A Tight Lower Bound for Determinization of Transition Labeled Büchi Automata

Thomas Colcombet, Konrad Zdanowski
CNRS

JAF28, Fontainebleau
June 18, 2009

Finite Automata

A *finite automaton* is a tuple $\mathcal{A} = (Q, \Sigma, I, \Gamma, \Delta)$, where:

- Q is a set of *states*,
- Σ is an *input alphabet*,
- I is a set of *initial states*,
- Γ is an *output alphabet*
- $\Delta \subseteq Q \times \Sigma \times \Gamma \times Q$ is the *transition relation*.

$\mathcal{A}$ is a non-deterministic transducer from Σ^ω to Γ^ω.

For $u \in \Sigma^\omega$, $\rho = (p_0, b_0, p_1)(p_1, b_1, p_2)(p_2, b_2, p_3)\ldots$
is a *run of $\mathcal{A}$ over u* if $p_0 \in I$ and $\forall i\,(p_i, u(i), b_i, p_{i+1}) \in \Delta$.

The *output of ρ*, $\mathrm{Out}(\rho)$ is the word $b_0 b_1 b_2 \ldots$.

Deterministic Finite Automata

$\mathcal{A}$ is deterministic if

- $card(I) = 1$,

- Δ is a functional relation: for each p and a there is at most one pair b, q such that $(p, a, b, q) \in \Delta$.

In this case we write $\delta(p, a)$ for q such that $(p, a, ., q) \in \Delta$.

This is naturaly extended to $\delta(p, u)$, for finite words $u \in \Sigma^*$.

Büchi Automata

$\mathcal{A} = (Q, \Sigma, I, \Gamma, \Delta)$ is a *Büchi automaton* if $\Gamma = \{0, 1\}$.

The Büchi language L_B is the set of words v in $\{0, 1\}^\omega$ such that v contains infinitely many zeros.

A run ρ of a Büchi automaton $\mathcal{A}$ is *accepting* if $\mathrm{Out}(\rho) \in L_B$.

$\mathcal{A}$ *accepts* $u \in \Sigma^\omega$ if there is an accepting run of $\mathcal{A}$ on u.

Rabin Automata

$\mathcal{A}$ is a *Rabin automaton* with h conditions (i.e, h Rabin pairs) if $\Gamma = \mathcal{P}(\{r_1, s_1, r_2, s_2, \ldots, r_h, s_h\})$.

The Rabin language L_R is the set of words $v \in \Gamma^\omega$ such that for some i, $r_i \in v(n)$ for finitely many n, and $s_i \in v(n)$ for infinitely many n.

A run ρ of a Rabin automaton $\mathcal{A}$ is *accepting* if $\mathrm{Out}(\rho) \in L_R$.

$\mathcal{A}$ *accepts* u if there is an accepting run of $\mathcal{A}$ on u.

Determinization problem for state labeled Büchi automata

Theorem 1 *For each Büchi automata $\mathcal{B}$ there exists a deterministic Rabin automaton $\mathcal{R}$ such that $L(\mathcal{B}) = L(\mathcal{R})$.*

Let n be the number of states of a given Büchi automata.

Safra (1988) provided a construction of $\mathcal{R}$ with at most $(12)^n n^{2n}$.

Piterman (2007) provided parity automaton of at most $2nn^n(n!)$ ($n! \approx (0.36n)^n$).

Schewe (2009) defines an automaton with $o((2.66n)^n)$ states (but for the cost of 2^{n-1} Rabin pairs).

Lower bounds for state labeled automata

The first lower bound was given by Löding (1999) of $n!$.

Yan (2006) gives a lower bound of $\Omega((0.76n)^n)$.

Results

We prove, for a certain function $hist(n)$, specified later, that the determinization problem for transition labeled Büchi automata with n states requires exactly $hist(n)$ (with transition labeled Rabin automata as output).

For input being state labeled Büchi automata with n states one can infer a lower bound $hist(n-1)$.

$hist(n) \in \Omega((1.64n)^n)$.

Full automata

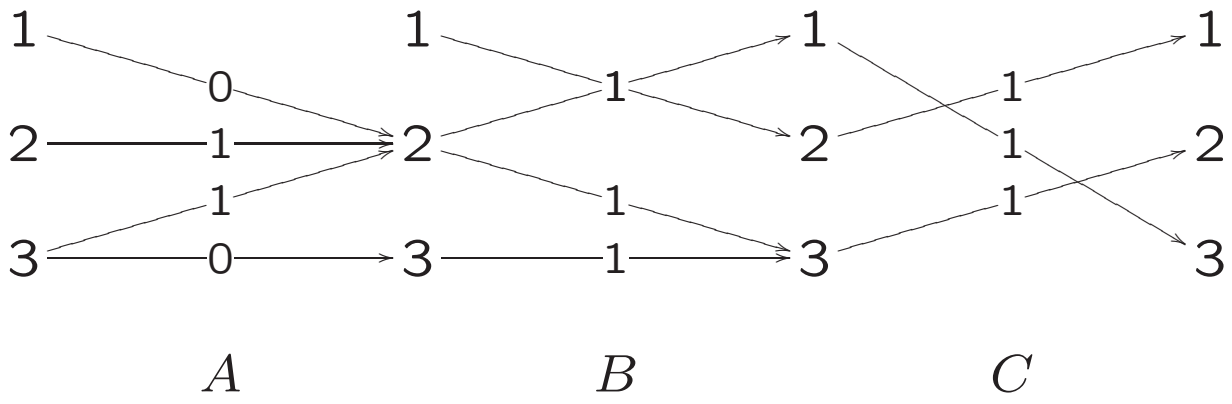
Let $Q = \{1, \ldots, n\}$.

$\mathcal{A}_n = (Q, \Sigma, Q, \{0, 1\}, \Delta)$ is a full automaton if

- $\Sigma = \mathcal{P}(Q \times \Gamma \times Q)$,
- $\Delta = \{(p, A, b, q) : (p, b, q) \in A\}$.

Lemma 2 (Yan'06) *The full automata are hardest to determinized.*

An example of a word



What is a tree?

A tree T is a subset of finite sequences of elements from ω closed on prefixes.

The empty sequence, ε, is a root of a tree.

Elements of T are nodes, for $x, xi, xj \in T$, xi is a child of x and xi, xj are siblings. For $i < j$, xi is an older sibling of xj.

We label nodes of T by nonempty subsets of a fixed set. A label of $x \in T$ is denoted by $T(x)$. For $x \notin T$, $T(x) = \emptyset$.

If x is a prefix of y, $x \preceq y$, then x is an ancestor of y and y is a descendant of x.

We assume that if $xi \in T$ then for all $j < i$, $xj \in T$.

Because of this, if we want to delate a node xj from T, we need to rename all the nodes of the form xiy, for $i > j$, into $x(i-1)y$.

Safra Style Detereminization by Schewe

Let us fix a Büchi automaton $\mathcal{B} = (Q, I, \Sigma, \{0, 1\}, \Delta)$ with n states. We look for a deterministic Rabin automaton $\mathcal{R}$ such that $L(\mathcal{B}) = L(\mathcal{R})$.

States of $\mathcal{R}$ are trees with nodes labeled by nonempty subsets of states of $\mathcal{B}$.

The initial state of the Rabin automaton is a one node tree labeled by the set of initial states of $\mathcal{B}$.

We denote this tree by T_0.

Conditions on states of $\mathcal{R}$

1. for each node $x \in T$, $T(x) \supsetneq \bigcup_{i \in \omega} T(xi)$,

2. labels of siblings are disjoint.

We call such trees *history trees*.

Let $hist(n)$ be the function which counts the number of history trees labeled by elements of an n-element set.

Lemma 3 *1.* $hist(n) \in \Omega((1.64n)^n)$ *(Bouvel, Rossin),*

2. $hist(n) \in o((1.65n)^n)$ *(Schewe).*

Some notation

Let $S \subseteq Q,\ A \in \Sigma$.

$$\Delta(S, A) = \{q : \exists p \in S \exists b\, (p, b, q) \in A\}\,,$$

$$\Delta_0(S, A) = \{q : \exists p \in S\, (p, 0, q) \in A\}\,.$$

Transition function

Assume that $\mathcal{R}$ in a state T reads a letter A.

1 For each node $x \in T$, replace the label of x with $\Delta(T(x), A)$.

2 For each node $x \in T$, originally labeled S, if $\Delta_0(S, A) \neq \emptyset$, form a new youngest child of x, label it with $\Delta_0(S, A)$.

3 For each $x \in T$, for each $q \in T(x)$, if q belongs to an older sibling of x, delete q from labels of x and its descendants.

4 Now, we contract the tree T obtained so far:

4.1 for each x with $T(x) \neq \emptyset$ and $T(x) = \bigcup_i T(xi)$ delete all strict descendants of x, call x *green*,

4.2 for each node x with $T(x) = \emptyset$, mark x and all the nodes in trees rooted at its younger siblings as red, delete the subtree rooted at x.

5 The output of the transition is a set $\mathcal{E}$ of what we call *events*. For each red node x we put $(x, A) \in \mathcal{E}$ and for each green x we put (x, E) in $\mathcal{E}$.

$\mathcal{R}$ accepts if there exists x such that (x, A) is generated only finitely many times and (x, E) is generated infinitely many times.

$\mathcal{R}$ accepts if there exists x such that (x, A) is generated only finitely many times and (x, E) is generated infinitely many times.

Why does it work?

If $\mathcal{B}$ accepts u with a computation ρ then consecutive states of ρ will end in a node x which will witness that $\mathcal{R}$ accepts u.

On the other hand, if, during a computation of $\mathcal{R}$: $T_0T_1T_2\ldots$, there is x as above then, using König's lemma, we can find accepting computation $\rho = q_0q_1q_2\ldots$ of $\mathcal{B}$ with $q_i \in T_i(x)$, for almost all i.

Main result

Let L_n be the language of the full Büchi automaton with n states.

Theorem 4 *Let $\mathcal{R}$ be a deterministic Rabin automaton accepting L_n. Then, $\mathcal{R}$ has at least $hist(n)$ states.*

Games

$\mathcal{G} = (V, V_E, V_A, q_0, \Gamma, \text{Move}, L)$ is game, where

- V is a set of *positions*, partitioned into the *positions for Eva* V_E and *positions for Adam* V_A,
- q_0 is the initial position of $\mathcal{G}$,
- Γ is the *labeling alphabet*,
- $\text{Move} \subseteq V \times \Gamma \times V$ is the set of possible *moves*,
- L is a winning condition.

During a play Adam and Eva makes moves from their positions forming and infinite sequence $\pi = (p_0, a_0, p_1, a_1, p_2, a_2, \dots)$ such that

- $p_0 = q_0$,
- $(p_i, a_i, p_{i+1}) \in \text{Move}$, for $i \in \omega$.

We call π a play.

Eva wins π if $a_0 a_1 a_2 \cdots \in L$.

Strategies

A strategy for the player X is a function which tells the player what moves X should choose depending on the finite history of moves played so far.

A play π is compatible with a strategy σ for X if each move of X conforms to σ, that is, for each $p_i \in V_X$,

$$(p_i, a_i, p_{i+1}) = \sigma(p_0, a_0, p_1, \ldots, p_i).$$

A strategy σ for X is winning if X wins each play compatible with σ.

σ is positional if it depends only on the actual position in the play, that is $\sigma(p_0, a_0, p_1, \ldots, p_i) = \sigma'(p_i)$.

A *strategy with memory* m *for Eva* is described as $\sigma = (M, \text{update}, \text{choice}, \text{init})$, where

1. $card(M) = m$,
2. update: $M \times \text{Move} \longrightarrow M$,
3. choice: $V_E \times M \longrightarrow \text{Move}$ and $\text{init} \in M$.

The mapping update is defined for moves, but can naturally be extended to paths in the game.

To a path $\pi = (p_0, a_0, \ldots, a_{n-1}, p_n)$ with $p_n \in V_E$, σ associates $\text{choice}(p_n, \text{update}(\text{init}, \pi))$.

A *positional strategy* corresponds to the case $m = 1$.

Eva wins a game with memory m if it has a winning strategy with memory m.

We call a game $\mathcal{G} = (V, V_E, V_A, q_0, \Gamma, \text{Move}, L)$ as an L-game.

If L is a Rabin language we say that $\mathcal{G}$ is a Rabin-game.

Theorem 5 (Klarlund94, Zielonka98) *For every Rabin-game, if Eva wins she can win using a positional strategy.*

Corollary 6 *Let L be accepted by a deterministic Rabin automaton with n states. If Eva wins an L-game then she wins with memory n.*

We will use this corollary as follows: if Eva wins an L-game, and requires memory n for that, then every deterministic Rabin-automaton for L has size at least n

An argument for Corollary 6

For a game $\mathcal{G} = (V, V_E, V_A, q_0, \Gamma, \text{Move}, L)$, let unfolding($\mathcal{G}$) be a tree T s.t. paths in T are all possible scenarios for $\mathcal{G}$.

Replace labels in each path in T, $a_0 \dots a_n \in \Gamma^*$ by the unique $b_0 \dots b_n$ which is output by a computation of a deterministic Rabin automaton recognizing L.

Then, we get from T a new game $\mathcal{G}'$ such that Eva wins $\mathcal{G}$ iff Eva wins $\mathcal{G}'$.

To win $\mathcal{G}'$ it is enough to have a positional strategy.

Then, to mimic this strategy in $\mathcal{G}$ Eva needs to compute $\mathcal{G}'$ "on the fly". To do this it suffices to keep track of the computation of $\mathcal{R}$ using memory of size $card(\mathcal{R})$.

Strategy for the proof

1. Take the full automaton $\mathcal{A}_n$ with n states, $L_n = L(\mathcal{A}_n)$.

2. Define an L_n-game $\mathcal{G}$ such that Eva wins $\mathcal{G}$.

3. Show that Eva does not win $\mathcal{G}$ with memory less than $hist(n)$.

A restricted case

Recall that an alphabet of a full automaton with set of states Q is $\Sigma = \mathcal{P}(Q \times \{0,1\} \times Q)$.

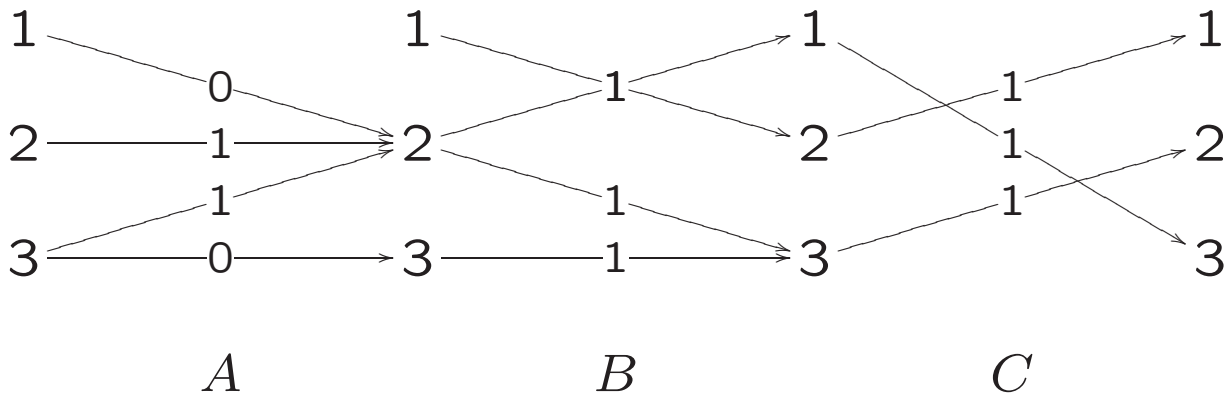
By induction we define Reach(u), for $u \in \Sigma^*$:

1. Reach$(\varepsilon) = Q$,
2. Reach$(ua) = \{q \in Q : \exists p \in \text{Reach}(u) \exists b\, (p,b,q) \in a\}$.

A word u, finite or infinite, is an S-stabilizer if for each v, nonempty prefix of u, we have Reach$(v) = S$.

Let L_n^S be the language of $\mathcal{A}_n$ restricted to S-stabilizers.

An example



Reach(A) $= \{2, 3\}$, Reach(B) $= \{1, 2, 3\}$, Reach(C) $= \{1, 2, 3\}$

Reach(AB) $= \{1, 3\}$, Reach(BC) $= \{1, 2, 3\}$

Reach(ABC) $= \{2, 3\}$

If a word u is an S stabilizer then during the computation $T_0T_1T_2\ldots$ of the Safra automaton on u, for each $i > 0$, $T_i(\varepsilon) = S$.

Let $\mathcal{H}_S$ be the set of history trees T such that $T(\varepsilon) = S$.

We prove first the following.

Lemma 7 *Every deterministic Rabin automaton accepting L_n^Q has size at least $|\mathcal{H}_Q|$.*

From the restricted to the general case

Let $\mathcal{R}$ be a deterministic Rabin automaton for L_n.

For $S \subseteq Q$, $R_S = \{q : \exists u \in (\Sigma_S)^* \, \delta(q_0, u) = q\}$.

Lemma 8 *If $L(\mathcal{R}) = L_n$, then, for $S \neq W$, $R_S \cap R_W = \emptyset$.*

If $\delta(q_0, u) = \delta(q_0, v)$ then

$$u\left\{r \xrightarrow{0} r\right\}^\omega \in L(\mathcal{R}) \iff v\left\{r \xrightarrow{0} r\right\}^\omega \in L(\mathcal{R}).$$

But if $r \in S \setminus W$, $u \in (\Sigma_S)^*$, $v \in \Sigma_W)^*$ then

$$u\left\{r \xrightarrow{0} r\right\}^\omega \in L_n \text{ and } v\left\{r \xrightarrow{0} r\right\}^\omega \notin L_n.$$

For each $S \subseteq Q$, R_S can be seen as deterministic Rabin automaton for a full Büchi automaton of cardinality $card(S)$ and set of states S.

Then, $L(\mathcal{R}_S) = L^S_{card(S)}$ and, by Lemma 7,

$$card(\mathcal{R}_S) = card(\mathcal{H}_S).$$

But

$$hist(n) = \sum_{\substack{S \subseteq \{1,\ldots,n\} \\ S \neq \emptyset}} card(\mathcal{H}_S)$$

The L_n^Q-game

$\mathcal{G}$= $(V, V_E, V_A, p_I, (\Sigma_Q)^*, \text{Move}, L_n^Q)$, where
1. $V_E = \{p_E\}$,
2. $V_A = \{p_I\} \cup \left\{p_T : T \in \mathcal{H}_Q\right\}$.

Let $\mathcal{R}$ be the deterministic automaton for $\mathcal{A}_n$ from Safra construction.

Recall that $\mathcal{E}(T, a)$ is the set of events generated by $\mathcal{R}$ while reading a in state T.

Let $\mathcal{E}(T, ua) = \mathcal{E}(T, u) \cup \mathcal{E}(\delta(T, u), a)$.

Let $T <_x T'$ if $T'(x) \subsetneq T(x)$ and, for all $y <_{\text{lex}} x$, $T'(y) = T(y)$.

The moves in $\mathcal{G}$ are the following:

1. $(p_I, u, p_E) \in$ Move, for all $u \in (\Sigma_Q)^*$,

2. $(p_E, \text{id}, p_T) \in$ Move, for $T \in \mathcal{H}_Q$, $\text{id} = \{(q, 1, q) : q \in Q\}$,

3. $(p_T, u, p_E) \in$ Move, for $T \in \mathcal{H}_Q$, $u \in (\Sigma_Q)^*$, provides that there exists a node $x \in \delta(T, u)$ such that either

   - $(x, E) \in \mathcal{E}(T, u)$, and for all $y \leq_{\text{lex}} x$, $(y, A) \notin \mathcal{E}(T, u)$ and $\delta(T, u)(y) = T(y)$ or;

   - $\delta(T, u) <_x T$ and for all $y <_{\text{lex}} x$, $(y, A) \notin \mathcal{E}(T, u)$.

$\mathcal{G}$ has a flower shape.

Firstly, Adam chooses to go to the central node p_E with an arbitrary word.

The central node p_E is controlled by Eva and she can decide to go to any of the petals p_T corresponding to a history tree T.

Then, Adam chooses a word u and come back to the center.

Adam's moves are restricted in the following way:

- he can play a word u in a petal p_T if it is possible to witness in the behaviour of $\mathcal{R}$ from the state T when reading u that it is profitable for Eva.

This witness takes the form of a node $x \in \delta(T, u)$ such that nothing happens for nodes $y \in \delta(T, u)$, $y <_{\mathsf{lex}} x$, and something good for Eva happened at x:

$$\text{either } (x, E) \in \delta(T, u) \text{ or } \delta(T, u) <_x T.$$

Eva wins $\mathcal{G}$ with memory $card(\mathcal{H}_Q)$

A winning strategy for Eva is as follows:

if $u = u_0u_1 \dots u_n$ is a word played so far and $T = \delta(T_0, u)$ then Eva chooses to go to p_T.

Then Adam has to move in such a way that

$$\text{either } (x, E) \in \delta(T, u) \text{ or } \delta(T, u) <_x T.$$

Each $\delta(T, u) <_x T$ requires moving some states to a lexicographically smaller position x.

Thus, after some finite time Adam has to use option $(x, E) \in \delta(T, u)$.

Eva really needs memory $card(\mathcal{H}_Q)$ to win

If Eva has memory $m < card(\mathcal{H}_Q)$ then her strategy will systematically omitt some petal p_T for $T \in \mathcal{H}_Q$.

Indeed, her moves depends only on the content of memory, so she may chooses only m different petals.

For $T \in \mathcal{H}_Q$, let $\mathcal{G}_T$ be a modification of $\mathcal{G}$ by removing the petal p_T.

Lemma 9 *For each $T \in \mathcal{H}_Q$, Adam wins $\mathcal{G}_T$.*

Proving lemma completes the proof of the main theorem.

A winnig strategy for Adam in $\mathcal{G}_T$

Lemma 10 *Let $T' \neq T$ be history trees in $\mathcal{H}_Q$. There exists a word u such that*

1. *$(p_{T'}, u, p_E)$ is a move in $\mathcal{G}$ and $\mathcal{G}_T$,*
2. *$T = \delta(T, u)$,*
3. *for all x, $(x, E) \notin \mathcal{E}(T, u)$.*

From p_I Adam plays a word u such that $\delta(T_0, u) = T$

Eva has to go to $p_{T'}$ for some $T' \neq T$.

Then, Adam play a word u from Lemma 10.

During all play there is no Eva good event (x, E), thus Adam wins.

Sketch of the proof of Lemma 10

The suffix ordering.

For two nodes $x, y \in T$, $x \leq_{\mathsf{suf}} y$ if

either $x <_{\mathsf{lex}} y$ and $\neg(x \prec y)$, or $y \prec x$.

$<_{\mathsf{suf}}$ is a strict version of $\leq_{\mathsf{suf}}$.

Suffix(T, ε) outputs nodes of T according to $\leq_{\mathsf{suf}}$.

Let children(T) be the number of children of the root of T.

Let child(T, i) be the i-th child of the root of T.

Then Suffix(T, x) can be writte as:

- For $i = 0, \ldots,$ children$(T) - 1$,
  Suffix(child(T, xi)).

- Write(x).

Let us fix $T, T' \in \mathcal{H}_Q$, $T \neq T'$.

p_T is a removed petal and $p_{T'}$ is a petal chosen by Eva.

Let $x_0, \ldots, x_n$ and $x'_0, \ldots, x'_{n'}$ be nodes of T and T', respectively, in the suffix order.

Let $\mathcal{L} = T(x_0), \ldots, T(x_n)$ and $\mathcal{L}' = T'(x'_0), \ldots, T'(x'_{n'})$ be corresponding labelings.

Let us observe that from $\mathcal{L}$, $\mathcal{L}'$ we can reconstruct T and T'.

This is so, because $x_i \prec x_j$ iff $T(x_j) \subsetneq T(x_i)$.

Since $T \neq T'$ and $\mathcal{L}$ and $\mathcal{L}'$ contain all information about T and T', there exists i such that:

- $T(x_i) \neq T'(x'_i)$,
- for all $j < i$, $T(x_j) = T(x'_j)$.

We take such an i and by a case analysis provide the suitable move for Adam.

Notation for letters in $\mathcal{P}(Q \times \Gamma \times Q)$

The default letter is id $= \{(q, 1, q) : q \in Q\}$.

For more complex letter, we use a notation $\{x_1, \ldots, x_n\}$ in which x_i is either $p \xrightarrow{b} q$, for $b \in \{0, 1\}$, or $p \rightarrow \perp$.

$\{x_1, \ldots, x_n\}$ describes the letter which contains:

- (p, b, q) if $p \xrightarrow{b} q = x_i$ for some i,
- $(p, 1, p)$ if none of the x_i's is of the form $p \xrightarrow{b} q$ or $p \rightarrow \perp$,
- no transition $(p, ., .)$ if some $x_i = p \rightarrow \perp$.

Recall that we want to provide u such that:

1. $(p_{T'}, u, p_E)$ is a move in $\mathcal{G}$ and $\mathcal{G}_T$,

2. $T = \delta(T, u)$,

3. for all x, $(x, E) \notin \mathcal{E}(T, u)$.

Recall that $x_0, \ldots, x_n$ and $x'_0, \ldots, x'_{n'}$ are nodes of T and T', respectively, in the suffix order.

We fix the smallest i such that $T(x_i) \neq T'(x'_i)$.

Let $\mathrm{New}(T, x) = T(x) \setminus \bigcup_{k \in \omega} T(xk)$.

Let $R = \mathrm{New}(T, x_i)$ and $R' = \mathrm{New}(T', x'_i)$.

If $R \neq R'$ we have three cases:

1. there are $q \in R \setminus R'$, $q' \in R' \setminus R$,

2. $R \subsetneq R'$,

3. $R' \subsetneq R$.

$R = \mathsf{New}(T, x_i)$ and $R' = \mathsf{New}(T', x'_i)$.

1. There are $q \in R \setminus R'$, $q' \in R' \setminus R$.

Then, $q' \notin \bigcup_{j \leq i} T(x_j)$ and $q \notin \bigcup_{j \leq i} T'(x'_j)$.

A good move for Adam is

$$M = \left\{ q' \xrightarrow{0} q, q' \xrightarrow{1} q' \right\}.$$

Let $y \in T'$, such that $q \in \mathsf{New}(T', y)$.

If $y <_{\mathsf{lex}} x_i$, let $y' = yk$ such that $y' \leq_{\mathsf{lex}} x_i$.

Either $(y, E) \in \mathcal{E}(T', M)$ or $\delta(T', M) <_{y'} T'$ $[T'(y') \subsetneq \delta(T', M)(y')]$.

If $\neg(y <_{\mathsf{lex}} x_i)$, then $\delta(T', M) <_{x_i'} T'$.

The other cases can be treated similarly

If New(T, x_i) = New(T', x'_i) then the difference between $T(x_i)$ and $T'(x'_i)$ must come from the difference in children of x_i in T and x'_i in T', respectively.

Again, a case analysis is needed.

Further work

1. An extension to determinizing Streett automata (determinization construction given by Safra).

2. Lower bounds for operations on tree automata.

3. Considering parity automata as output.

4. Considering state labeled automata as output.

Thank you.