussinie and Schnoebelen [5] have confirmed the conjecture partially. They showed EXPTIME-hardness for all relations between simulation preorder and bisimilarity, and EXPSPACE-hardness on the 'trace equivalence end' of van Glabbeek's spectrum. We plan to explore the probable possibility that our construction can be 'lifted' (i.e., programmed concisely by a concurrent system of finite agents), which would settle Rabinovich's conjecture completely.

**Acknowledgement.** We thank Philippe Schnoebelen for fruitful discussions.

## References


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

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

[3] H. Hüttel and S. Shukla. On the complexity of deciding behavioural equivalences and preorders. Technical Report SUNYA-CS-96-03, State University of New York at Albany, Dec. 28, 1996. Also available at `http://www.cs.auc.dk/~hans/Publications/pubs.html`.

[4] P. Kanellakis and S. Smolka. CCS expressions, finite state processes and three problems of equivalence. *Information and Computation*, 86:43–68, 1990.

[5] F. Laroussinie and P. Schnoebelen. The state explosion problem from trace to bisimulation equivalence. In *Proc. FoSSaCS 2000*, volume 1784 of *Lecture Notes in Computer Science*, pages 192–207. Springer Verlag, 2000.

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

[7] R. Paige and R. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16:937–989, 1987.

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

[9] R. van Glabbeek. The linear time – branching time spectrum. In *Proceedings CONCUR'90*, volume 458 of *Lecture Notes in Computer Science*, pages 278–297, Amsterdam, 1990. Springer-Verlag.