The state explosion problem from trace to bisimulation equivalence

F. Laroussinie and Ph. Schnoebelen

Lab. Spécification & Vérification, ENS de Cachan & CNRS UMR 8643,
61, av. Pdt. Wilson, 94235 Cachan Cedex France
email: {fl,phs}@lsv.ens-cachan.fr

Abstract. We show that any relation between the simulation preorder and bisimilarity is EXPTIME-hard when systems are given as networks of finite state systems (or equivalently as automata with boolean variables, etc.). We also show that any relation between trace inclusion and ready trace equivalence or possible-futures equivalence is EXPSPACE-hard for these systems.
These results match the already known upper bounds and partially answer a conjecture by Rabinovich. They strongly suggest that there is no way to escape the state explosion problem when checking behavioural relations.
For the branching-time relations, our proof uses a new construction that immediately applies to timed automata, a family of systems for which these complexity results are new.

1 Introduction

The *model-checking approach* to automated or computer-aided verification is now widely recognized as a promising development for system design, especially in the area of critical systems [CGL96]. The main practical limitation of model-checking is the well-known *state explosion problem*: the systems we check are built by composing several subsystems, they use variables and/or clocks, and a flat equivalent transition system would have an exponential number of states. Therefore, even if model-checking flat systems is tractable, verifying non-flat systems has been a major challenge since the beginning.

The state explosion problem can be considered from a pragmatic or from a theoretical angle. The pragmatical approach aims, e.g., at designing symbolic methods that may bypass the state explosion in many practical cases [BCM+92]. The theoretical approach studies the structural complexity of model-checking *non-flat systems*, i.e. systems described as combinations of finite-state components. The goal here is to understand better which verification problems have to face state explosion in an intrinsic way, which special way of combining subsystems could avoid state explosion, and what are the theoretical limits of all approaches, even the best pragmatical ones.

But what exactly are these non-flat systems ? Different models exist: synchronized products of finite-state automata are a natural possibility, automata acting on boolean variables are another one, as well as 1-safe Petri nets. From a structural complexity perspective, these brands of non-flat systems can all be succinctly encoded into each other and the complexity results hold robustly across many variant presentations. In this paper we consider synchronized products of automata (see section 2) but we keep the more general terminology of "non-flat systems" in this introduction.

An overview of existing results. The literature is limited but the main questions have been answered:

Classical verification problems: The complexity classes of the main questions for non-flat systems, like reachability, termination, deadlock-freedom, etc., are known (e.g., these three examples are PSPACE-complete). Most of these problems have been investigated in the framework of 1-safe Petri nets, where they were natural questions since the beginning. An excellent survey is [Esp98].

Temporal logic: Model-checking PLTL, CTL, or CTL* formulas on non-flat systems is PSPACE-complete [KVW98]. Model-checking the branching-time mu-calculus is EXPTIME-complete, even when restricted to the alternation-free fragment [Rab97b,KVW98].

Behavioural equivalences and preorders: Trace equivalence of non-flat systems is EXPSPACE-complete [Rab97a] while bisimilarity is EXPTIME-complete [JM96], as is simulation equivalence [HKV97].

Behavioural equivalences. This third set of problems is where the existing results are the most incomplete when assessing the state explosion problem. One of the difficulties here is that the linear time – branching time spectrum contains dozens of different semantical equivalences [Gla90] (cf. Fig. 1).

However, some general methods apply to several equivalences at once:
(1) [JM96] shows EXPTIME-completeness of seven truly concurrent variants of bisimulation. One single construction suffices for the lower bounds since all seven equivalences coincide in the absence of concurrency.
(2) [Rab97a] shows that *all equivalences lying between trace equivalence and bisimilarity* are PSPACE-hard. Note that this apply to all classical equivalences from [Gla90] and also to any new equivalence, however fancy, one would care to define [1].

Rabinovich's result is impressive, even more since it has been convincingly argued [Gla90,Pnu85,Mil89] that any interesting equivalence lies between these two extremes. However, the result is not optimal since not one relation between trace equivalence and bisimilarity is known to be in PSPACE for non-flat systems. Indeed, [Rab97a] conjectures that all these equivalences are EXPTIME-hard.

[1] A similar approach appears in [Jan95] where a single construction shows undecidability, over P/T nets, of all equivalences between trace equivalence and bisimilarity.

Our contribution. We partially answer Rabinovich's conjecture. We prove EXPTIME-hardness of all equivalences (actually any relation) lying between the simulation preorder and strong bisimilarity, and EXPSPACE-hardness of all equivalences (actually any relation) lying between trace inclusion and ready trace equivalence or possible-futures equivalence.

These results have several important corollaries. First, they close (on non-flat systems) the gap between lower-bound and upper-bound for the 11 relations van Glabbeek singles out as most fundamental in his linear time – branching time spectrum.

Secondly, they entail EXPTIME-hardness (over non-flat systems) of all model-checking problems for temporal or modal logics able to specify bisimilarity or simulation. For example, since the branching-time modal mu-calculus can state bisimilarity through a simple (modal depth 2) formula [And93], EXPTIME-hardness of bisimilarity entails EXPTIME-hardness of model-checking mu-calculus formula over non-flat systems (a result already known from [KVW98,Rab97b]).

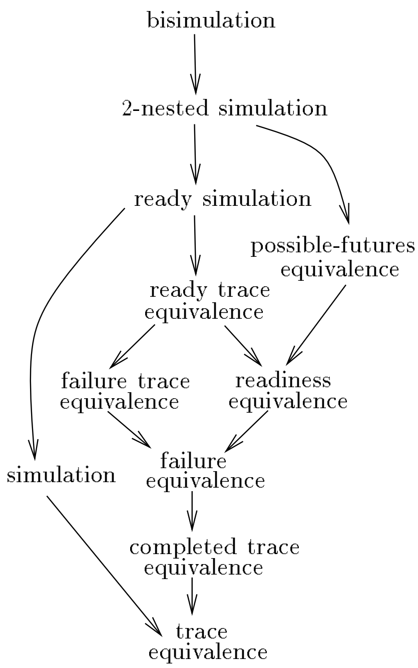

Fig. 1. The linear time – branching time spectrum [Gla90]

Finally, our technique is interesting in itself: our construction for the branching-time relations differs from the approach in [JM96] [2]. It originates from our investigations of complexity questions for Timed Automata [AL99] and readily gives EXPTIME-hardness of all relations between (strong timed–) bisimilarity and simulation. To our knowledge this is the first complexity characterization of behavioural equivalences over these models.

Plan of the paper. We first give basic definitions on (flat and non-flat) systems, the behavioural equivalences we need (§ 2) and alternating Turing machines (§ 3). We then prove our generic EXPTIME lower bound (§ 4) and our generic EXPSPACE lower bound (§ 5). Upper bounds are given when they match the lower bounds.

Acknowledgments. This work owes much to the comments and suggestions we got from L. Jategaonkar and the anonymous referees who saw an earlier version.

[2] Additionally, an incorrect labeling of the nets and the omission of some crucial part of the construction make the proof of Theo. 5.7. in [JM96] hard to repair [Jat99].

2 Equivalences and preorders between non-flat systems

Transition systems. A flat (transition) system is a tuple $C = \langle \Sigma, Q, \longrightarrow \rangle$ where Σ is a finite alphabet, Q is a finite set of states, and $\longrightarrow \subseteq Q \times \Sigma \times Q$ is the transition relation. The size $|C|$ of C is $|\Sigma| + |Q| + |\longrightarrow|$. As usual, we write $q \xrightarrow{a} q'$ when $(q, a, q') \in \longrightarrow$, and let *ready*$(q)$ denote the *set of actions ready in* q, i.e. $\{a \in \Sigma \mid q \xrightarrow{a} q' \text{ for some } q'\}$.

Traces. A *trace* from $q \in Q$ is any $w = a_1 \dots a_n \in \Sigma^*$ such that there exists $q_0, q_1, \dots, q_n \in Q$ with $q = q_0$ and $q_{i-1} \xrightarrow{a_i} q_i$ for $i = 1, \dots, n$ (written $q_0 \xrightarrow{w} q_n$). If $X_i = \mathit{ready}(q_i)$ for $i = 0, \dots, n$, then $(X_0, a_1, X_1, a_2, X_2, \dots, a_n, X_n)$ is a *ready trace* from q. We write $Tr(q)$ (resp. $RT(q)$, $PF(q)$) for the set of traces (resp. ready traces, possible futures) from q (where $(w, S) \in \Sigma^* \times \mathcal{P}(\Sigma^*)$ is a *possible future* of p if there exists q s.t. $p \xrightarrow{w} q$ and $S = Tr(q)$). Trace inclusion, trace equality, ready trace equivalence and possible-futures equivalence, denoted $\subseteq_{Tr}$, $=_{Tr}$, $=_{RT}$ and $=_{PF}$, have the obvious definition.

Bisimulations. A *simulation* over C is any $R \subseteq Q \times Q$ satisfying the following transfer property: for all qRq' and $q \xrightarrow{a} r$, there is a $q' \xrightarrow{a} r'$ s.t. $q'Rr'$. A *bisimulation* is any symmetric simulation. The largest simulation over C exists, is denoted $\sqsubseteq$, and is called the simulation preorder. The largest bisimulation is denoted $\underline{\leftrightarrow}$ and is called bisimilarity.

The hierarchy of equivalences. [Gla90,Gla93] survey the main behavioural equivalences (and preorders) used in the semantics of concurrent systems. Van Glabbeek list dozens of different possibilities between the weakest (trace equivalence) and the strongest (bisimilarity). The most important stepping stones in this hierarchy are given in Fig. 1.

As usual, for any such behavioural relation $\mathcal{R}$, we write $(C, q)\,\mathcal{R}\,(C', q')$ when $q\,\mathcal{R}\,q'$ inside a disjoint sum system $C + C'$. We write $C\,\mathcal{R}\,C'$ when C and C' come with (often implicit) initial states, and $(C, q_0)\,\mathcal{R}\,(C', q_0')$.

Non-flat systems. A non-flat system is a product of flat systems. Formally, it is a vector $\mathcal{S} = (C_1, \dots, C_k)$ where $C_i = \langle \Sigma_i, Q_i, \longrightarrow_i \rangle$ for $i = 1, \dots, k$. The *flattening* $C_{\mathcal{S}}$ of $\mathcal{S}$ is the transition system $\langle \Sigma, Q, \longrightarrow \rangle$ given by $\Sigma \stackrel{\text{def}}{=} \Sigma_1 \cup \dots \cup \Sigma_k$, $Q \stackrel{\text{def}}{=} Q_1 \times \dots \times Q_k$ and where $\longrightarrow$ is the set of all triples $((q_1, \dots, q_k), a, (r_1, \dots, r_k))$ from $Q \times \Sigma \times Q$ s.t. for $i = 1, \dots, k$ either $q_i \xrightarrow{a}_i r_i$ or ($a \notin \Sigma_i$ and $q_i = r_i$). In this paper we only need *binary* synchronization, i.e. where a given a belongs to at most two different Σ_i.

For a behavioural relation $\mathcal{R}$, deciding whether $\mathcal{S}\mathcal{R}\mathcal{S}'$ means deciding whether $C_{\mathcal{S}}\,\mathcal{R}\,C_{\mathcal{S}'}$. A *naive algorithm* for this problem is any algorithm that computes $C_{\mathcal{S}}$ and $C_{\mathcal{S}'}$. Let $|\mathcal{S}| \stackrel{\text{def}}{=} |C_1| + \dots + |C_k|$. Then $|C_{\mathcal{S}}|$ is $O(|C_1| \times \dots \times |C_k|)$, hence $O(2^{|\mathcal{S}|})$. This is known as *state explosion*.

Non-flat systems in the literature. [Rab97b] uses automata acting on boolean variables (or more generally on variables with a finite domain), [Esp98,JM96] use 1-safe labeled Petri nets. Product of finite automata are sometimes called *concurrent systems* [KVW98,Rab97a]. When relabeling of actions is allowed, any of these models can be directly translated into any other [3]. For maximal generality, we prove our hardness results for products without relabeling (and our upper bounds through naive algorithms that easily handle relabelings) which is one more way we strengthen the results from [Rab97a].

3 Alternating machines

An *Alternating Turing Machine* [CKS81] (an ATM for short) is a tuple $A = \langle Q, \Sigma, \delta, l, q_0, q_F \rangle$ where $Q = \{q, \dots\}$ is the *set of states*, $\Sigma = \{a, ..\}$ is the tape *alphabet* containing a special *blank symbol* (denoted by $\diamondsuit$), $\delta \subseteq Q \times \Sigma \times Q \times \Sigma \times \{L, R\}$ is the set of *transitions*, $q_0 \in Q$ is the *initial state*, $q_F \in Q$ is the final (accepting) state and $l : Q \to \{\vee, \wedge\}$ labels each state as either disjunctive or conjunctive.

Q is thus partitioned by l into $Q_\vee$ and $Q_\wedge$. We use letters $r, r', \dots$ to denote conjunctive states, $s, s', \dots$ for disjunctive states and $q, q', \dots$ for both. W.l.o.g. we require that $q_0, q_F \in Q_\vee$, that $\Sigma = \{a, b\}$, that each $q \neq q_F$ is the source of a transition, and that an ATM has clean alternation, i.e. it moves from disjunctive to conjunctive states and *vice versa*. We assign to each transition in δ a number $k \in \{1, \dots, |\delta|\}$ and we will denote by t_k the k-th transition.

A *configuration* of A (also called an *instantaneous description*, or an *i.d.*) is a triple $\alpha = (q, i, w)$ where $q \in Q$ is the *current state*, $w \in \Sigma^*$ is a word describing the *tape content*, and $0 < i \leq |w|$ is the *position of the head* on the tape (i.e. A is currently seeing $w(i)$). We use letters $\beta, \dots$ to denote disjunctive i.d.'s (that is, i.d.'s with a disjunctive control state) and $\gamma, \dots$ for the conjunctive i.d.'s. An i.d. (q, i, w) is *final* iff $q = q_F$.

An ATM moves like an usual non-deterministic TM: if $\alpha = (q, i, w)$, $w(i) = a$ and $(q, a, q', b, D) \in \delta$, then A may move from α to $\alpha' = (q', i', w')$, written $\alpha \longrightarrow \alpha'$, where w' is w updated by writing a b in position i and i' is $i + 1$ if $D = R$ or $i - 1$ if $D = L$. (As usual, if i' falls outside of w', we pad w' with an extra $\diamondsuit$ and perhaps readjust i'.) We say α' is a *successor* of α : there can only be a finite number of such successors.

The moves of an ATM starting from some i.d. α_0 can be arranged into a tree: the root node is labeled with α_0, and any node labeled by some α has one child for every α' s.t. $\alpha \longrightarrow \alpha'$. The order of the branches is not relevant so that there is only one tree starting from a given α_0. We call it the *run* of A from α_0.

[3] There exist other varieties of non-flat systems. Quite often they rely on a direct synchronization mechanism and can be accounted for in our formalism. The few exceptions (e.g., the Message Sequence Charts of [MPS98] or the Communicating Hierarchical State Machine of [AKY99]) have only recently been considered from a complexity-theoretic point of view, and they are obvious candidates for continuations of our work.

The run of A on some input word x is its run from $\alpha(x) \stackrel{\text{def}}{=} (q_0, 1, x)$. Note that a run may be infinite, and that a node is a leaf if and only if it is labeled by a configuration without any successor.

For ATM's, *accepting runs* are defined by seeing the run as an AND-OR tree. Formally, for $n \in \mathbb{N}$, we say a run rooted at some disjunctive β is accepting in n steps iff it is a final configuration or $n \geq 1$ and *one* of its children is accepting in $n-1$ steps, while a run rooted at some conjunctive [4] γ is accepting in n steps iff $n \geq 1$ and *all* its children are accepting in $n-1$ steps (and there is at least one child). We say A accepts x in n steps iff the run from $\alpha(x)$ is accepting in n steps. A word x is accepted by A iff there exists $n \geq 0$ s.t. A accepts x in n steps.

We say A is *linearly-bounded* on x if any configuration (q, w, i) in the run of A on some x has $|w| \leq |x|$ (that is, the machine never uses more tape than what is needed by the input). A classical result says that the problem LB-ATM-ACCEPT :

input: an ATM A and a word $x \in \Sigma^*$ s.t. A is linearly-bounded on x,

output: yes iff A accepts x, no otherwise.

is EXPTIME-complete.

4 EXPTIME-hard relations

Theorem 4.1. *Any relation lying between the simulation preorder and bisimilarity is EXPTIME-hard on non-flat systems.*

This is our main technical result and the rest of this section is devoted to the proof, a logspace reduction from LB-ATM ACCEPT. The proof of EXPTIME-hardness of bisimilarity in [JM96] is also based on a reduction from LB-ATM ACCEPT but, as mentioned in the introduction, the encoding is quite different.

The proofs of the next two lemmas assume familiarity in handling simulations and bisimulations.

4.1 Modeling an ATM by a non-flat system

Let A, w_0 be an ATM with a word of length n such that $A = \langle Q, \Sigma, \delta, l, q_0, q_F\rangle$ is linearly-bounded on w_0. We build a concurrent system $\mathcal{S}_{A,w_0} = (\mathcal{B}, C_1, \ldots, C_n)$ which models the run of A over w_0. Each C_i models the i-th tape cell: it can be in state a or b, and its initial state is $w_0(i)$. The tape cell synchronizes with the head of the ATM, hence for each transition $t_k = (q, e, q', e', d) \in \delta$, C_i has a transition $e \xrightarrow{t_k,i} e'$. See Fig. 2 for an example of C_i component for a set of transitions.

$\mathcal{B}$ is the control part of A. Write $Q^- = \{q^- \mid q \in Q\}$ (resp. $Q^+ = \{q^+ \ldots\}$) for a set of copies of states from Q tagged by a "$-$" (resp. by a "$+$"). The states of $\mathcal{B}$ is $Q_{\mathcal{B}} \stackrel{\text{def}}{=} Q^- \times \{1, \ldots, n\} \cup Q^+_\vee \times \{1, \ldots, n\} \cup Q^+_\wedge \times \{1, \ldots, n\} \times \{t_1, \ldots, t_{|\delta|}\}$.

[4] Remember $q_F \notin Q_\wedge$.

$t_1 = (s_1, a, r_1, b, R)$
$t_2 = (s_1, b, r_2, b, R)$
$t_3 = (r_1, a, s_2, a, L)$
$t_4 = (r_1, a, s_3, b, R)$
$t_5 = (r_2, b, s_3, a, R)$
$t_6 = (r_2, a, s_4, a, L)$
$\dots$

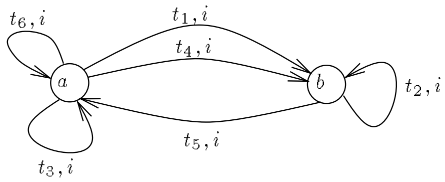

Fig. 2. System C_i

A state (q^-, i) of $\mathcal{B}$ encodes a control state of A and a position of A's head over the tape. For each $t_k = (q, e, q', e', d) \in \delta$ and for any i, $\mathcal{B}$ has a transition $(q^-, i) \xrightarrow{t_k, i} (q'^-, i + d)$ where $i + d$ denotes $i + 1$ (resp. $i - 1$) if $d = R$ and $i < n$ (resp. $d = L$ and $i > 1$).

These transitions are called "type 1" and they synchronize with the corresponding transitions from the C_i's: a transition labeled "t_k, i" is enabled in $\mathcal{S}_{A,w_0}$ iff the current control state is q, the position of the head is i, and if C_i contains the right value. Firing this transition modifies the value of C_i, the control state and the head position so that the behaviour of A and its tape is faithfully emulated by the type 1 transitions.

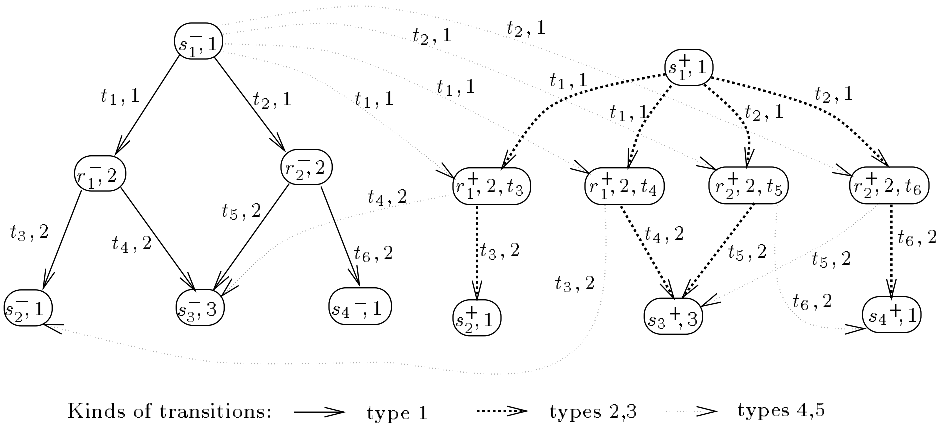

Fig. 3. (Part of) system $\mathcal{B}$

An example of such type 1 transitions is displayed in the left part of Fig. 3, assuming δ as in the previous example.

The q^+ states behave slightly differently. To begin with, disjunctive and conjunctive states are not dealt in the same way. Assume $s \in Q_\vee$ is disjunctive. For each $t_k = (s, e, r, e', d)$ and $t_{k'} = (r, f, s', f', d')$ in δ, for any i, $\mathcal{B}$ has a (type 2) transition $(s^+, i) \xrightarrow{t_k,i} (r^+, i+d, t_{k'})$. These transitions correspond to "firing t_k and picking $t_{k'}$ as next transition".

The transitions for conjunctive states are defined accordingly: if t_k is some (r, e, s, e', d), then $\mathcal{B}$ has (type 3) transitions $(r^+, i, t_k) \xrightarrow{t_k,i} (s^+, i+d)$.

Here is the idea behind these transitions: assume an i.d. s, i, w is not accepting. A strategy for establishing this had an opponent firing any transition t_k from s, and the defender may picking a $t_{k'}$ (one must exist) leading to a rejecting i.d., etc. In $\mathcal{S}_{A,w_0}$, this strategy is implemented by having s^- (the opponent) firing a type 1 transition t_k, forcing s^+ to fire the type 2 t_k transition that selects $t_{k'}$ for the next move.

The (type 4) transitions move from s^- states to the Q^+ part: formally $\mathcal{B}$ has all $(s^-, i) \xrightarrow{t_k,i} (r^+, i', t_{k'})$ s.t. $(s^+, i) \xrightarrow{t_k,i} (r^+, i', t_{k'})$. The purpose is to allow in s^- everything allowed in s^+.

The (type 5) transitions allow firing any $t_{k'}$ from a (r^+, i, t_k) where $\mathcal{B}$ has already commited to t_k. Assume $t_k = (r, e, s, e', d)$, then for all $t_{k'} \neq t_k$ starting from r, i.e. $t_{k'}$ is some (r, f, s', f', d'), $\mathcal{B}$ has one of the following transitions depending on the values of e and f:

- $(r^+, i, t_k) \xrightarrow{t_{k'},i} (s'^-, i+d')$ if $e = f$. For example, see transition "$t_4, 2$" from $(r_1^+, 2, t_3)$ or transition "$t_3, 2$" from $(r_1^+, 2, t_4)$ in Fig. 3.
- $(r^+, i, t_k) \xrightarrow{t_{k'},i} (s'^+, i+d')$ if $e \neq f$. For example, see transition "$t_6, 2$" from $(r_2^+, 2, t_5)$ or transition "$t_5, 2$" from $(r_2^+, 2, t_6)$ in Fig. 3.

Intuitively if t_k and $t_{k'}$ are both enabled (according to the value of C_i), then the transition leads to the corresponding s^- state, otherwise it leads to the corresponding s^+ state.

Finally, $\mathcal{B}$ has special (type 6) transitions in (q_F^-, i) states: $(q_F^-, i) \xrightarrow{\mathsf{acc}} (q_F^-, i)$. These are the only transitions without synchronization, they do not exist from the q_F^+ states and they distinguish between the Q^- and the Q^+ parts of $\mathcal{B}$.

With this we have completed the description of $\mathcal{B}$. The size of $\mathcal{S}_{A,w_0}$ is $O(n \times |Q| \times |\delta|^2)$ and $\mathcal{S}_{A,w_0}$ can be built using only four counters, that is in space $\ln(n) + \ln(|Q|) + 2\ln(|\delta|)$.

4.2 Relating A on w_0 and $\mathcal{S}_{A,w_0}$

A configuration of $\mathcal{S}_{A,w_0}$ has the form $\langle \rho, e_1, \ldots, e_n \rangle$ where ρ is a $\mathcal{B}$ state and $e_i \in \{a, b\}$ is a C_i state. We write such a configuration as $\langle \rho, w \rangle$ where $w \in \Sigma^n$ is given by $w(i) = e_i$ for $i = 1, \ldots, n$.

$\langle \rho, w \rangle$ is said to be disjunctive (resp. conjunctive) depending on whether ρ contains a disjunctive or conjunctive state of A.

We now link i.d.'s of A and configurations of $\mathcal{S}_{A,w_0}$: Given an i.d. $\alpha = (q, i, w)$, α^- denotes the $\mathcal{S}_{A,w_0}$ configuration $\langle (q^-, i), w \rangle$. Given a disjunctive i.d. $\beta = (s, i, w)$, β^+ represents $\langle (s^+, i), w \rangle$. Given a conjunctive i.d. $\gamma = (r, i, w)$ and a transition t_k whose source node is r, γ_k^+ denotes $\langle (r^+, i, t_k), w \rangle$.

Lemma 4.2. *If A does not accept w_0, then $\langle(q_0^-,1),w_0\rangle \underline{\leftrightarrow} \langle(q_0^+,1),w_0\rangle$.*

Proof. Remember that $\beta,\ldots$ (resp. $\gamma,\ldots$) denote disjunctive (resp. conjunctive) i.d.'s. Consider the following relation R between the configurations of $\mathcal{S}_{A,w_0}$:

$$R \stackrel{\text{def}}{=} \{(\beta^-,\beta^+) \mid \beta \text{ is rejecting}\} \cup \mathtt{Id} \cup \\ \{(\gamma^-,\gamma_k^+) \mid \gamma \longrightarrow_k \beta \text{ and } \gamma \text{ and } \beta \text{ are rejecting}\}$$

We can show that R is bisimulation: since A rejects w_0, then $(\langle(q_0^-,1),w_0\rangle, \langle(q_0^+,1),w_0\rangle) \in R$, and it remains to check that R has the transfer property in both directions (see Appendix A for details).

Lemma 4.3. *If A accepts w_0, then $\langle(q_0^-,1),w_0\rangle \not\sqsubseteq \langle(q_0^+,1),w_0\rangle$.*

Proof. By induction on the number of steps for accepting. See Appendix B.

Corollary 4.4. *For any relation $\mathcal{R}$ s.t. $\underline{\leftrightarrow} \subseteq \mathcal{R} \subseteq \sqsubseteq$, A does not accept w_0 iff $\langle(q_0^-,1),w_0\rangle\mathcal{R}\langle(q_0^+,1),w_0\rangle$.*

which concludes the proof of Theorem 4.1.

4.3 Upper bounds

Theorem 4.1 is in a sense optimal since the lower bounds it exhibits are optimal for the relations singled out in Fig. 1:

Theorem 4.5. *Bisimulation, 2-nested simulation, ready simulation, and simulation on non-flat systems are EXPTIME-complete.*

Proof (sketch). There only remain to show membership in EXPTIME. In all four cases this can be done by a reduction to model checking of a simple branching-time mu-calculus formula. Such a reduction expresses a relation $\mathcal{R}$ via a mu-calculus formula $\varphi_{\mathcal{R}}$ in a way s.t. $C\mathcal{R}C'$ iff $(\tilde{C},\tilde{C'}) \models \varphi_{\mathcal{R}}$ where $\tilde{C}$ is a variant of C where the actions have been relabeled to avoid conflicts with C'. For bisimulation this is done in [And93], and the same technique apply to the other equivalences. We then rely on EXPTIME-completeness of mu-calculus model-checking for non-flat systems [KVW98,Rab97b]. $\square$

4.4 Extension to Timed Automata

Timed Automata [AD94] can be seen as a special kind of non-flat systems. We denote by $\mathbb{N}$ the set of natural numbers and by $\mathbb{R}$ the set of non–negative real numbers. If $Cl = \{x, y, \ldots\}$ is a set of clocks, $\mathcal{L}(Cl)$ denotes *clocks constraints* over Cl, that is the set of formulas built using boolean connectives over atomic formulas of the form $x \bowtie m$ or $x - y \bowtie m$ with $x, y \in Cl$, $m \in \mathbb{N}$ and $\bowtie \in \{=, <, >, \leq, \geq\}$. A time assignment v for Cl is a function from Cl to $\mathcal{R}$. We denote by $\mathcal{R}^{Cl}$ the set of time assignments for Cl. For $v \in \mathcal{R}^{Cl}$ and $d \in \mathcal{R}$, $v + d$ denotes the time assignment which maps each clock x in Cl to the value $v(x) + d$. For

$Cl' \subseteq Cl$, $[Cl' \leftarrow 0]v$ denotes the assignment for Cl which maps each clock in Cl' to the value 0 and agrees with v over $Cl \backslash Cl'$. Given a condition $g \in \mathcal{L}(Cl)$ and a time assignment $v \in \mathcal{R}^{Cl}$, we note $v \models g$ when g holds for v. We define the timed automata (TA):

Definition 4.6. *A timed automaton TA over Σ is a tuple $\langle N, \eta_0, Cl, E\rangle$ where N is a finite set of nodes, $\eta_0 \in N$ is the initial node, Cl is a finite set of clocks, $E \subseteq N \times \mathcal{L}(Cl) \times \Sigma \times 2^{Cl} \times N$ corresponds to the set of edges: $e = \langle \eta, g, a, r, \eta'\rangle \in E$ represents an edge from the node η to the node η' with action a, r denotes the set of clocks to be reset and g is the enabling condition (the guard) over the clocks of TA. We use the notation $\eta \xrightarrow{g,a,r} \eta'$.*

A *configuration* of TA is a pair (η, v) where η is a node of TA and v a time assignment for Cl. Informally, the system starts at node η_0 with the assignment v_0 which maps all clocks to 0. The values of the clocks may increase synchronously with time. At any time, the automaton whose current node is η can change node by following an edge $\langle \eta, g, a, r, \eta'\rangle \in E$ provided the current values of the clocks satisfy g. With this transition the clocks in r get reset to 0. Let Θ denote the set of delay actions $\{\varepsilon(d) \mid d \in \mathbb{R}\}$. Formally the semantics of TA is defined as a *labeled timed transition system*:

Definition 4.7. *A labeled timed transition system over Σ is a tuple $\mathcal{S} = \langle S, s_0, \longrightarrow\rangle$, where S is a set of states, s_0 is the initial state, $\longrightarrow \subseteq S \times (\Sigma \cup \Theta) \times S$ is a transition relation. We require that for any $s \in S$ and $d \in \mathcal{R}$, there exists a unique state s^d such that $s \xrightarrow{\varepsilon(d)} s^d$ and that $(s^d)^e = s^{d+e}$.*

The labeled timed transition system associated with TA is $\langle S_{TA}, s_0, \longrightarrow_{TA}\rangle$, where S_{TA} is the set of configuration of TA, s_0 is the initial configuration (η_0, v_0), and $\longrightarrow_{TA}$ is the transition relation defined as follows:

$$
\begin{array}{lll}
(\eta, v) \xrightarrow{a} (\eta', v') & \text{iff} & \exists\ \langle \eta, g, a, r, \eta'\rangle \in E \ \text{ s.t. } \ v \models g \ \text{ and } \ v' = [r \leftarrow 0]v \\
(\eta, v) \xrightarrow{\varepsilon(d)} (\eta', v') & \text{iff} & \eta = \eta' \ \text{ and } \ v' = v + d
\end{array}
$$

The standard notion of bisimulation (and simulation) can be naturally extended to timed systems [Čer93]: A strong timed (bi)simulation between TA and TB is a (bi)simulation between the associated labeled timed transition systems.

Theorem 4.8. *Any relation lying between the simulation preorder and bisimilarity is EXPTIME-hard on timed automata.*

Proof (sketch). Let A, w_0 be an ATM with a word of length n. We transform the automaton $\mathcal{B}$ defined in section 4.1 into a timed automaton TB in such a way that the clocks Cl of TB encode the tape content. This encoding is used in [AL99]. The transitions of TB use guards over the clocks to ensure a correct behavior, and reset operations are used to modify the tape content according to the performed transition. Therefore we obtain a single timed automaton instead of a parallel composition of finite automata.

TB uses $2n+1$ clocks: $\{x_1, \ldots, x_n, y_1, \ldots, y_n, t\}$. The clock t is used to ensure a delay of length 1 between two transitions of TB. The clocks x_i and y_i encode the value of the i-th tape cell by the following convention: $C_i = a$ (resp. $C_i = b$) iff $x_i = y_i$ (resp. $x_i < y_i$). Let t_i be the ATM transition (q, e, q', e', d) : the transition $(q, k) \xrightarrow{t_i, k} (q', k+d)$ we used in $\mathcal{B}$ is replaced by a transition $(q, k) \xrightarrow{t=1 \wedge g, t_i, r} (q', k+d)$ where g is $x_i = y_i$ (resp. $x_i < y_i$) if $e = a$ (resp. $e = b$), the reset set r is $\{t, x_i\}$ (resp. $\{t, x_i, y_i\}$) if $e' = b$ (resp. $e' = a$). The initialization of the tape with the input word w_0 can be encoded by adding the transitions $init^- \xrightarrow{t=1, s_0, r_w} (q_0^-, 1)$ and $init^+ \xrightarrow{t=1, s_0, r_{w_0}} (q_0^+, 1)$, where $r_{w_0} = \{t\} \cup \{x_i \mid w_0(i) = b\}$. The acc transition of $\mathcal{B}$ are kept in TB.
Lemmas 4.2 and 4.3 still hold for the initial configurations $(init^-, u_0)$ and $(init^+, v_0)$ where u_0 and v_0 map any clock in Cl to 0.

Remark 4.9. Note that bisimulation and simulation for timed automata are EXPTIME-complete since the model-checking problem for the timed μ-calculus (which allows to express bisimilarity and similarity) is EXPTIME-complete [AL99].

5 EXPSPACE-hard relations

Theorem 5.1. *Any relation lying between trace inclusion and the intersection of ready trace equivalence and possible-futures equivalence is EXPSPACE-hard on non-flat systems.*

Proof (sketch). We adapt the proof, from [JM96], that trace inclusion is EXPSPACE-hard on non-flat systems.

Their proof is a reduction from the problem of deciding whether the language defined by a regular expression with interleaving is Σ^*, which is known to be EXPSPACE-complete [MS94]. Given any regular expression e built from $\{\cup, *, ., \|\}$ with $|e| = n$, Jategaonkar and Meyer build a non-flat system $Net(e)$ over the alphabet $\Sigma \cup \{\mathbf{1}, \surd\}$ s.t. $Tr(Net(e))$ is (the prefix-closure of) $\{\mathbf{1}^{4n} a_1 \mathbf{1}^{4n} \ldots \mathbf{1}^{4n} a_k \mathbf{1}^{4n} \surd \mid a_1 \ldots a_k \in L(e)\}$.

Let $=_{RT.PF}$ be the equivalence defined as the intersection of $=_{RT}$ and $=_{PF}$. We can modify the previous model in a simple way to obtain $Net(e, n)$ with $n \geq |e|$ so that, for $L(e) = \Sigma^*$ iff $Net(e, n) =_{RT.PF} Net(\Sigma^*, n)$ iff $Net(\Sigma^*, n) \subseteq_{Tr} Net(e, n)$. This will entail the result.

The main idea is to add a state end from which the enabled transitions are labeled by $\Sigma \cup \{\mathbf{1}\}$ and lead to end. From any state q, we add transitions $q \xrightarrow{\Sigma} end$. By this way we have that $RT(Net(e, n))$ is (the prefix-closure of)

$$\{(\overbrace{\{\mathbf{1}\}, \mathbf{1}, \ldots, \{\mathbf{1}\}, \mathbf{1}}^{4n}, \Sigma, a_1, \{\mathbf{1}\}, \mathbf{1}, \ldots, \Sigma, a_k, \{\mathbf{1}\}, \mathbf{1}, \ldots, \{\surd\}, \surd, \emptyset) \mid a_1 \ldots a_k \in L(e)\}$$
$$\bigcup \{(\overbrace{\{\mathbf{1}\}, \mathbf{1}, \ldots, \{\mathbf{1}\}, \mathbf{1}}^{4n}, \Sigma, b_1, \{\mathbf{1}\}, \mathbf{1}, \ldots, \Sigma, b_k, \{\mathbf{1}\}) \mid b_1, \ldots, b_k \in \Sigma\}$$

and $PF(Net(e,n))$ is the set of pairs (w,S) s.t. $w = \mathbf{1}^{4n}a_1\mathbf{1}^{4n}\ldots\mathbf{1}^k \in Tr(Net(e,n))$ and S is the prefix-closure of $S_1 \cup S_2$ or S_2 with:

$$S_1 = \{w' \mid w.w'\surd \in Tr(Net(e,n))\}$$
$$S_2 = \{w' = \mathbf{1}^{4n-k}b_1\mathbf{1}^{4n}\ldots b_l\mathbf{1}^{4n} \mid b_1\ldots b_l \in \Sigma^*\}$$

Note that for $(w,S) \in PF(Net(\Sigma^*,n))$, S is the prefix-closure of:

$$\{w' = \mathbf{1}^{4n-k}b_1\mathbf{1}^{4n}\ldots b_l\mathbf{1}^{4n}\surd \mid b_1\ldots b_l \in \Sigma^*\}$$

Clearly, $Net(\Sigma^*,n) \subseteq_{Tr} Net(e,n)$ iff $Net(e,n) =_{RT} Net(\Sigma^*,n)$ iff $Net(e,n) =_{PF} Net(\Sigma^*,n)$ iff $L(e) = \Sigma^*$. This gives the result. □

5.1 Upper bounds

Theorem 5.1 is in a sense optimal since the lower bounds it exhibits are optimal for the relations singled out in Fig. 1:

Theorem 5.2. *Possible-futures equivalence, ready trace equivalence, failure trace equivalence, readiness equivalence, failures equivalence, completed trace equivalence and trace equivalence on non-flat systems are EXPSPACE-complete.*

Proof (sketch). We only need to prove membership in EXPSPACE. In all cases, this can be done by the naive algorithm, noting that the problems are in PSPACE for flat systems (by simple reductions to language equivalence of non-deterministic automata).

6 Conclusion

We have shown that for non-flat systems, any relation between the simulation preorder and bisimilarity is EXPTIME-hard, and that any relation between trace inclusion and ready trace equivalence is EXPSPACE-hard.

This is a partial answer to the questions raised by Rabinovich [Rab97a] [5]. Indeed, these results cover a large array of relations, and they give lower bounds matching the (obvious) upper bounds in the 11 relations van Glabbeek singles out as most prominent in his branching time – linear time spectrum.

For the EXPTIME-hard relations, our construction also applies to timed automata, where the lower bounds were not known.

This theoretical study has practical implications. It strongly suggests that there is no way to escape state explosion when checking non-flat systems for some behavioural relation, at least not by some smart choice of which behavioural equivalence is chosen [6]. Attempts at general solutions should rather aim at finding a smart limitation of how non-flat systems may be described. In such a

[5] Additionally, our hardness results do not need his *hide* operator to further relabel the products of systems.

[6] Since our results are not a complete answer, we cannot rule out the dim possibility that some PSPACE-easy relation exist in the branching time–linear time spectrum.

quest, one should aim at forbidding the construction of our $\mathcal{S}_{A,w_0}$ system (or any reasonably succinct equivalent encoding).

A related idea is to focus on the complexity of deciding whether $\mathcal{S} \sim C$ *where* C *is a fixed system* and where $\mathcal{S}$ is then the only input. For this measure, called *implementation complexity*, the results are no longer uniform. For example, for simulation we have that deciding whether $\mathcal{S} \sqsubseteq C$ is still EXPTIME-complete [HKV97] while for bisimulation, we have the following:

Proposition 6.1. *When* C *is fixed, deciding whether* $\mathcal{S} \underline{\leftrightarrow} C$ *is PSPACE-complete.*

Proof (Sketch). PSPACE membership combines the ability to build a CTL formula Φ_C such that $\mathcal{S} \underline{\leftrightarrow} C$ iff $\mathcal{S} \models \Phi_C$ [BCG88] and the fact that CTL model checking of non-flat systems is PSPACE-complete [KVW98]. PSPACE-hardness is by reduction of the reachability problem in $\mathcal{S}$.

References

[AD94] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[AKY99] R. Alur, S. Kannan, and M. Yannakakis. Communicating hierarchical state machines. In *Proc. 26th Int. Coll. Automata, Languages, and Programming (ICALP'99), Prague, Czech Republic, July 1999*, volume 1644 of *Lecture Notes in Computer Science*, pages 169–178. Springer, 1999.

[AL99] L. Aceto and F. Laroussinie. Is your model checker on time ? In *Proc. 24th Int. Symp. Math. Found. Comp. Sci. (MFCS'99), Szklarska Poreba, Poland, Sep. 1999*, volume 1672 of *Lecture Notes in Computer Science*, pages 125–136. Springer, 1999.

[And93] H. R. Andersen. *Verification of Temporal Properties of Concurrent Systems.* PhD thesis, Aarhus University, Denmark, June 1993. Available as DAIMI PB–445.

[BCG88] M. C. Browne, E. M. Clarke, and O. Grumberg. Characterizing finite Kripke structures in propositional temporal logic. *Theoretical Computer Science*, 59(1–2):115–131, 1988.

[BCM+92] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, 1992.

[Čer93] K. Čerāns. Decidability of bisimulation equivalence for parallel timer processes. In *Proc. 4th Int. Workshop Computer Aided Verification (CAV'92), Montreal, Canada, June–July 1992*, volume 663 of *Lecture Notes in Computer Science*, pages 302–315. Springer, 1993.

[CGL96] E. M. Clarke, O. Grumberg, and D. Long. Model-checking. In M. Broy, editor, *Deductive Program Design, Proc. NATO-ASI Summer School, Marktoberdorf, Germany, 26 July - 7 Aug 1994*, volume F-152 of *NATO ASI Series*. Springer, 1996.

[CKS81] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[Esp98] J. Esparza. Decidability and complexity of Petri net problems — an introduction. In *Advances in Petri Nets 1998*, volume 1491 of *Lecture Notes in Computer Science*, pages 374–428. Springer, 1998.

[Gla90] R. J. van Glabbeek. The linear time – branching time spectrum. In *Proc. Theories of Concurrency (CONCUR'90), Amsterdam, NL, Aug. 1990*, volume 458 of *Lecture Notes in Computer Science*, pages 278–297. Springer, 1990.

[Gla93] R. J. van Glabbeek. The linear time – branching time spectrum II: The semantics of sequential systems with silent moves. In *Proc. 4th Int. Conf. Concurrency Theory (CONCUR'93), Hildesheim, Germany, Aug. 1993*, volume 715 of *Lecture Notes in Computer Science*, pages 66–81. Springer, 1993.

[HKV97] D. Harel, O. Kupferman, and M. Y. Vardi. On the complexity of verifying concurrent transition systems. In *Proc. 8th Int. Conf. Concurrency Theory (CONCUR'97), Warsaw, Poland, Jul. 1997*, volume 1243 of *Lecture Notes in Computer Science*, pages 258–272. Springer, 1997.

[Jan95] P. Jančar. Undecidability of bisimilarity for Petri nets and some related problems. *Theoretical Computer Science*, 148(2):281–301, 1995.

[Jat99] L. Jategaonkar. Personal communication, August 1999.

[JM96] L. Jategaonkar and A. R. Meyer. Deciding true concurrency equivalences on safe, finite nets. *Theoretical Computer Science*, 154(1):107–143, 1996.

[KVW98] O. Kupferman, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking, 1998. Full version of the CAV'94 paper, accepted for publication in J. ACM.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice Hall Int., 1989.

[MPS98] A. Muscholl, D. Peled, and Z. Su. Deciding properties for message sequence charts. In *Proc. Int. Conf. Foundations of Software Science and Computation Structures (FOSSACS'98), Lisbon, Portugal, Mar.-Apr. 1999*, volume 1378 of *Lecture Notes in Computer Science*, pages 226–242. Springer, 1998.

[MS94] A. J. Mayer and L. J. Stockmeyer. Word problems—this time with interleaving. *Information and Computation*, 115(2):293–311, 1994.

[Pnu85] A. Pnueli. Linear and branching structures in the semantics and logics of reactive systems. In *Proc. 12th Coll. Automata, Languages and Programming (ICALP'85), Nafplion, Greece, Jul. 1985*, volume 194 of *Lecture Notes in Computer Science*, pages 15–32. Springer, 1985.

[Rab97a] A. Rabinovich. Complexity of equivalence problems for concurrent systems of finite agents. *Information and Computation*, 139(2):111–129, 1997.

[Rab97b] A. Rabinovich. Symbolic model checking for μ-calculus requires exponential time. Tech. report, Dept. Comp. Sci., Tel Aviv University, Israel, August 1997.

A Proof of Lemma 4.2

We have to show that R has the transfer property in both directions:
1. Consider a pair $(\beta^-, \beta^+) \in R$, with $\beta = (s, i, w)$ a rejecting (disjunctive) i.d. β is not a final configuration, no `acc` transition is enabled from β^-: and we just have to check the transfer property for transitions labeled by t_k, i:

- Assume a type 1 move $\beta^- \xrightarrow{t_k,i} \langle (r^-, i'), w' \rangle$. In A, this corresponds to $(s, i, w) \longrightarrow_k (r, i', w')$. Let γ be (r, i', w'). Since β is disjunctive and rejecting, γ is rejecting. Thus there exists a move $\gamma \longrightarrow_{k'} \beta'$ s.t. β' is rejecting. Moreover the (type 2) transition $\beta^+ \xrightarrow{t_k,i} \gamma^+_{k'}$ is allowed in $\mathcal{S}_{A,w_0}$. Therefore, for any transition $\beta^- \xrightarrow{t_k,i} \gamma^-$, there exists $\beta^+ \xrightarrow{t_k,i} \gamma^+_{k'}$ s.t. $(\gamma^-, \gamma^+_{k'}) \in R$.

– The other possible moves for this pair are type 4 transitions $\beta^- \longrightarrow \gamma^+_{k'}$ and type 2 transitions $\beta^+ \longrightarrow \gamma^+_{k'}$: they can be imitated by the other side, relying on $(\gamma^+_{k'}, \gamma^+_{k'}) \in \mathtt{Id} \subseteq R$.

2. Consider a pair $(\gamma^-, \gamma^+_k) \in R$ with $\gamma = (r, i, w)$. γ is a rejecting i.d. and there exists a move $\gamma \longrightarrow_k \beta$ leading to a rejecting $\beta = (s, i', w')$. We check the transfer property:

– Assume a type 1 move $\gamma^- \xrightarrow{t_{k'},i} \beta^-$. Then $\gamma \longrightarrow_{k'} \beta$ and either $k = k'$ or $k \neq k'$. If $k = k'$, then $\gamma^+ \xrightarrow{t_{k'},i} \beta^+$ and since β is not accepting, $(\beta^-, \beta^+) \in \mathcal{R}$. When $k \neq k'$, both t_k and $t_{k'}$ are enabled from γ, so that t_k and $t_{k'}$ require the same letter on the tape cell, and there exists a type 5 move $\gamma^+ \xrightarrow{t_{k'},i} \beta^-$. We use the fact that $(\beta^-, \beta^-) \in \mathtt{Id} \subseteq \mathcal{R}$.
– Assume a type 3 move $\gamma^+_k \xrightarrow{t_k,i} \beta^+$ with $\beta = (s^+, i', w')$, it can be simulated by $\gamma^- \xrightarrow{t_k,i} \beta^-$ because β is not accepting, so that $(\beta^-, \beta^+) \in R$.
– Other moves from γ^+_k reach a β^- (because t_k is enabled from γ) and can be easily imitated.

3. Finally, the pairs from $\mathtt{Id}$ obviously enjoy the transfer property.

B Proof of Lemma 4.3

We show by induction on l that

1. If β is accepting in l steps, then $\beta^- \not\sqsubseteq \beta^+$.
2. If $\gamma = (r, i, w)$ is accepting in l steps, then $\gamma^- \not\sqsubseteq \gamma^+_k$ for any k s.t. the source node of t_k is r.

– $l = 0$: if β accepts in 0 steps, then it is final and $\beta^- \xrightarrow{\mathtt{acc}}$ cannot be matched from β^+. A conjunctive configuration γ cannot be accepting in 0 steps.
– Assume the property holds for any $l' \leq l$. We have two cases:
  - A disjunctive β accepts in $l + 1$ steps. Then there exists t_k s.t. $\beta \longrightarrow_k \gamma$ where $\gamma = (r, i, w)$ is accepting in l steps. In $\mathcal{S}_{A,w_0}$, the transition $\beta^- \xrightarrow{t_k,i} \gamma^-$ has to be matched by a transition labeled with "t_k, i" which leads to a configuration $\gamma^-_{k'}$ and, by i.h., $\gamma^- \not\sqsubseteq \gamma^+_{k'}$.
  - A conjunctive $\gamma = (r, i, w)$ accepts in $l + 1$ steps. We must show that $\gamma^- \not\sqsubseteq \gamma^+_k$ for any k s.t. t_k starts from r. There are two cases:
    * t_k is enabled from γ: since γ accepts, any move from γ leads to an i.d. accepting in l steps. In $\mathcal{S}_{A,w_0}$, the transition $\gamma^- \xrightarrow{k,i} \beta^-$ can only be matched from γ^+_k by $\gamma^+_k \xrightarrow{k,i} \beta^+$ and, by i.h., $\beta^- \not\sqsubseteq \beta^+$.
    * t_k is not enabled from γ: any transition $t_{k'}$ enabled from γ leads to an i.d. accepting in l steps and one such a transition exists. In $\mathcal{S}_{A,w_0}$, the move $\gamma^- \xrightarrow{k',i} \beta^-$ can only be matched from γ^+_k by $\gamma^+_k \xrightarrow{k',i} \beta^+$ and, by i.h., $\beta^- \not\sqsubseteq \beta^+$.

    In both cases, we found a transition from γ^- which cannot be simulated from γ^+ and then $\gamma^- \not\sqsubseteq \gamma^+_k$.

Now, since we assume that $(q_0, 1, w_0)$ is accepting, the proof is complete.