Unfolding Synthesis of Asynchronous Automata*

Nicolas BAUDRU & Rémi MORIN

Laboratoire d'Informatique Fondamentale de Marseille
39 rue Frédéric Joliot-Curie, F-13453 Marseille cedex 13, France

Abstract. Zielonka's theorem shows that each regular set of Mazurkiewicz traces can be implemented as a system of synchronized processes provided with some distributed control structure called an asynchronous automaton. This paper gives a new algorithm for the synthesis of a non-deterministic asynchronous automaton from a regular Mazurkiewicz trace language. Our approach is based on an unfolding procedure that improves the complexity of Zielonka's and Pighizzini's techniques: Our construction is polynomial in terms of the number of states but still double-exponential in the size of the alphabet. As opposed to Métivier's work, our algorithm does not restrict to acyclic dependence alphabets.

Introduction

One of the major contributions in the theory of Mazurkiewicz traces [5] characterizes regular languages by means of asynchronous automata [17] which are devices with a distributed control structure. So far all known constructions of asynchronous automata from regular trace languages are quite involved and yield an exponential explosion of the number of states [7, 12]. Furthermore conversions of non-deterministic asynchronous automata into deterministic ones rely on Zielonka's time-stamping function [8, 13] and suffer from the same state-explosion problem. Interestingly heuristics to build small deterministic asynchronous automata were proposed in [15].

Zielonka's theorem and related techniques are fundamental tools in concurrency theory. For instance they are useful to compare the expressive power of classical models of concurrency such as Petri nets, asynchronous systems, and concurrent automata [10, 16]. These methods have been adapted already to the construction of communicating finite-state machines from collections of message sequence charts [1, 6, 11].

In this paper we give a new construction of a non-deterministic asynchronous automaton. Our algorithm starts from the specification of a regular trace language in the form of a possibly non-deterministic automaton. The latter is unfolded inductively on the alphabet into an automaton that enjoys several structural properties (Section 3). Next this unfolding automaton is used as the common skeleton of all local processes of an asynchronous automaton (Section 2). Due to the structural properties of the unfolding this asynchronous automaton accepts precisely the specified regular trace language.

We show that the number of local states built is polynomial in the number of states in the specification and double-exponential in the size of the alphabet (Subsection 3.4). Therefore *our approach subsumes the complexity of Zielonka's and Pighizzini's constructions* (Subsection 1.3).

* Supported by the ANR project SOAPDC

1 Background and main result

In this paper we fix a finite alphabet Σ provided with a total order $\sqsubseteq$. An automaton over a subset $T \subseteq \Sigma$ is a structure $\mathcal{A} = (Q, \imath, T, \longrightarrow, F)$ where Q is a *finite* set of states, $\imath \in Q$ is an initial state, $\longrightarrow \subseteq Q \times T \times Q$ is a set of transitions, and $F \subseteq Q$ is a subset of final states. We write $q \xrightarrow{a} q'$ to denote $(q, a, q') \in \longrightarrow$. An automaton $\mathcal{A}$ is called *deterministic* if we have $q \xrightarrow{a} q' \wedge q \xrightarrow{a} q'' \Rightarrow q' = q''$. For any word $u = a_1 ... a_n \in \Sigma^\star$, we write $q \xrightarrow{u} q'$ if there are some states $q_0, q_1, ..., q_n \in Q$ such that $q = q_0 \xrightarrow{a_1} q_1 ... q_{n-1} \xrightarrow{a_n} q_n = q'$. The language $L(\mathcal{A})$ accepted by some automaton $\mathcal{A}$ consists of all words $u \in \Sigma^\star$ such that $\imath \xrightarrow{u} q$ for some $q \in F$. A subset of words $L \subseteq \Sigma^\star$ is *regular* if it is accepted by some automaton.

1.1 Mazurkiewicz traces

We fix an *independence relation* $\parallel$ over Σ, that is, a binary relation $\parallel \subseteq \Sigma \times \Sigma$ which is irreflexive and symmetric. For any subset of actions $T \subseteq \Sigma$, the *dependence graph* of T is the undirected graph (V, E) whose set of vertices is $V = T$ and whose edges denote dependence, i.e. $\{a, b\} \in E \Leftrightarrow a \not\parallel b$.

The *trace equivalence* $\sim$ associated with the independence alphabet $(\Sigma, \parallel)$ is the least congruence over $\Sigma^\star$ such that $ab \sim ba$ for all pairs of independent actions $a \parallel b$. For a word $u \in \Sigma^\star$, the *trace* $[u] = \{v \in \Sigma^\star \mid v \sim u\}$ collects all words that are equivalent to u. We extend this notation from words to sets of words in a natural way: For all $L \subseteq \Sigma^\star$, we put $[L] = \{v \in \Sigma^\star \mid \exists u \in L, v \sim u\}$.

A *trace language* is a subset of words $L \subseteq \Sigma^\star$ that is closed for trace equivalence: $u \in L \wedge v \sim u \Rightarrow v \in L$. Equivalently we require that $L = [L]$. As usual a trace language L is called *regular* if it is accepted by some automaton.

1.2 Asynchronous systems vs. asynchronous automata

Two classical automata-based models are known to correspond to regular trace languages. Let us first recall the basic notion of an asynchronous system [3].

DEFINITION 1.1. *An automaton $\mathcal{A} = (Q, \imath, \Sigma, \longrightarrow, F)$ over the alphabet Σ is called an* asynchronous system *over $(\Sigma, \parallel)$ if we have*

ID: $q_1 \xrightarrow{a} q_2 \wedge q_2 \xrightarrow{b} q_3 \wedge a \parallel b$ *implies* $q_1 \xrightarrow{b} q_4 \wedge q_4 \xrightarrow{a} q_3$ *for some* $q_4 \in Q$.

The Independent Diamond property ID ensures that the language $L(\mathcal{A})$ of any asynchronous system is closed for the commutation of independent adjacent actions. Thus it is a regular trace language. Conversely it is easy to observe that *any regular trace language is the language of some deterministic asynchronous system.*

We recall now a more involved model of communicating processes known as asynchronous automata [17]. A finite family $\delta = (\Sigma_k)_{k \in K}$ of subsets of Σ is called a *distribution of* $(\Sigma, \parallel)$ if we have $a \not\parallel b \Leftrightarrow \exists k \in K, \{a, b\} \subseteq \Sigma_k$ for all actions $a, b \in \Sigma$. Note that each subset Σ_k is a clique of the dependence graph $(\Sigma, \not\parallel)$ and a distribution δ is simply a clique covering of $(\Sigma, \not\parallel)$. We fix an arbitrary distribution $\delta = (\Sigma_k)_{k \in K}$ in the rest of this paper. We call *processes* the elements of K. The *location* $\mathrm{Loc}(a)$ of an action $a \in \Sigma$ consists of all processes $k \in K$ such that $a \in \Sigma_k$: $\mathrm{Loc}(a) = \{k \in K \mid a \in \Sigma_k\}$.

DEFINITION 1.2. *An* asynchronous automaton *over the distribution* $(\Sigma_k)_{k\in K}$ *consists of a family of finite sets of states* $(Q_k)_{k\in K}$*, a family of initial local states* $(\imath_k)_{k\in K}$ *with* $\imath_k \in Q_k$*, a subset of final global states* $F \subseteq \prod_{k\in K} Q_k$*, and a transition relation* $\partial_a \subseteq \prod_{k\in \mathrm{Loc}(a)} Q_k \times \prod_{k\in \mathrm{Loc}(a)} Q_k$ *for each action* $a \in \Sigma$*.*

The set of *global states* $Q = \prod_{k\in K} Q_k$ can be provided with a set of global transitions $\longrightarrow$ in such a way that an asynchronous automaton is viewed as a particular automaton. Given an action $a \in \Sigma$ and two global states $q = (q_k)_{k\in K}$ and $r = (r_k)_{k\in K}$, we put $q \xrightarrow{a} r$ if $((q_k)_{k\in \mathrm{Loc}(a)}, (r_k)_{k\in \mathrm{Loc}(a)}) \in \partial_a$ and $q_k = r_k$ for all $k \in K \setminus \mathrm{Loc}(a)$. The initial global state $\imath$ consists of the collection of initial local states: $\imath = (\imath_k)_{k\in K}$. Then the *global automaton* $\mathcal{A} = (Q, \imath, \Sigma, \longrightarrow, F)$ satisfies Property ID of Def. 1.1. Thus it is an asynchronous system over $(\Sigma, \|)$ and $L(\mathcal{A})$ is a regular trace language. An asynchronous automaton is *deterministic* if its global automaton is deterministic, i.e. the local transition relations ∂_a are partial functions.

1.3 Main result and comparisons to related works

Although deterministic asynchronous automata appear as a restricted subclass of deterministic asynchronous systems, Zielonka's theorem asserts that any regular trace language can be implemented in the form of a deterministic asynchronous automaton.

THEOREM 1.3. *[17] For any regular trace language* L *there exists a deterministic asynchronous automaton whose global automaton* $\mathcal{A}$ *satisfies* $L = L(\mathcal{A})$*.*

In [12] a complexity analysis of Zielonka's construction is detailed. Let $|Q|$ be the number of states of the minimal deterministic automaton that accepts L and $|K|$ be the number of processes. Then the number of local states built by Zielonka's technique in each process $k \in K$ is $|Q_k| \leqslant 2^{O(2^{|K|}.|Q|\log|Q|)}$. The simplified construction by Cori et al. in [4] also suffers from this exponential state-explosion [5].

Another construction proposed by Pighizzini [14] builds a non-deterministic asynchronous automaton from particular rational expressions. This simpler approach proceeds inductively on the structure of the rational expression. Each step can easily be shown to be polynomial. In particular the number of local states in each process is (at least) *doubled* by each restricted iteration. Consequently in some cases the number of local states in each process is *exponential* in the length of the rational expression.

In the present paper we give a new construction that is *polynomial in* $|Q|$ (Th. 4.9): It produces $|Q_k| \leqslant (3.|\Sigma|.|Q|)^d$ local states for each process, where $d = 2^{|\Sigma|}$, $|\Sigma|$ is the size of Σ, and $|Q|$ is the number of states of some (possibly non-deterministic) asynchronous system that accepts L.

With the help of two simple examples we present our new approach in the next section. It consists basically in two steps: A *naive construction* applied on an *unfolded* automaton. Comparisons with known techniques is hard since this twofold approach has no similarity with previous methods. On the other hand we have applied recently our unfolding strategy in the framework of Message Sequence Charts [2]. We believe also that our approach can be strengthened in order to build *deterministic* asynchronous automata from deterministic asynchronous systems with a similar complexity cost.

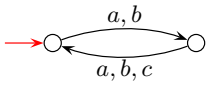

FIG. 1. Asynchronous system $\mathcal{A}$

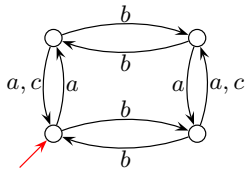

FIG. 2. Asynchronous system $\mathcal{A}'$

2 Twofold strategy

In this section we fix a (possibly non-deterministic) automaton $\mathcal{A} = (Q, \imath, \Sigma, \longrightarrow, F)$ over the alphabet Σ. We fix also a distribution $\delta = (\Sigma_k)_{k\in K}$ of $(\Sigma, \|)$. We introduce a basic construction of a *projected asynchronous automaton* $\widehat{\mathcal{A}}$ associated with $\mathcal{A}$. In general $L(\widehat{\mathcal{A}}) \neq L(\mathcal{A})$ even if we assume that $\mathcal{A}$ satisfies Axiom ID of Def. 1.1. Our strategy will appear as a method to unfold an *asynchronous system* $\mathcal{A}$ into a larger automaton $\mathcal{A}_{\mathrm{Unf}}$ that represents the same language: $[L(\mathcal{A}_{\mathrm{Unf}})] = L(\mathcal{A})$ and such that the projected asynchronous automaton of the unfolding $\mathcal{A}_{\mathrm{Unf}}$ yields a correct implementation: $L(\widehat{\mathcal{A}_{\mathrm{Unf}}}) = [L(\mathcal{A}_{\mathrm{Unf}})] = L(\mathcal{A})$. Note that $\mathcal{A}_{\mathrm{Unf}}$ will not fulfill ID in general.

The construction of the *projected asynchronous automaton* $\widehat{\mathcal{A}}$ over δ from the automaton $\mathcal{A}$ proceeds as follows. First the local states are copies of states of $\mathcal{A}$: We put $Q_k = Q$ for each process $k \in K$. The initial state $(\imath, ..., \imath)$ consists of $|K|$ copies of the initial state of $\mathcal{A}$. Moreover for each $a \in \Sigma$, the pair $((q_k)_{k\in \mathrm{Loc}(a)}, (r_k)_{k\in \mathrm{Loc}(a)})$ belongs to the transition relation ∂_a if there exist two states $q, r \in Q$ and a transition $q \xrightarrow{a} r$ in $\mathcal{A}$ such that the two following conditions are satisfied:

- for all $k \in \mathrm{Loc}(a)$, $q_k \xrightarrow{u} q$ in $\mathcal{A}$ for some word $u \in (\Sigma \setminus \Sigma_k)^\star$;
- for all $k \in \mathrm{Loc}(a)$, $r_k = r$; in particular all r_k are equal.

To conclude this definition, a global state $(q_k)_{k\in K}$ is *final* if there exists a final state $q \in F$ such that for all $k \in K$ there exists a path $q_k \xrightarrow{u} q$ in $\mathcal{A}$ for some word $u \in (\Sigma \setminus \Sigma_k)^\star$. The next result can be proved straightforwardly.

PROPOSITION 2.1. *We have* $L(\mathcal{A}) \subseteq [L(\mathcal{A})] \subseteq L(\widehat{\mathcal{A}})$.

EXAMPLE 2.2. We consider the independence alphabet $(\Sigma, \|)$ where $\Sigma = \{a, b, c\}$, $a\|b$ but $a \not\|c \not\|b$. Let $\mathcal{A}$ be the asynchronous system depicted in Fig. 1 and δ be the distribution with two processes $\Sigma_a = \{a, c\}$ and $\Sigma_b = \{b, c\}$. We assume here that all states of $\mathcal{A}$ are final. Then we get $L(\widehat{\mathcal{A}}) = \Sigma^\star$ whereas the word cc does not belong to $L(\mathcal{A})$. Consider now the asynchronous system $\mathcal{A}'$ depicted in Fig. 2. We can check that $L(\mathcal{A}') = L(\mathcal{A})$ and $L(\widehat{\mathcal{A}'}) = L(\mathcal{A})$.

This example shows that for some automata $\mathcal{A}$ the naive construction of the projected asynchronous automaton does not provide a correct implementation. However it is possible to unfold the automaton $\mathcal{A}$ to get a larger automaton $\mathcal{A}'$ for which the naive construction is correct. The aim of this paper is to show that this unfolding process is feasible with a polynomial cost for any asynchronous system $\mathcal{A}$.

3 Unfolding algorithm

In the rest of the paper we fix some asynchronous system $\mathcal{A} = (Q, \imath, \Sigma, \longrightarrow, F)$ that is possibly non-deterministic. The aim of this section is to associate $\mathcal{A}$ with a family of automata called *boxes* and *triangles* which are defined inductively. The last box built by this construction is called the *unfolding* of $\mathcal{A}$ (Def. 3.1).

Boxes and triangles are related to $\mathcal{A}$ by means of morphisms which are defined as follows. Let $\mathcal{A}_1 = (Q_1, \imath_1, T, \longrightarrow_1, F_1)$ and $\mathcal{A}_2 = (Q_2, \imath_2, T, \longrightarrow_2, F_2)$ be two automata over a subset of actions $T \subseteq \Sigma$. A *morphism* $\sigma : \mathcal{A}_1 \to \mathcal{A}_2$ *from* $\mathcal{A}_1$ *to* $\mathcal{A}_2$ is a mapping $\sigma : Q_1 \to Q_2$ from Q_1 to Q_2 such that $\sigma(\imath_1) = \imath_2$, $\sigma(F_1) \subseteq F_2$, and $q_1 \xrightarrow{a}_1 q_1'$ implies $\sigma(q_1) \xrightarrow{a}_2 \sigma(q_1')$. In particular, we have then $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$.

Now boxes and triangles are associated with an initial state that may not correspond to the initial state of $\mathcal{A}$. They are associated also with a subset of actions $T \subseteq \Sigma$. For these reasons, for any state $q \in Q$ and any subset of actions $T \subseteq \Sigma$, we let $\mathcal{A}_{T,q}$ denote the automaton $(Q, q, T, \longrightarrow_T, F)$ where $\longrightarrow_T$ is the restriction of $\longrightarrow$ to the transitions labeled by actions in T: $\longrightarrow_T = \longrightarrow \cap (Q \times T \times Q)$.

In this section we shall define the box $\square_{T,q}$ for all states $q \in Q$ and all subsets of actions $T \subseteq \Sigma$. The box $\square_{T,q}$ is a pair $(\mathcal{B}_{T,q}, \beta_{T,q})$ where $\mathcal{B}_{T,q}$ is an automaton over T and $\beta_{T,q} : \mathcal{B}_{T,q} \to \mathcal{A}_{T,q}$ is a morphism. Similarly, we shall define the triangle $\triangle_{T,q}$ for all states q and all *non-empty* subsets of actions T. The triangle $\triangle_{T,q}$ is a pair $(\mathcal{T}_{T,q}, \tau_{T,q})$ where $\mathcal{T}_{T,q}$ is an automaton over T and $\tau_{T,q} : \mathcal{T}_{T,q} \to \mathcal{A}_{T,q}$ is a morphism.

The *height* of a box $\square_{T,q}$ or a triangle $\triangle_{T,q}$ is the cardinality of T. Boxes and triangles are defined inductively on the height. We first define the box $\square_{\emptyset,q}$ for all states $q \in Q$. Next triangles of height h are built upon boxes of height $g < h$ and boxes of height h are built upon either triangles of height h or boxes of height $g < h$, whether the dependence graph $(T, \not\parallel)$ is connected or not.

The base case deals with the boxes of height 0. For all states $q \in Q$, the box $\square_{\emptyset,q}$ consists of the morphism $\beta_{\emptyset,q} : \{q\} \to Q$ that maps q to itself together with the automaton $\mathcal{B}_{\emptyset,q} = (\{q\}, q, \emptyset, \emptyset, F_{\emptyset,q})$ where $F_{\emptyset,q} = \{q\}$ if $q \in F$ and $F_{\emptyset,q} = \emptyset$ otherwise. In general a state of a box or a triangle is final if it is associated with a final state of $\mathcal{A}$.

DEFINITION 3.1. *The* unfolding $\mathcal{A}_{\mathrm{Unf}}$ *of* $\mathcal{A}$ *is the box* $\mathcal{B}_{\Sigma,\imath}$.

3.1 Building triangles from boxes

Triangles are made of boxes of lower height. Boxes are inserted into a triangle recursively on the height along a tree-like structure and several copies of the same box may appear within a triangle. We want to keep track of this structure in order to prove properties of triangles (and boxes) inductively: This enables us to allow for different copies of the same box within a triangle.

To do this, each state of a triangle is associated with a *rank* $k \in \mathbb{N}$ such that all states with the same rank come from the same copy of the same box. It is also important to keep track of the height each state comes from, because boxes of a triangle are inserted recursively on the height. For these reasons, a state of a triangle $\triangle_{T^\circ,q^\circ} = (\mathcal{T}_{T^\circ,q^\circ}, \tau_{T^\circ,q^\circ})$ is encoded as a quadruple $v = (w, T, q, k)$ such that w is a

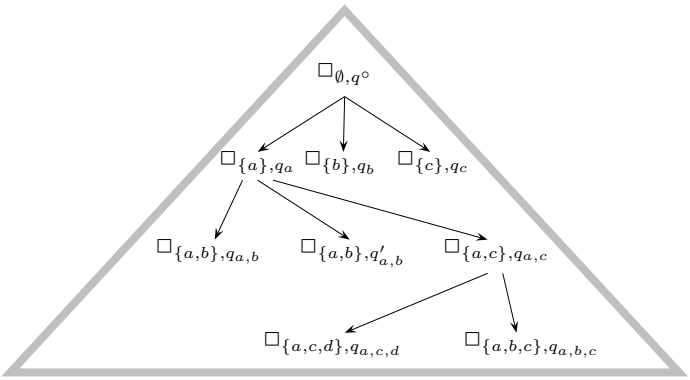

FIG. 3. Tree structure of triangles $\triangle_{T^\circ,q^\circ}$ with $T^\circ = \{a, b, c, d\}$

state from the box $\square_{T,q}$ with height $h = |T|$ and v is added to the triangle within the k-th box inserted into the triangle. Moreover this box is a copy of $\square_{T,q}$. In that case the state v maps to $\tau_{T^\circ,q^\circ}(v) = \beta_{T,q}(w)$, that is, the insertion of boxes preserves the correspondance to the states of $\mathcal{A}$. Moreover the morphism τ_{T°,q° of a triangle $\triangle_{T^\circ,q^\circ}$ is encoded in the data structure of its states.

We denote by $\mathcal{B}' = \textsc{Mark}(\mathcal{B}, T, q, k)$ the generic process that creates a copy $\mathcal{B}'$ of an automaton $\mathcal{B}$ by replacing each state w of $\mathcal{B}$ by $v = (w, T, q, k)$. The construction of the triangle $\triangle_{T^\circ,q^\circ}$ starts with using this marking procedure and building a copy $\textsc{Mark}(\square_{\emptyset,q^\circ}, \emptyset, q^\circ, 1)$ of the base box $\square_{\emptyset,q^\circ}$ which gets rank $k = 1$ and whose marked initial state $(\imath_{\square,\emptyset,q^\circ}, \emptyset, q^\circ, 1)$ becomes the initial state of $\triangle_{T^\circ,q^\circ}$. Along the construction of this triangle, an integer variable k counts the number of boxes already inserted in the triangle to make sure that all copies inserted get distinct ranks. The construction of the triangle $\triangle_{T^\circ,q^\circ}$ proceeds by successive insertions of copies of boxes according to the single following rule.

RULE 3.2. *A new copy of the box $\square_{T',q'}$ is inserted into the triangle $\triangle_{T^\circ,q^\circ}$ in construction if there exists a state $v = (w, T, q, l)$ in the triangle in construction and an action $a \in \Sigma$ such that*

- T_1: *$\beta_{T,q}(w) \xrightarrow{a} q'$ in the automaton $\mathcal{A}_{T^\circ,q^\circ}$;*
- T_2: *$T' = T \cup \{a\}$ and $T \subset T' \subset T^\circ$;*
- T_3: *no a-transition relates sofar v to the initial state of some copy of the box $\square_{T',q'}$ in the triangle in construction.*

In that case some a-transition is added in the triangle in construction from v to the initial state of the new copy of the box $\square_{T',q'}$.

Note here that Condition T_2 ensures that inserted boxes have height at most $|T^\circ| - 1$. By construction all copies of boxes inserted in a triangle are related in a tree-like structure built along the application of the above rules. It is easy to implement the

construction of a triangle from boxes as specified by the insertion rules above by means of a list of inserted boxes whose possible successors have not been investigated, in a depth-first-search or breadth-first-search way. Condition T_2 ensures also that if a new copy of the box $\square_{T',q'}$ is inserted and connected from $v = (w, T, q, l)$ then $T \subset T' \subset T^\circ$. This shows that this insertion process eventually stops and the resulting tree has depth at most $|T^\circ - 1|$. Moreover, since we start from the empty box and transitions in boxes $\square_{T,q}$ carry actions from T, we get the next obvious property.

LEMMA 3.3. *If a word $u \in \Sigma^\star$ leads in the triangle $\triangle_{T^\circ,q^\circ}$ from its initial state to some state $v = (w, T, q, l)$ then $u \in T^\star$ and all actions from T appear in u.*

Note also that it is easy to check that the mapping τ_{T°,q° induced by the data structure builds a morphism from $\mathcal{T}_{T^\circ,q^\circ}$ to $\mathcal{A}_{T^\circ,q^\circ}$. For latter purposes we define the list of missing transitions to state $q' \in Q$ in the triangle $\triangle_{T^\circ,q^\circ}$ as follows.

DEFINITION 3.4. *Let $T^\circ \subseteq \Sigma$ be a subset of actions and q°, q' be two states of $\mathcal{A}$. The set of missing transitions* MISSING(T°, q°, q') *consists of all pairs (v, a) where $v = (w, T, q, l)$ is a state of $\triangle_{T^\circ,q^\circ}$ and a is an action such that*

- *$\beta_{T,q}(w) \xrightarrow{a} q'$ in the automaton $\mathcal{A}_{T^\circ,q^\circ}$;*
- *$T \subset T \cup \{a\} = T^\circ$.*

Note here that the insertion rule T_2 for triangles forbids to insert a copy of the box $\mathcal{B}_{T^\circ,q'}$ and to connect its initial state with a transition labeled by a from state v. Note also that $|\text{MISSING}(T^\circ, q^\circ, q')|$ is less than the number of states in $\triangle_{T^\circ,q^\circ}$.

3.2 Building boxes from triangles

As announced in the introduction of this section the construction of the box $\square_{T^\circ,q^\circ}$ depends on the connectivity of the dependence graph of T°. Assume first that $T^\circ \subseteq \Sigma$ is not connected. Let T_1 denote the connected component of $(T^\circ, \not\parallel)$ that contains the least action $a \in T^\circ$ w.r.t. the total order $\sqsubseteq$ over Σ. We put $T_2 = T^\circ \setminus T_1$. The construction of the box $\square_{T^\circ,q^\circ}$ starts with building a copy of the box $\square_{T_2,q^\circ}$. Next for each state w of $\square_{T_2,q^\circ}$ and each transition $\beta_{T_2,q^\circ}(w) \xrightarrow{a} q$ in $\mathcal{A}_{T^\circ,q^\circ}$ with $a \in T_1$, the algorithm adds some a-transition from the copy of w to the initial state of a new copy of $\square_{T_1,q}$.

We come now to the definition of boxes associated with a *connected* set of actions. This part is more subtle than the two previous constructions which have a tree-structure and create no new loop. Let $T^\circ \subseteq \Sigma$ be a connected (non-empty) subset of actions. Basically the box $\square_{T^\circ,q^\circ}$ collects all triangles $\triangle_{T^\circ,q}$ for all states $q \in Q$. Each triangle is replicated a fixed number of times and copies of triangles are connected in some very specific way. We adopt a data structure similar to triangles (and unconnected boxes). A node w of a box $\square_{T^\circ,q^\circ}$ is a quadruple (v, T°, q, k) where v is a node of the triangle $\triangle_{T^\circ,q}$ and $k \in \mathbb{N}$. The rank k will allow us to distinguish between different copies of the same triangle within a box.

The construction of the box $\square_{T^\circ,q^\circ}$ consists in two steps. First m copies of each triangle $\triangle_{T^\circ,q}$ are inserted in the box and the first copy of $\triangle_{T^\circ,q^\circ}$ gets rank 1; moreover the first copy of its initial state is the initial state of the box. The value of m will be discussed below. In a second step some transitions are added to connect these triangles to each other according to the single following rule.

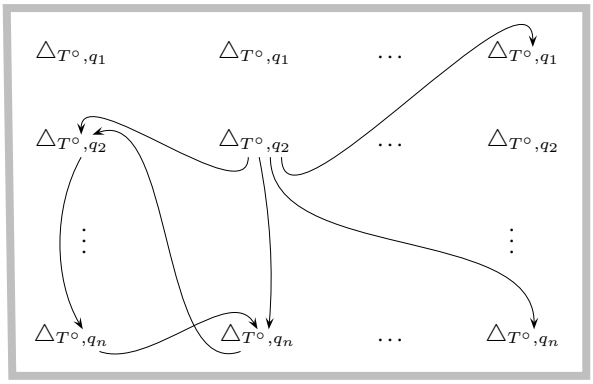

FIG. 4. Square structure of a box $\square_{T^\circ,q^\circ}$ with T° connected

RULE 3.5. *For each triangle $\triangle_{T^\circ,q}$, for each state $q' \in Q$, and for each missing transition $(v,a) \in \text{MISSING}(T^\circ,q,q')$ we add some a-transition from each copy of state v to the initial state of some copy of triangle $\triangle_{T^\circ,q'}$. In this process of connecting triangles we obey to the two following requirements:*

C_1*: No added transition connects two states from the same copy of a triangle.*
C_2*: At most one transition connects one copy of $\triangle_{T^\circ,q}$ to one copy of $\triangle_{T^\circ,q'}$.*

Condition C_1 requires that there is no added transition from state (v,T°,q,l) with rank l to the (initial) state $(\imath_{\triangle,T^\circ,q},T^\circ,q,l)$. To do so it is sufficient to have two copies of each triangle. Condition C_2 ensures that if we add from a copy of $\triangle_{T^\circ,q}$ of rank l some transition $(v_1,T^\circ,q,l) \xrightarrow{a_1} (\imath_{\triangle,T^\circ,q'},T^\circ,q',l')$ and some transition $(v_2,T^\circ,q,l) \xrightarrow{a_2} (\imath_{\triangle,T^\circ,q'},T^\circ,q',l')$ to the same copy of $\triangle_{T^\circ,q'}$ then $v_1 = v_2$ and $a_1 = a_2$. Recall that the number of added transitions from a fixed copy of $\triangle_{T^\circ,q}$ to copies of $\triangle_{T^\circ,q'}$ is $|\text{MISSING}(T^\circ,q,q')|$. Altogether it is sufficient to take

$$m = \max_{q,q' \in Q} |\text{MISSING}(T^\circ,q,q')| + 1 \tag{1}$$

From the definition of missing transitions (Def. 3.4) it follows that the data-structure defines a morphism from the box $\square_{T^\circ,q^\circ}$ to $\mathcal{A}_{T^\circ,q^\circ}$. Furthermore Definition 3.4 and Lemma 3.3 yield easily the following useful property.

LEMMA 3.6. *Within a box $\square_{T^\circ,q^\circ}$ associated with a* connected *set of actions T°, if a non-empty word $u \in \Sigma^\star$ leads from the initial state of a triangle to the initial state of a triangle then the alphabet of u is precisely T°.*

3.3 Some notations and a useful observation

First, for each path $s = q \xrightarrow{u} q'$ in some automaton $\mathcal{G}$ over Σ and for each action $a \in \Sigma$ we denote by $s|a$ the sequence of transitions labeled by a that appear along s.

Let T be a non-empty subset of Σ. Let v be a state from the triangle $\mathcal{T}_{T,q}$. By construction of $\mathcal{T}_{T,q}$, v is a quadruple (w, T', q', k') such that w is a state from the box $\square_{T',q'}$ and $k' \in \mathbb{N}$. Then we say that the *box location* of v is $l^{\square}(v) = (T', q', k')$. We define the *sequence of boxes* $\mathbb{L}^{\square}(s)$ visited along a path $s = v \xrightarrow{u} v'$ in $\mathcal{T}_{T,q}$ as follows:

- If the length of s is 0 then s corresponds to a state v of $\mathcal{T}_{T,q}$ and $\mathbb{L}^{\square}(s) = l^{\square}(v)$.
- If s is a product $s = s' \cdot t$ where t is the transition $v'' \xrightarrow{a} v'$ then $\mathbb{L}^{\square}(s) = \mathbb{L}^{\square}(s')$ if $l^{\square}(v'') = l^{\square}(v')$ and $\mathbb{L}^{\square}(s) = \mathbb{L}^{\square}(s').l^{\square}(v')$ otherwise.

Similarly we define the sequence of boxes $\mathbb{L}^{\square}(s)$ visited along a path s in a box $\mathcal{B}_{T,q}$ where T is an *unconnected* set of actions and the sequence of triangles $\mathbb{L}^{\triangle}(s)$ visited along a path s in a box $\mathcal{B}_{T,q}$ where T is a non-empty *connected* set of actions.

By means of Lemma 3.6 the next fact is easy to show.

LEMMA 3.7. *Let T be a non-empty* connected *set of actions. Let $a \in T$ be some action. Let $s_1 = v \xrightarrow{u_1} v'$ and $s_2 = v \xrightarrow{u_2} v'$ be two paths from v to v' in a box $\mathcal{B}_{T,q}$. If $s_1|a = s_2|a$ then $\mathbb{L}^{\triangle}(s_1) = \mathbb{L}^{\triangle}(s_2)$.*

3.4 Complexity of this unfolding construction

For all naturals $n \geqslant 0$ we denote by $\mathfrak{B}_n$ the maximal number of states in a box $\mathcal{B}_{T,q}$ with $|T| = n$ and $q \in Q$. Similarly for all naturals $n \geqslant 1$ we denote by $\mathfrak{T}_n$ the maximal number of states in a triangle $\mathcal{T}_{T,q}$ with $|T| = n$ and $q \in Q$. Noteworthy $\mathfrak{B}_0 = 1$ and $\mathfrak{T}_1 = 1$. Moreover $\mathfrak{T}_n$ is non-decreasing because the triangle $\triangle_{T',q}$ is a subautomaton of the triangle $\triangle_{T,q}$ as soon as $T' \subseteq T$. In the following we assume $2 \leqslant n \leqslant |\Sigma|$. Consider some subset $T \subseteq \Sigma$ with $|T| = n$. Each triangle $\mathcal{T}_{T,q}$ is built inductively upon boxes of height $h \leqslant n-1$. We distinguish two kinds of boxes. First boxes of height $h < n-1$ are inserted. Each of these boxes appears also in some triangle $\mathcal{T}_{T',q}$ with $T' \subset T$ and $|T'| = n-1$. Each of these triangles is a subautomaton of $\mathcal{T}_{T,q}$ with at most $\mathfrak{T}_{n-1}$ states. Moreover there are only n such triangles which give rise to at most $n.\mathfrak{T}_{n-1}$ states built along this first step. Second, boxes of height $n-1$ are inserted and connected to states inserted at height $n-2$. Each of these states belongs to some box $\square_{T',q'}$ with $|T'| = n-2$; it gives rise to at most $2.|Q|$ boxes at height $n-1$ because $|T \setminus T'| = 2$: This produces at most $2.|Q|.\mathfrak{B}_{n-1}$ new states. Altogether we get

$$\mathfrak{T}_n \leqslant n.\mathfrak{T}_{n-1}.(1 + 2.|Q|.\mathfrak{B}_{n-1}) \leqslant 3.|\Sigma|.|Q|.\mathfrak{T}_{n-1}.\mathfrak{B}_{n-1} \qquad (2)$$

Assume now $1 \leqslant n \leqslant |\Sigma|$ and consider a connected subset $T \subseteq \Sigma$ with $|T| = n$. Then each box $\mathcal{B}_{T,q}$ is built upon all triangles $\mathcal{T}_{T,q'}$ of height n. It follows from (1) that $m \leqslant \mathfrak{T}_n + 1 \leqslant 2.\mathfrak{T}_n$. Therefore the box $\mathcal{B}_{T,q}$ contains at most $2.\mathfrak{T}_n$ copies of each triangle $\mathcal{T}_{T,q'}$. It follows that we have $(*)$ $|\mathcal{B}_{T,q}| \leqslant 2.|Q|.\mathfrak{T}_n^2$. Consider now a *non-connected* subset $T \subseteq \Sigma$ with $|T| = n$. Then $\mathcal{B}_{T,q}$ consists of at most $1 + (n-1).Q.\mathfrak{B}_{n-1}$ boxes of height at most $n-1$. Therefore we have also $(**)$ $|\mathcal{B}_{T,q}| \leqslant |\Sigma|.|Q|.\mathfrak{B}_{n-1}^2$. From (2), $(*)$, and $(**)$ we get the next result by an immediate induction.

LEMMA 3.8. *If $1 \leqslant n \leqslant |\Sigma|$ then $\mathfrak{T}_n \leqslant (3.|\Sigma|.Q)^{2^n - 1}$ and $\mathfrak{B}_n \leqslant (3.|\Sigma|.Q)^{2^n - 1}$.*

As a consequence the unfolding automaton $\mathcal{A}_{\mathrm{Unf}}$ has at most $(3.|\Sigma|.Q)^{2^{|\Sigma|}-1}$ states.

4 Properties of the unfolding construction

In this section we fix a regular trace language L over the independence alphabet $(\Sigma, \|)$. We assume that the possibly non-deterministic automaton $\mathcal{A}$ fulfills Property ID of Def. 1.1 and satisfies $L(\mathcal{A}) = L$.

4.1 Arched executions for boxes and triangles

Let $\mathcal{G}$ be some automaton over Σ and $\widehat{\mathcal{G}}$ be its projected asynchronous automaton. For each global state q of $\widehat{\mathcal{G}}$ we denote by $q{\downarrow}k$ the local state of process k in q. Let q_1 and q_2 be two global states of $\widehat{\mathcal{G}}$. A *true step* $q_1 \xrightarrow{a} q_2$ of action a from q_1 to q_2 in $\widehat{\mathcal{G}}$ consists of a transition $q \xrightarrow{a} r$ in $\mathcal{G}$ such that $q_1{\downarrow}k = q_2{\downarrow}k$ for all $k \notin \mathrm{Loc}(a)$, $q_1{\downarrow}k = q$ and $q_2{\downarrow}k = r$ for all $k \in \mathrm{Loc}(a)$. If $q_1{\downarrow}j = q_2{\downarrow}j$ for all processes $j \neq k$, $q_1{\downarrow}k \xrightarrow{a} q_2{\downarrow}k$, and $k \notin \mathrm{Loc}(a)$ then $q_1 \xrightarrow{\varepsilon} q_2$ is called a *ε-step* of process k from q_1 to q_2.

DEFINITION 4.1. *An* execution *of $u \in \Sigma^\star$ from q to q' in $\widehat{\mathcal{G}}$ is a sequence of n true or ε-steps $q_{i-1} \xrightarrow{x_i} q_i$ such that $q_0 = q$, $q_n = q'$, and $u = x_1 ... x_n$ with $x_i \in \Sigma \cup \{\varepsilon\}$.*

For each execution s of $u \in \Sigma^\star$ and each process $k \in K$ we denote by $s{\downarrow}k$ the path of $\mathcal{G}$ followed by process k along s. For each state q of $\mathcal{G}$ we denote by $\widehat{q}$ the global state of $\widehat{\mathcal{G}}$ such that each process is at state q. A global state is *coherent* if it is equal to some $\widehat{q}$. Notice that the initial state and all final states of $\widehat{\mathcal{G}}$ are coherent. An execution from q_1 to q_2 in $\widehat{\mathcal{G}}$ is called *arched* if both q_1 and q_2 are coherent. The next observation shows how arched executions are related to the language of $\widehat{\mathcal{G}}$.

PROPOSITION 4.2. *For all words $u \in \Sigma^\star$ we have $u \in L(\widehat{\mathcal{G}})$ if and only if there exists an arched execution of u from the initial state of $\widehat{\mathcal{G}}$ to some of its final states.*

The following result expresses a main property of boxes: It asserts that active processes visit the same sequence of triangles along an arched execution within a box.

PROPOSITION 4.3. *Let $\mathcal{B}_{T,q}$ be a box with T a* connected *set of actions and s be an* arched *execution in $\widehat{\mathcal{B}_{T,q}}$. Then $\mathbb{L}^{\triangle}(s{\downarrow}k) = \mathbb{L}^{\triangle}(s{\downarrow}k')$ for all $k, k' \in \mathrm{Loc}(T)$.*

Proof. Since T is connected it is sufficient to show that for all actions $a \in T$ and for all processes $k, k' \in \mathrm{Loc}(a)$ we have $\mathbb{L}^{\triangle}(s{\downarrow}k) = \mathbb{L}^{\triangle}(s{\downarrow}k')$. So we fix an action $a \in T$ and two processes $k, k' \in \mathrm{Loc}(a)$. Since k and k' synchronize on the same transition at each occurrence of a, we have $(s{\downarrow}k)|a = (s{\downarrow}k')|a$. It follows by Lemma 3.7 that $\mathbb{L}^{\triangle}(s{\downarrow}k) = \mathbb{L}^{\triangle}(s{\downarrow}k')$. ∎

Since processes in $K \setminus \mathrm{Loc}(T)$ are involved in ε-steps only, we can change their behavior within an execution of u without affecting the resulting word u and we get:

PROPOSITION 4.4. *Let $\mathcal{B}_{T,q}$ be a box with T some connected set of actions and s be an* arched *execution of u in $\widehat{\mathcal{B}_{T,q}}$ from $\widehat{w}$ to $\widehat{w}'$. Then there exists some arched execution s° of u in $\widehat{\mathcal{B}_{T,q}}$ from $\widehat{w}$ to $\widehat{w}'$ such that $\mathbb{L}^{\triangle}(s^\circ{\downarrow}k) = \mathbb{L}^{\triangle}(s^\circ{\downarrow}j)$ for all $j, k \in K$.*

Noteworthy it is easy to adapt these remarks to unconnected boxes and triangles due to their tree-like structure.

4.2 A technical lemma and a key property

We come now to the main technical lemma of this paper. It completes Proposition 4.4 and asserts that we can split any arched execution s associated with a box $\mathcal{B}_{T,q}$ with a connected set of actions T into an equivalent series of arched executions $s_0 \cdot t_1 \cdot s_1 \cdot t_2 \cdot ... \cdot t_n \cdot s_n$ where each s_i is an arched execution within a component triangle and each t_i corresponds to the unique added transition from a triangle to another triangle. This decomposition will allow us to reason about arched executions inductively on the construction of the unfolding.

LEMMA 4.5. *Let $\mathcal{B}_{T,q}$ be a box with T some connected set of actions and $l_1, ..., l_n$ be a sequence of triangle locations within $\mathcal{B}_{T,q}$ with $n \geqslant 2$. Let $s : \widehat{w} \xrightarrow{u} \widehat{w}'$ be an* arched *execution of u in $\widehat{\mathcal{B}_{T,q}}$ from $\widehat{w}$ to $\widehat{w}'$ such that $\mathbb{L}^{\triangle}(s^{\circ}\downarrow k) = l_1...l_n$ for all $k \in K$. Then there exist a transition $w_2 \xrightarrow{a} w_3$ in $\mathcal{B}_{T,q}$ and three arched executions $s_1 : \widehat{w} \xrightarrow{u_1} \widehat{w}_2$, $s_2 : \widehat{w}_2 \xrightarrow{u_2} \widehat{w}_3$, and $s_3 : \widehat{w}_3 \xrightarrow{u_3} \widehat{w}'$ such that*

- *$\mathbb{L}^{\triangle}(s_1\downarrow k) = l_1$ for all $k \in K$;*
- *$s_2\downarrow k = w_2 \xrightarrow{a} w_3$ for all $k \in K$;*
- *$\mathbb{L}^{\triangle}(s_3\downarrow k) = l_2...l_n$ for all $k \in K$;*
- *$(s_1 \cdot s_2 \cdot s_3\downarrow k) = (s\downarrow k)$ for all $k \in K$;*
- *$s_1 \cdot s_2 \cdot s_3$ is an execution of $u_1.u_2.u_3$ in $\widehat{\mathcal{B}_{T,q}}$ and $u_1.u_2.u_3 \sim u$.*

Intuitively we require that all processes leave together the first triangle along the unique transition $w_2 \xrightarrow{a} w_3$ that leads from l_1 to l_2. This result relies on Lemma 3.6 and the two properties C_1 and C_2 of Rule 3.5.

Observe now that the tree-structure of triangles and boxes associated to unconnected sets of actions ensures that we can state a similar result for all triangles and all boxes.

LEMMA 4.6. *Let $T \subseteq \Sigma$ be a non-empty subset of actions. If s is an arched execution of u from $\widehat{w}_1$ to $\widehat{w}_2$ in $\widehat{\mathcal{B}_{T,q}}$ (resp. $\widehat{\mathcal{T}_{T,q}}$) then there exists some path $w_1 \xrightarrow{u'} w_2$ in $\mathcal{B}_{T,q}$ (resp. $\mathcal{T}_{T,q}$) such that $u' \sim u$.*

Proof. Observe first that this property holds also trivially for the empty boxes $\mathcal{B}_{\emptyset,q}$. We proceed now by induction on the size of T along the construction of triangles and boxes. Let $n = |T|$. Assume that the property holds for all triangles $\mathcal{T}_{T^\dagger,q}$ with $|T^\dagger| \leqslant n$. Assume also that T is connected. By Lemma 4.5 we can split the execution s into an equivalent series of arched executions $s_0 \cdot t_1 \cdot s_1 \cdot t_2 \cdot ... \cdot t_n \cdot s_n$ where each s_i is an arched execution within a component triangle and each t_i corresponds to the unique added transition from a triangle to another triangle. By induction hypothesis each s_i corresponds to a path in the corresponding triangle. In that way we get a path for the sequence s. The case where T is not connected is similar due to the tree-structure of these boxes. The case of triangles is also similar. ∎

Recall now that arched executions are closely related to the langage of the projected asynchronous automaton (Prop. 4.2). As an immediate corollary we get the following key result.

PROPOSITION 4.7. *We have $L(\widehat{\mathcal{A}_{\mathrm{Unf}}}) \subseteq [L(\mathcal{A}_{\mathrm{Unf}})]$.*

4.3 Main result

Due to the morphisms from boxes and triangles to asynchronous systems $\mathcal{A}_{T,q}$ we have the inclusion relation $[L(\mathcal{B}_{T,q})] \subseteq L(\mathcal{A}_{T,q})$ for each box $\mathcal{B}_{T,q}$ and similarly $[L(\mathcal{T}_{T,q})] \subseteq L(\mathcal{A}_{T,q})$ for each triangle $\mathcal{T}_{T,q}$. We can check by an easy induction that boxes satisfy the converse inclusion relation, which leads us to the next statement.

PROPOSITION 4.8. *We have* $[L(\mathcal{A}_{\mathrm{Unf}})] = L(\mathcal{A})$.

We come to the main statement of this paper.

THEOREM 4.9. *The asynchronous automaton* $\widehat{\mathcal{A}_{\mathrm{Unf}}}$ *satisfies* $L(\widehat{\mathcal{A}_{\mathrm{Unf}}}) = L(\mathcal{A})$. *Moreover the number of states in each process is* $|Q_k| \leqslant (3.|\Sigma|.|Q|)^d$ *where* $d = 2^{|\Sigma|}$.

Proof. By Proposition 4.8 we have $[L(\mathcal{A}_{\mathrm{Unf}})] = L(\mathcal{A})$. By Proposition 2.1 we have also $[L(\mathcal{A}_{\mathrm{Unf}})] \subseteq L(\widehat{\mathcal{A}_{\mathrm{Unf}}})$ hence $L(\mathcal{A}) \subseteq L(\widehat{\mathcal{A}_{\mathrm{Unf}}})$. Now Proposition 4.7 shows that $L(\widehat{\mathcal{A}_{\mathrm{Unf}}}) \subseteq [L(\mathcal{A}_{\mathrm{Unf}})] = L(\mathcal{A})$. The complexity result follows from Lemma 3.8. ∎

References

1. Baudru N. and Morin R.: *Safe Implementability of Regular Message Sequence Charts Specifications*. Proc. of the ACIS 4th Int. Conf. SNDP (2003) 210–217
2. Baudru N. and Morin R.: *The Synthesis Problem of Netcharts*. (2006) – Submitted
3. Bednarczyk M.A.: *Categories of Asynchronous Systems*. PhD thesis in Computer Science (University of Sussex, 1988)
4. Cori R., Métivier Y. and Zielonka W.: *Asynchronous mappings and asynchronous cellular automata*. Inform. and Comput. **106** (1993) 159–202
5. Diekert V. and Rozenberg G.: *The Book of Traces*. (World Scientific, 1995)
6. Genest B., Muscholl A. and Kuske D.: *A Kleene Theorem for a Class of Communicating Automata with Effective Algorithms*. DLT, LNCS **3340** (2004) 30–48
7. Genest B. and Muscholl A.: *Constructing Exponential-size Deterministic Zielonka Automata*. Technical report (2006) – 12 pages
8. Klarlund N., Mukund M. and Sohoni M.: *Determinizing Asynchronous Automata*. ICALP, LNCS **820** (1994) 130–141
9. Métivier Y.: *An algorithm for computing asynchronous automata in the case of acyclic non-commutation graph*. ICALP, LNCS **267** (1987) 226–236
10. Morin R.: *Concurrent Automata vs. Asynchronous Systems*. LNCS **3618** (2005) 686–698
11. Mukund M., Narayan Kumar K. and Sohoni M.: *Synthesizing distributed finite-state systems from MSCs*. CONCUR, LNCS **1877** (2000) 521–535
12. Mukund M. and Sohoni M.: *Gossiping, Asynchronous Automata and Zielonka's Theorem*. Report TCS-94-2, SPIC Science Foundation (Madras, India, 1994)
13. Muscholl A.: *On the complementation of Büchi asynchronous cellular automata*. ICALP, LNCS **820** (1994) 142–153
14. Pighizzini G.: *Synthesis of Nondeterministic Asynchronous Automata*. Algebra, Logic and Applications, vol. **5** (1993) 109–126
15. Ştefănescu A., Esparza J. and Muscholl A.: *Synthesis of distributed algorithms using asynchronous automata*. CONCUR, LNCS **2761** (2003) 20–34
16. Thiagarajan P.S.: *Regular Event Structures and Finite Petri Nets: A Conjecture*. Formal and Natural Computing, LNCS **2300** (2002) 244–256
17. Zielonka W.: *Notes on finite asynchronous automata*. RAIRO, Theoretical Informatics and Applications **21** (Gauthiers-Villars, 1987) 99–135