Equivalence Checking of Non-Flat Systems Is EXPTIME-hard*

Zdeněk Sawa

Dept. of Computer Science, FEI,
Technical University of Ostrava,
17. listopadu 15,
CZ-708 33 Ostrava, Czech Republic
Zdenek.Sawa@vsb.cz

Abstract. The equivalence checking of systems that are given as a composition of interacting finite-state systems is considered. It is shown that the problem is *EXPTIME*-hard for any notion of equivalence that lies between bisimulation equivalence and trace equivalence, as conjectured by Rabinovich (1997). The result is proved for parallel composition of finite-state systems where hiding of actions is allowed, and for 1-safe Petri nets. The technique of the proof allows to extend this result easily to other types of 'non-flat' systems.

Key words: equivalence checking, finite-state systems, complexity

1 Introduction

One problem that naturally arises in the area of automatic verification of systems is the problem of equivalence checking. This problem can be stated as follows: given two descriptions of labelled transition systems, decide if the systems behave equivalently.

Many different types of equivalences were proposed in the literature, and some of the most prominent were organized by van Glabbeek [10] into linear time – branching time spectrum. All these equivalences lie between bisimulation equivalence (which is the finest of these equivalences) and trace equivalence (which is the coarsest).

We call a finite transition system that is given explicitly a *flat* transition system. A *non-flat* system is a system given as a composition of interacting flat systems. The set of global states of a non-flat system can be exponentially larger than the sum of sizes of its parts. This phenomenon is known as a state explosion and presents the main challenge in the design of efficient algorithms for verification of non-flat systems.

* This work was sponsored by the Grant Agency of the Czech Republic, Grant No. 201/03/1161

Overview of existing results. Rabinovich [6] considered a composition of finite-state systems that synchronize on identical actions and where some actions may be 'hidden' in the sense that they are replaced with invisible τ actions. He proved that equivalence checking is *PSPACE*-hard for such systems for any relation between bisimilarity and trace equivalence, and that the problem is *EXPSPACE*-complete for trace equivalence. He also mentioned that the problem is *EXPTIME*-complete for bisimilarity and conjectured that the problem is in fact *EXPTIME*-hard for any relation between bisimilarity and trace equivalence.

Laroussinie and Schnoebelen [5] approved the Rabinovich's conjecture for all relations that lie between bisimilarity and simulation preorder. The non-flat systems, used in their proof, synchronize on identical actions and do not use hiding. It is not possible to extend their result to all equivalences between bisimilarity and trace equivalence, because for example trace equivalence can be decided in *PSPACE* for this model, as was proved in [8]. See also [9] for results for other types of 'trace-like' equivalences and non-flat systems. Other type of non-flat systems are 1-safe Petri nets. See [4] for some results concerning them, in particular, deciding of bisimilarity is *EXPTIME*-complete for 1-safe Petri nets.

Our contribution. The Rabinovich's conjecture is approved in this paper for *all* relations between bisimilarity and trace preorder, not only for relations between bisimilarity and simulation preorder. We show that equivalence checking is *EXPTIME*-hard for any such relation if the considered model is a parallel composition of finite-state systems with hiding, the model for which Rabinovich formulated his conjecture in [6].

To simplify the proof, a new auxiliary model called reactive linear bounded automaton (RLBA) is introduced in this paper. Reactive linear bounded automata can be easily modeled by different types of non-flat systems, for example by parallel compositions of finite-state systems with hiding, or by 1-safe Petri nets. The *EXPTIME*-hardness result is shown for RLBA first, and then it is extended to other types of non-flat systems that are able to model RLBA.

From the construction in the proof we also obtain a simpler proof of the result, shown in [7], that equivalence checking is *PTIME*-hard for flat systems for every relation between bisimilarity and trace preorder.

Overview of the paper. Section 2 contains some necessary definitions. The outline of the proof is presented in Section 3. Reactive linear bounded automata are introduced in Section 4, together with the description how they can be transformed into other non-flat systems. The proof of *PTIME*-hardness of equivalence checking in case of flat systems is presented in Section 5. The construction in this proof forms a base of the more complicated construction described in Section 6 where we show that equivalence checking is *EXPTIME*-hard for reactive linear bounded automata.

Acknowledgements. I would like to thank P. Schnoebelen for drawing my attention to equivalence checking of non-flat systems, and P. Jančar for fruitful discussions and comments.

2 Basic Definitions

2.1 Labelled Transition Systems

A *labelled transition system* (LTS) is a tuple $\mathcal{T} = (S, \mathcal{A}, \longrightarrow)$ where S is a set of *states*, $\mathcal{A}$ is the finite set of *actions*, and $\longrightarrow \subseteq S \times \{\mathcal{A} \cup \{\tau\}\} \times S$ is a *transition relation* where $\tau \notin \mathcal{A}$ is a special *invisible* action. We write $s \xrightarrow{a} s'$ instead of $(s, a, s') \in \longrightarrow$. Only *finite-state* LTSs where S is finite are considered in this paper.

An LTS $\mathcal{T} = (S, \mathcal{A}, \longrightarrow)$ where S, $\mathcal{A}$ and $\longrightarrow$ are given explicitly is called *flat system* (FS) and the size $|\mathcal{T}|$ of FS $\mathcal{T}$ is $|S| + |\mathcal{A}| + |\longrightarrow|$.

More complicated LTSs can be created from FSs by parallel composition and hiding. In the parallel composition a visible action a is executed iff every LTS that has a in its alphabet executes it. Invisible actions are not synchronized, that is, when an LTS executes the invisible action τ, other LTSs do nothing. Formally, the *parallel composition* $\mathcal{T}_1 \parallel \cdots \parallel \mathcal{T}_n$ of LTSs $\mathcal{T}_1, \ldots, \mathcal{T}_n$ where $\mathcal{T}_i = (S_i, \mathcal{A}_i, \longrightarrow_i)$ for each $i \in I$ where $I = \{1, \ldots, n\}$, is the LTS $(S, \mathcal{A}, \longrightarrow)$ where $S = S_1 \times \cdots \times S_n$, $\mathcal{A} = \mathcal{A}_1 \cup \cdots \cup \mathcal{A}_n$, and $(s_1, \ldots, s_n) \xrightarrow{a} (s'_1, \ldots, s'_n)$ iff either

- $a \in \mathcal{A}$ and for every $i \in I$: if $a \in \mathcal{A}_i$, then $s_i \xrightarrow{a} s'_i$, and if $a \notin \mathcal{A}_i$, then $s_i = s'_i$,
- $a = \tau$ and for some $i \in I$ is $s_i \xrightarrow{\tau} s'_i$ and $s_j = s'_j$ for each $j \in I$ such that $j \neq i$.

Tuples from $S_1 \times \cdots \times S_n$ are called *global states*. In this paper only *binary* synchronizations are needed, where any $a \in \mathcal{A}$ belongs to at most two different $\mathcal{A}_i$.

Hiding of actions removes a set of visible actions from the alphabet of an LTS and relabels corresponding transitions with the invisible action τ. Formally, *hide* $\mathcal{B}$ *in* $\mathcal{T}_1$, where $\mathcal{T}_1$ is an LTS $(S_1, \mathcal{A}_1, \longrightarrow_1)$ and $\mathcal{B} \subseteq \mathcal{A}_1$, denotes the LTS $(S, \mathcal{A}, \longrightarrow)$ where $S = S_1$, $\mathcal{A} = \mathcal{A}_1 - \mathcal{B}$, and $s \xrightarrow{a} s'$ iff there is some $a' \in (\mathcal{A}_1 \cup \{\tau\})$ such that $s \xrightarrow{a'} s'$ and either $a \notin \mathcal{B}$ and $a = a'$ or $a' \in \mathcal{B}$ and $a = \tau$.

A *parallel composition with hiding* (PCH) is an LTS $\mathcal{T}$ given in the form *hide* $\mathcal{B}$ *in* $(\mathcal{T}_1 \parallel \cdots \parallel \mathcal{T}_n)$ where $\mathcal{T}_1, \ldots, \mathcal{T}_n$ are FSs. The size $|\mathcal{T}|$ of PCH $\mathcal{T}$ is $|\mathcal{T}_1| + \cdots + |\mathcal{T}_n| + |\mathcal{B}|$.

Other type of non-flat systems are 1-safe Petri nets. A *labelled net* is a tuple $\mathcal{N} = (P, T, F, \lambda)$, where P and T are finite sets of *places* and *transitions*, $F \subseteq (S \times T) \cup (T \times S)$ is the *flow relation*, and $\lambda : T \to \mathcal{A}$ is a labelling function that associates to each transition t a label $\lambda(t)$ taken from some given set of actions $\mathcal{A}$. Pairs from F are called *arcs*. We identify F with its characteristic function $(P \times T) \cup (T \times P) \to \{0, 1\}$. A *marking* is a mapping $M : P \to \mathbb{N}$. A *labelled Petri net* is a pair $N = (\mathcal{N}, M_0)$ where $\mathcal{N}$ is a labelled net and M_0 is the initial marking. A transition t is *enabled* at a marking M if $M(p) > 0$

for every p such that $(p,t) \in F$. If t is enabled in M, then it can *fire* and its firing leads to the successor marking M' which is defined for every place p by $M'(p) = M(p) + F(t,p) - F(p,t)$. Given $a \in \mathcal{A}$, we denote by $M \xrightarrow{a} M'$ that there is some transition t such that M enables transition t, the marking reached by the firing of t is M', and $\lambda(t) = a$. A Petri net is *1-safe* if $M(p) \leq 1$ for every place p and every reachable marking M.

Let $\mathcal{T} = (S, \mathcal{A}, \longrightarrow)$ be an LTS. A *trace* from $s \in S$ is any $w = a_1 \dots a_n \in \mathcal{A}^*$ such that there is a sequence $s_0, s_1, \dots, s_n \in S$ where $s_0 = s$ and $s_{i-1} \xrightarrow{a_i} s_i$ for every $1 \leq i \leq n$. The set of all traces from s is denoted $Tr(s)$. States s, s' are in *trace preorder*, written $s \sqsubseteq_{tr} s'$, iff $Tr(s) \subseteq Tr(s')$. States s, s' are *trace equivalent* iff $s \sqsubseteq_{tr} s'$ and $s' \sqsubseteq_{tr} s$.

A bisimulation over $\mathcal{T}$ is any relation $\mathcal{R} \subseteq S \times S$ satisfying the following two conditions for each $s, t \in S$ such that $s\mathcal{R}t$:

- if $s \xrightarrow{a} s'$ for some s', then $t \xrightarrow{a} t'$ for some t' such that $s'\mathcal{R}t'$, and
- if $t \xrightarrow{a} t'$ for some t', then $s \xrightarrow{a} s'$ for some s' such that $s'\mathcal{R}t'$.

(It is said that $s \xrightarrow{a} s'$ is *matched* by $t \xrightarrow{a} t'$, resp. $t \xrightarrow{a} t'$ by $s \xrightarrow{a} s'$.) States s, s' are *bisimilar*, written $s \sim s'$, iff there exists some bisimulation $\mathcal{R}$ such that $s\mathcal{R}s'$.

Let $\mathcal{R}$ be some binary relation defined over states of LTSs, such that $s \sim s'$ implies $s\mathcal{R}s'$, and $s\mathcal{R}s'$ implies $s \sqsubseteq_{tr} s'$, i.e., $\sim \subseteq \mathcal{R} \subseteq \sqsubseteq_{tr}$. The problem FS-EQ$_\mathcal{R}$ is defined as:

INSTANCE: An FS $\mathcal{T}$ and its two states s and s'.
QUESTION: Is $s \,\mathcal{R}\, s'$?

the problem PCH-EQ$_\mathcal{R}$ as:

INSTANCE: A PCH $\mathcal{T}$ and its two global states $(s_1, \dots, s_n)$ and $(s'_1, \dots, s'_n)$.
QUESTION: Is $(s_1, \dots, s_n) \,\mathcal{R}\, (s'_1, \dots, s'_n)$?

and the problem PN-EQ$_\mathcal{R}$ as:

INSTANCE: A labelled net $\mathcal{N}$ with two markings M, M', such that $(\mathcal{N}, M)$ and $(\mathcal{N}, M')$ are 1-safe Petri nets.
QUESTION: Is $M \,\mathcal{R}\, M'$?

The main results of the paper show that FS-EQ$_\mathcal{R}$ is *PTIME*-hard, and PCH-EQ$_\mathcal{R}$ and PN-EQ$_\mathcal{R}$ are *EXPTIME*-hard for any $\mathcal{R}$ satisfying $\sim \subseteq \mathcal{R} \subseteq \sqsubseteq_{tr}$.

2.2 Alternating Graphs

In the proof of *PTIME*-hardness of FS-EQ$_\mathcal{R}$ we show a logspace reduction from the *Alternating Graph Problem* (AGP) that is known to be *PTIME*-complete, see for example [2]. The definition of this problem follows.

An *alternating graph* is a directed graph $G = (V, E, t)$ where V is a finite set of nodes, $E \subseteq V \times V$ is a set of edges, and $t : V \to \{\wedge, \vee\}$ is a labelling function that partitions V into sets $V_\wedge$ and $V_\vee$ of disjunctive and conjunctive nodes. The set of successors of a node v, i.e., the set $\{v' \in V \mid (v, v') \in E\}$, is denoted by $\sigma(v)$.

The set of *successful* nodes W is the least subset of V with the property that if a node v is conjunctive and all nodes from $\sigma(v)$ are in W, or disjunctive and at least one node from $\sigma(v)$ is in W, then v also belongs to W. AGP is the problem whether a given node is successful:

INSTANCE: An alternating graph $G = (V, E, t)$ and a node $v \in V$.
QUESTION: Is v successful?

Let $\mathcal{P}(V)$ be the power set of V. Notice that W is the least fixed point of a function $f : \mathcal{P}(V) \to \mathcal{P}(V)$ where for $U \subseteq V$ is $v \in f(U)$ iff either $v \in V_\wedge$ and $(\forall v' \in \sigma(v) : v' \in U)$, or $v \in V_\vee$ and $(\exists v' \in \sigma(v) : v' \in U)$.

Let us have a node that has no successors. If this node is conjunctive, it is called an *accepting* node, and otherwise it is called a *rejecting* node. Notice that accepting nodes are always successful, rejecting nodes are never successful, and that W is nonempty iff G contains at least one accepting node.

2.3 Alternating Linear Bounded Automata

In the proof we use a logspace reduction from a well known *EXPTIME*-complete problem that is called ALBA-ACCEPT in this paper. It is a problem of deciding if a given alternating linear bounded automaton accepts a given word. Its definition follows, see [1] for further details.

A *linear bounded automaton* (LBA) is a tuple $A = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$, where Q is a set of control states, Σ is an input alphabet, Γ is a tape alphabet, $\delta \subseteq (Q - \{q_{acc}, q_{rej}\}) \times \Gamma \times Q \times \Gamma \times \{-1, 0, +1\}$ is a set of transitions, $q_0, q_{acc}, q_{rej} \in Q$ are an initial, accepting and rejecting state. The alphabet Γ contains left and right endmarkers $\vdash$ and $\dashv$.

A *configuration* of A is a triple $\alpha = (q, w, i)$ where q is the current state, $w = a_1 a_2 \ldots a_n$ is the tape content, and $1 \leq i \leq |w|$ is the head position. Only configurations where $w = \vdash w' \dashv$ and endmarkers do not occur in w' are allowed. The size $|\alpha|$ of α is $|w|$. A configuration $\alpha' = (q', w', i')$ is a *successor* of $\alpha = (q, w, i)$, written $\alpha \vdash_A \alpha'$ (or just $\alpha \vdash \alpha'$ when A is obvious), iff $(q, a, q', a', d) \in \delta$, w contains a on position i, $i' = i + d$, and w' is obtained from w by writing a' on position i. Endmarkers may not be overwritten, and the machine is constrained never to move left of the $\vdash$ nor right of the $\dashv$. Notice that when $\alpha \vdash \alpha'$, then $|\alpha| = |\alpha'|$. The initial configuration for an input $w \in \Sigma^*$ is $\alpha_{ini}(w) = (q_0, \vdash w \dashv, 1)$. A configuration is *accepting* iff $q = q_{acc}$, and *rejecting* iff $q = q_{rej}$.

An *alternating LBA* (ALBA) is an LBA extended with a function $l : Q \to \{\wedge, \vee\}$ that labels each state as either conjunctive or disjunctive. We extend l to configurations in an obvious manner and so also configurations are labeled as conjunctive and disjunctive. A configuration is *successful* iff it is either accepting, or disjunctive with at least one successful successor, or conjunctive with all successors successful. An ALBA A accepts an input $w \in \Sigma^*$ iff $\alpha_{ini}(w)$ is successful.

The problem ALBA-ACCEPT is defined as:

INSTANCE: An ALBA A and a word $w \in \Sigma^*$.
QUESTION: Does A accept w?

Notice that there is a close relationship between AGP and ALBA-ACCEPT. A computation of an ALBA can be viewed as an alternating graph where successful nodes correspond to successful configurations. The size of this graph can be exponentially larger than the size of the corresponding instance of ALBA-ACCEPT.

3 Outline of the Proof

Reactive linear bounded automata (RLBA) are introduced in Section 4 and it is shown that they can be modeled by other types of non-flat systems, in particular by PCH and by 1-safe Petri nets. An RLBA is similar to a usual LBA, but is intended to generate an LTS, not to accept or reject an input. The equivalence checking problem where the instance is an RLBA and two of its configurations is denoted RLBA-EQ$_{\mathcal{R}}$ in this paper.

The main technical result of the paper shows that RLBA-EQ$_{\mathcal{R}}$ is *EXPTIME*-hard for any relation $\mathcal{R}$ satisfying $\sim \subseteq \mathcal{R} \subseteq \sqsubseteq_{tr}$. From this follows *EXPTIME*-hardness of equivalence checking for every model that is able to model an RLBA.

To show *EXPTIME*-hardness of RLBA-EQ$_{\mathcal{R}}$, we present a logspace reduction from ALBA-ACCEPT. The construction in the proof is based on a simpler construction that can be used to show *PTIME*-hardness of FS-EQ$_{\mathcal{R}}$. This simpler construction is presented in Section 5, where we show a logspace reduction from AGP to the complement of FS-EQ$_{\mathcal{R}}$ that works for any $\mathcal{R}$ such that $\sim \subseteq \mathcal{R} \subseteq \sqsubseteq_{tr}$. The basic idea is to construct an LTS with two distinguished states s, s', such that $s \not\sqsubseteq_{tr} s'$ if the answer to the original problem is YES, and $s \sim s'$ otherwise. The same construction can be used for every $\mathcal{R}$, because $s \not\sqsubseteq_{tr} s'$ implies that not $s\mathcal{R}s'$, and $s \sim s'$ implies $s\mathcal{R}s'$. The same idea was also used for example in [3] and [6]. We can conclude that the complement of FS-EQ$_{\mathcal{R}}$ is *PTIME*-hard for any $\mathcal{R}$, and so also FS-EQ$_{\mathcal{R}}$ is *PTIME*-hard because *PTIME* is closed under complement. *PTIME*-hardness of FS-EQ$_{\mathcal{R}}$ was already proved in [7], but the reduction presented here is simpler.

Now consider ALBA-ACCEPT, a well known *EXPTIME*-complete problem. A computation of an ALBA can be viewed as an alternating graph, where successful nodes correspond to successful configurations, and this allows us to 'shift' the previous result 'higher' in the complexity hierarchy. We will construct an RLBA that will model the LTS which we would obtain when we would apply the above mentioned reduction to the alternating graph corresponding to the computation of the ALBA. Moreover, logarithmic space will be sufficient for the construction of this RLBA from the instance of ALBA-ACCEPT.

4 Reactive Linear Bounded Automata

Reactive linearly bounded automata are introduced in this section. A *reactive linear bounded automaton* (RLBA) is like a usual LBA, but it has special control states, called *reactive* states, where it can perform actions from some given set of actions $\mathcal{A}$. Only the control state is changed after performing such actions, neither the tape content nor the head position is modified. The other control states are called *computational* and RLBA performs steps as a usual LBA in them. Each such step is represented as the invisible action τ.

Formally, an RLBA is a tuple $B = \{Q, \Gamma, \delta, \mathcal{A}, l, \longrightarrow\}$, where the meaning of Q, Γ and δ is the same as in a usual LBA, $\mathcal{A}$ is the finite set of actions, the function $l : Q \to \{r, c\}$ partitions Q into sets Q_r and Q_c of reactive and computational states, and $\longrightarrow \subseteq Q_r \times (\mathcal{A} \cup \{\tau\}) \times Q$ is the transition relation (we write $q \xrightarrow{a} q'$ instead of $(q, a, q') \in \longrightarrow$). It is also required that if $(q, b, q', b', d) \in \delta$ then $q \in Q_c$. The definition of a configuration and a successor relation is the same as for a usual LBA.

An RLBA B generates an LTS $\mathcal{T}(B) = (S, \mathcal{A}, \longrightarrow)$, where S is the set of configurations of B, and $(q, w, i) \xrightarrow{a} (q', w', i')$ iff either $q \in Q_c$, $(q, w, i) \vdash (q', w', i')$ and $a = \tau$, or $q \in Q_r$, $q \xrightarrow{a} q'$, $w = w'$ and $i = i'$.

For each $\mathcal{R}$, such that $\sim \subseteq \mathcal{R} \subseteq \sqsubseteq_{tr}$, we can define the problem RLBA-EQ$_{\mathcal{R}}$:

INSTANCE: An RLBA B and its two configurations α, α' of size n.
QUESTION: Is $\alpha \, \mathcal{R} \, \alpha'$?

An RLBA with a configuration of size n can be easily modeled by various non-flat systems, as two following lemmas show.

Lemma 1. *There is a logspace reduction from* RLBA-EQ$_{\mathcal{R}}$ *to* PCH-EQ$_{\mathcal{R}}$.

Proof. Let us have an RLBA B and two its configurations of size n. We construct a PCH $\mathcal{T}$ of the form *hide* $\mathcal{B}$ *in* $(\mathcal{T}_c \parallel \mathcal{T}_1 \parallel \cdots \parallel \mathcal{T}_n)$ which models the LTS generated by B. In particular, $\mathcal{T}_c$ models the control unit, and $\mathcal{T}_1, \ldots, \mathcal{T}_n$ model the tape cells of B. A state of $\mathcal{T}_c$ represents the current control state and head position, and a state of $\mathcal{T}_i$ represents the symbol on the i-th position of the tape.

Let $I = \{1, \ldots, n\}$ be the set of all possible positions of the head. For each $i \in I$ is $\mathcal{T}_i = (S_i, \mathcal{A}_i, \longrightarrow_i)$ where $S_i = \Gamma$, $\mathcal{A}_i = \{\langle b, b', i\rangle \mid b, b' \in \Gamma\}$, and $\longrightarrow_i$ contains transitions $b \xrightarrow{\langle b,b',i\rangle} b'$ for each $b, b' \in \Gamma$.

In $\mathcal{T}_c = (S_c, \mathcal{A}_c, \longrightarrow_c)$ is $S_c = \{\langle q, i\rangle \mid q \in Q, i \in I\}$ and $\mathcal{A}_c = \mathcal{A} \cup \mathcal{A}_0 \cup \cdots \cup \mathcal{A}_n$ (w.l.o.g. we can assume that $\mathcal{A} \cap \mathcal{A}_i = \emptyset$ for each $i \in I$). To $\longrightarrow_c$ we add for each

$(q,b,q',b',d) \in \delta$ and $i \in I$, such that $i+d \in I$, a transition $\langle q,i\rangle \xrightarrow{\langle b,b',i\rangle} \langle q', i+d\rangle$, and for each $q \in Q_r$, $q' \in Q$, $a \in (Act \cup \{\tau\})$ and $i \in I$ where $q \xrightarrow{a} q'$ we add a transition $\langle q,i\rangle \xrightarrow{a} \langle q',i\rangle$.

The set $\mathcal{B}$ in $\mathcal{T}$ is defined as $\mathcal{A}_1 \cup \cdots \cup \mathcal{A}_n$. Each configuration $\alpha = (q, a_1 a_2 \cdots a_n, i)$ has a corresponding global state $g(\alpha) = (\langle q,i\rangle, a_1, a_2, \ldots, a_n)$. As can be easily checked, $\alpha \xrightarrow{a} \alpha'$ in B iff $g(\alpha) \xrightarrow{a} g(\alpha')$ in $\mathcal{T}$, and so $\alpha \mathcal{R} \alpha'$ in B iff $g(\alpha)\mathcal{R}g(\alpha')$ for any $\mathcal{R}$ such that $\sim \subseteq \mathcal{R} \subseteq \sqsubseteq_{tr}$. It is obvious that $\mathcal{T}$ can be constructed from B in a logarithmic space. □

Lemma 2. *There is a logspace reduction from* RLBA-EQ$_\mathcal{R}$ *to* PN-EQ$_\mathcal{R}$.

Proof. Let us have an instance of RLBA-EQ$_\mathcal{R}$, i.e. an RLBA B and two of its configurations of size n. We construct a labelled net as follows. Let $I = \{1,\ldots,n\}$. The set of places will be $Q \cup \{\langle a,i\rangle \mid a \in \Gamma,\, i \in I\} \cup I$.

For each $(q,b,q',b',d) \in \delta$ and $i \in I$ where $q \in Q_c$ and $i+d \in I$ we add a transition $t = \langle q,b,q',b',i,i+d\rangle$ labelled with τ together with incoming arcs (q,t), $(\langle b,i\rangle,t)$, and (i,t), and outgoing arcs (t,q'), $(t,\langle s',i\rangle)$, and $(t,i+d)$.

For each $q,q' \in Q$ and $a \in (\mathcal{A} \cup \{\tau\})$ where $q \in Q_r$ and $q \xrightarrow{a} q'$ we add a new transition $t = \langle q,a,q'\rangle$ labelled with a together with an incoming arc $\langle q,t\rangle$ and an outgoing arc $\langle t,q'\rangle$.

For a configuration $\alpha = \{q, a_1 a_2 \cdots a_n, i\}$ we define a corresponding marking M_α where $M_\alpha(p) = 1$ if p is q, i, or $\langle a_j, j\rangle$ where $j \in I$, and $M_\alpha(p) = 0$ otherwise. It is easy to check that $\alpha \xrightarrow{a} \alpha'$ iff $M_\alpha \xrightarrow{a} M_{\alpha'}$. □

5 Construction for Flat Systems

A logspace reduction from AGP to the complement of FS-EQ$_\mathcal{R}$ is presented in this section. For a given alternating graph $G = (V,E,t)$ with a distinguished node x we construct a corresponding LTS $\mathcal{T}_G = (S, \mathcal{A}, \longrightarrow)$ with two distinguished states $s, s' \in S$ such that $s \not\sqsubseteq_{tr} s'$ if x is successful, and $s \sim s'$ otherwise.

The set of states S is V. For each $v \in V$ we define a set of corresponding actions $Act(v)$. If $v \in V_\wedge$, then $Act(v) = \{\langle v\rangle\}$, and if $v \in V_\vee$, then $Act(v) = \{\langle v,i\rangle \mid 1 \leq i \leq |\sigma(v)|\}$. The set of actions $\mathcal{A}$ is $\bigcup_{v \in V} Act(v)$. We assume w.l.o.g. that successors of each node are ordered in some fixed order. The i-th successor of v where $i \in \{1,\ldots,|\sigma(v)|\}$ is denoted by $\sigma_i(v)$.

The transition relation contains transitions of three types:

1. $v \xrightarrow{\langle a\rangle} v$ for each $v \in V$ and $a \in \mathcal{A}$ such that $a \notin Act(v)$.
2. $v \xrightarrow{\langle v,i\rangle} v'$ for each $v \in V_\vee$ and $i \in \{1,\ldots,|\sigma(v)|\}$ where $v' = \sigma_i(v)$.
3. $v \xrightarrow{\langle u\rangle} u'$ for each $v \in V$, $u \in V_\wedge$ and $u' \in V$ such that $u' \in \sigma(u)$.

We may assume w.l.o.g. that G contains at least one rejecting node z. The instance of FS-EQ$_{\mathcal{R}}$ then consists of $\mathcal{T}_G$ and states z and x, where x is the distinguished node from the instance of AGP.

Proposition 3. *If $v \in V$ is not successful then $z \sim v$.*

Proof. It is sufficient to show that $\{(z,v) \mid v \in (V-W)\} \cup \mathit{Id}$ is a bisimulation (Id denotes the identity relation $\{(v,v) \mid v \in V\}$). Let us consider some pair (z,v) where $v \in (V-W)$, and a transition $v \xrightarrow{a} v'$. This transitions is either of:

- type 1 and then it is matched by $z \xrightarrow{a} z$ of type 1, because $\mathit{Act}(z) = \emptyset$ and so $z \xrightarrow{a} z$ for every $a \in \mathcal{A}$,
- type 2 and then $v \in V_\vee$ and because v is unsuccessful, each $v' \in \sigma(v)$ is also unsuccessful, and so $v \xrightarrow{a} v'$ is matched by $z \xrightarrow{a} z$,
- type 3 and then it can be matched by $z \xrightarrow{a} v'$ of type 3.

Now consider a transition of the form $z \xrightarrow{a} z'$. It is of:

- type 1 and then $z' = z$ and either $a \notin \mathit{Act}(v)$, and $z \xrightarrow{a} z$ is matched by $v \xrightarrow{a} v$ of type 1, or $a \in \mathit{Act}(v)$ and there are two possibilities:
  - if $v \in V_\vee$ then each $v' \in \sigma(v)$ is unsuccessful since v is unsuccessful, and so $z \xrightarrow{a} z$ can be matched by $v \xrightarrow{a} v'$ of type 2,
  - if $v \in V_\wedge$ then there is at least one unsuccessful $v' \in \sigma(v)$, and so $z \xrightarrow{a} z$ can be matched by $v \xrightarrow{a} v'$ of type 3,
- type 2, but this is not possible as $\mathit{Act}(z) = \emptyset$,
- type 3 and it can be matched by $v \xrightarrow{a} z'$ of type 3. □

Proposition 4. *There is $w \in \mathcal{A}^*$ such that if $v \in V$ is successful, then $w \notin \mathit{Tr}(v)$.*

Proof. As W can be computed as the least fixed point of f, we can define a sequence $W_0 \subseteq W_1 \subseteq W_2 \subseteq \cdots$ of subsets of W where $W_0 = \emptyset$ and $W_{i+1} = f(W_i)$ for $i > 0$. For each $v \in W$ there is some least i such that $v \in W_i$. This i is denoted $\mathit{rank}(v)$. Let $m = |W|$, and let $v_1, v_2, \ldots, v_m$ be the nodes in W ordered by their rank, i.e., if $i < j$ then $\mathit{rank}(v_i) \leq \mathit{rank}(v_j)$.

Let us consider a word $w_m = a_m a_{m-1} \cdots a_1$ where $a_i = \langle v_i \rangle$ if $v_i \in V_\wedge$, and if $v_i \in V_\vee$ then $a_i = \langle v_i, k \rangle$ where we choose k such that $v' = \sigma_k(v_i)$ is successful and $\mathit{rank}(v') < \mathit{rank}(v_i)$ (obviously there is at least one such v'). We show that $w_m \notin \mathit{Tr}(v)$ for any successful node v. In particular, for each $i \leq m$ we show that $w_i = a_i a_{i-1} \cdots a_1 \notin \mathit{Tr}(v_j)$ if $j \leq i$. We proceed by induction on i and in the proof we use the following simple observation: $w_i \notin \mathit{Tr}(v)$ iff for each v' such that $v \xrightarrow{a_i} v'$ is $w_{i-1} \notin \mathit{Tr}(v')$.

The base case $(i = 0)$ is trivial. In the induction step we consider $i > 0$ and show that the proposition holds for every v_j where $1 \leq j \leq i$.

If $v_i \in V_\vee$ then $a_i = \langle v_i, k\rangle$. Any transition of the form $v_j \xrightarrow{a_i} v'$ is either of type 1, and then $v' = v_j$ and $j < i$, and by induction hypothesis $w_{i-1} \notin Tr(v')$, or of type 2, and then $v' = \sigma_k(v_i)$, so v' is successful and $rank(v') < rank(v)$, and by induction hypothesis $w_{i-1} \notin Tr(v')$.

If $v_i \in V_\wedge$ then $a_i = \langle v_i\rangle$. Any transition of the form $v_j \xrightarrow{a_i} v'$ is either of type 1, and then $v' = v_j$ and $j < i$, and by induction hypothesis $w_{i-1} \notin Tr(v')$, or of type 3, and then $v' \in \sigma(v_i)$ and so v' is successful and $rank(v') < rank(v_i)$, so by induction hypothesis $w_{i-1} \notin Tr(v')$. $\square$

Notice that $z \xrightarrow{a} z$ for each $a \in \mathcal{A}$, because $Act(z) = \emptyset$, and so $Tr(z) = \mathcal{A}^*$. From this and Proposition 4 we have that $z \not\sqsubseteq_{tr} x$ if x is successful. On the other hand, from Proposition 3 we have that $z \sim x$ if x is not successful, and so the described construction is correct.

The described reduction can be obviously performed in a logarithmic space. Since the problem AGP is *PTIME*-complete and *PTIME* is closed under complement, we obtain the following result:

Lemma 5. FS-EQ$_\mathcal{R}$ *is PTIME-hard for any* $\mathcal{R}$ *such that* $\sim\subseteq \mathcal{R} \subseteq\sqsubseteq_{tr}$.

6 Construction for Non-Flat Systems

The description of the reduction from ALBA-ACCEPT to the complement of RLBA-EQ$_\mathcal{R}$ consists of several steps that are summarized in the following figure where $\overline{\text{FS-EQ}_\mathcal{R}}$ and $\overline{\text{RLBA-EQ}_\mathcal{R}}$ denote the complements of FS-EQ$_\mathcal{R}$ and RLBA-EQ$_\mathcal{R}$:

$$\begin{tikzcd}
\begin{array}{c}\text{ALBA-ACCEPT}\\(A,w_0)\end{array} \arrow[rr, "(3)", "\text{LOGSPACE red.}"'] \arrow[d, dashed, "(2)"'] & & \begin{array}{c}\overline{\text{RLBA-EQ}_\mathcal{R}}\\(B)\end{array} \arrow[d, dotted, no head, "\sim\ (5)"] \\
\begin{array}{c}\text{AGP}\\(G_A)\end{array} \arrow[r, "(1)", "\text{LOGSPACE red.}"'] & \begin{array}{c}\overline{\text{FS-EQ}_\mathcal{R}}\\(\mathcal{T}_A)\end{array} \arrow[r, dashed, "(4)"] & \begin{array}{c}\overline{\text{FS-EQ}_\mathcal{R}}\\(\mathcal{T}'_A)\end{array}
\end{tikzcd}$$

The reduction (1) from AGP to $\overline{\text{FS-EQ}_\mathcal{R}}$ can be applied to the alternating graph G_A that corresponds (2) to the ALBA A in the instance of ALBA-ACCEPT. We obtain an LTS $\mathcal{T}_A$. From the instance of ALBA-ACCEPT we construct (3) a RLBA B that models $\mathcal{T}_A$ in the sense, that after we apply a certain kind of transformation (4) to $\mathcal{T}_A$, we obtain an LTS $\mathcal{T}'_A$ bisimilar (5) with B. It will be proved that the transformation (4) preserves some important properties, in particular, states that were bisimilar are bisimilar after the transformation, and states that were not in trace preorder are not in trace preorder after the transformation. Bisimilarity (5) implies that the same is true for corresponding configurations of B, from which the correctness of the construction (3) follows. The *EXPTIME*-hardness of $\overline{\text{RLBA-EQ}_{\mathcal{R}}}$ implies the *EXPTIME*-hardness of RLBA-EQ$_{\mathcal{R}}$ since *EXPTIME* is closed under complement.

The rest of the paper is devoted to the description of a logspace reduction from ALBA-ACCEPT to $\overline{\text{RLBA-EQ}_{\mathcal{R}}}$.

Let an ALBA $A = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$ with a word $w_0 \in \Sigma^*$ be an instance of ALBA-ACCEPT. We can assume that transitions in δ are ordered and that this ordering determines the order of successors of a configuration. For simplicity we can assume w.l.o.g. that each configuration of A, which is not accepting nor rejecting, has exactly two successors, and that $l(q_{acc}) = \wedge$ and $l(q_{rej}) = \vee$. Let *Conf* be the set of all configurations of A of size $n = |w_0| + 2$, and let $Conf_\wedge$, $Conf_\vee$, and $Conf_{rej}$ be the sets of conjunctive, disjunctive, and rejecting configurations of size n, respectively. Notice that any configuration reachable from $\alpha_0 = \alpha_{ini}(w_0)$ is of size n.

The ALBA A has a corresponding alternating graph $G_A = (V, E, t)$, where $V = Conf$, $(\alpha, \alpha') \in E$ iff $\alpha \vdash \alpha'$, and $t(\alpha) = l(\alpha)$ for each $\alpha \in Conf$. Notice that a configuration α is successful in A iff the node α is successful in G_A, and that A accepts w iff the node α_0 is successful.

When we apply the logspace reduction described in Section 5 to G_A with a node α_0, we obtain the LTS $\mathcal{T}_A = (S_A, \mathcal{A}_A, \longrightarrow_A)$, where $S_A = Conf$, $Act(\alpha) = \{\langle\alpha\rangle\}$ for each $\alpha \in Conf_\wedge$, $Act(\alpha) = \{\langle\alpha, i\rangle \mid i \in \{1,2\}\}$ for each $\alpha \in Conf_\vee - Conf_{rej}$, $Act(\alpha) = \emptyset$ for each $\alpha \in Conf_{rej}$, $\mathcal{A}_A = \bigcup_{\alpha \in Conf} Act(\alpha)$, and $\longrightarrow_A$ contains the following transitions for each $\alpha \in Conf$:

1. $\alpha \xrightarrow{x}_A \alpha$ for each $x \in (\mathcal{A}_A - Act(\alpha))$,
2. $\alpha \xrightarrow{\langle\alpha,i\rangle}_A \alpha'$ if $\alpha \in Conf_\vee$, and α' is the i-th successor of α,
3. $\alpha \xrightarrow{\langle\beta\rangle}_A \beta'$ for each $\beta \in Conf_\wedge$ and $\beta' \in Conf$ such that $\beta \vdash \beta'$.

Let $\alpha_{rej} \in Conf_{rej}$ be some rejecting configuration. The states α_{rej} and α_0 are the two distinguished states with the property, that if A accepts w_0, then $\alpha_{rej} \not\sqsubseteq_{tr} \alpha_0$, and $\alpha_{rej} \sim \alpha_0$ otherwise.

An RLBA $B = (Q_B, \Gamma_B, \delta_B, \mathcal{A}_B, l_B, \longrightarrow_B)$ that in some sense 'models' $\mathcal{T}_A$ will be constructed. The RLBA B will be described only informally, but it should be clear from this description how to construct it. In fact B models an LTS that we obtain from $\mathcal{T}_A$ by a transformation illustrated in Figure 1.

Figure 1 shows only transitions going from one state, but the same transformation is performed for all states and transitions. In this simplified example is $\mathcal{A}_A = \{a, b, c, d\}$. At first, the non-deterministic choice is postponed. Notice that that a new state is added for each action in $\mathcal{A}_A$. Next, each action from $\mathcal{A}_A$ is replaced by sequence of actions from some 'small' alphabet $\mathcal{A}_B$. In our example is $\mathcal{A}_B = \{0, 1\}$ and a, b, c, d are replaced with 00, 01, 10 and 11. Invisible actions

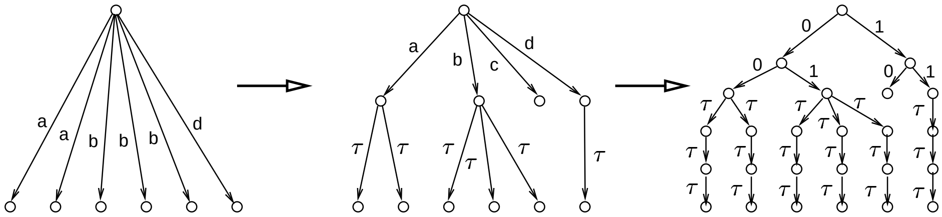

Fig. 1. The transformation performed on $\mathcal{T}_A$

representing non-deterministic choice are replaced with sequences of τ actions of some fixed length m (in our example $m = 3$). This kind of transformation is described more formally in the next subsection.

Configurations of A can be written as words in an alphabet $\Delta = (Q \times \Gamma) \cup \Gamma$, where occurrence of the symbol from $Q \times \Gamma$ denotes the position of the head (there must be exactly one such symbol in the word). A word from Δ^* corresponding to a configuration α is denoted by $desc(\alpha)$. Actions from $Act(\alpha)$ are replaced with sequences of actions corresponding to $desc(\alpha)$ in B. In particular, $\mathcal{A}_B = \Delta_{\mathcal{A}} \cup \{1, 2\}$ where $\Delta_{\mathcal{A}} = \Delta - \{(q_{rej}, a) \mid a \in \Gamma\}$. Actions from $\{1, 2\}$ are used to identify a successor of a disjunctive configuration.

B has a tape with two tracks, denoted track 1 and 2, respectively. A current state α of $\mathcal{T}_A$ is is stored as a word $desc(\alpha)$ on track 1. B also needs to store information about the label of a transition that $\mathcal{T}_A$ performs. The configuration from the label of the transition is stored on track 2. Formally this means that $\Gamma_B = (\Delta \times \Delta) \cup \{\vdash, \dashv\}$. See Figure 2 for an example:

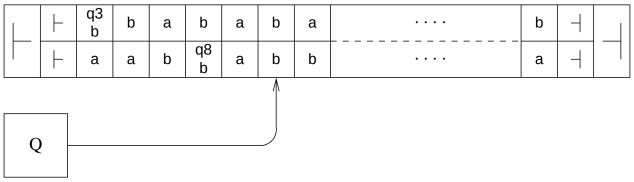

Fig. 2. An example of a configuration of the RLBA B

As mentioned above, a transition of $\mathcal{T}_A$ labelled with an action from $Act(\beta)$ is represented in B as a sequence of transitions. Each such sequence start and ends in a configuration where track 1 contains the current state α of $\mathcal{T}_A$ and where the head of B points to the first symbol of $desc(\alpha)$, i.e., it is on position 2. The contents of track 2 is not important, since it will be overwritten. The sequence of transitions of B corresponding to one transition of $\mathcal{T}_A$ has two phases (denoted as phase 1 and phase 2):

1. An actions representing symbols of $desc(\beta)$ are performed one by one and the corresponding symbols are stored on track 2. The head of B goes from left to right.
2. One of the possibilities is chosen non-deterministically (possibilities depend on some properties of α and β that are described below, information about these properties can be kept in the control unit of B):
   (a) The head of B moves back to the left endmarker without changing anything.
   (b) A chosen successor of β is stored on track 1 while the head returns back to the left endmarker. This involves copying of track 2 to track 1 with the necessary modifications on positions where β and its successor differ.

The three following steps are performed for each symbol a of $desc(\beta)$ during phase 1:

- the symbol from track 1 is read into the control unit,
- an action a is performed, and remembered in the control unit,
- a is written on track 2 and the head moves to the next cell.

This means that actions $\tau a \tau$ are performed for each symbol a. Phase 1 ends when the right endmarker $\dashv$ is reached. If $\beta \in Conf_{\vee}$, then phase 1 includes also an action $a \in \{1, 2\}$ identifying a successor of β. This number is stored in the control unit of B.

The possible choices at the start of phase 2 depend only on whether $\alpha = \beta$, and on the type of β (if it is accepting, conjunctive or disjunctive). This information can be stored in the control unit of B. To find out if $\alpha = \beta$, notice that we can compare symbols on tracks 1 and 2 during phase 1. The possible non-deterministic choices are the following: if β is disjunctive, the successor of β that was chosen at the end of phase 1 can be stored on track 1, and if β is conjunctive, the non-deterministically chosen successor of β can be stored on track 1. The choice (a), i.e., to keep track 1 intact, is possible only when $\alpha \neq \beta$. Notice that if β is accepting, there are no successors of β and so there are no transitions possible when $\alpha = \beta$.

B can be constructed in such a way that only valid configurations can be written on track 2 during phase 1, and some fixed number m of steps is always performed during phase 2, where $m \in \mathcal{O}(n)$. In particular, we can put $m = 2n + 4$, because two steps are needed to copy one symbol from track 2 to track 1, and we need two additional steps to modify track 1 to reflect one step of A. We also need additional steps at the beginning and at the end of phase 2.

6.1 Decomposition of Transitions

In this subsection we describe the transformation performed on $\mathcal{T}_A$ more formally and we show that it preserves some important properties of the original LTS.

Let us have an LTS $\mathcal{T} = (S, \mathcal{A}, \longrightarrow)$, a set of actions $\mathcal{A}'$, some positive integer m and a mapping $h : \mathcal{A} \to \mathcal{A}'^*$ such that $h(a)$ is not prefix of $h(a')$ if $a \neq a'$. Let $H = \{h(a) \mid a \in \mathcal{A}\}$ and let $\mathit{Pref}(H)$ be the set of all prefixes of words from H.

We can construct a new LTS $\mathcal{T}' = (S', \mathcal{A}', \longrightarrow')$ where $S' = \{\langle s, w\rangle \mid s \in S, w \in \mathit{Pref}(H)\} \cup \{\langle s, i\rangle \mid s \in S, 0 \leq i < m\}$, where we identify the the states $\langle s, 0\rangle$ and $\langle s, \varepsilon\rangle$ (i.e., $\langle s, 0\rangle$ and $\langle s, \varepsilon\rangle$ are the same state), and where $\longrightarrow'$ contains transitions:

- $\langle s, w\rangle \xrightarrow{a} \langle s, wa\rangle$ for each $s \in S$, $w \in \mathcal{A}'^*$ and $a \in \mathcal{A}'$ such that $wa \in \mathit{Pref}(H)$,
- $\langle s, w\rangle \xrightarrow{\tau} \langle s', m-1\rangle$ for each $s, s' \in S$ and $a \in \mathcal{A}$ such that $s \xrightarrow{a} s'$ and $h(a) = w$,
- $\langle s, i\rangle \xrightarrow{\tau} \langle s, i-1\rangle$ for each $s \in S$ and $0 < i < m$.

For each state $s \in S$ in $\mathcal{T}$ there is a corresponding state $\langle s, \varepsilon\rangle \in S'$ in $\mathcal{T}'$.

Lemma 6. *For each $s, s' \in S$: if $s \sim s'$, then $\langle s, \varepsilon\rangle \sim \langle s', \varepsilon\rangle$, and if $s \not\sqsubseteq_{tr} s'$, then $\langle s, \varepsilon\rangle \not\sqsubseteq_{tr} \langle s', \varepsilon\rangle$.*

Proof (sketch). To prove the first part of the lemma, it is sufficient to show that $R = \{(\langle s, w\rangle, \langle t, w\rangle) \mid s \sim t, w \in \mathit{Pref}(H)\} \cup \{(\langle s, i\rangle, \langle t, i\rangle) \mid s \sim t, 0 < i < m\}$ is a bisimulation.

To prove the second part, let us define a mapping $\hat{h} : \mathcal{A}^* \to \mathcal{A}'^*$ such that $\hat{h}(\varepsilon) = \varepsilon$, and $\hat{h}(aw) = h(a)\tau^m \hat{h}(w)$. By induction on $|w|$ we can show that for every $s \in S$ and $w \in \mathcal{A}^*$ is $w \in \mathit{Tr}(s)$ iff $\hat{h}(w) \in \mathit{Tr}(\langle s, \varepsilon\rangle)$.

If $s \not\sqsubseteq_{tr} s'$ then there is some $w \in \mathcal{A}^*$ such that $w \in \mathit{Tr}(s)$ and $w \notin \mathit{Tr}(s')$. This implies that $\hat{h}(w) \in \mathit{Tr}(\langle s, \varepsilon\rangle)$ and $\hat{h}(w) \notin \mathit{Tr}(\langle s', \varepsilon\rangle)$. $\square$

6.2 Correctness of the Construction of the RLBA

Theorem 7. *The problem* RLBA-EQ$_{\mathcal{R}}$ *is EXPTIME-hard for any $\mathcal{R}$ such that $\sim \subseteq \mathcal{R} \subseteq \sqsubseteq_{tr}$.*

Proof. Let us return to the construction of B and consider the corresponding $\mathcal{T}_A$. We define a mapping $h : \mathcal{A}_A \to \mathcal{A}_B^*$ such that $h(\langle \alpha\rangle) = \tau a_1 \tau\tau a_2 \tau \cdots \tau a_n \tau$ for $\alpha \in \mathit{Conf}_\wedge$, and $h(\langle \alpha, i\rangle) = \tau a_1 \tau\tau a_2 \tau \ldots \tau a_n \tau i$ for $\alpha \in \mathit{Conf}_\vee - \mathit{Conf}_{rej}$, where $\mathit{desc}(\alpha) = a_1 a_2 \cdots a_n$. We apply the transformation described in the previous subsection with h and $m = 2n + 4$ to $\mathcal{T}_A$, and we obtain $\mathcal{T}_A'$. It is straightforward to create a bisimulation that relates configurations of B and states of $\mathcal{T}_A'$.

States α_{rej} and α_0 from $\mathcal{T}_A$ correspond to a rejecting, resp. initial, configuration of A. If A accepts w, then $\alpha_{rej} \not\sqsubseteq_{tr} \alpha_0$ in $\mathcal{T}_A$, and so $\langle \alpha_{rej}, \varepsilon\rangle \not\sqsubseteq_{tr} \langle \alpha_0, \varepsilon\rangle$ in $\mathcal{T}_A'$, and if A does not accept w, then $\alpha_{rej} \sim \alpha_0$ in $\mathcal{T}_A$ and $\langle \alpha_{rej}, \varepsilon\rangle \sim \langle \alpha_0, \varepsilon\rangle$ in $\mathcal{T}_A'$

by Lemma 6. The same holds for the corresponding configurations of B. This shows that the described construction is correct.

RLBA B with two configurations can be constructed from an instance of ALBA-ACCEPT in a logarithmic space, since it is obvious that some fixed number of pointers pointing to symbols in the instance would be sufficient for the construction. The problem ALBA-ACCEPT is *EXPTIME*-complete and *EXPTIME* is closed under complement. $\square$

So from Theorem 7 and Lemmas 1 and 2 we obtain the main result of the paper:

Theorem 8. *The problems* PCH-EQ$_{\mathcal{R}}$ *and* PN-EQ$_{\mathcal{R}}$ *are EXPTIME-hard for any* $\mathcal{R}$ *such that* $\sim \subseteq \mathcal{R} \subseteq \sqsubseteq_{tr}$.

References

1. A.K. Chandra, D.C. Kozen, and L.J. Stockmeyer. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133, January 1981.
2. N. Immerman. *Descriptive Complexity*, pages 53–54. Springer-Verlag, 1998.
3. P. Jančar. Nonprimitive recursive complexity and undecidability for petri net equivalences. *Theoretical Computer Science*, 256:23–30, 2001.
4. L. Jategaonkar and A. R. Meyer. Deciding true concurrency equivalences on safe, finite nets. *Theoretical Computer Science*, 154(1):107–143, January 1996.
5. F. Laroussinie and P. Schnoebelen. The state explosion problem from trace to bisimulation equivalence. In *Proc. FOSSACS'2000 (Berlin, Germany, Mar.-Apr. 2000)*, volume 1784, pages 192–207. Springer, 2000.
6. A. Rabinovich. Complexity of equivalence problems for concurrent systems of finite agents. *Information and Computation*, 139(2):111–129, 15 December 1997.
7. Z. Sawa and P. Jančar. *P*-hardness of equivalence testing on finite-state processes. In *Proc. SOFSEM 2001 (Piestany, Slovak Rep., November 2001)*, volume 2234 of *Lecture Notes in Computer Science*, page 326. Springer, 2001.
8. S. K. Shukla, H. B. Hunt, D. J. Rosenkrantz, and R. E. Stearns. On the complexity of relational problems for finite state processes. In *Proc. ICALP'96 (Paderborn, Germany)*, volume 1099 of *Lecture Notes in Computer Science*, pages 466–477. Springer-Verlag, 1996.
9. A. Valmari and A. Kervinen. Alphabet-based synchronisation is exponentially cheaper. In *Proc. CONCUR 2002*, volume 2421 of *Lecture Notes in Computer Science*, page 161. Springer, 2002.
10. R.J. van Glabbeek. The Linear Time - Branching Time Spectrum. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings of CONCUR '90, Theories of Concurrency: Unification and Extension*, volume 458 of *Lecture Notes in Computer Science*, pages 278–297. Springer-Verlag, Berlin, 1990.