Symbolic Determinisation of Extended Automata

Thierry Jéron, Hervé Marchand, Vlad Rusu

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES

PUBLICATION INTERNE
N^{o} 1776

SYMBOLIC DETERMINISATION OF EXTENDED AUTOMATA

THIERRY JÉRON , HERVÉ MARCHAND , VLAD RUSU

ISSN 1166-8687

IRISA
CAMPUS UNIVERSITAIRE DE BEAULIEU - 35042 RENNES CEDEX - FRANCE

Institut de Recherche en Informatique et Systèmes Aléatoires
Campus de Beaulieu – 35042 Rennes Cedex – France
Tél. : (33) 02 99 84 71 00 – Fax : (33) 02 99 84 71 71
http://www.irisa.fr

Symbolic Determinisation of Extended Automata

Thierry Jéron , Hervé Marchand , Vlad Rusu

Systèmes communicants
Projet VerTeCs

Publication interne n1776 — January 2006 — 24 pages

Abstract: We define a symbolic determinisation procedure for a class of infinite-state systems, which consists of automata extended with symbolic variables that may be infinite-state. The subclass of extended automata for which the procedure terminates is characterised as *bounded lookahead extended automata*. It corresponds to automata for which, in any location, the observation of a bounded-length trace is enough to infer the first transition actually taken. We discuss applications of the algorithm to the verification, testing, and diagnosis of infinite-state systems.

Key-words: symbolic automata, determinisation

(Résumé : tsvp)

Centre National de la Recherche Scientifique
(UMR 6074) Université de Rennes 1 – Insa de Rennes

Institut National de Recherche en Informatique
et en Automatique – unité de recherche de Rennes

Déterminisation symboliques d'automates étendus

Résumé : Nous introduisons une procédure de déterminisation symbolique pour une classe de systèmes infinis. Il s'agit d'automates étendus avec des variables, qui communiquent avec l'environnement au moyen d'actions de synchronisation. Nous caractérisons précisément la sous-classe pour laquelle la procédure termine, et donnons des conditions suffisantes décidables pour la terminaison. Nous discutons les applications de cette procédure au test de conformité et au diagnostic de systèmes infinis.

Mots clés : automates symboliques, déterminisation

1 Introduction

Most existing models of computation are nondeterministic, but they include restricted, deterministic versions as subclasses. A natural question is comparing the expressiveness of the general, nondeterministic class with that of the corresponding restricted, deterministic subclass. For example, it is well known that nondeterministic and deterministic *finite automata* on *finite words* are equivalent, but for finite automata on *infinite* words, the equivalence depends on the acceptance condition (e.g., *Müller* versus *Büchi* acceptance); and for *pushdown* and *timed automata*, the nondeterministic version is strictly more expressive than the deterministic one [3, 1].

Besides this theoretical interest, the distinction between nondeterministic and deterministic models has practical consequences. For example, *verification* consists in checking whether an *implementation* of a system satisfies a *specification*; both views of the system are modeled by automata of some kind. This problem can be seen as a language inclusion problem, which in turn can be encoded into a language emptyness problem (i.e., checking the emptyness of the language recognised by a product between the implementation and the *complement* of the specification). The complement of the specification is an automaton that accepts exactly the words that are rejected by the specification, and is easily computed if the specification is deterministic (by complementing the specification's acceptance condition). Otherwise, if the specification is nondeterministic, it has to be *determinised*, i.e., turned into an equivalent deterministic machine.

Hence, determinisation is an important operation in formal verification. It is also important in other fields such as *conformance testing* and *fault diagnosis* where deterministic testers (resp. diagnosers) have to be derived from specifications that are, in general nondeterministic due to, e.g., partial observation.

In this paper we define a determinisation operation for a class of infinite-state systems, which consists of extended automata operating on symbolic variables and communicating with the environment via synchronising actions. Variants of this model are often encountered in the literature and can be used, e.g., for the formal specification of reactive systems. The determinisation procedure consists in iterating a sequence of *local determinisation steps*, which postpone operations on the variables until it becomes clear which exact operations should have been performed. The subclass of extended automata on which the procedure terminates is characterised as *bounded-lookahead* automata, for which the observation of a bounded-length trace is enough to infer the first transition actually taken.

The result is nontrivial because the order in which local determinisation steps are iterated has a strong influence on termination. The main difficulty was to find an order for which the bounded lookahead decreases at each iteration, thus ensuring termination of the procedure.

The rest of the paper is organised as follows. We first introduce extended automata and the determinisation operation by means of examples. Then, in Section 2 we formally define the syntax and semantics of extended automata, and in Section 3 the determinisation operation is formally defined. The operation may not terminate in general, and in Section 4 the subclass for which the procedure does terminate is precisely characterised via necessary and sufficient conditions. However, these conditions are undecidable, hence, we also provide sufficient, decidable conditions for termination. In Section 5 we discuss applications of our procedure to the verification, testing, and diagnosis of reactive systems, and conclude in Section 6. The Appendix contains detailed proofs of the results.

Example 1 (extended automata, determinisation) *Figure 1 (left) depicts an extended automaton $\mathcal{S}$. In location l_0, the action a occurs. If $x \geq 0$ then the control goes to location l_1 and the variable x is decreased by 1, and if $x \leq 0$ then the control goes to location l_2 and x is increased by 2. Clearly, if $x = 0$ then the next control location and the next value of x are not uniquely defined: the system is nondeterministic.*

The right-hand side of Figure 1 depicts the automaton det($\mathcal{S}$) obtained after determinising $\mathcal{S}$. Intuitively, the locations l_1 and l_2, which could be nondeterministically chosen as the next control location after an action a, are merged into one new location denoted by $\langle l_0, \langle l_1, l_2 \rangle\rangle$. A new transition labeled by a goes from l_0 to $\langle l_0, \langle l_1, l_2 \rangle\rangle$. This transition is taken if a occurs, and if x satisfies the disjunction

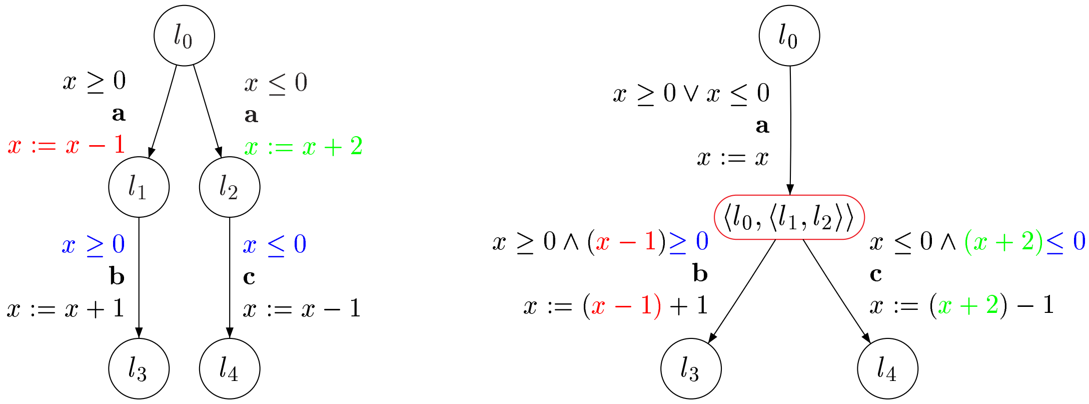

Figure 1: Left: extended automaton $\mathcal{S}$ Right: extended automaton $det(\mathcal{S})$

$x \geq 0 \vee x \leq 0$ *(which actually simplifies to* true*). This condition is the disjunction of the guards of the two transitions involved in the nondeterministic choice in* $\mathcal{S}$*.*

Note, however, that those transitions perform different assignments to variables: $x := x - 1$ *for one, and* $x := x + 2$ *for the other. Hence, the new transition from* l_0 *to* $\langle l_0, \langle l_1, l_2 \rangle\rangle$ *of* $det(\mathcal{S})$ *does not "know" which assignment to perform. To solve this problem, the idea is to* postpone *assignments until it becomes clear which one of the transitions of the nondeterministic choice was actually taken, and then to "catch up" with the assignments in order to preserve the semantics.*

Hence, if b *occurs after* a*, then the transition from* l_0 *to* l_1 *was taken (hence,* $x := x - 1$ *sould have been performed), but if* c *occurs after* a*, the transition from* l_0 *to* l_2 *was taken (hence* $x := x + 2$ *should have been performed). Note how the assignments are simulated in* $det(\mathcal{S})$*: the transition labeled by* b *(resp. by* c*) has* $x - 1$ *(resp.* $x + 2$*) substituted for* x *in its guard and assignments. To match the behaviour of* $\mathcal{S}$*, in which the transition labeled by* b *(resp.* c*) are fireable only after a transition labeled* a *has been fired with* $x \geq 0$ *(resp.* $x \leq 0$*) holding, the guard of the transition labeled by* b *(resp.* c*) in* $det(\mathcal{S})$ *is strengthened by* $x \leq 0$ *(resp.* $x \geq 0$*).*

2 Extended automata

Extended automata consist of a finite control structure and a finite set of typed variables V. Each variable $x \in V$ takes values in some domain dom_x. A *valuation* ν of the variables V is a function that associates to each variable $x \in V$ a value $\nu(x) \in dom_x$. The set of valuations of the variables V is denoted by $\mathcal{V}$. In the sequel, a predicate P over variables V is often identified with its set of "solutions", i.e., the set of valuations $\mathcal{V}' \subseteq \mathcal{V}$ of the variables V for which P is *true.*

Definition 1 (extended automaton) *An extended automaton (sometimes refered to simply as an* automaton*) is a tuple* $\mathcal{S} = \langle V, \Theta, L, l^0, \Sigma, \mathcal{T} \rangle$*:*

- V *is a finite set of typed* variables
- Θ *is the* initial condition*, a predicate on* V*, assumed to have a unique solution* $\nu_0 \in \mathcal{V}$
- L *is a nonempty, finite set of* locations *and* $l^0 \in L$ *is the* initial location*,*
- Σ *is a nonempty, finite* alphabet of actions*,*

- $\mathcal{T}$ *is a set of* transitions. *Each transition* $t \in \mathcal{T}$ *is associated with a tuple* $\langle \mathsf{o}_t, G_t, a_t, A_t, \mathsf{d}_t \rangle$, *where*
  - $\mathsf{o}_t \in L$ *is called the* origin *of the transition,*
  - G_t *is a Boolean expression over variables* V, *called the* guard,
  - $a_t \in \Sigma$ *is called the* action *of the transition,*
  - A_t *is the* assignment *of the transition: a set of expressions of the form* $(x := A^x)_{x\in V}$ *where, for each* $x \in V$, *the right-hand side* A^x *of the assignment* $x := A^x$ *is an expression on* V,
  - $\mathsf{d}_t \in L$ *is called the* destination *of the transition.*

We sometimes write $t : \langle \mathsf{o}, G, a, A, \mathsf{d} \rangle$ to emphasise the tuple associated to t. By slight abuse of notation, we shall denote by $\circ$ an operation of syntactical substitution: a guard G (or an assignment A) is composed with another assignment A' by replacing in G (resp. in the right-hand side of A) all the variables by their corresponding right-hands sides from A'. Examples of such substitutions in guards and assignments have been given in Example 1 above.

The semantics of extended automata is described by labelled transitions systems.

Definition 2 (Labelled Transition System (LTS)) *A Labelled Transition System is a tuple* $S = \langle Q, Q^0, \Lambda, \rightarrow \rangle$ *where* Q *is a set of* states, $Q^0 \subseteq Q$ *is the set of* initial states, Λ *is a set of* labels, *and* $\rightarrow \subseteq Q \times \Lambda \times Q$ *is the* transition relation.

The LTS semantics of an extended automaton enumerates the valuations $\mathcal{V}$ of its variables V. For an expression E involving (a subset of) V, and for $\nu \in \mathcal{V}$, we denote by $E(\nu)$ the value obtained by substituting in E each variable x by its value $\nu(x)$.

Definition 3 (Semantics of extended automata) *The semantics of an extended automaton* $\mathcal{S} = \langle V, \Theta, L, l^0, \Sigma, \mathcal{T} \rangle$ *is an LTS* $[\![\mathcal{S}]\!] = \langle Q, Q^0, \Lambda, \rightarrow \rangle$, *where*

- *the set of states is* $Q = L \times \mathcal{V}$,
- *the initial state is* $q^0 = \langle l_0, \nu_0 \rangle$, *where* ν_0 *is the unique valuation satisfying* Θ,
- *the set of labels is* $\Lambda = \mathcal{T}$,
- $\rightarrow$ *is the smallest relation in* $Q \times \Lambda \times Q$ *defined by the following rule:*

$$\frac{\langle l, \nu \rangle, \langle l', \nu' \rangle \in Q \quad t : \langle l, G, a, A, l' \rangle \in \mathcal{T} \quad G(\nu) = true \quad \nu' = A(\nu)}{\langle l, \nu \rangle \xrightarrow{t} \langle l', \nu' \rangle}$$

The rule says that the transition $t : \langle l, G, a, A, l' \rangle$ is *fireable* in a state $\langle l, \nu \rangle$ if the guard G evaluates to *true* when the variables evaluate according to ν; then the transition takes the system to the state $\langle l', \nu' \rangle$ where the assignment A of the transition maps the valuation ν to ν'.

We extend this notion to sequences of transitions $\sigma = t_1 \cdot t_2 \cdots t_n \in \mathcal{T}^*$, saying that σ is *fireable* in a state $q \in Q$ if there exists states $q_1 = q, q_2, \ldots q_n \in Q$ such that $\forall i = 1 \ldots n-1$, $q_i \xrightarrow{t_i} q_{i+1}$. We then write $q \xrightarrow{\sigma}$ to say that σ is fireable in q. The transition sequence σ is *initially fireable* if it is fireable in the initial state q_0. A state q is *reachable* if there exists an initially fireable transition sequence σ leading to it, i.e., $\exists \sigma \in \mathcal{T}^*, q_0 \xrightarrow{\sigma} q$. We denote by $\mathit{Reach}(\mathcal{S})$ the set of reachable states. For a sequence $\sigma = t_1 \cdots t_n \in \mathcal{T}^n$ $(n \geq 1)$, we let $\mathit{first}(\sigma) \triangleq t_1$.

Definition 4 (trace) *The* trace *of a transition sequence* $\sigma = t_1 \cdot t_2 \cdots t_n$ *is the projection* $\mathit{trace}(\sigma) = a_{t_1} \cdot a_{t_2} \cdots a_{t_n}$ *of* σ *on the set* Σ *of actions. The* set of traces *of an extended automaton* $\mathcal{S}$ *is the set of traces of initially fireable transition sequences and is denoted by* $\mathit{Traces}(\mathcal{S})$.

3 Local Determinisation

Intuitively, an extended automaton is *deterministic* if in each location, the guards of the transitions labeled by the same action are mutually exclusive. *Determinising* an extended automaton $\mathcal{S}$ means computing a deterministic extended automaton $det(\mathcal{S})$ with the same traces as $\mathcal{S}$.

Definition 5 (deterministic extended automaton) *An extended automaton $\langle V, \Theta, L, l^0, \Sigma, \mathcal{T}\rangle$ is deterministic in a location $l \in L$ if for all actions $a \in \Sigma$ and each pair $t_1 : \langle l, G_1, a, A_1, l_1\rangle$ and $t_2 : \langle l, G_2, a, A_2, l_2\rangle$ of transitions with origin l and labeled by a, the conjunction of the guards $G_1 \wedge G_2$ is unsatisfiable. The automaton is deterministic if it is deterministic in all locations $l \in L$.*

It is assumed that the guards are written in a theory where satisfiability is decidable, such as, e.g., combinations of quantifier-free Presurger arithmetic formulas, arrays, and lists. Such formulas are expressive enough to encode the most common data structures, and their satisfiability is decidable using, e.g., the classical Nelson-Oppen combination of decision procedures [6]. Note that determinism does not take reachability of states into account. However, since extended automata have a unique initial state, the definition of determinism is equivalent to the fact that the semantics of a deterministic extended automaton is a deterministic LTS in the usual sense.

Exemple 1 shows that determinising two transitions consists in merging the two transitions into a new one, and propagating guards and assignments onto transitions following them (cf. Figure 1). Formally, $follow(t) \triangleq \{t' \in \mathcal{T} | \mathsf{o}_{t'} = \mathsf{d}_t\}$. We also denote by Id_V the *identity* assignments over variables V, i.e., $x := x$ for each $x \in V$.

Definition 6 (determinising two transitions) *Let $\mathcal{S}$ be an extended automaton, and let $t_1, t_2 \in \mathcal{T}$ be two transitions with same origin $\mathsf{o} = \mathsf{o}_{t_1} = \mathsf{o}_{t_2}$ and same action $a = a_{t_1} = a_{t_2}$. The automaton $det_2(\mathcal{S}, t_1, t_2)$ is defined as follows. If $\mathsf{G}_{t_1} \wedge \mathsf{G}_{t_2}$ is unsatisfiable then $det_2(\mathcal{S}, t_1, t_2) = \mathcal{S}$, otherwise,*

- $V_{det_2(\mathcal{S},\,t)} = V_{\mathcal{S}}$
- $\Theta_{det_2(\mathcal{S},\,t)} = \Theta_{\mathcal{S}}$
- $L_{det_2(\mathcal{S},\,t)} = L_{\mathcal{S}} \cup \{\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle\}$*, where* $\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle$ *is a new location*
- $l^0_{det_2(\mathcal{S},\,t)} = l^0_{\mathcal{S}}$
- $\Sigma_{det_2(\mathcal{S},\,t)} = \Sigma_{\mathcal{S}}$
- $\mathcal{T}_{det_2(\mathcal{S},\,t)} = \mathcal{T}_{\mathcal{S}} \setminus \{t_1, t_2\} \cup \{t_{1,2}\} \cup T_1 \cup T_2$*, where*
  - $t_{1,2} = \langle \mathsf{o}, \mathsf{G}_{t_1} \vee \mathsf{G}_{t_2}, a, Id_V, \langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle\rangle$,
  - *for* $i = 1, 2$, $T_i = \bigcup_{t' \in follow(t_i)} \{modify_i(t')\}$*, where* $modify_i(t') : \langle\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle, \mathsf{G}_{t_i} \wedge \mathsf{G}_{t'} \circ \mathsf{A}_{t_i}, a_{t'}, \mathsf{A}_{t'} \circ \mathsf{A}_{t_i}, \mathsf{d}_{t'}\rangle$.

The transitions t_1 and t_2 in $\mathcal{S}$ are replaced in $det_2(\mathcal{S}, t_1, t_2)$ by the set of transitions $\{t_{1,2}\} \cup T_1 \cup T_2$. The transition $t_{1,2}$ leads from the common origin o of t_1, t_2 to the new location $\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle$; its guard is the *disjunction* of those of t_1, t_2; hence, $t_{1,2}$ can be fired whenever t_1 or t_2 can. However, $t_{1,2}$ does not perform any of the assignments of t_1, t_2 because it does not "know" which ones to perform. The assignments are postponed onto *copies* of the transitions $t' \in follow(t_i)$ $(i = 1, 2)$, *modified* in order to "catch up" with the effect of transition t_i:

- the guard $\mathsf{G}_{modify_i(t')}$ equals $\mathsf{G}_{t_i} \wedge \mathsf{G}_{t'} \circ \mathsf{A}_{t_i}$. Intuitively, this amounts to *firing the transition* $modify_i(t')$ in $det_2(\mathcal{S}, t_1, t_2)$, *under exactly the same conditions* as the transition t' in $\mathcal{S}$: the conjunct G_{t_i} "recalls" that t_i should have been fired before t', and by composing $\mathsf{G}_{t'}$ with A_{t_i}, the effect of t_i on the variables is simulated before the guard of t' is evaluated.

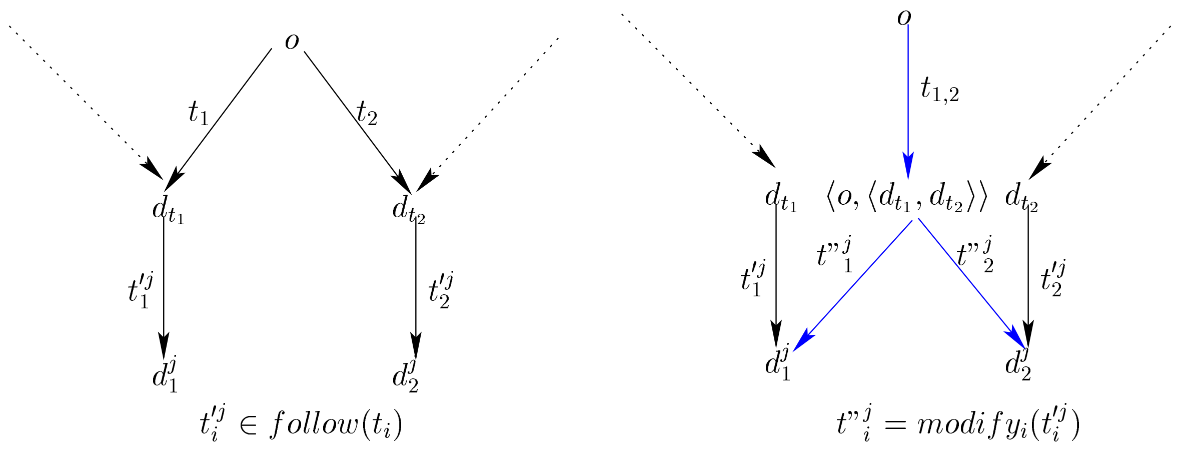

Figure 2: Determinising 2 transitions: (left) before (right) after.

- $\mathsf{A}_{modify_i(t')}$ performs the assignments of $\mathsf{A}_{t'}$ *composed* with the assignments A_{t_i}. In this way, the cumulated effect on the variables of firing in sequence t_i then t' in $\mathcal{S}$ is simulated.

Then, local determinisation consists in determinising transitions involved into nondeterminism:

Definition 7 (Local determinisation in location) *The* local determinisation in location l *of an extended automaton* $\mathcal{S} = \langle V, \Theta, L, l^0, \Sigma, \mathcal{T} \rangle$*, where* $l \in L$ *and* $a \in \Sigma$*, is defined as follows. Let* $\mathcal{T}_l \subseteq \mathcal{T}$ *be the set of all transitions with origin* l*, then:*

- $det(\mathcal{S}, l) = \mathcal{S}$ *if for every pair of same-labeled distinct transitions* t_1*,* $t_2 \in \mathcal{T}_l$*, the formula* $\mathsf{G}_{t_1} \wedge \mathsf{G}_{t_2}$ *is unsatisfiable;*

- *otherwise, choose two distinct transitions* $t_1, t_2 \in \mathcal{T}_l$ *such that* $\mathsf{a}_{t_1} = \mathsf{a}_{t_2}$*,* $\mathsf{G}_{t_1} \wedge \mathsf{G}_{t_2}$ *is satisfiable, and let* $det(\mathcal{S}, l) = det(det_2(\mathcal{S}, t_1, t_2), l)$*.*

The operation terminates: the set of pairs of nondeterministic transitions decreases.

4 Bounded-Lookahead Extended Automata

We now know how to eliminate nondeterminism from a location $l \in L_{\mathcal{S}}$. Then, to eliminate the nondeterminism globally from $\mathcal{S}$, one should iterate $det(\mathcal{S}, l)$ for all $l \in L_{\mathcal{S}}$. However, local determinisation creates new locations, which may themselves be nondeterministic and have to be determinised, which may give rise to yet another set of nondeterministic locations, etc. This raises the question of whether the global determinisation process ever terminates. In this section we define a global determinisation procedure that we show to terminate exactly for the class of *bounded lookahead* extended automata. Intuitively, an automaton is deterministic with lookahead n if any nondeterministic choice can be resolved by looking n actions ahead.

Definition 8 (bounded lookahead) *An extended automaton* $\mathcal{S} = \langle V, \Theta, L, q^0, \Sigma, \mathcal{T} \rangle$ *has lookahead* $n \in \mathbb{N}$ *in a state* $q \in Q_{[\![\mathcal{S}]\!]}$ *if* $\forall \sigma_1, \sigma_2 \in \mathcal{T}^{n+1}$*.* $q \xrightarrow{\sigma_1} \wedge q \xrightarrow{\sigma_2} \wedge trace(\sigma_1) = trace(\sigma_2) \Rightarrow first(\sigma_1) = first(\sigma_2)$*. The automaton has lookahead* n *in a set* $Q' \subseteq Q_{[\![\mathcal{S}]\!]}$ *of states if it has lookahead* n *in every* $q \in Q'$*. Finally,* $\mathcal{S}$ *has bounded lookahead if, for some* $n \in \mathbb{N}$*,* $\mathcal{S}$ *has lookahead* n *in the whole set* $Q_{[\![\mathcal{S}]\!]}$*.*

We shall find it convenient to define the lookahead of a *location* of an automaton.

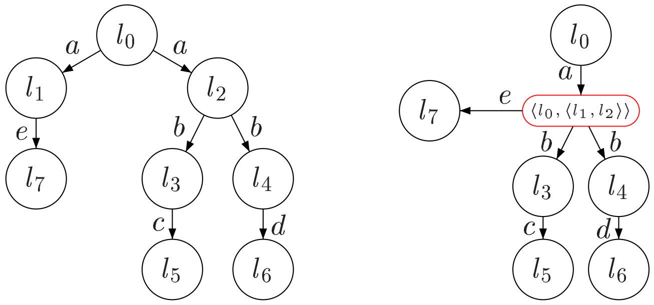

Figure 3: Inherited nondeterminism may not decrease global lookahead.

Definition 9 ((smallest) lookahead in location) *An automaton $\mathcal{S}$ has lookahead n in location $l \in L$ if $\mathcal{S}$ has lookahead n in the set $\{\langle l, \nu\rangle | \nu \in \mathcal{V}\}$. $\mathcal{S}$ has* smallest *lookahead $n \in \mathbb{N}$ in a given location l if it has lookahead n in l, and does not have lookahead $n-1$ in l. We denote by $look(l, \mathcal{S}) \in \mathbb{N}$ the smallest lookahead of location l in $\mathcal{S}$ (if it exists), otherwise, $look(l, \mathcal{S}) \triangleq \infty$.*

For example, the automaton depicted in the left-hand side of Figure 3 has $look = 1$ in l_0, because, when e occurs, the left-hand side a-labeled transition must have been fired, but when b occurs, the right-hand side a-labeled transition has been fired.

On the other hand, the automaton depicted in the left-hand side of Figure 4 does not have $look = 1$ in l_0, because the occurence of b does not reveal which of the a-labeled transitions was fired. However, the following action (either c or d) reveals all the past trace, hence, $look = 2$ in l_0 for the given automaton.

Definition 10 (global lookahead of automaton) $look(\mathcal{S}) \triangleq max_{l \in L_{\mathcal{S}}}\{look(l, \mathcal{S})\}$.

Clearly, a location l is deterministic in an automaton $\mathcal{S}$ iff $look(l, \mathcal{S}) = 0$; and the automaton $\mathcal{S}$ itself is deterministic iff $look(\mathcal{S}) = 0$.

The following proposition says that the lookahead of an automaton does not increase by local determinisation (all proofs can be found in the Appendix).

Proposition 1 (Global lookahead does not increase) $look(det(\mathcal{S}, l)) \leq look(\mathcal{S})$.

The following examples show that $look(\mathcal{S})$ may or may not decrease with local determinisation. Consider the automaton on the left-hand side of Figure 3, which has global lookahead 1. Determinising in l_0 leaves the automaton in the right-hand side, which still has the same global lookahead! The determinisation in l_0 in Figure 4, however, decreases the global lookahead of the automaton from 2 to 1.

The difference between these situations is the following: in Figure 3, the determinisation step has merged the *nondeterministic* location l_2 into the *new* location $\langle l_0, \langle l_1, l_2\rangle\rangle$, hence, the resulting automaton has *inherited* (in a sense that will be made precise below) the nondeterminism that l_2 had; because of that nondeterminism, the global lookahead has not decreased. On the other hand, the determinisation step in Fig. 4 does not have this problem: both l_1, l_2 are deterministic, and, even though the new location $\langle l_0, \langle l_1, l_2\rangle\rangle$ is nondeterministic, the nondeterminism is *created* by the fact that l_1, l_2 bring one b-labeled transition each.

Definition 11 (created/inherited nondeterminism) *Let $\mathcal{S}$ be an extended automaton and t_1, t_2 be two transitions of $\mathcal{S}$ involved into a nondeterminism in $\mathsf{o}_{t_1} = \mathsf{o}_{t_2} = \mathsf{o}$. Let $\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle$ be the new location resulting from the determinisation $det_2(\mathcal{S}, t_1, t_2)$, and assume that $\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle$ is nondeterministic in $det_2(\mathcal{S}, t_1, t_2)$. We say that this nondeterminism is* created *if both d_{t_1}, d_{t_2} are deterministic in $\mathcal{S}$, otherwise, the nondeterminism is* inherited.

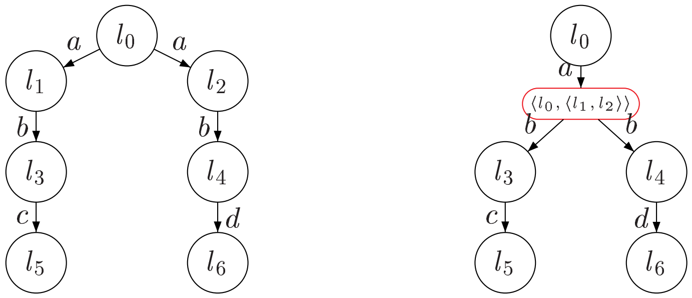

Figure 4: Created nondeterminism has decreased global lookahead.

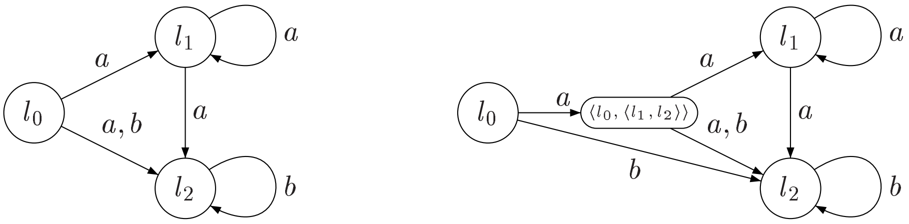

Figure 5: Depth-first determinisation may not terminate.

Now, consider a global determinisation procedure that performs local determinisation steps in a *breadth-first* order: the first iteration determinises the nondeterministic locations of the original automaton, and each subsequent iteration determinises the new nondeterministic locations, generated during the iteration that preceded it.
Figure 3 also illustrates the first iteration of such a *breadth-first* procedure on the automaton in the left-hand side. The resulting automaton is depicted on the right-hand side. Both automata have the same global lookahead =1. Hence, the lookahead cannot be used as a decreasing measure to ensure the termination of the procedure.

Even worse, applying local determinisations in a *depth-first* order (i.e., determinising new nondeterministic locations as soon as they are created) may not terminate, even when the automaton has bounded lookahead. An example is shown in Figure 5: the automaton in the left-hand side has global lookahead 1, and, by determinising in l_0, one obtains the automaton depicted in the right-hand side of the figure, which contains a sub-automaton isomorphic the automaton in the left-hand side, with global lookahead still 1. After determinising in the newly created location, the sub-automaton is still there, and remains present all through the process of *depth-first* determinisation, which, in this case, clearly does not terminate.

Hence, applying local determinisation steps in depth-first or in breadth-first order does not lead, in general, to a terminating global determinisation procedure.

However, Proposition 2 below (whose proof can be found in the Appendix) shows that if an iteration of a breadth-first procedure only gives rise to *created* nondeterminism, the global lookahead does decrease.

Proposition 2 (Global lookahead decreases if all new nondeterminism is created) *Let $\mathcal{S}'$ be an automaton obtained by determinising all nondeterministic locations $\{l_1, \dots l_k\}$ of an automaton $\mathcal{S}$*

in an arbitrary order, (i.e., $\mathcal{S}_0 = \mathcal{S}$, $\forall i \leq k-1, \mathcal{S}_{i+1} = det(\mathcal{S}_i, l_i)$, and $\mathcal{S}' = \mathcal{S}_k$). If none of these local determinisation steps gave rise to inherited nondeterminism, then $look(\mathcal{S}') < look(\mathcal{S})$.

To ensure that all new nondeterminism is created, one must determinise locations whose *direct successors* are deterministic. But now we are faced with another difficulty: if the automaton has cycles in which *every location is nondeterministic*, it is impossible to choose a location on the cycle to start determinising with! This will lead us to "breaking" such cycles by determinising one location on each of them.

Definition 12 *A location l' is a direct successor of a location l in $\mathcal{S}$ if there exists $t \in \mathcal{T}_\mathcal{S}$ such that $\mathsf{o}_t = l$ and $\mathsf{d}_t = l'$. A* cycle *is a sequence $c = t_1 \cdot t_2 \cdots t_n \in \mathcal{T}^*$ such that $\forall i = 1, \ldots n-1, \mathsf{d}_{t_i} = \mathsf{o}_{t_{i+1}}$, and $\mathsf{d}_{t_n} = \mathsf{o}_{t_1}$. The cycle is* elementary *if moreover $\forall i, j = 1, \ldots n-1, i < j \Rightarrow \mathsf{d}_{t_j} \neq \mathsf{o}_{t_i}$ holds. We say $l \in c$ if $\exists i \in \{1, \ldots n\}. l = \mathsf{d}_{t_i}$, denote by $\mathcal{C}(\mathcal{S})$ the set of cycles of $\mathcal{S}$, and by $\mathcal{C}(\mathcal{S}, l) = \{c \in \mathcal{C}(\mathcal{S}) | l \in c\}$.*

Definition 13 (nondeterministic cycle) *A cycle c is* nondeterministic *if $\forall l \in c$, l is nondeterministic. We denote by $\mathcal{N}(\mathcal{S})$ the set of nondetermnistic cycles of $\mathcal{S}$.*

Lemma 1 *For $\mathcal{S}$ an automaton and all locations $l \in L_\mathcal{S}$, $\mathcal{C}(det(\mathcal{S}, l), l) \cap \mathcal{N}(det(\mathcal{S}, l)) = \emptyset$, and $\forall c' \in \mathcal{C}(\mathcal{S}). c' \notin \mathcal{C}(\mathcal{S}, l) \wedge c' \notin \mathcal{N}(\mathcal{S}) \Rightarrow c' \in \mathcal{C}(det(\mathcal{S}, l)) \setminus \mathcal{N}(det(\mathcal{S}, l))$.*

Proof : For the first statement, note that l is deterministic in $det(\mathcal{S}, l)$, hence, by definition, a cycle $c \in \mathcal{C}(det(\mathcal{S}, l), l)$ cannot be nondeterministic in $det(\mathcal{S}, l)$, i.e., it cannot be in $\mathcal{N}(det(\mathcal{S}, l))$. For the second statement, the left-hand side of the implication means that the cycle $c' \in \mathcal{C}(\mathcal{S})$ does not visit l, but visits some other location l' which is *deterministic* in $\mathcal{S}$. Determinisation in l leaves c' unchanged, thus, $c' \in \mathcal{C}(det(\mathcal{S}, l))$, and l' is still deterministic in $det(\mathcal{S}, l)$, hence, $c' \notin \mathcal{N}(det(\mathcal{S}, l))$.□

Lemma 1 says that cycles visiting l in $det(\mathcal{S}, l)$ are *not* nondeterministic, and cycles c' that do not visit l and that are *not* nondeterministic in $\mathcal{S}$ are still *not* nondeterministic cycles of $det(\mathcal{S}, l)$. *The consequences are that determinising one location per* elementary *nondeterministic cycle generates an automaton without* any *nondeterministic cycles, and determinisation does not add new nondeterministic cycles.*

We now introduce our global determinisation procedure (Figure 6), which starts by "breaking" all elementary nondeterministic cycles, by determinising one location on each.

Theorem 1 (termination, sufficient condition) *$det(\mathcal{S})$ terminates if $look(\mathcal{S}) < \infty$.*

Proof : By Lemma 1 and Proposition 1, the elimination of nondeterministic cycles (first `while` loop in Figure 6) terminates and does not increase $look(\mathcal{S})$. Consider the sets $L'' \subseteq L'$ computed at each new iteration of the inner (third) `while` loop.

Note that $L' \neq \emptyset$ and $L'' = \emptyset$ implies that there exists a nondeterministic cycle in $\mathcal{S}_n$. Indeed, assume $l_1 \in L'$, then $L'' = \emptyset$ implies $l_1 \notin L''$, which implies that l_1 has a direct successor $l_2 \in L_{\mathcal{S}_n}$ where $\mathcal{S}'_n$ is also nondeterministic, which implies again $l_2 \in L'$. The process continues, and we eventually build a nondeterministic cycle in $\mathcal{S}_n$, which is impossible since all nondeterministic cycles were eliminated.

Inside the inner `while` loop, $L' \neq \emptyset$, and by the above reasoning, $L'' \neq \emptyset$. Hence, the `choose` l operation (from L'') inside the loop is always possible, and then determinising in location l decreases the cardinal of L' by one. Since L' is finite ($L' \subseteq L_{\mathcal{S}_n}$) and its cardinal decreases, eventually $L' = \emptyset$ and the inner `while` loop terminates. $L' = \emptyset$ also means that at the end of the inner `while` loop, $\mathcal{S}'_n$ is deterministic in all locations $L_{\mathcal{S}_n}$, hence, nondeterministic locations in $\mathcal{S}'_n$ are *new*.

For termination of the outer `while` loop, we prove $look(\mathcal{S}_{n+1}) < look(\mathcal{S}_n)$. We know that after the inner loop, the nondeterministic locations in $\mathcal{S}'_n$ are *new* (in $L_{\mathcal{S}'_n} \setminus L_{\mathcal{S}_n}$) and cannot have inherited

```
Procedure $det(\mathcal{S})$
while $\mathcal{C} := \{c \in \mathcal{N}(\mathcal{S}) | c$ elementary$\} \neq \emptyset$ do
  choose $c \in \mathcal{C}$; choose $l \in c$; $\mathcal{S} := det(\mathcal{S}, l)$
endwhile
$n := 0$; $\mathcal{S}_n := \mathcal{S}$
while $\mathcal{S}_n$ is nondeterministic do
    $\mathcal{S}'_n := \mathcal{S}_n$
    while $L' := \{l \in L_{\mathcal{S}_n} | \mathcal{S}'_n$ is nondeterministic in $l\} \neq \emptyset$ do
        $L'' := \{l' \in L' | \mathcal{S}'_n$ is deterministic in all direct successors of $l'\})$
        choose $l \in L''$
        $\mathcal{S}'_n$:= $det(\mathcal{S}'_n, l)$
    endwhile
    $\mathcal{S}_n := \mathcal{S}'_n$; $n := n + 1$
endwhile
return $\mathcal{S}_n$.
```

Figure 6: Global determinisation procedure *det*()

nondeterminism, because they were generated by determinising locations in L'', whose direct successors are, by construction, deterministic. Finally, by Proposition 2, $look(\mathcal{S}'_n) < look(\mathcal{S}_n)$, and $\mathcal{S}_{n+1}$ becomes $\mathcal{S}'_n$ after n is incremented, and the proof is done. □

The fact that bounded lookahead is *necessary* for termination is based on:

Proposition 3 (*look*() decreases by at most 1) *$look(det_2(\mathcal{S}, t_1, t_2)) \geq look(\mathcal{S}) - 1$.*

Then, a finite sequence of $det_2()$ operations cannot decrease lookahead from ∞ to 0:

Theorem 2 (necessary condition) *If $det(\mathcal{S})$ terminates then $look(\mathcal{S}) < \infty$.*

This concludes the study of the procedure's *termination*. It also *preserves traces:*

Theorem 3 *If $det(\mathcal{S})$ terminates then $Traces(det(\mathcal{S})) = Traces(\mathcal{S})$.*

All proofs can be found in the Appendix.

The determinisation procedure can be improved using approximate reachability analysis. Assume that an over-approximation $Reach^{\alpha} \supseteq Reach(\mathcal{S})$ of the reachable set of states is known (e.g., by abstract interpretation [2]). Moreover, assume that this set is described using a formula in the same logic as the autoamaton's guards, which we have assumed to be decidable for satisfiability (cf. Section 2). Then, Definition 5 of a deterministic extended automaton can be weakened, by requiring that $Reach^{\alpha} \wedge \mathsf{G}_{t_1} \wedge \mathsf{G}_{t_2}$ be unsatisfiable (instead of $\mathsf{G}_{t_1} \wedge \mathsf{G}_{t_2}$ unsatisfiable).

This new definition of determinism increases the suclass of extended automata on which the determinisation procedure terminates. The procedure now terminates for automata satisfying a modified definition of bounded lookahead, which, intuitively, requires only states in the set $Reach^{\alpha}$ (instead of $Q_{[\![\mathcal{S}]\!]}$) to have bounded lookahead.

Checking for Bounded Lookahead. The bounded lookahead condition is clearly undecidable for extended automata. We now give a sufficient criterion for this condition. We need a notion of *product* of extended automata:

Definition 14 (Synchronous Product) *Two automata $\mathcal{S}_j = \langle V_j, \Theta_j, L_j, l_j^0, \Sigma_j, \mathcal{T}_j\rangle$ $(j = 1, 2)$ are* compatible *if $V_1 \cap V_2 = \emptyset$ and $\Sigma_1 = \Sigma_2$. The synchronous product $\mathcal{S} = \mathcal{S}_1 || \mathcal{S}_2$ of two compatible automata $\mathcal{S}_1, \mathcal{S}_2$ is the automaton $\langle V\Theta, L, l^0, \Sigma, \mathcal{T}\rangle$ with: $V = V_1 \cup V_2$, $\Theta = \Theta_1 \wedge \Theta_2$, $L = L_1 \times L_2$, $l^0 = \langle l_1^0, l_2^0\rangle$, $\Sigma = \Sigma_1 = \Sigma_2$, and the set $\mathcal{T}$ of transitions of the composed system is the smallest set defined by the rule:*

$$\frac{t_1 : \langle l_1, a, G_1, A_1, q_1'\rangle \in \mathcal{T}_1 \quad t_2 : \langle l_2, a, G_2, A_2, l_2'\rangle \in \mathcal{T}_2}{t : \langle\langle l_1, l_2\rangle, a, G_1 \wedge G_2, A_1 \cup A_2, \langle l_1', l_2'\rangle\rangle \in \mathcal{T}}$$

Then, the bounded lookahead condition for an extended automaton can be equivalently formulated as follows. Consider an extended automaton $\mathcal{S} = \langle V, \Theta, L, l^0, \Sigma, \mathcal{T}\rangle$, and let the *primed copy* $\mathcal{S}'$ of $\mathcal{S}$ be the automaton obtained by "priming" all the components of $\mathcal{S}$ except the alphabet Σ, i.e., $\mathcal{S}' = \langle V', \Theta', L', l'^0, \Sigma, \mathcal{T}'\rangle$, where $V' = \{v' | v \in V\}$, $L' = \{l' | l \in L\}$, and for states $q' = (\langle l, \nu\rangle)' = \langle l', \nu'\rangle$ where ν' is the same valuation as ν, but for variables V', i.e., $\forall x' \in V'$, $\nu'(x') \triangleq \nu(x)$.

Proposition 4 (checking for bounded lookahead) *An extended automaton $\mathcal{S}$ has bounded lookahead iff, for all $q, q_1, q_2 \in Q_{[\![\mathcal{S}]\!]}$ and distinct transitions $t_1, t_2 \in \mathcal{T}_\mathcal{S}$ with $\mathsf{a}_{t_1} = \mathsf{a}_{t_2}$, if $q \xrightarrow{t_1}_\mathcal{S} q_1 \wedge q \xrightarrow{t_2}_\mathcal{S} q_2$ then there exists no infinite execution in $\mathcal{S}||\mathcal{S}'$ starting from (q_1, q_2'), where $\mathcal{S}'$ denotes the primed copy of $\mathcal{S}$.*

The conditions of Proposition 4 are decidable if $\mathcal{S}$ is finite-state but are not decidable in general. For infinite-state extended automata $\mathcal{S}$, we can build finite-state abstractions $\mathcal{S}^\alpha$ that simulate the transition sequences σ of $\mathcal{S}$ (i.e., whenever $q \xrightarrow{\sigma} q'$ holds in $\mathcal{S}$, $\alpha(q) \xrightarrow{\alpha(\sigma)} \alpha(q')$ holds in $\mathcal{S}^\alpha$). The bounded lookahead conditions of Proposition 4 can be then automatically checked on $\mathcal{S}^\alpha$, and, if they hold, the simulation property guarantees that they also hold on $\mathcal{S}$. This gives a sufficient criterion for bounded lookahead, which is, in general, not necessary ($\mathcal{S}^\alpha$ may contain cycles not present in $\mathcal{S}$), and whose precision can be improved by taking more precise abstractions $\mathcal{S}^\alpha$.

5 Applications of Determinisation

Verification. A standard verification problem is that of trace (or language) inclusion: given two systems $\mathcal{I}$ (the *implementation*) and $\mathcal{S}$ (the *specification*), decide whether *Traces*$(\mathcal{I}) \subseteq$ *Traces*$(\mathcal{S})$. When $\mathcal{I}$, $\mathcal{S}$ are extended automata and $\mathcal{S}$ is deterministic, the problem reduces to a reachability problem in the extended automaton $\mathcal{I}||\overline{\mathcal{S}}$, where $\overline{\mathcal{S}}$ is obtained from $\mathcal{S}$ by adding a new location *fail* $\notin L$, and for each $l \in L$ and $a \in \Sigma$, a new transition with origin l, destination *fail*, action a, identity assignments, and guard $\bigwedge_{t:\langle l, a, G_t, A_t, l_t'\rangle \in \mathcal{T}} \neg G_t$. The new transitions allow actions in $\overline{\mathcal{S}}$ whenever they are not allowed in $\mathcal{S}$. Hence, when $\mathcal{S}$ is deterministic, *Traces*$(\mathcal{I}) \subseteq$ *Traces*$(\mathcal{S})$ iff no location in the set $\{\langle l, \mathit{fail} | l \in L_\mathcal{I}\rangle\}$ is reachable in $\mathcal{I}||\overline{\mathcal{S}}$.

When $\mathcal{S}$ is *not* deterministic, the above statement is incorrect. Let $\mathcal{S}$ be the nondeterministic automaton in the left-hand side of Figure 3. A naive application of the completion operation on $\mathcal{S}$ builds a transition labeled b from l_1 to *fail*, suggesting that $a \cdot b$ is not a trace of $\mathcal{S}$, which is obviously false. In particular, verification would wrongly declare erroneous an implementation that exhibits the trace $a \cdot b$. Hence, to be adequate for verification, $\mathcal{S}$ has to be *determinised* before being completed.

Conformance Testing is a functional testing that consists in comparing a black-box implementation $\mathcal{I}$ to a formal specification $\mathcal{S}$ according a conformance relation. The implementation is a black box, i.e., only its interface (input and output alphabet) is known. In [7] we show that conformance of an implementation $\mathcal{S}$ to a specification according to the standard **ioco** relation [9] is equivalent to the fact that running a *canonical tester* in parallel with the implementation never reaches a certain set of locations. The tester can be automatically computed from the specification using operations similar to the completion operation, defined above, and, of course, determinisation. Without determinisation, the tester might wrongly declare non-conformant an implementation that is conformant to the specification (a phenomenon similar to that exhibited by the trace $a \cdot b$, noted in the previous paragraph).

Diagnosis. The determinisation problem for extended automata also has a close relationship with diagnosis for discrete event systems [8]. For instance, an extended automaton with bounded lookahead can be seen as an automaton in which nondeterministic choices are diagnosable; and checking membership to the class of bounded lookahead automata can be reduced to a diagnosability problem in this model. Also, the sufficient criterion for bounded lookahead (around Proposition 4) was inspired by the algorithm used to check diagnosability [5], based on the search of specific cycles in a product of the specification with itself.

Conversely, it could be profitable to re-define diagnosability in terms of our *bounded lookahead* condition, in order to capture a notion of diagnosability for richer, infinite-state models. Finally, the construction of a diagnoser from an automaton specifying a plant and a fault model is based on determinisation: one has to determinise the plant "decorated" with past occurrences of (unobservable) faults. Our determinisation procedure then constitutes a basic block for the construction of diagnosers from plants specified as extended automata, thus extending the works on diagnosis to more expressive, infinite-state models.

6 Conclusion and Future Work

In this paper we present a determinisation procedure for extended automata and prove that the procedure terminates exactly for the class of extended automata *with bounded lookahead*. The intuition behind this class is that in any location, for any trace, there exists a bounded number of steps after which the first transition taken is uniquely identified. Technical difficulties for proving termination arise from the fact that the order in which elementary determinisation steps are applied has a strong influence on termination. The main dificulty was to find an adequate order, for which the bounded lookahead provides a decreasing measure.

The models of extended automata considered in this paper only have observable actions. One can also consider models with *internal* (unobservable) actions. In this case, determinisation first consists in an extended *ϵ-closure* generalising that of finite automata. The extended ϵ-closure algorithm is then based on the propagation of guards and actions onto the next transitons labeled by observable actions [10], and terminates iff there are no cycles of transitions labeled by internal actions.

The present work was initially motivated by *conformance testing*, more specifically, model-based testing based on the **ioco** theory [9]. In this framework, off-line test generation (computation of test cases from specifications) involves *determinising* the specification in order to compute the next possible observable actions after each trace, and, therefore, to obtain *deterministic* test cases [4]. In that work, we consider an extension of the model presented here (actions are either inputs or outputs and may carry communication parameters), which can be handled by a small modification of our determinisation procedure. The procedure also has potentially interesting application in the verification and diagnosis of infinite-state systems.

References

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

[2] Patrick Cousot and Radhia Cousot. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *4th POPL, Los Angeles, CA*, pages 238–252, January 1977.

[3] John E. Hopcroft, Rajeev Motwani, Rotwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computability.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.

[4] B. Jeannet, T. Jéron, V. Rusu, and E. Zinovieva. Symbolic test selection based on approximate analysis. In *11th Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'05), Edinburgh (Scottland)*, volume 3440 of *LNCS*, April 2005.

[5] S. Jiang, Z. Huang, V. Chandra, and R. Kumar. A polynomial time algorithm for diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 46(8):1318–1321, August 2001.

[6] Greg Nelson and Derek C. Oppen. Simplification by cooperating decision procedures. *ACM Trans. Program. Lang. Syst.*, 1(2):245–257, 1979.

[7] Vlad Rusu, Hervé Marchand, and Thierry Jéron. Automatic verification and conformance testing for validating safety properties of reactive systems. In John Fitzgerald, Andrzej Tarlecki, and Ian Hayes, editors, *Formal Methods 2005 (FM05)*, LNCS. Springer, July 2005.

[8] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Failure diagnosis using discrete event models. *Proceedings of the IEEE Transactions on Automatic Control*, 4(2):105–124, 1996.

[9] Jan Tretmans. Test generation with inputs, outputs and repetitive quiescence. *Software - Concepts and Tools*, 17(3):103–120, 1996.

[10] E. Zinovieva. *Méthodes symboliques pour la génération de tests de systèmes réactifs comportant des données,*. PhD thesis, Univ. of Rennes, Nov. 2004.

7 Appendix

We prove the partial correctness and termination of the global determinisation procedure. Most results in this Appendix are based on defining and proving properties of functions between sequences of transitions in the original and the determinised automaton. We shall use the following notations: for a sequence σ, $\mathit{first}(\sigma)$, $\mathit{snd}(\sigma)$, and $\mathit{last}(\sigma)$ respectively define the first, second, and last element of σ, if they exist, otherwise, by convention, they return the empty sequence ϵ.

7.1 Partial Correctness

The local determinisation operation $\mathit{det}_2(\mathcal{S}, t_1, t_2)$ (cf. Definition 6) adds a transition $t_{1,2}$ to the transitions of $\mathcal{S}$ and modifies the guards and assignments of transitions that follow t_1, t_2 in $\mathcal{S}$.

The following lemma states some basic properties of the $\mathit{det}_2(\mathcal{S}, t_1, t_2)$ operation. Essentialy, the first item says that the new transition $t_{1,2}$ is fireable in $\mathit{det}_2(\mathcal{S}, t_1, t_2)$ whenever t_1 or t_2 are fireable in $\mathcal{S}$, but it does not necesarilly lead to the same state as some assignments to variables are postponed. The second item says that the sequence of transitions $t_{1,2} \cdot \mathit{modify}_i(t'_i)$ (where $\mathit{modify}_i(t'_1)$ is a copy of a transition t'_i following t_i ($i = 1, 2$), adequately modified to include the previously postponed assignments) performs a faithful simulation of $t_i \cdot t'_i$.

Lemma 2 *With the notations of Definition 6, for all states $q, q' \in Q_{[\![\mathcal{S}]\!]}$,*

(i) $q \xrightarrow{t_{1,2}}_{\mathit{det}_2(\mathcal{S}, t_1, t_2)}$ *iff* $q \xrightarrow{t_1}_{\mathcal{S}}$ *or* $q \xrightarrow{t_2}_{\mathcal{S}}$

(ii) $\forall t'_i \in \mathit{follow}(t_i)$, $i = 1, 2$, $q \xrightarrow{t_i \cdot t'_i}_{\mathcal{S}} q'$ *iff* $q \xrightarrow{t_{1,2} \,\cdot\, \mathit{modify}_i(t_i)}_{\mathit{det}_2(\mathcal{S}, t_1, t_2)} q'$.

Proof : For statement (i): let $q = \langle l, \nu \rangle$, then $q \xrightarrow{t_{1,2}}_{\mathit{det}_2(\mathcal{S}, t_1, t_2)}$ iff (a) $\mathsf{o}_{t_{1,2}} = l$ and (b) $\nu \models \mathsf{G}_{t_{1,2}}$. Since $\mathsf{o}_{t_1} = \mathsf{o}_{t_2} = \mathsf{o}_{t_{1,2}}$ and $\mathsf{G}_{t_{1,2}} = \mathsf{G}_{t_1} \vee \mathsf{G}_{t_2}$, the conjunction of (a) and (b) is equivalent to ($q \xrightarrow{t_1}_{\mathcal{S}}$ or $q \xrightarrow{t_1}_{\mathcal{S}}$), which proves statement (i).

For statement (ii): let $t'_i \in \mathit{follow}(t_i)$. We have $q \xrightarrow{t_i \cdot t'_i}_{\mathcal{S}} q'$ iff there exists a state q'' such that (α) $q \xrightarrow{t_i}_{\mathcal{S}} q''$ and (β) $q'' \xrightarrow{t'_i}_{\mathcal{S}} q'$. Now, ($\alpha$) is equivalent to $q = \langle \mathsf{o}_{t_i}, \nu \rangle$, $q'' = \langle \mathsf{d}_{t_i}, \nu'' \rangle$ for some valuations ν, ν'' such that $\nu \models \mathsf{G}_{t_i}$ and $\nu'' = \mathsf{A}_{t_i}(\nu)$; and ($\beta$) si equivalent to $q'' = \langle \mathsf{o}_{t'_i}, \nu'' \rangle$, $q' = \langle \mathsf{d}_{t'_i}, \nu' \rangle$ with $\nu'' \models \mathsf{G}_{t'_i}$ and $\nu' = \mathsf{A}_{t'_i}(\nu'')$. Since $\mathsf{o}_{t'_i} = \mathsf{d}_{t_i}$ we obtain overall that ($\alpha \wedge \beta$) is equivalent to $q = \langle \mathsf{o}_{t_i}, \nu \rangle$, $q' = \langle \mathsf{d}_{t'_i}, \nu' \rangle$ such that $\nu \models \mathsf{G}_{t_i}$ and $\mathsf{A}_{t_i}(\nu) \models \mathsf{G}_{t'_i}$ and $\nu' = (\mathsf{A}_{t'_i} \circ \mathsf{A}_{t_i})(\nu)$. Now, since $\mathsf{A}_{t_i}(\nu) \models \mathsf{G}_{t'_i}$ is equivalent to $\nu \models \mathsf{G}_{t'_i} \circ \mathsf{A}_{t_i}$, we further obtain that ($\alpha \wedge \beta$) can be equivalently rewritten into (δ): $\nu \models \mathsf{G}_{t_i}$ and (γ): $\nu \models \mathsf{G}_{t_i} \wedge (\mathsf{G}_{t'_i} \circ \mathsf{A}_{t_i})$ and (η): $\nu' = (\mathsf{A}_{t'_i} \circ \mathsf{A}_{t_i})(\nu)$.

Now, consider Definition 6 of the $\mathit{det}_2(\mathcal{S}, t_1, t_2)$ operation - specifically, the construction of the transitions $t_{1,2}$ and $\mathit{modify}_i(t'_i)$. We have

- $t_{1,2} : \langle \mathsf{o}, \mathsf{G}_{t_1} \vee \mathsf{G}_{t_2}, a, Id_V, \langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2} \rangle \rangle \rangle$,
- $\mathit{modify}_i(t'_i) : \langle \langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2} \rangle \rangle, \mathsf{G}_{t_i} \wedge \mathsf{G}_{t'_i} \circ \mathsf{A}_{t_i}, a_{t'_i}, \mathsf{A}_{t'_i} \circ \mathsf{A}_{t_i}, \mathsf{d}_{t'_i} \rangle$.

From this, we obtain that ($\delta \wedge \gamma \wedge \eta$) is equivalent to $q \xrightarrow{t_{1,2}}_{\mathit{det}_2(\mathcal{S}, t_1, t_2)} \tilde{q} \xrightarrow{\mathit{modify}_i(t'_i)}_{\mathit{det}_2(\mathcal{S}, t_1, t_2)} q'$, for some state $\tilde{q}$ (here, $\tilde{q} = \langle \langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2} \rangle \rangle, \nu \rangle$, which is equivalent to $q \xrightarrow{t_{1,2} \,\cdot\, \mathit{modify}_i(t'_i)}_{\mathit{det}_2(\mathcal{S}, t_1, t_2)} q'$, i.e., to the right-hand side of (ii).

We have obtained that the left-hand side of (ii) is equivalent to ($\alpha \wedge \beta$), which is equivalent to ($\delta \wedge \gamma \wedge \eta$), which is equivalent to the right-hand side of (ii), which concludes our proof. □

The following definition will allow, in particular, to re-formulate the definition of bounded lookahead (Definition 8) in a manner sometimes more convenient in subsequent proofs.

Definition 15 (executable sequence with given trace) *For an extended automaton $\mathcal{S}$, a state $q \in Q_{[\![\mathcal{S}]\!]}$, and a word $w \in \Sigma_{\mathcal{S}}^*$, we denote by $seq_q(\mathcal{S}, w)$ the set $\{\sigma \in \mathcal{T}^* | q \xrightarrow{\sigma}_{\mathcal{S}} \wedge trace(\sigma) = w\}$ of executable sequences starting from q and whose trace is w.*

Then, Definition 8 is equivalently reformulated as follows: a state $q \in Q_{[\![\mathcal{S}]\!]}$ has lookahead n if, for all words in $w \in \Sigma_{\mathcal{S}}^{n+1}$, every pair of sequences $\sigma_1, \sigma_2 \in seq_q(\mathcal{S}, w)$ satisfies $first(\sigma_1) = first(\sigma_2)$. In the rest of this section we define two functions between sequences of transitions, which will serve as a basic tool for proving both partial correctness and termination.

Definition 16 (direct function) *With the notations of Definition 6 we define a function $h : \mathcal{T}_{\mathcal{S}}^* \to \mathcal{T}_{det_2(\mathcal{S},\,t_1,\,t_2)}^*$, obtained by applying the following rules in the given order:*

1. *$h(\epsilon) = \epsilon$*
2. *$h(t \cdot \sigma) = t \cdot h(\sigma)$ for all transitions $t \notin \{t_1, t_2\}$*
3. *$h(\sigma \cdot t_1) = h(\sigma \cdot t_2) = h(\sigma) \cdot t_{1,2}$ if $last(\sigma) \notin \{t_1, t_2\}$*
4. *$\forall t_i' \in follow(t_i)$ $(i = 1, 2)$, $h(\sigma \cdot t_i \cdot t_i' \cdot \sigma') = h(\sigma) \cdot t_{1,2} \cdot modify_i(t_i') \cdot h(\sigma')$ if $last(\sigma) \notin \{t_1, t_2\}$.*

The second rule maps each transition not involved into the considered nondeterminism to itself. The third rule deals with sequences ending with transitions t_1 or t_2, which are mapped to $t_{1,2}$, except if they are preceded by one of t_1, t_2, in which case the fourth rule may apply. The fourth rule replaces patterns of the form $t_1 \cdot t_1'$, with $t_1' \in follow(t_1)$, by patterns $t_{1,2} \cdot modify_1(t_1')$ (and similarly for t_2). The constraint $last(\sigma) \notin \{t_1, t_2\}$ ensures that the replacement is done as early as possible.

Lemma 3 *With the notations of Definitions 6 and 16, for each $w \in \Sigma_{\mathcal{S}}^*$ and state $q \in Q_{[\![\mathcal{S}]\!]}$, the function h defines a mapping between $seq_q(\mathcal{S}, w)$ and $seq_q(det_2(\mathcal{S}, t_1, t_2), w)$.*

Proof : We have to prove $h(seq_q(\mathcal{S}, w)) \subseteq seq_q(det_2(\mathcal{S}, t_1, t_2), w)$. We prove the following slightly stronger statements by well-founded induction over σ:

(i) $\forall q, q' \in Q_{[\![\mathcal{S}]\!]}$, $\sigma \in \mathcal{T}_{\mathcal{S}}^*$, $q \xrightarrow{\sigma}_{\mathcal{S}} q'$ and $last(\sigma) \notin \{t_1, t_2\}$ implies $q \xrightarrow{h(\sigma)}_{det_2(\mathcal{S},\,t_1,\,t_2)} q'$

(ii) $\forall q \in Q_{[\![\mathcal{S}]\!]}$, $\sigma \in \mathcal{T}_{\mathcal{S}}^*$, $q \xrightarrow{\sigma}_{\mathcal{S}}$ and $last(\sigma) \in \{t_1, t_2\}$ implies $q \xrightarrow{h(\sigma)}_{det_2(\mathcal{S},\,t_1,\,t_2)}$.

For statement (i): if $\sigma = \epsilon$ it is trivial, otherwise, let $last(\sigma) = t \in \mathcal{T}_{\mathcal{S}} \setminus \{t_1, t_2\}$. Let then σ_1 be the longest prefix of σ not containing any of t_1, t_2. Then, $h(\sigma_1) = \sigma_1$. If $\sigma_1 = \sigma$ then (i) is proved, because in this case $q \xrightarrow{\sigma}_{\mathcal{S}} q'$ is equivalent to $q \xrightarrow{h(\sigma)}_{det_2(\mathcal{S},\,t_1,\,t_2)} q'$ - the sequence $\sigma = \sigma_1 = h(\sigma) = h(\sigma_1)$ visits a part of the system outside of t_1, t_2, on which determinisation of t_1, t_2 has no influence. Otherwise, $\sigma = \sigma_1 \cdot t_i \cdot t_i' \cdot \sigma_2 \cdot t$ with $t_i' \in follow(t_i)$, $i = 1, 2$. By the definition of h, $h(\sigma) = h(\sigma_1) \cdot t_{1,2} \cdot modify_i(t_i') \cdot h(\sigma_2) \cdot t$.

Let then q_1, q_2, q_3 be the states such that $q \xrightarrow{\sigma_1}_{\mathcal{S}} q_1 \xrightarrow{t_1 \cdot t_i'}_{\mathcal{S}} q_2 \xrightarrow{\sigma_2}_{\mathcal{S}} q_3 \xrightarrow{t}_{\mathcal{S}} q'$. By induction hypothesis, definition of h, and Lemma 2, $q \xrightarrow{h(\sigma_1)}_{det_2(\mathcal{S},\,t_1,\,t_2)} q_1 \xrightarrow{t_{1,2} \,\cdot\, modify_i(t_i')}_{det_2(\mathcal{S},\,t_1,\,t_2)} q_2 \xrightarrow{h(\sigma_2)}_{det_2(\mathcal{S},\,t_1,\,t_2)} q_3 \xrightarrow{t}_{det_2(\mathcal{S},\,t_1,\,t_2)} q'$, i.e., $q \xrightarrow{h(\sigma)}_{det_2(\mathcal{S},\,t_1,\,t_2)} q'$ and the proof is done.

For statement (ii): if $\sigma = \epsilon$ it is trivial, otherwise, let σ_2 be the longest suffix of σ that consists *only* of transitions in $\{t_1, t_2\}$. Then, $\sigma = \sigma_1 \cdot \sigma_2$ and $last(\sigma_1) \notin \{t_1, t_2\}$. There are two cases:

- if the length of σ_2 is *even*, then σ_2 can be decomposed as $\sigma_2 = \sigma_2^1 \cdot \sigma_2^2 \cdots \sigma_2^n$ $(n \geq 1)$ where each σ_2^j is of the form $t_i \cdot t_1'$ with $t_i' \in follow(t_i)$, $i = 1, 2$. By the definition of h, $h(\sigma) = h(\sigma_1) \cdot \sigma_2'^1 \cdots \sigma_2'^n$ where each $\sigma_2'^j$ is of the form $t_{1,2} \cdot modify(t_i')$, $i = 1, 2$. By item (i) above and Lemma 2, whenever $q \xrightarrow{\sigma_1}_{\mathcal{S}} q_1 \xrightarrow{\sigma_2^1}_{\mathcal{S}} q_2 \cdots q_n \xrightarrow{\sigma_2^n}_{\mathcal{S}} q'$ holds, $q \xrightarrow{h(\sigma_1)}_{det_2(\mathcal{S},\,t_1,\,t_2)} q_1 \xrightarrow{\sigma_2'^1}_{det_2(\mathcal{S},\,t_1,\,t_2)} q_2 \cdots q_n \xrightarrow{\sigma_2'^n}_{det_2(\mathcal{S},\,t_1,\,t_2)} q'$ holds as well, which implies that $q \xrightarrow{h(\sigma)}_{det_2(\mathcal{S},\,t_1,\,t_2)}$ and the proof is done in this case.

- if the length of σ_2 is *odd*, then σ_2 can be decomposed as $\sigma_2 = \sigma_2^1 \cdot \sigma_2^2 \cdots \sigma_2^n \cdot t$ $(n \geq 1)$ where each σ_2^j is of the form $t_i \cdot t'_1$ with $t'_i \in \mathit{follow}(t_i)$, $i = 1, 2$, and $t \in \{t_1, t_2\}$. By the definition of h, $h(\sigma) = h(\sigma_1) \cdot \sigma_2'^1 \cdots \sigma_2'^n \cdot t_{1,2}$ where, again, each $\sigma_2'^j$ is of the form $t_{1,2} \cdot \mathit{modify}(t'_i)$, $i = 1, 2$. By item (i) above, and Lemma 2, whenever $q \xrightarrow{\sigma_1}_{\mathcal{S}} q_1 \xrightarrow{\sigma_2^1}_{\mathcal{S}} q_2 \cdots q_n \xrightarrow{\sigma_2^n}_{\mathcal{S}} q_{n+1} \xrightarrow{t}_{\mathcal{S}}$ holds, $q \xrightarrow{h(\sigma_1)}_{det_2(\mathcal{S},\,t_1,\,t_2)} q_1 \xrightarrow{\sigma_2'^1}_{det_2(\mathcal{S},\,t_1,\,t_2)} q_2 \cdots q_n \xrightarrow{\sigma_2'^n}_{det_2(\mathcal{S},\,t_1,\,t_2)} q_{n+1} \xrightarrow{t_{1,2}}_{det_2(\mathcal{S},\,t_1,\,t_2)}$ holds as well, hence, $q \xrightarrow{h(\sigma)}_{det_2(\mathcal{S},\,t_1,\,t_2)}$ and the proof of this case and of (ii) are done.

Finally, let $\sigma \in \mathit{seq}_q(\mathcal{S}, w)$.

- if $\sigma = \epsilon$ then clearly $h(\sigma) \in \mathit{seq}_q(\mathit{det}_2(\mathcal{S}, t_1, t_2), w)$.
- otherwise, if $\mathit{last}(\sigma) \notin \{t_1, t_2\}$, then item (i) above implies $h(\sigma) \in \mathit{seq}_q(\mathit{det}_2(\mathcal{S}, t_1, t_2), w)$.
- otherwise, if $\mathit{last}(\sigma) \in \{t_1, t_2\}$, then item (ii) above implies $h(\sigma) \in \mathit{seq}_q(\mathit{det}_2(\mathcal{S}, t_1, t_2), w)$.

Overall, $h(\mathit{seq}_q(\mathcal{S}, w)) \subseteq \mathit{seq}_q(\mathit{det}_2(\mathcal{S}, t_1, t_2), w)$. □

We now define an *inverse function*, between sequences of $\mathit{det}_2(\mathcal{S}, t_1, t_2)$ and sequences of $\mathcal{S}$. This new function is *not* the inverse of h, because h is not injective as it maps both t_1, t_2 to $t_{1,2}$.

Definition 17 (inverse function) *With the notations of Definition 6, we define a function* $\varphi : \mathcal{T}^*_{det_2(\mathcal{S},\,t_1,\,t_2)} \to \mathcal{T}^*_{\mathcal{S}}$ *by:*

- $\varphi(t_{1,2}) = \varphi(\epsilon) = \epsilon$
- $\varphi(\mathit{modify}_i(t'_i)) = t_i \cdot t'_i$, $\forall t'_i \in \mathit{follow}(t_i)$, $i = 1, 2$
- $\varphi(t) = t$ *for all other transitions in* $\mathcal{T}_{det_2(\mathcal{S},\,t_1,\,t_2)}$
- $\varphi(\sigma \cdot \sigma') = \varphi(\sigma) \cdot \varphi(\sigma')$.

Lemma 4 *With the notations of Definitions 6 and 17, for each* $w \in \Sigma^*_{\mathcal{S}}$ *and state* $q \in Q_{[\![\mathcal{S}]\!]}$*, the function* φ *defines a mapping between* $\mathit{seq}_q(\mathit{det}_2(\mathcal{S}, t_1, t_2), w)$ *and* $\mathit{seq}_q(\mathcal{S}, w)$.

Proof : First, note that the lemma deals with the states $Q_{[\![\mathcal{S}]\!]}$ of the original automaton, not with the states of the determinised automaton. The determinised automaton has one more location (and corresponding states) which are dealt with by a subsequent lemma.

We have to prove $\varphi(\mathit{seq}_q(\mathit{det}_2(\mathcal{S}, t_1, t_2), w)) \subseteq (\mathit{seq}_q(\mathcal{S}, w))$. We prove the following slightly stronger statements by well-founded induction over σ:

(α) $\forall q, q' \in Q_{[\![\mathcal{S}]\!]}$, $\sigma \in \mathcal{T}^*_{det_2(\mathcal{S},\,t_1,\,t_2)}$, $q \xrightarrow{\sigma}_{det_2(\mathcal{S},\,t_1,\,t_2)} q'$ and $\mathit{last}(\sigma) \neq t_{1,2}$ implies $q \overset{\varphi(\sigma)}{\to}_{\mathcal{S}} q'$

(β) $\forall q \in Q_{[\![\mathcal{S}]\!]}$, $\sigma \in \mathcal{T}^*_{det_2(\mathcal{S},\,t_1,\,t_2)}$, $q \xrightarrow{\sigma}_{det_2(\mathcal{S},\,t_1,\,t_2)}$ and $\mathit{last}(\sigma) = t_{1,2}$ implies $q \overset{\varphi(\sigma)}{\to}_{\mathcal{S}}$.

To prove (α): let σ_1 be the longest prefix of σ not contaning $t_{1,2}$. Hence, σ_1 does not contain any transition in $\mathit{modify}_i(\mathit{follow}(t_i))$, because otherwise it would also contain their prececessor $t_{1,2}$.[1]

Hence, by the definition of φ, $\varphi(\sigma_1) = \sigma_1$. If $\sigma_1 = \sigma$, whenever $q \xrightarrow{\sigma}_{det_2(\mathcal{S},\,t_1,\,t_2)} q'$ holds, $q \overset{\varphi(\sigma)}{\to}_{\mathcal{S}} q'$ holds as well - because in this case σ visits a portion of the automaton $\mathit{det}_2(\mathcal{S}, t_1, t_2)$ where determinisation had no influence - and (α) is proved in this case.

Otherwise, σ_1 is a strict prefix of σ. Since $\mathit{last}(\sigma) \neq t_{1,2}$, we have $\sigma = \sigma_1 \cdot t_{1,2} \cdot \mathit{modify}_i(t'_i) \cdot \sigma_2$, for some $t'_i \in \mathit{follow}(t_i)$ and $i = 1, 2$, and $\mathit{last}(\sigma_2) \neq t_{1,2}$. Then, $\varphi(\sigma) = \sigma_1 \cdot t_i \cdot t'_i \cdot \varphi(\sigma_2)$. By Lemma 2

[1] Here, the fact that $q \in Q_{[\![\mathcal{S}]\!]}$ is important, because this means in particular that σ does not start in the "new" location of $\mathit{det}_2(\mathcal{S}, t_1, t_2)$, hence, it cannot start with a transition in $\mathit{modify}_i(\mathit{follow}(t_i))$ and contradict the statement that all transitions in $\mathit{modify}_i(\mathit{follow}(t_i))$ are preceded by $t_{1,2}$

and the induction hypothesis of (α), if $q \xrightarrow{\sigma_1}_{det_2(\mathcal{S},t_1,t_2)} q_1 \xrightarrow{t_{1,2} \cdot modify_i(t'_i)}_{det_2(\mathcal{S},t_1,t_2)} q_2 \xrightarrow{\sigma_2}_{det_2(\mathcal{S},t_1,t_2)} q'$ holds, $q \xrightarrow{\sigma_1}_{\mathcal{S}} q_1 \xrightarrow{t_i \cdot t'_i}_{\mathcal{S}} q_2 \xrightarrow{\varphi(\sigma_2)}_{\mathcal{S}} q'$ holds as well, and (α) is proved.

To prove (β): in this case we can decompose σ as $\sigma = \sigma' \cdot t_{1,2}$, and $last(\sigma') \neq t_{1,2}$, because by construction $t_{1,2}$ is not a self-loop - its origin is a location of $\mathcal{S}$ but its destination is the new location $\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2} \rangle\rangle$ which is not a location of $\mathcal{S}$. Hence, we can apply (α) that we have proved above. Let then q' be such that $q \xrightarrow{\sigma'}_{det_2(\mathcal{S},t_1,t_2)} q' \xrightarrow{t_{1,2}}_{det_2(\mathcal{S},t_1,t_2)}$, by (α) we have $q \xrightarrow{\varphi(\sigma')}_{\mathcal{S}} q'$, and since $\varphi(\sigma) = \varphi(\sigma' \cdot t_{1,2}) = \varphi(\sigma') \cdot \varphi(t_{1,2}) = \varphi(\sigma')$, we have also $q \xrightarrow{\varphi(\sigma)}_{\mathcal{S}} q'$, and (β) is proved.

Finally, let $\sigma \in seq_q(det_2(\mathcal{S}, t_1, t_2), w)$.

- if $\sigma = \epsilon$ then trivially $\varphi(\sigma) \in seq_q(\mathcal{S}, w)$
- otherwise, if $last(\sigma) \neq t_{1,2}$ then, by (α), $\varphi(\sigma) \in seq_q(\mathcal{S}, w)$
- otherwise, if $last(\sigma) = t_{1,2}$ then, by (β), $\varphi(\sigma) \in seq_q(\mathcal{S}, w)$.

Overall, we obtain $\varphi(seq_q(det_2(\mathcal{S}, t_1, t_2), w)) \subseteq (seq_q(\mathcal{S}, w))$. □

Corollary 1 ($det_2()$ preserves traces) *$Traces(det_2(\mathcal{S}, t_1, t_2)) = Traces(\mathcal{S})$.*

Proof : $\subseteq$: let $w \in Traces(det_2(\mathcal{S}, t_1, t_2))$. Then, by Definitions 4, 15, there exists a sequence $\sigma' \in seq_{q^0}(det_2(\mathcal{S}, t_1, t_2), w)$, where q^0 is the initial state of $det_2(\mathcal{S}, t_1, t_2)$, which by Definition 6 of the $det_2()$ operation is also the initial state of $\mathcal{S}$. By Lemma 4, the sequence $\varphi(\sigma')$ is in $seq_{q^0}(\mathcal{S}, w)$; hence, $w \in Traces(\mathcal{S})$.

$\supseteq$: let $w \in Traces(\mathcal{S})$. By Definitions 4, 15, there exists a sequence $\sigma' \in seq_{q^0}(\mathcal{S})$ for w, and q^0 is initial for both $\mathcal{S}$ and $det_2(\mathcal{S}, t_1, t_2)$. By Lemma 3, $\sigma' = h(\sigma) \in seq_{q^0}(det_2(\mathcal{S}, t_1, t_2), w)$ is a sequence of trace w in $det_2(\mathcal{S}, t_1, t_2)$; hence, $w \in Traces(det_2(\mathcal{S}, t_1, t_2))$. □

Our global determinisation procedure $det(\mathcal{S})$ is just an iteration of local determininisation steps in some particular order. Hence, we immediately have

Theorem 3 (Partial correctness) *If $det(\mathcal{S})$ terminates then $Traces(det(\mathcal{S})) = Traces(\mathcal{S})$.*

7.2 Lemmas for Termination: Sufficient Condition

In this section we assume an extended automaton $\mathcal{S}$ with bounded lookahead, i.e., $look(\mathcal{S}) < \infty$.

We study the lookaheads of locations in $det_2(\mathcal{S}, t_1, t_2)$ and compare them with those of $\mathcal{S}$. Section 7.2.1 deals with the locations $l \in L_{\mathcal{S}}$ of $\mathcal{S}$, and Section 7.2.2 deals with the new location $\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2} \rangle\rangle$ of $det_2(\mathcal{S}, t_1, t_2)$, where o is the common origin of t_1, t_2 and $\mathsf{d}_{t_1}, \mathsf{d}_{t_2}$ are their respective destinations. We use the direct function h and inverse function φ (Definitions 16, 17).

7.2.1 Lookahead in common locations of $\mathcal{S}$ and $det_2(\mathcal{S}, t_1, t_2)$

We distinguish two cases, depending on whether $l = \mathsf{o}$ (o is the origin of t_1, t_2) or $l \in L_{\mathcal{S}} \setminus \{\mathsf{o}\}$.

Lemma 5 *For each state $q = \langle l, \nu \rangle \in Q_{[\![\mathcal{S}]\!]}$ with $l \in L_{\mathcal{S}} \setminus \{\mathsf{o}\}$, each word $w \in \Sigma^+_{\mathcal{S}}$, and each pair of sequences $\sigma_1, \sigma_2 \in seq_q(\mathcal{S}, w)$: $first(\sigma_1) \neq first(\sigma_2) \Rightarrow first(h(\sigma_1)) \neq first(h(\sigma_2))$.*

Proof : Since $l \in L_{\mathcal{S}} \setminus \{\mathsf{o}\}$, $first(\sigma_1)$, $first(\sigma_2)$ are not in $\{t_1, t_2\}$. Thus, $first(h(\sigma_1)) = first(\sigma_1)$ and $first(h(\sigma_2)) = first(\sigma_2)$, which immediately proves the result. □

A similar lemma holds for the inverse function φ:

Lemma 6 *For each state $q = \langle l, \nu \rangle \in Q_{[\![\mathcal{S}]\!]}$ with $l \in L_{\mathcal{S}} \setminus \{\mathsf{o}\}$, each word $w \in \Sigma^+_{\mathcal{S}}$, and each pair of sequences $\sigma_1, \sigma_2 \in seq_q(det_2(\mathcal{S}, t_1, t_2), w)$: $first(\sigma_1) \neq first(\sigma_2) \Rightarrow first(\varphi(\sigma_1)) \neq first(\varphi(\sigma_2))$.*

Proof : Since $l \in L_{\mathcal{S}} \setminus \{\mathsf{o}\}$, $\mathit{first}(\sigma_1)$, $\mathit{first}(\sigma_2)$ are not in $\{t_{1,2}\} \cup \bigcup_{t'_i \,\in\, follow(t_i),\, i \,=\, 1,\, 2} \{\mathit{modify}_i(t'_i)\}$. Thus, $\mathit{first}(\varphi(\sigma_1)) = \mathit{first}(\sigma_1)$ and $\mathit{first}(\varphi(\sigma_2)) = \mathit{first}(\sigma_2)$, which immediately proves the result. □

Lemmas 5, 6 together with $h(\mathit{seq}_q(\mathcal{S}, w)) \subseteq \mathit{seq}_q(\mathit{det}_2(\mathcal{S}, t_1, t_2), w)$ and $\varphi(\mathit{seq}_q(\mathit{det}_2(\mathcal{S}, t_1, t_2), w)) \subseteq \mathit{seq}_q(\mathcal{S}, w)$ from Lemmas 3, 4 allow to prove

Lemma 7 $\forall l \in L_{\mathcal{S}} \setminus \{\mathsf{o}\}, \mathit{look}(l, \mathcal{S}) = \mathit{look}(l, \mathit{det}_2(\mathcal{S}, t_1, t_2))$.

Proof : ($\leq$) We first prove the following fact
(i) $\forall n \in \mathbb{N}$, $\mathit{det}_2(\mathcal{S}, t_1, t_2)$ *does not have* lookahead n in $l \Rightarrow \mathcal{S}$ *does not have* lookahead n in l.

Indeed, if $\mathit{det}_2(\mathcal{S}, t_1, t_2)$ does not have lookahead n in l then there exists a state $q = \langle l, \nu \rangle$, a word w of length $n+1$, and two sequences $\sigma_1, \sigma_2 \in \mathit{seq}_q(\mathit{det}_2(\mathcal{S}, t_1, t_2), w)$ such that $\mathit{first}(\sigma_1) \neq \mathit{first}(\sigma_2)$. By Lemma 6, their images by φ, which are in $\mathit{seq}_q(\mathcal{S}, w)$, also differ on the same transition, which means that $\mathcal{S}$ *does not have* lookahead n in l, and (i) is proved.

The statement (i) can be rephrased as $\forall n \in \mathbb{N}$, $\mathcal{S}$ has lookahead n in $l \Rightarrow \mathit{det}_2(\mathcal{S}, t_1, t_2)$ has lookahead n in l. In particular, for $n_0 = \mathit{look}(l, \mathcal{S}) \in \mathbb{N}$ (we have assumed bounded lookahead in this section) $\mathcal{S}$ does have lookahead n_0, which by (i) implies that $\mathit{det}_2(\mathcal{S}, t_1, t_2)$ has lookahead n_0 as well, hence, the *smallest* lookahead $\mathit{look}(l, \mathit{det}_2(\mathcal{S}, t_1, t_2))$ satisfies $\mathit{look}(l, \mathit{det}_2(\mathcal{S}, t_1, t_2)) \leq n_0 = \mathit{look}(l, \mathcal{S})$.

($\geq$) The proof is almost symmetrical to the one for the ($\leq$) inequality. We start by proving
(ii) $\forall n \in \mathbb{N}$, $\mathcal{S}$ *does not have* lookahead n in $l \Rightarrow \mathit{det}_2(\mathcal{S}, t_1, t_2)$ *does not have* lookahead n in l.

Indeed, if $\mathcal{S}$ does not have lookahead n in l then there exists a state $q = \langle l, \nu \rangle$, a word w of length $n+1$, and two sequences $\sigma_1, \sigma_2 \in \mathit{seq}_q(\mathcal{S}, w)$ such that $\mathit{first}(\sigma_1) \neq \mathit{first}(\sigma_2)$. By Lemma 5, their images by h, which are in $\mathit{seq}_q(\mathit{det}_2(\mathcal{S}, t_1, t_2), w)$, also differ on the same transition, which means that $\mathit{det}_2(\mathcal{S}, t_1, t_2)$ *does not have* lookahead n in l, and (ii) is proved.

The statement (ii) can be rephrased as $\forall n \in \mathbb{N}$, $\mathit{det}_2(\mathcal{S}, t_1, t_2)$ has lookahead n in $l \Rightarrow \mathcal{S}$ has lookahead n in l. In particular, let $n_1 = \mathit{look}(l, \mathit{det}_2(\mathcal{S}, t_1, t_2))$. Then, $n_1 \in \mathbb{N}$, as $n_1 \leq n_0 \in \mathbb{N}$, which we know from the ($\leq$) inequality. Hence, $\mathit{det}_2(\mathcal{S}, t_1, t_2)$ does have lookahead n_1, which by (ii) implies that $\mathcal{S}$ has lookahead n_1 as well, hence, the *smallest* lookahead $\mathit{look}(l, \mathcal{S})$ satisfies $\mathit{look}(l, \mathcal{S}) \leq n_1 = \mathit{look}(l, \mathit{det}_2(\mathcal{S}, t_1, t_2))$. □

We now turn to the case $l = \mathsf{o}$. The next lemma says that Lemma 6 also "holds" for $l = \mathsf{o}$:

Lemma 8 *For each* $q = \langle \mathsf{o}, \nu \rangle \in Q_{[\![\mathcal{S}]\!]}$, $w \in \Sigma^+_{\mathcal{S}}$, *and* $\sigma_1, \sigma_2 \in \mathit{seq}_q(\mathit{det}_2(\mathcal{S}, t_1, t_2), w)$, $\mathit{first}(\sigma_1) \neq \mathit{first}(\sigma_2) \Rightarrow \mathit{first}(\varphi(\sigma_1)) \neq \mathit{first}(\varphi(\sigma_2))$.

Proof : We first note that in the location o, σ_1, σ_2 may only start either with $t_{1,2}$, or with a transition $t \in \mathcal{T}_{\mathcal{S}} \setminus \{t_{1,2}, t_1, t_2\}$ - as t_1, t_2 are not transitions of $\mathit{det}_2(\mathcal{S}, t_1, t_2)$.

Let us now enumerate the possible values of $\mathit{first}(\varphi(\sigma_1))$ (and similarly for $\mathit{first}(\varphi(\sigma_2))$):

1. if $\sigma_1 = t \cdot \sigma'_1$ with $t \in \mathcal{T}_{\mathcal{S}} \setminus \{t_{1,2}, t_1, t_2\}$, then $\varphi(\sigma_1) = t \cdot \varphi(\sigma'_1)$
2. if σ_1 starts with $t_{1,2}$, then either $\sigma_1 \equiv t_{1,2}$, in which case $\varphi(\sigma_1) = \epsilon$ and by convention $\mathit{first}(\varphi(\sigma_1)) = \epsilon$, or σ_1 is of the form $\sigma_1 = t_{1,2} \cdot \mathit{modify}_i(t'_i) \cdot \sigma'_1$, for some $t'_i \in follow(t_i)$ and $i = 1, 2$. Then, $\varphi(\sigma_1) = t_i \cdot t'_i \cdot \varphi(\sigma'_1)$, and then $\mathit{first}(\varphi(\sigma_1)) = t_i \in \{t_1, t_2\}$.

Then, for two sequences σ_1, σ_2 as in the hypothesis of our lemma, we have the following possibilities:

- σ_1 starts with $t_{1,2}$ and σ_2 starts with $t \in \mathcal{T}_{\mathcal{S}} \setminus \{t_{1,2}, t_1, t_2\}$. Then, $\mathit{first}(\varphi(\sigma_1)) \in \{\epsilon, t_1, t_2\}$ and $\mathit{first}(\varphi(\sigma_2)) = t \in \mathcal{T}_{\mathcal{S}} \setminus \{t_{1,2}, t_1, t_2\}$: clearly, the two cannot be equal;
- a symmetrical situation obtained by switching the roles of σ_1, σ_2, with similar outcome that $\mathit{first}(\varphi(\sigma_1)) \neq \mathit{first}(\varphi(\sigma_2))$;
- σ_1, σ_2 start with two different transitions, respectively, t', t'' in $\mathcal{T}_{\mathcal{S}} \setminus \{t_{1,2}, t_1, t_2\}$, in which case $\mathit{first}(\varphi(\sigma_1)) = t' \neq t'' = \mathit{first}\varphi(\sigma_2))$. □

This and the inclusion $\varphi(seq_q(det_2(\mathcal{S}, t_1, t_2), w)) \subseteq seq_q(\mathcal{S}, w)$ from Lemma 4 allows us to prove, by complete analogy with the ($\leq$) part of the proof of Lemma 7

Lemma 9 *$look(\mathsf{o}, det_2(\mathcal{S}, t_1, t_2)) \leq look(\mathsf{o}, \mathcal{S})$.*

This concludes the study of the relations between smallest lookaheads of locations in $L_{\mathcal{S}}$ before an after the det_2 operation.

7.2.2 Lookahead in the new location created by local determinisation

We still have to consider the lookahead of the *new* location $\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2} \rangle \rangle$ created by the determinisation of two transitions t_1, t_2 with common origin o, common action a, and destinations $\mathsf{d}_{t_1}, \mathsf{d}_{t_2}$ (cf. Definition 6).

First, we prove that if a transition can be fired from a state of the form $\langle \langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2} \rangle \rangle, \nu \rangle$ (where $\nu \in \mathcal{V}$ is a valuation of the variables), then *two* transitions can be fired in sequence from o:

Lemma 10 *$\forall t'_i \in follow(t_i), i = 1, 2,$ $\langle \langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2} \rangle \rangle, \nu \rangle \xrightarrow{modify_i(t'_i)}_{det_2(\mathcal{S}, t_1, t_2)} q'$ implies $\langle \mathsf{o}, \nu \rangle \xrightarrow{t_i \cdot t'_i}_{\mathcal{S}} q'$.*

Proof : By Definition 6 of the $det_2(\mathcal{S}, t_1, t_2)$ the transitions $t_{1,2}$ and $modify_i(t'_i)$ are defined as follows:

- $t_{1,2} : \langle \mathsf{o}, \mathsf{G}_{t_1} \vee \mathsf{G}_{t_2}, a, Id_V, \langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2} \rangle \rangle \rangle$,
- $modify_i(t'_i) : \langle \langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2} \rangle \rangle,\ \mathsf{G}_{t_i} \wedge \mathsf{G}_{t'_i} \circ \mathsf{A}_{t_i}, a_{t'_i}, \mathsf{A}_{t'_i} \circ \mathsf{A}_{t_i}, \mathsf{d}_{t'_i} \rangle$.

Let now $\langle \langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2} \rangle \rangle, \nu \rangle \xrightarrow{modify_i(t'_i)}_{det_2(\mathcal{S}, t_1, t_2)} q'$. Then, $\nu \models \mathsf{G}_{modify_i(t'_1)} = \mathsf{G}_{t_i} \wedge \mathsf{G}_{t'_i} \circ \mathsf{A}_{t_i}$, and $q' = \langle \mathsf{d}_{t'_i}, (\mathsf{A}_{t'} \circ \mathsf{A}_{t_i})(\nu) \rangle$. Let then $\nu'' = \mathsf{A}_{t_i}(\nu)$ and $q'' = \langle \mathsf{d}_{t_i}, \nu'' \rangle$. From $\nu \models \mathsf{G}_{t'_i} \circ \mathsf{A}_{t_i}$, we obtain $\nu'' = \mathsf{A}_{t_i}(\nu) \models \mathsf{G}_{t'_i}$ and since $\mathsf{d}_{t_i} = \mathsf{o}_{t'_i}$ we have $q'' \xrightarrow{t'_i}_{\mathcal{S}} q'$. Also, we have $\nu \models \mathsf{G}_{t_i}$ and since $\nu'' = \mathsf{A}_{t_i}(\nu)$ we also get $\langle \mathsf{o}, \nu \rangle \xrightarrow{t_i}_{\mathcal{S}} q''$. Overall, $\langle \mathsf{o}, \nu \rangle \xrightarrow{t_i}_{\mathcal{S}} q'' \xrightarrow{t'_i}_{\mathcal{S}} q'$, and the proof is done. □

Lemma 11 *With the notations of Definitions 6, 17, for each trace $w \in \Sigma_{\mathcal{S}}^+$ and valuation $\nu \in \mathcal{V}$, the function φ defines a mapping from $seq_{\langle \langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2} \rangle \rangle, \nu \rangle}(det_2(\mathcal{S}, t_1, t_2), w)$ to $seq_{\langle \mathsf{o}, \nu \rangle}(\mathcal{S}, a \cdot w)$, where a is the common action of t_1, t_2.*

Proof : We prove that for all $\nu \in \mathcal{V}$, $\sigma \in seq_{\langle \langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2} \rangle \rangle, \nu \rangle}(det_2(\mathcal{S}, t_1, t_2), w)$, and $q' \in Q_{[\![det_2(\mathcal{S}, t_1, t_2)]\!]}$ the two statements hold:

(i) if $\langle \langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2} \rangle \rangle, \nu \rangle \xrightarrow{\sigma}_{det_2(\mathcal{S}, t_1, t_2)} q'$ and $\sigma \neq \epsilon$ and $last(\sigma) \neq t_{1,2}$ then $\langle o, \nu \rangle \overset{\varphi(\sigma)}{\rightarrow}_{\mathcal{S}} q'$

(ii) if $\langle \langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2} \rangle \rangle, \nu \rangle \xrightarrow{\sigma}_{det_2(\mathcal{S}, t_1, t_2)}$ and $\sigma \neq \epsilon$ and $last(\sigma) = t_{1,2}$ then $\langle o, \nu \rangle \overset{\varphi(\sigma)}{\rightarrow}_{\mathcal{S}}$.

To prove (i), we first note that since σ starts in the location $\langle \langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2} \rangle \rangle$, it can be decomposed as $\sigma = modify_i(t'_i) \cdot \sigma'$ for some $t'_i \in follow(t_i)$ and $i = 1, 2$. We now proceed by well-founded induction on σ. Let σ_2 the longest suffix of σ that does not contain $t_{1,2}$.

- If $\sigma_2 = \sigma$ then their subsequence σ' does not contain $t_{1,2}$, hence, it does not contain any of the transitions in $modify_i(follow(t_i))$, $i = 1, 2$ (otherwise it would contain their predecessor $t_{1,2}$ as well). Hence, σ' visits only a part of $det_2(\mathcal{S}, t_1, t_2)$ where determinisation had no effect, and $\varphi(\sigma) = t_i \cdot t'_i \cdot \sigma'$. Now, using these observations and Lemma 10, we obtain that if $\langle \langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2} \rangle \rangle, \nu \rangle \overset{modify_i(t'_i)}{\rightarrow}_{det_2(\mathcal{S}, t_1, t_2)} q_1 \xrightarrow{\sigma'}_{det_2(\mathcal{S}, t_1, t_2)} q'$ holds, $\langle \mathsf{o}, \nu \rangle \overset{t_i \cdot t'_i}{\rightarrow}_{\mathcal{S}} q_1 \xrightarrow{\sigma'}_{\mathcal{S}} q'$ holds, and (i) is proved in this case.

- otherwise, σ_2 is a strict prefix of σ, and we have the decomposition $\sigma = \mathit{modify}_i(t'_i) \cdot \sigma_1 \cdot \sigma_2$, with $\mathit{last}(\sigma_1) = t_{1,2}$, and σ_2 is nonempty (otherwise the whole sequence would terminate by $t_{1,2}$, prohibited in (i)). Hence, we also have $\mathit{first}(\sigma_2) = \mathit{modify}_j(t'_j)$ for some $t'_j \in \mathit{follow}(t_j)$ and $j = 1, 2$, because $t_{1,2}$ may only be followed by such transitions. Overall, $\sigma = \mathit{modify}_i(t'_i) \cdot \sigma'_1 \cdot t_{1,2} \cdot \mathit{modify}_j(t'_j) \cdot \sigma'_2$, hence, $\varphi(\sigma) = t_i \cdot t'_t \cdot \varphi(\sigma'_1) \cdot t_j \cdot t'_j \cdot \varphi(\sigma'_2)$.

  Assume now $\langle\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle, \nu\rangle \xrightarrow{\sigma}_{det_2(\mathcal{S},t_1,t_2)} q'$, i.e., $\langle\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle, \nu\rangle \xrightarrow{modify_i(t'_i)}_{det_2(\mathcal{S},t_1,t_2)} q_1 \xrightarrow{\sigma'_1}_{det_2(\mathcal{S},t_1,t_2)} q_2 \xrightarrow{t_{1,2} \cdot modify_j(t'_j)}_{det_2(\mathcal{S},t_1,t_2)} q_3 \xrightarrow{\sigma'_2}_{det_2(\mathcal{S},t_1,t_2)} q'$ for some states q_1, q_2, q_3.

  – by induction hypothesis, $\langle \mathsf{o}, \nu\rangle \xrightarrow{t_i \cdot t'_i}_{\mathcal{S}} q_1 \xrightarrow{\varphi(\sigma'_1)}_{\mathcal{S}} q_2$. Note that the conditions for applying the induction hypothesis hold: as $\sigma = \mathit{modify}_i(t'_i) \cdot \sigma'_1 \cdot t_{1,2} \cdot \mathit{modify}_j(t'_j) \cdot \sigma'_2$, we have $\mathit{last}(\sigma'_1) \neq t_{1,2}$ (otherwise σ would contain the subsequence $t_{1,2} \cdot t_{1,2}$, impossible since $t_{1,2}$ is not a self loop), and $\varphi(\mathit{modify}_i(t'_i) \cdot \sigma'_1) = t_i \cdot t'_i \cdot \varphi(\sigma'_1)$.

  – by Lemma 2 item (i), $q_2 \xrightarrow{t_j \cdot t'_j}_{\mathcal{S}} q_3$,

  – and by statement (α) established in the proof of Lemma 4, $q_3 \xrightarrow{\varphi(\sigma'_2)}_{\mathcal{S}} q'$. Note that the conditions for applying the statement hold: as $\sigma = \mathit{modify}_i(t'_i) \cdot \sigma'_1 \cdot t_{1,2} \cdot \mathit{modify}_j(t'_j) \cdot \sigma'_2$, we have $\mathit{last}(\sigma'_2) = \mathit{last}(\sigma) \neq t_{1,2}$ and $q_3, q' \in Q_{[\![\mathcal{S}]\!]}$ (as the locations of q_3, q' cannot be the new location $\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle$, because the target of $\mathit{modify}_j(t'_j)$ cannot be this new location - $t_{1,2}$ is the only transition with destination $\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle$, and $\mathit{last}(\sigma'_2) \neq t_{1,2}$).

  Overall, from $\langle\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle, \nu\rangle \xrightarrow{\sigma}_{det_2(\mathcal{S},t_1,t_2)} q'$, that is, from

  $\langle\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle, \nu\rangle \xrightarrow{modify_i(t'_i)}_{det_2(\mathcal{S},t_1,t_2)} q_1 \xrightarrow{\sigma'_1}_{det_2(\mathcal{S},t_1,t_2)} q_2 \xrightarrow{t_{1,2} \cdot modify_j(t'_j)}_{det_2(\mathcal{S},t_1,t_2)} q_3 \xrightarrow{\sigma'_2}_{det_2(\mathcal{S},t_1,t_2)} q'$,

  we have obtained

  $\langle \mathsf{o}, \nu\rangle \xrightarrow{t_i \cdot t'_i}_{\mathcal{S}} q_1 \xrightarrow{\varphi(\sigma'_1)}_{\mathcal{S}} q_2 \xrightarrow{t_j \cdot t'_j}_{\mathcal{S}} q_3 \xrightarrow{\varphi(\sigma'_2)}_{\mathcal{S}} q'$,

  that is,

  $\langle o, \nu\rangle \xrightarrow{\varphi(\sigma)}_{\mathcal{S}} q'$, and (i) is proved.

We now prove (ii). If $\mathit{last}(\sigma) = t_{1,2}$ then $\sigma = \sigma' \cdot t_{1,2}$ and $\mathit{last}(\sigma') \neq t_{1,2}$, because the new transition $t_{1,2}$ is not a self-loop: its origin is in $L_{\mathcal{S}}$ and its destination is the new location $\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle \notin L_{\mathcal{S}}$. Now, let q' be the state such that $\langle\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle, \nu\rangle \xrightarrow{\sigma'}_{det_2(\mathcal{S},t_1,t_2)} q' \xrightarrow{t_{1,2}}_{det_2(\mathcal{S},t_1,t_2)}$. As $\mathit{last}(\sigma') \neq t_{1,2}$, we can apply the result (i), proved above, and obtain $\langle \mathsf{o}, \nu\rangle \xrightarrow{\varphi(\sigma')}_{\mathcal{S}} q'$, which can be also written as $\langle \mathsf{o}, \nu\rangle \xrightarrow{\varphi(\sigma')}_{\mathcal{S}} q' \xrightarrow{\varphi(t_{1,2})}_{\mathcal{S}} q'$, because $\varphi(t_{1,2}) = \epsilon$. Hence, $\langle \mathsf{o}, \nu\rangle \xrightarrow{\varphi(\sigma)}_{\mathcal{S}}$, which concludes the proof of (ii).

Finally, we note again that, for all $\sigma \in \mathcal{T}^+_{det_2(\mathcal{S},t_1,t_2)}$ starting in the location $\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle$, $\mathit{first}(\sigma) = \mathit{modify}_i(t'_i)$ for some $t'_i \in \mathit{follow}(t_i)$, $i = 1, 2$, and therefore $\mathit{first}(\varphi(\sigma)) \in \{t_1, t_2\}$. Using (i), (ii), and the above observation, we conclude $\varphi(\mathit{seq}_{\langle\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle, \nu\rangle}(det_2(\mathcal{S}, t_1, t_2), w)) \subseteq \mathit{seq}_{\langle \mathsf{o}, \nu\rangle}(\mathcal{S}, a \cdot w)$, where a is the common action of t_1, t_2, and the proof is done. □

Lemma 12 *For all* $\nu \in \mathcal{V}, w \in \Sigma^m_{\mathcal{S}}$ *(*$m \geq 1$*), and* $\sigma_1, \sigma_2 \in \mathit{seq}_{\langle\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle, \nu\rangle}(det_2(\mathcal{S}, t_1, t_2), w)$

$$\mathit{first}(\sigma_1) \neq \mathit{first}(\sigma_2) \Rightarrow (\mathit{first}(\varphi(\sigma_1)) \neq \mathit{first}(\varphi(\sigma_2)) \vee \mathit{snd}(\varphi(\sigma_1)) \neq \mathit{snd}(\varphi(\sigma_2)))$$

Proof : As they start in $\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle$, the first transitions of σ_1, σ_2 are one of the *modified* transitions following the new transition $t_{1,2}$. We then have the following cases:

1. $\mathit{first}(\sigma_1) = \mathit{modify}_1(t'_1)$ and $\mathit{first}(\sigma_2) = \mathit{modify}_1(t''_1)$ for some $t'_1, t''_1 \in \mathit{follow}(t_1)$. By hypothesis, $\mathit{first}(\sigma_1) \neq \mathit{first}(\sigma_2)$, hence, $t'_1 \neq t''_1$. Then, $\varphi(\sigma_1) = t_1 \cdot t'_1 \cdot \varphi(\mathit{rest}(\sigma_1))$, and $\varphi(\sigma_2) = t_1 \cdot t''_1 \cdot \varphi(\mathit{rest}(\sigma_2))$, and we have $\mathit{snd}(\varphi(\sigma_1)) = t'_1 \neq t''_1 = \mathit{snd}(\varphi(\sigma_2))$.

2. $\mathit{first}(\sigma_1) = \mathit{modify}_2(t'_2)$ and $\mathit{first}(\sigma_2) = \mathit{modify}_2(t''_2)$ for some $t'_2, t''_2 \in \mathit{follow}(t_2)$: this case is similar to the previous one.

3. $\mathit{first}(\sigma_1) = \mathit{modify}_1(t'_1)$ and $\mathit{first}(\sigma_2) = \mathit{modify}_2(t'_2)$ for some $t'_1 \in \mathit{follow}(t_1)$, $t'_2 \in \mathit{follow}(t_2)$. Then, $\varphi(\sigma_1) = t_1 \cdot t'_1 \cdot \varphi(\mathit{rest}(\sigma_1))$, and $\varphi(\sigma_2) = t_2 \cdot t'_2 \cdot \varphi(\mathit{rest}(\sigma_2))$, and we have $\mathit{first}(\varphi(\sigma_1)) = t_1 \neq t_2 = \mathit{first}(\varphi(\sigma_2))$.

4. $\mathit{first}(\sigma_1) = \mathit{modify}_2^2(t'_2)$ and $\mathit{first}(\sigma_2) = \mathit{modify}_2^1(t'_2)$ for some $t'_2 \in \mathit{follow}(t_2)$, $t'_1 \in \mathit{follow}(t_1)$: this case is similar to the previous one, and the proof is done. □

Corollary 2 *$\forall n \in \mathbb{N}$ ($\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2} \rangle\rangle$ has lookahead n in $\mathit{det}_2(\mathcal{S}, t_1, t_2)$) $\Leftarrow$ ((o has lookahead $n+1$ in $\mathcal{S}$) $\wedge$ (d_{t_1} has lookahead n in $\mathcal{S}$) $\wedge$ (d_{t_2} has lookahead n in $\mathcal{S}$)).*

Proof : The property has the form $A \Leftarrow (B \wedge C \wedge D)$; we prove $\neg A \Rightarrow (\neg B \vee \neg C \vee \neg D)$ using Lemmas 11, 12. For this assume ($\neg A$): $\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2} \rangle\rangle$ does *not* have lookahead n in $\mathit{det}_2(\mathcal{S}, t_1, t_2)$). Then, by Definition 9, there exists a valuation $\nu \in \mathcal{V}$ such that the state $q \triangleq \langle\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle, \nu\rangle$ does not have lookahead n in $\mathit{det}_2(\mathcal{S}, t_1, t_2)$). By Definition 8, there exist $\sigma_1, \sigma_2 \in \mathcal{T}^{n+1}_{\mathit{det}_2(\mathcal{S}, t_1, t_2)}$ such that $\mathit{trace}(\sigma_1) = \mathit{trace}(\sigma_2) = w$ for some $w \in \Sigma^{n+1}_{\mathcal{S}}$, $q \xrightarrow{\sigma_1}_{\mathit{det}_2(\mathcal{S}, t_1, t_2)}$, $q \xrightarrow{\sigma_2}_{\mathit{det}_2(\mathcal{S}, t_1, t_2)}$, and yet $\mathit{first}(\sigma_1) \neq \mathit{first}(\sigma_2)$.

By Lemmas 11, 12, there exist $\sigma'_1 = \varphi(\sigma_1), \sigma'_2 = \varphi(\sigma_2) \in \mathit{seq}_{\langle \mathsf{o}, \nu\rangle}(\mathcal{S}, a \cdot w)$ with $\mathit{first}(\sigma'_1) \neq \mathit{first}(\sigma'_2)$ or $\mathit{snd}(\sigma'_1) \neq \mathit{snd}(\sigma'_2)$. $\sigma'_2, \sigma'_2 \in \mathit{seq}_{\langle \mathsf{o}, \nu\rangle}(\mathcal{S}, a \cdot w)$ also implies that $\mathit{trace}(\sigma'_1) = \mathit{trace}(\sigma'_2) = a \cdot w$, and the length of σ'_1, σ'_2 is $n+2$.

- if $\mathit{first}(\sigma'_1) \neq \mathit{first}(\sigma'_2)$, σ'_1, σ'_2 are witnesses for o *does not have lookahead* $n+1$ *in* $\mathcal{S}$ *(i.e., $\neg B$)*
- otherwise, if $\mathit{first}(\sigma'_1) = \mathit{first}(\sigma'_2)$ but $\mathit{snd}(\sigma'_1) \neq \mathit{snd}(\sigma'_2)$, then either $\mathit{first}(\sigma'_1) = \mathit{first}(\sigma'_2) = t_1$ or $\mathit{first}(\sigma'_1) = \mathit{first}(\sigma'_2) = t_2$. Then, $\mathit{snd}(\sigma'_1) \neq \mathit{snd}(\sigma'_2)$ means that $\mathit{rest}(\sigma'_1)$, $\mathit{rest}(\sigma'_1)$ are witnesses for the property d_{t_1} *or* d_{t_2} *do not have lookahead* n *in* $\mathcal{S}$, i.e., $(\neg C \vee \neg D)$. □

Corollary 3 $\mathit{look}(\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle, \mathit{det}_2(\mathcal{S}, t_1, t_2)) \leq \mathit{max}\{(\mathit{look}(\mathsf{o}, \mathcal{S}) - 1), \mathit{look}(\mathsf{d}_{t_1}, \mathcal{S}), \mathit{look}(\mathsf{d}_{t_2}, \mathcal{S}))\}$.

Proof : Let $n_1 = \mathit{max}\{(\mathit{look}(\mathsf{o}, \mathcal{S}) - 1), \mathit{look}(\mathsf{d}_{t_1}, \mathcal{S}), \mathit{look}(\mathsf{d}_{t_2}, \mathcal{S})\}$. Corollary 2 has the form $\forall n. B(n) \wedge C(n) \wedge D(n) \Rightarrow A(n)$. By Definition 9, $B(n_1)$,$C(n_1)$, and $D(n_1)$ all hold, hence, by Corollary 2, $A(n_1)$ holds, and using Definition 9 again, $\mathit{look}(\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle, \mathit{det}_2(\mathcal{S}, t_1, t_2)) \leq n_1$. □

Corollary 4 *$\mathit{look}(\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle, \mathit{det}_2(\mathcal{S}, t_1, t_2)) \leq \mathit{look}(\mathcal{S})$, and $\mathit{look}(\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle, \mathit{det}_2(\mathcal{S}, t_1, t_2)) < \mathit{look}(\mathcal{S})$ holds if $\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle$ does not have inherited nondeterminism.*

Proof : The first inequality is trivial using Definition 10 and Corollary 3. For the second one, $\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle$ does not have inherited nondeterminism means by Definition 11 that $\mathit{look}(\mathsf{d}_{t_1}, \mathcal{S}) = \mathit{look}(\mathsf{d}_{t_2}, \mathcal{S}) = 0$. Then, Corollary 4 becomes $\mathit{look}(\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle, \mathit{det}_2(\mathcal{S}, t_1, t_2)) \leq \mathit{look}(\mathsf{o}, \mathcal{S}) - 1$ which, by Definition 10 is $< \mathit{look}(\mathcal{S})$. □

Corollary 5 *$\mathit{look}(\mathit{det}_2(\mathcal{S}, t_1, t_2)) \leq \mathit{look}(\mathcal{S})$.*

Proof : After determinisation, the new location $\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2}\rangle\rangle$ has smallest lookahead $\leq \mathit{look}(\mathcal{S})$ by Corollary 4, and the smallest lookaheads of all other locations are also $\leq \mathit{look}(\mathcal{S})$ before determinisation and did not grow afterwards (Lemmas 7, 9). Hence, for the *maximum* $\mathit{look}(\mathit{det}_2(\mathcal{S}, t_1, t_2))$ of all those values, $\mathit{look}(\mathit{det}_2(\mathcal{S}, t_1, t_2)) \leq \mathit{look}(\mathcal{S})$ holds. □

The last two propositions in this section are stated in the paper, where they were directly used for proving termination of the global determinisation procedure. We prove them below:

Proposition 1 (*look*() does not grow with determinisation) $look(det(\mathcal{S}, l)) \leq look(\mathcal{S})$.

Proof : Direct consequence of Corollary 5.

Proposition 2 (*look*() decreases if all new nondeterminism is created) *Let $\mathcal{S}'$ be an automaton obtained by determinising all nondeterministic locations $\{l_1, \dots l_k\}$ of an automaton $\mathcal{S}$ in an arbitrary order, (i.e., $\mathcal{S}_0 = \mathcal{S}$, $\forall i \leq k-1, \mathcal{S}_{i+1} = det(\mathcal{S}_i, l_i)$, and $\mathcal{S}' = \mathcal{S}_k$). If none of these local determinisation steps gave rise to inherited nondeterminism, then $look(\mathcal{S}') < look(\mathcal{S})$.*

Proof : After determinisation, the set $L_{\mathcal{S}'}$ of locations is partitioned into three sets:

- the original nondeterministic locations $\{l_1, \dots l_k\}$, now deterministic, with $look = 0$
- the original deterministic locations $L_{\mathcal{S}} \setminus \{l_1, \dots l_k\}$, with $look = 0$
- the locations in $L_{\mathcal{S}'} \setminus L_{\mathcal{S}}$ generated in the determinisation process, which are either deterministic (with $look = 0$) or have *created* nondeterminism, and hence, using Corollary 4, have smallest lookahead strictly smaller that $look(\mathcal{S})$.

Overall, the maximum $look(\mathcal{S}')$ of these values satisfies $look(\mathcal{S}') < look(\mathcal{S})$. □

This concludes the proof of the fact that bounded lookahead is a *sufficient* condition for termination of our determinisation procedure.

7.3 Lemmas for the necessary condition

We now turn to proving the fact that bounded lookahead is also a *necessary* condition for termination of our determinisation procedure. The main result in this section is Proposition 3, which is used in the paper to prove the necessary condition. We first prove two technical lemmas.

Lemma 13 *Consider two distinct transitions $t_1, t_2 \in \mathcal{T}_\mathcal{S}$ with common origin o, a state $\langle \mathsf{o}, \nu \rangle \in Q_{[\![\mathcal{S}]\!]}$, a word $w \in \Sigma_\mathcal{S}^n$ $(n \geq 2)$, and two sequences $\sigma_1, \sigma_2 \in seq_{\langle \mathsf{o}, \nu \rangle}(\mathcal{S}, w)$ such that $first(\sigma_1) = t_1$, $first(\sigma_2) = t_2$. Let $\sigma_1' \triangleq rest(h(\sigma_1)), \sigma_2' \triangleq rest(h(\sigma_2))$, where h is the function from Definition 16. Then, $\sigma_1', \sigma_2' \in seq_{\langle\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2} \rangle\rangle, \nu\rangle}(det_2(\mathcal{S}, t_1, t_2), rest(w))$, and $first(\sigma_1') \neq first(\sigma_2')$.*

Proof : As σ_1, σ_2 have length ≥ 2, we have $\sigma_1 = t_1 \cdot t_1' \cdot \sigma_1''$ and $\sigma_1 = t_2 \cdot t_2' \cdot \sigma_2''$, for some transitions $t_1' \in follow(t_1)$, $t_2 \in follow(t_2)$, and sequences of transitions σ_1'', σ_2''. By definition of h, $h(\sigma_1) = t_{1,2} \cdot modify_1(t_1') \cdot h(\sigma_2'')$ and $h(\sigma_2) = t_{1,2} \cdot modify_2(t_2') \cdot h(\sigma_2'')$ Then, as $\sigma_1' \triangleq rest(h(\sigma_1)), \sigma_2' \triangleq rest(h(\sigma_2))$, we have $first(\sigma_1') = modify_1(t_1')$ and $first(\sigma_2') = modify_2(t_2')$, which are distinct identifiers whenever t_1, t_2 are distinct. This proves the second part of the lemma.

Also, from $\sigma_1 \in seq_{\langle \mathsf{o}, \nu \rangle}(\mathcal{S}, w)$ we obtain that $\langle \mathsf{o}, \nu \rangle \overset{t_1 \cdot t_1'}{\rightarrow}_\mathcal{S} q_1' \overset{\sigma_1''}{\rightarrow}_\mathcal{S} q_1''$ for some states q_1', q_1''. Using Lemma 2 and the definition of h, we have $\langle \mathsf{o}, \nu \rangle \overset{t_{1,2} \cdot modify_1(t_1')}{\rightarrow}_{det_2(\mathcal{S}, t_1, t_2)} q_1' \overset{h(\sigma_1'')}{\rightarrow}_{det_2(\mathcal{S}, t_1, t_2)} q_1''$, which can be rewritten as $\langle \mathsf{o}, \nu \rangle \overset{t_{1,2}}{\rightarrow}_{det_2(\mathcal{S}, t_1, t_2)} \tilde{q} \overset{\sigma_1'}{\rightarrow}_{det_2(\mathcal{S}, t_1, t_2)} q_1''$, where $\sigma_1' = modify_1(t_1') \cdot h(\sigma_1'')$ and, since $t_{1,2}$ performs identity assignments, the state $\tilde{q} = \langle\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2} \rangle\rangle, \nu\rangle$.

This implies $\sigma_1' \in seq_{\langle\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2} \rangle\rangle, \nu\rangle}(det_2(\mathcal{S}, t_1, t_2), rest(w))$. The same reasoning can be used to prove $\sigma_2' \in seq_{\langle\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2} \rangle\rangle, \nu\rangle}(det_2(\mathcal{S}, t_1, t_2), rest(w))$. □

Lemma 14 *Consider two distinct transitions $t_1, t_2 \in \mathcal{T}_\mathcal{S}$ with common origin o, a state $\langle \mathsf{o}, \nu \rangle \in Q_{[\![\mathcal{S}]\!]}$, a word $w \in \Sigma_\mathcal{S}^n$ $(n \geq 2)$, and two sequences $\sigma_1, \sigma_2 \in seq_{\langle \mathsf{o}, \nu \rangle}(\mathcal{S}, w)$ such that $first(\sigma_1) \neq first(\sigma_2)$, and at least one of $first(\sigma_1), first(\sigma_2)$ does not belong to $\{t_1, t_2\}$. Then, $first(h(\sigma_1)) \neq first(h(\sigma_2))$.*

Proof : Assume $\mathit{first}(\sigma_1) \notin \{t_1, t_2\}$. Then, $\mathit{first}(h(\sigma_1)) = \mathit{first}(\sigma_1) \notin \{t_1, t_2\}$. Then, either $\mathit{first}(\sigma_2) \notin \{t_1, t_2\}$ and then $\mathit{first}(h(\sigma_2)) = \mathit{first}(\sigma_2) \neq \mathit{first}(\sigma_1) = \mathit{first}(h(\sigma_1))$ and in this case the lemma is proved, or $\mathit{first}(\sigma_2) \in \{t_1, t_2\}$, in which case $\mathit{first}(h(\sigma_2)) = t_{1,2}$, which is also different from $\mathit{first}(h(\sigma_1)) = \mathit{first}(\sigma_1)$ as $t_{1,2} \notin \mathcal{T}_{\mathcal{S}}$ but $\mathit{first}(h(\sigma_1)) = \mathit{first}(\sigma_1) \in \mathcal{T}_{\mathcal{S}}$. □

Proposition 3 $\mathit{look}(\mathit{det}_2(\mathcal{S}, t_1, t_2)) \geq \mathit{look}(\mathcal{S}) - 1$.

Proof (Sketch). We base the proof on the following **Claim**: $\forall n \in \mathbb{N} \cup \{\infty\}$, if $\mathit{det}_2(\mathcal{S}, t_1, t_2)$ has lookahead n then $\mathcal{S}$ has lookahead $n+1$. For assume the **Claim** holds, and let $n_0 = \mathit{look}(\mathit{det}_2(\mathcal{S}, t_1, t_2))$, then, $\mathit{det}_2(\mathcal{S}, t_1, t_2)$ has lookahead n_0, hence, by the **Claim**, $\mathcal{S}$ has lookahead n_0+1, and then $\mathit{look}(\mathcal{S}) \leq n_0 + 1 = \mathit{look}(\mathit{det}_2(\mathcal{S}, t_1, t_2)) + 1$, and our Proposition is proved.

We now establish the **Claim**. It turns out that it is easier to prove the equivalent *statement*:

$\forall m \in \mathbb{N} \cup \{\infty\}$, if $\mathcal{S}$ does *not* have lookahead $m+1$ then $\mathit{det}_2(\mathcal{S}, t_1, t_2)$ does *not* have lookahead m.

We first consider the case $m = \infty$. The *statement* then says that if $\mathcal{S}$ has *finite* lookahead then so does $\mathit{det}_2(\mathcal{S}, t_1, t_2)$, and this is a consequence of Proposition 1 above.

The remaining case is $m \in \mathbb{N}$. Assume then the left-hand side of the implication in our *statement* holds. Then, $\exists q = \langle l, \nu \rangle \in Q_{[\![\mathcal{S}]\!]}. \exists w \in \Sigma_{\mathcal{S}}^{m+2}. \exists \sigma_1, \sigma_2 \in \mathit{seq}_q(\mathcal{S}, w). \mathit{first}(\sigma_1) \neq \mathit{first}(\sigma_2)$.

Using the direct function h (cf. Definition 16), we obtain the sequences $\sigma_1' = h(\sigma_1)$, $\sigma_2' = h(\sigma_2)$ which, by Lemma 3, are in $\mathit{seq}_q(\mathit{det}_2(\mathcal{S}, t_1, t_2), w)$. We distinguish several cases, depending on the location l in the state $\langle l, \nu \rangle$:

- if $l \neq \mathsf{o}$ (where o is the common origin of transitions t_1, t_2) then, by Lemma 5, $\mathit{first}(\sigma_1') \neq \mathit{first}(\sigma_2')$, which, together with $\sigma_1', \sigma_2' \in \mathit{seq}_q(\mathit{det}_2(\mathcal{S}, t_1, t_2), w)$ implies that $q = \langle l, \nu \rangle$ does not have lookahead $m+1$ in $\mathit{det}_2(\mathcal{S}, t_1, t_2)$, hence, q does not have lookahead m either, which implies that $\mathit{det}_2(\mathcal{S}, t_1, t_2)$ does not have lookahead m, and the *statement* is proved in this case.

- if $l = \mathsf{o}$, then we have again two subcases:

  - if $\{\mathit{first}(\sigma_1), \mathit{first}(\sigma_2)\} = \{t_1, t_2\}$, then consider the sequences $\sigma_1'' = \mathit{rest}(h(\sigma_1))$, $\sigma_2'' = \mathit{rest}(h(\sigma_2))$. By Lemma 13, $\sigma_1'', \sigma_2'' \in \mathit{seq}_{\langle\langle \mathsf{o}, \langle \mathsf{d}_{t_1}, \mathsf{d}_{t_2} \rangle\rangle, \nu\rangle}(\mathit{det}_2(\mathcal{S}, t_1, t_2), w')$ and $\mathit{first}(\sigma_1'') \neq \mathit{first}(\sigma_2'')$, where $w' = \mathit{rest}(w)$. Hence, we have two executable sequences of length $m+1$ starting from a state of $\mathit{det}_2(\mathcal{S}, t_1, t_2)$ with same trace w'. This implies that $\mathit{det}_2(\mathcal{S}, t_1, t_2)$ does *not* have lookahead m, which proves our *statement* in this case.
  - if $\{\mathit{first}(\sigma_1), \mathit{first}(\sigma_2)\} \neq \{t_1, t_2\}$, then, by Lemma 14, $\mathit{first}(h(\sigma_1)) \neq \mathit{first}(h(\sigma_2))$. Hence, we have two executable sequences $h(\sigma_1)$, $h(\sigma_2)$, which differ on their first transition, starting from $\langle \mathsf{o}, \nu \rangle$, of same trace w, and length $m+2$. This means that $\mathit{det}_2(\mathcal{S}, t_1, t_2)$ does not have lookahead $m+1$ and then it does not have lookahead m either - the *statement* is proved in this case as well and concludes the proof. □