A Tight Lower Bound for Determinization of Transition Labeled Büchi Automata

Thomas Colcombet Konrad Zdanowski

LIAFA, CNRS
University Denis Diderot, Paris

ICALP, Rhodos 2009

Outline

Outline

1. Description of the Problem
   - Determinization Theorem
   - Upper Bounds
   - Lower Bounds
   - Our Results

Theorem (McNughton'1966)

For every Büchi automaton $\mathcal{A}$ on infinite word there exists a deterministic (Müller) automaton $\mathcal{B}$ such that $L(\mathcal{A}) = L(\mathcal{B})$.

We consider state complexity for a deterministic Rabin automaton equivalent to a given Büchi automaton.

Outline

1. Description of the Problem
   - Determinization Theorem
   - Upper Bounds
   - Lower Bounds
   - Our Results

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

- Safra (1988) provided a construction of $\mathcal{R}$ with at most $(12n^2)^n$.
- Piterman (2007) provided parity automaton of at most $2n(0.36n^2)^n$.
- Schewe (2009) defines an automaton with $o((2.66n)^n)$ states (but for the cost of 2^{n-1} Rabin pairs).

Outline

1. Description of the Problem
   - Determinization Theorem
   - Upper Bounds
   - Lower Bounds
   - Our Results

- The first lower bound was given by Löding (1999) of $n! \approx (0.36n)^n$.
- Yan (2006) gives a lower bound of $\Omega((0.76n)^n)$.

Outline

1. Description of the Problem
   - Determinization Theorem
   - Upper Bounds
   - Lower Bounds
   - Our Results

- We consider transition labeled rather than state labeled automata.
- The upper bound given by Schewe (2009) is $hist(n) \in o((1.65n)^n)$.

Theorem

The lower bound for the determinization problem for Büchi automata is $hist(n)$.

The lower and upper bounds are now the same.

Outline

2. Basic Notions
   - Finite Transducers
   - Büchi Aceptance Conditions
   - Rabin Aceptance Condition

Definition

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

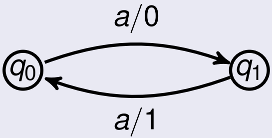
- *Q is a set of states,*
- *Σ is an input alphabet,*
- *I is a set of initial states,*
- *Γ is an output alphabet*
- *$\Delta \subseteq Q \times \Sigma \times \Gamma \times Q$ is the transition relation.*

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

Example

Let $\Sigma = \{a\}, \Gamma = \{0, 1\}, \Delta = \{(q_0, a, 0, q_1), (q_1, a, 1, q_0)\}$.

We write $\mathcal{A}$ *as*



Example

Let $\Sigma = \{a\}$, $\Gamma = \{0, 1\}$, $\Delta = \{(q_0, a, 0, q_1), (q_1, a, 1, q_0)\}$.

We write $\mathcal{A}$ *as*

$$\begin{tikzcd} q_0 \arrow[r, bend left, "a/0"] & q_1 \arrow[l, bend left, "a/1"] \end{tikzcd}$$

Then, the computation ρ *of* $\mathcal{A}$ *on* a^ω *is*

$$\begin{array}{ccccc} a & a & a & a & \cdots \\ q_0 & & & & \end{array}$$

Example

Let $\Sigma = \{a\}, \Gamma = \{0,1\}, \Delta = \{(q_0, a, 0, q_1), (q_1, a, 1, q_0)\}$.

We write $\mathcal{A}$ *as*

$$\begin{tikzcd} q_0 \arrow[r, bend left, "a/0"] & q_1 \arrow[l, bend left, "a/1"] \end{tikzcd}$$

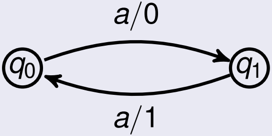
Then, the computation ρ *of* $\mathcal{A}$ *on* a^ω *is*

$$\begin{array}{ccccccccc} a & & a & & a & & a & & \cdots \\ q_0 & & q_1 & & & & & & \\ & 0 & & & & & & & \end{array}$$

Example

Let $\Sigma = \{a\}, \Gamma = \{0, 1\}, \Delta = \{(q_0, a, 0, q_1), (q_1, a, 1, q_0)\}$.

We write $\mathcal{A}$ *as*



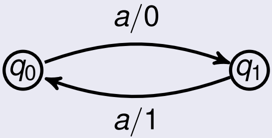
Then, the computation ρ *of* $\mathcal{A}$ *on* a^ω *is*

$$\begin{array}{ccccccccc} a & & a & & a & & a & & \cdots \\ q_0 & & q_1 & & q_0 & & & & \\ & 0 & & 1 & & & & & \end{array}$$

Example

Let $\Sigma = \{a\}, \Gamma = \{0, 1\}, \Delta = \{(q_0, a, 0, q_1), (q_1, a, 1, q_0)\}$.

We write $\mathcal{A}$ *as*



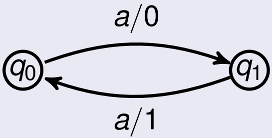
Then, the computation ρ *of* $\mathcal{A}$ *on* a^ω *is*

$$\begin{array}{ccccccccc} a & & a & & a & & a & & \cdots \\ q_0 & & q_1 & & q_0 & & q_1 & & \\ & 0 & & 1 & & 0 & & & \end{array}$$

Example

Let $\Sigma = \{a\}, \Gamma = \{0, 1\}, \Delta = \{(q_0, a, 0, q_1), (q_1, a, 1, q_0)\}$.

We write $\mathcal{A}$ *as*



Then, the computation ρ *of* $\mathcal{A}$ *on* a^ω *is*

$$\begin{array}{ccccccccc} a & & a & & a & & a & & \cdots \\ q_0 & & q_1 & & q_0 & & q_1 & & \cdots \\ & 0 & & 1 & & 0 & & 1 & \end{array}$$

Example

Let $\Sigma = \{a\}, \Gamma = \{0, 1\}, \Delta = \{(q_0, a, 0, q_1), (q_1, a, 1, q_0)\}$.

We write $\mathcal{A}$ *as*

$$\begin{tikzcd} q_0 \arrow[r, bend left, "a/0"] & q_1 \arrow[l, bend left, "a/1"] \end{tikzcd}$$

Then, the computation ρ *of* $\mathcal{A}$ *on* a^ω *is*

$$\begin{array}{ccccccccc} a & & a & & a & & a & & \cdots \\ q_0 & & q_1 & & q_0 & & q_1 & & \cdots \\ & 0 & & 1 & & 0 & & 1 & \end{array}$$

The output of ρ, $Out(\rho)$, *is the word* $(01)^\omega$.

Outline

Definition

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

Definition

- $\mathcal{A} = (Q, \Sigma, I, \Gamma, \Delta)$ *is a* *Büchi automaton* *if* $\Gamma = \{0, 1\}$.
- *The* *Büchi language* L_B *is the set of words from* $\{0, 1\}^\omega$ *which contains infinitely many zeros.*

Definition

- $\mathcal{A} = (Q, \Sigma, I, \Gamma, \Delta)$ *is a* ***Büchi automaton*** *if* $\Gamma = \{0, 1\}$.
- *The* ***Büchi language*** L_B *is the set of words from* $\{0, 1\}^\omega$ *which contains infinitely many zeros.*
- *A* ***transition*** *of the form* $(p, a, 0, q)$ *is* ***accepting***.

Definition

- $\mathcal{A} = (Q, \Sigma, I, \Gamma, \Delta)$ *is a* ***Büchi automaton*** *if* $\Gamma = \{0, 1\}$.
- *The* ***Büchi language*** L_B *is the set of words from* $\{0, 1\}^\omega$ *which contains infinitely many zeros.*
- *A* ***transition*** *of the form* $(p, a, 0, q)$ *is* ***accepting***.
- *A* ***run*** ρ *of a Büchi automaton* $\mathcal{A}$ *is* ***accepting*** *if* $Out(\rho) \in L_B$.

Definition

- $\mathcal{A} = (Q, \Sigma, I, \Gamma, \Delta)$ *is a* ***Büchi automaton*** *if* $\Gamma = \{0, 1\}$.
- *The* ***Büchi language*** L_B *is the set of words from* $\{0, 1\}^\omega$ *which contains infinitely many zeros.*
- *A* ***transition*** *of the form* $(p, a, 0, q)$ *is* ***accepting***.
- *A* ***run*** ρ *of a Büchi automaton* $\mathcal{A}$ *is* ***accepting*** *if* $Out(\rho) \in L_B$.
- $\mathcal{A}$ ***accepts*** $u \in \Sigma^\omega$ *if there is an accepting run of* $\mathcal{A}$ *on* u.

Outline

2. Basic Notions
   - Finite Transducers
   - Büchi Aceptance Conditions
   - Rabin Aceptance Condition

Definition

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

Definition

- $\mathcal{A}$ *is a* *Rabin automaton* *with* *h Rabin conditions* *if* $\Gamma = \mathcal{P}(\{r_1, s_1, r_2, s_2, \ldots, r_h, s_h\})$.
- *A Rabin condition* (r_i, s_i) *is* *satisfied by* $v \in \Gamma^\omega$ *if*

$$\exists^\infty k\ s_i \in v(k) \text{ and } \neg\exists^\infty k\ r_i \in v(k).$$

Definition

- *$\mathcal{A}$ is a Rabin automaton with h Rabin conditions if $\Gamma = \mathcal{P}(\{r_1, s_1, r_2, s_2, \ldots, r_h, s_h\})$.*
- *A Rabin condition (r_i, s_i) is satisfied by $v \in \Gamma^\omega$ if*

$$\exists^\infty k \; s_i \in v(k) \text{ and } \neg\exists^\infty k \; r_i \in v(k).$$

- *The Rabin language L_R is the set of words $v \in \Gamma^\omega$ such that some Rabin condition is satisfied by v.*

Definition

- $\mathcal{A}$ *is a* *Rabin automaton* *with* *h Rabin conditions* *if* $\Gamma = \mathcal{P}(\{r_1, s_1, r_2, s_2, \ldots, r_h, s_h\})$.
- *A Rabin condition* (r_i, s_i) *is* *satisfied by* $v \in \Gamma^\omega$ *if*

$$\exists^\infty k \; s_i \in v(k) \text{ and } \neg\exists^\infty k \; r_i \in v(k).$$

- *The* *Rabin language* L_R *is the set of words* $v \in \Gamma^\omega$ *such that some Rabin condition is satisfied by* v.
- *A run* ρ *of a Rabin automaton* $\mathcal{A}$ *is* *accepting* *if* $Out(\rho) \in L_R$.

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

Outline

3 Safra determinization by Schewe

- Safra/Schewe Construction of the Rabin Automaton
- Acceptance Condition
- Tree Ordering

- We fix a Büchi automaton $\mathcal{A}$ with the set of states Q.
- $\mathcal{R}$ is a deterministic Rabin automaton from Safra/Schewe construction such that $L(\mathcal{R}) = L(\mathcal{A})$, δ_R is its transition function and $\mathcal{E}(q, a)$ is its output while reading a in a state q.

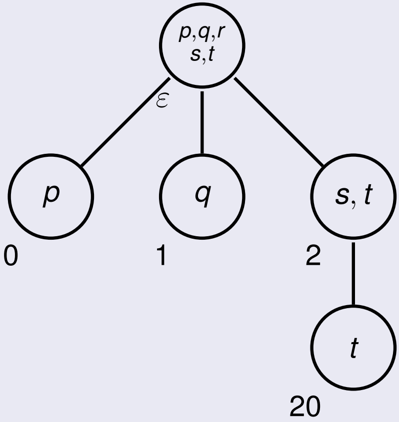
Definition (States of Safra/Schewe Automaton)

The states of $\mathcal{R}$ are finite trees labeled by nonempty subsets of Q such that

- *for each $x \in T$, $T(x) \supsetneq \bigcup_{i \in \omega} T(xi)$,*
- *for each $x \in T$, for each $j \neq i$, $T(xj) \cap T(xi) = \emptyset$.*

After Schewe, we call such trees history trees.

Example (of a history tree)



- *Nodes in a history tree represent some possible computations of the Büchi automaton $\mathcal{A}$.*
- *States labeling a given node are current states of computations of $\mathcal{A}$ represented by this node.*

There are $2^{card(Q)-1}$ possible nodes in a history tree but each history tree may have at most $card(Q)$ nodes.

Theorem

1. $hist(n) \in o((1.65n)^n)$ *(Schewe).*
2. $hist(n-1) \in \Omega((1.64n)^n)$ *(Bouvel, Rossin),*

Outline

- Recall that $\mathcal{E}(T, a)$ is the set of output events while reading a letter a in a state T.
- During a transition automaton produces two kinds of events: (x, A) and (x, E), where x is a node in history tree.
- Automaton accepts if for some x, (x, E) is output infinitely often and (x, A) is output only finitely often.

Outline

3. Safra determinization by Schewe
   - Safra/Schewe Construction of the Rabin Automaton
   - Acceptance Condition
   - Tree Ordering

We define a partial ordering on the set of history trees.

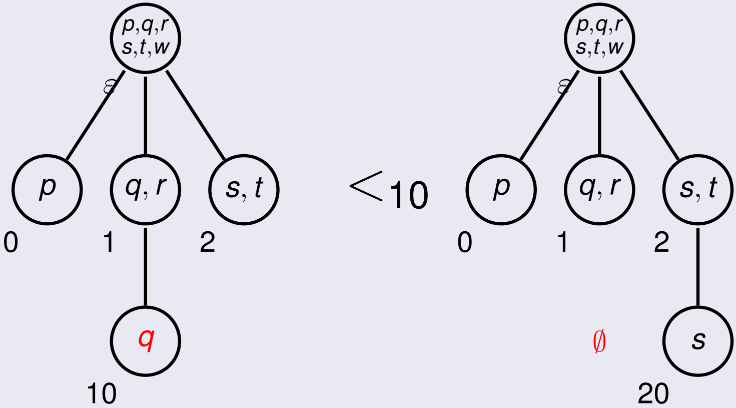
Definition

Let T, T' be history trees and let x be a possible node in a history tree.
T is strictly smaller than T' (at a position x), $T <_x T'$, if $T'(x) \subsetneq T(x)$ and for all $y <_{lex} x$, $T'(y) = T(y)$.

The intuition behind this definition is that

- for some node x, T has more possible computations which are kept at this node
- T and T' are equally good for all lexicographically smaller nodes.

Example



Outline

4. Notions for Proof
   - Theorem
   - Full Automata
   - Games and Lower Bounds

Theorem

The lower bound for the determinization problem for Büchi automata is $hist(n)$.

The above lower bound is exact.

Outline

4 Notions for Proof
- Theorem
- Full Automata
- Games and Lower Bounds

Definition (Full Automaton)

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 (Yan'06)

The full automata are hardest to determinized with respect to state complexity of an output automaton.

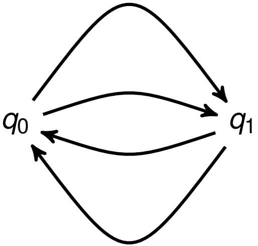
Thus, the automaton $\mathcal{A}$ is the full automaton $\mathcal{A}_n$ and $L = L(\mathcal{A}_n)$.

Outline

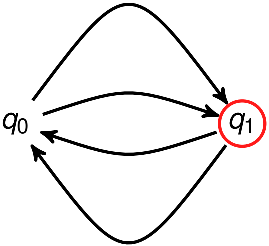
4 Notions for Proof

- Theorem
- Full Automata
- Games and Lower Bounds

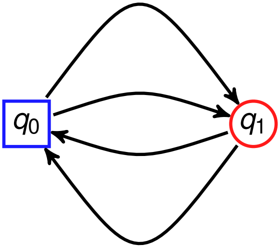
- A *L*-game $\mathcal{G} = (V, V_E, V_A, q_0, Move, \Gamma, L)$ can be seen as a directed graph



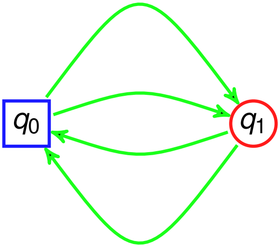
- A *L*-game $\mathcal{G} = (V, V_E, V_A, q_0, Move, \Gamma, L)$ can be seen as a directed graph
- with nodes V_E belonging to Eve



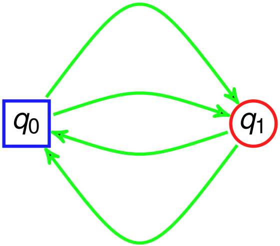
- A L-game $\mathcal{G} = (V, V_E, V_A, q_0, Move, \Gamma, L)$ can be seen as a directed graph
- with nodes V_E belonging to Eve and nodes V_A belonging to Adam.



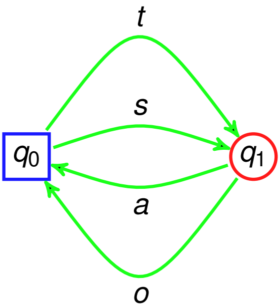
- A L-game $\mathcal{G} = (V, V_E, V_A, q_0, Move, \Gamma, L)$ can be seen as a directed graph
- with nodes V_E belonging to Eve and nodes V_A belonging to Adam.
- Players move sequentially starting from q_0 choosing some edge to go. A player X moves from nodes in V_X.



- A *L-game* $\mathcal{G} = (V, V_E, V_A, q_0, Move, \Gamma, L)$ can be seen as a directed graph
- with nodes V_E belonging to Eve and nodes V_A belonging to Adam.
- Players move sequentially starting from q_0 choosing some edge to go. A player X moves from nodes in V_X.
- During each move they produce some output $a_i \in \Gamma$.



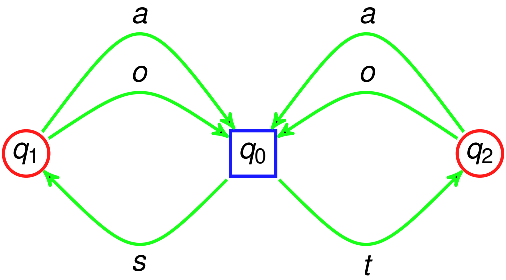
- A *L-game* $\mathcal{G} = (V, V_E, V_A, q_0, Move, \Gamma, L)$ can be seen as a directed graph
- with nodes V_E belonging to Eve and nodes V_A belonging to Adam.
- Players move sequentially starting from q_0 choosing some edge to go. A player X moves from nodes in V_X.
- During each move they produce some output $a_i \in \Gamma$.
- The result of the play is an infinite word $a_0 a_1 a_2 \ldots$ Eve wins if $a_0 a_1 a_2 \cdots \in L$.



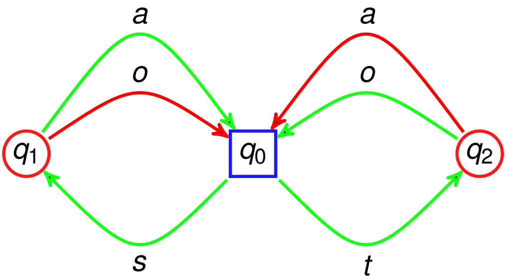
Definition

A strategy for a player X is a function which depending on the finite history of the play tells X what move X should take if the current position is in V_X.
A strategy σ is wining if X wins every play provided that he/she plays according to σ.
A strategy is positional if the choice of σ depends only on the current position.

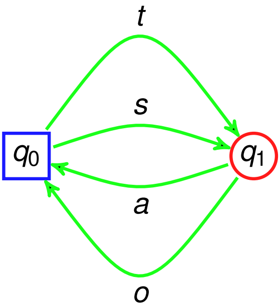
Let the winning condition be $L = (ta + so)^{\omega}$. Then, Eve wins the game below with a positional strategy.



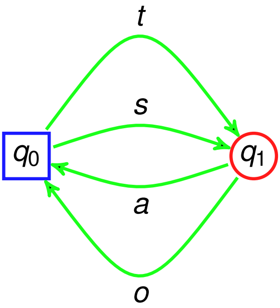
Let the winning condition be $L = (ta + so)^\omega$. Then, Eve wins the game below with a positional strategy.



Let the winning condition be $L = (ta + so)^\omega$. Eve wins the game below but she has no positional strategy.



Let the winning condition be $L = (ta + so)^{\omega}$. Eve wins the game below but she has no positional strategy.



To win Eve needs to remember the last Adam's move.

Definition

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

- $card(M) = m$,
- $update\colon M \times Move \longrightarrow M$,
- $choice\colon V_E \times M \longrightarrow Move$ *and* $init \in M$.

During a play Eva updates a content of her memory after every played move according to a function update.

Her moves depend only on her actual position and the present content of the memory.

A *positional strategy* corresponds to the case $card(M) = 1$.

Definition

A game $\mathcal{G} = (V, V_E, V_A, q_0, Move, \Gamma, L)$ is Rabin if L is a Rabin language.

Theorem (Klarlund94, Zielonka98)

For every Rabin-game, if Eve wins she can win using a positional strategy.

Corollary

Let L be accepted by a deterministic Rabin automaton with n states. If Eve 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

Outline

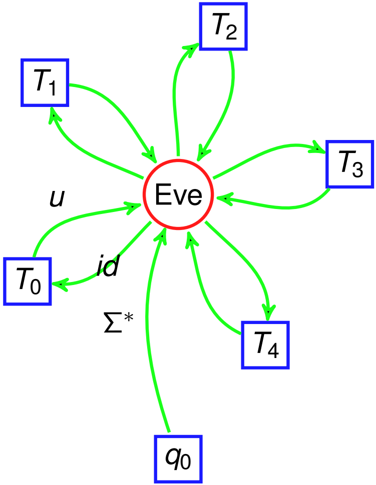
5 Proof
- An Proof Outline
- The Game
- Eva wins with memory *hist*(n)
- Eve looses without memory *hist*(n)

- Now, we will define a game $\mathcal{G}$ with the wining condition L – the full automata language.
- Then we show that Eve wins $\mathcal{G}$ with memory $hist(n)$ and that she looses with less memory.
- This proves that any deterministic Rabin automaton for L has size at least $hist(n)$.

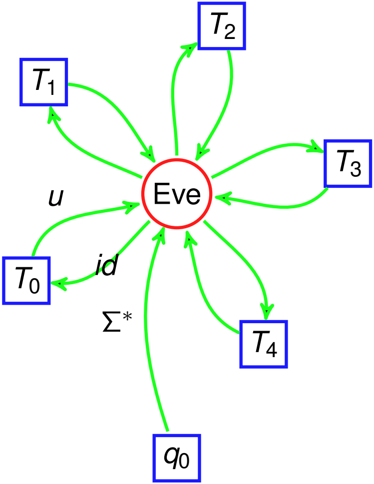
Outline

5 Proof

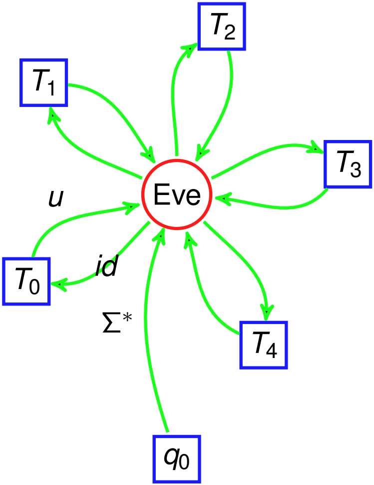
- An Proof Outline
- The Game
- Eva wins with memory $hist(n)$
- Eve looses without memory $hist(n)$



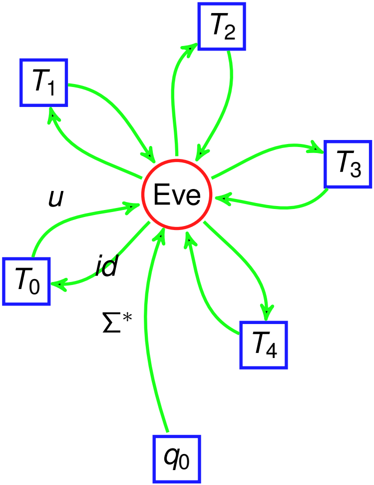
- Petals of the flower game are indexed by all history trees.



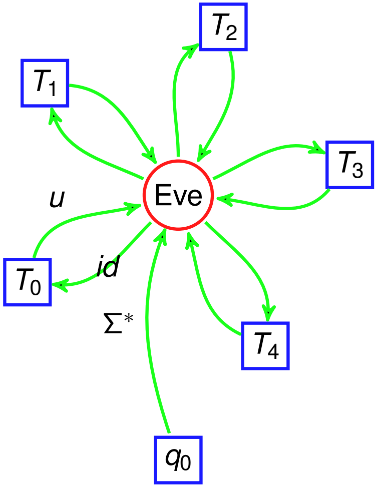
- Petals of the flower game are indexed by all history trees.
- In the first move of $\mathcal{G}$, Adam moves to the center and outputs an arbitrary word from Σ^*.



- Petals of the flower game are indexed by all history trees.
- In the first move of $\mathcal{G}$, Adam moves to the center and outputs an arbitrary word from Σ^*.
- Then, Eve chooses to go to some petal T_i using neutral letter



- Petals of the flower game are indexed by all history trees.
- In the first move of $\mathcal{G}$, Adam moves to the center and outputs an arbitrary word from Σ^*.
- Then, Eve chooses to go to some petal T_i using neutral letter
- Adam returns to the center by playing a word u which fulfills some restrictions.

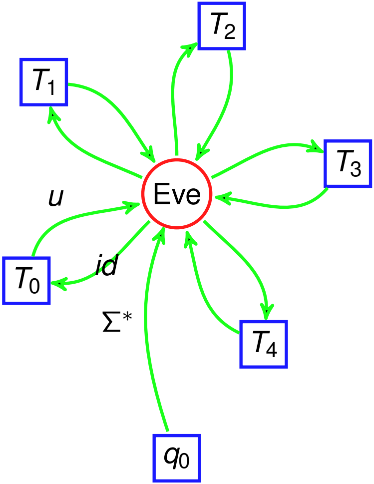


Adam may play from T a word u provided that there exists a node $x \in \delta_R(T, u)$ such that

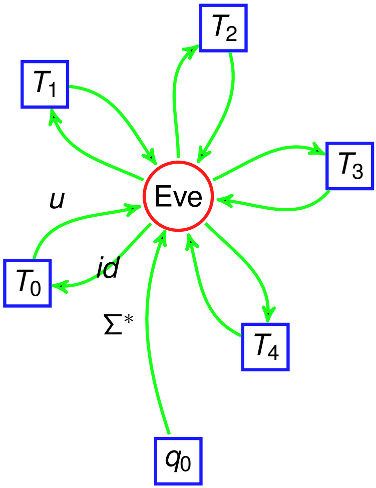
- $(x, A) \notin \mathcal{E}(T, u)$.
- either $(x, E) \in \mathcal{E}(T, u)$ or $\delta_R(T, u) <_x T$.
- for all $y <_{lex} x$, $T(y) = \delta_R(T, u)$ and $(y, A) \notin \mathcal{E}(T, u)$,

Outline

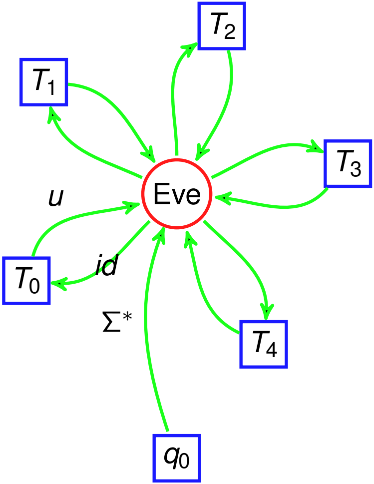
5 Proof
- An Proof Outline
- The Game
- Eva wins with memory *hist*(n)
- Eve looses without memory *hist*(n)



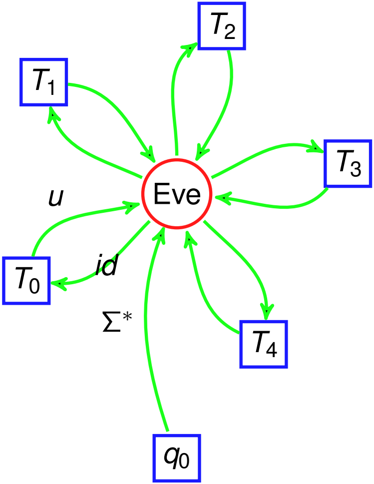
- If the word played so far is $u = u_0 \dots u_k$ then Eve ought to go to $T = \delta_{\mathcal{R}}(T_0, u)$, a current state of the computation of $\mathcal{R}$ on u.



- If the word played so far is $u = u_0 \ldots u_k$ then Eve ought to go to $T = \delta_{\mathcal{R}}(T_0, u)$, a current state of the computation of $\mathcal{R}$ on u.
- While returning with u_{k+1} Adam has to produce x such that $\delta_{\mathcal{R}}(T, u_{k+1}) <_x T$ or $(x, E) \in \mathcal{E}(T, u)$.



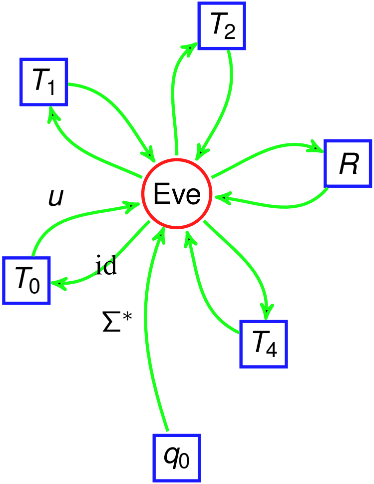
- If the word played so far is $u = u_0 \ldots u_k$ then Eve ought to go to $T = \delta_{\mathcal{R}}(T_0, u)$, a current state of the computation of $\mathcal{R}$ on u.
- While returning with u_{k+1} Adam has to produce x such that $\delta_{\mathcal{R}}(T, u_{k+1}) <_x T$ or $(x, E) \in \mathcal{E}(T, u)$.
- Thus, there will be a $<_{lex}$-min x such that (x, E) is generated infinitely often.



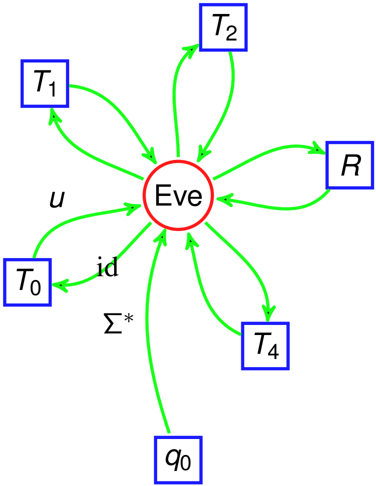
- If the word played so far is $u = u_0 \dots u_k$ then Eve ought to go to $T = \delta_{\mathcal{R}}(T_0, u)$, a current state of the computation of $\mathcal{R}$ on u.
- While returning with u_{k+1} Adam has to produce x such that $\delta_{\mathcal{R}}(T, u_{k+1}) <_x T$ or $(x, E) \in \mathcal{E}(T, u)$.
- Thus, there will be a $<_{lex}$-min x such that (x, E) is generated infinitely often.
- $\mathcal{R}$ accepts an infinite word $u_0 u_1 \dots u_k \dots$ and Eve wins!

Outline

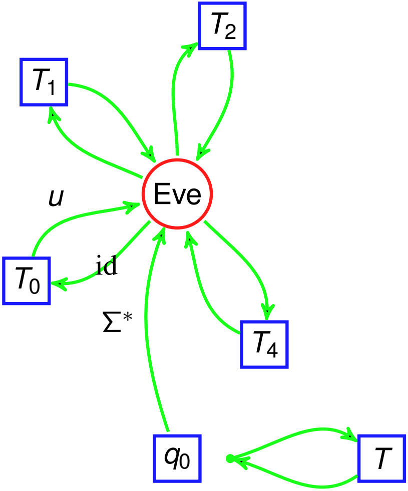
5. Proof
   - An Proof Outline
   - The Game
   - Eva wins with memory $hist(n)$
   - Eve looses without memory $hist(n)$



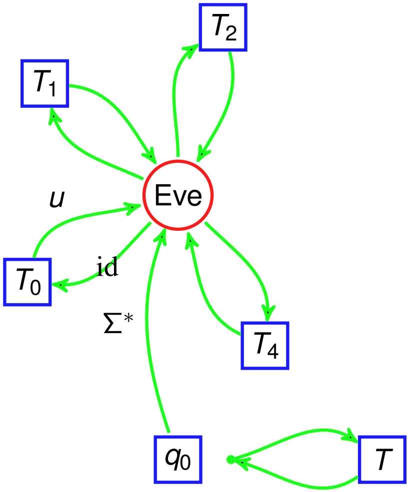
- If Eve has less memory then she will omit some petal R during her turn of the game.



- If Eve has less memory then she will omit some petal R during her turn of the game.
- This corresponds to removing one petal from the game.



- If Eve has less memory then she will omit some petal R during her turn of the game.
- This corresponds to removing one petal from the game.



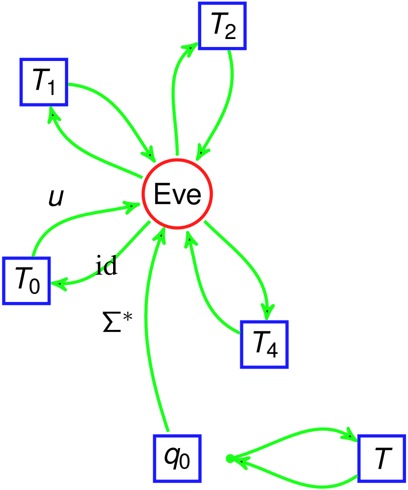
- If Eve has less memory then she will omit some petal R during her turn of the game.
- This corresponds to removing one petal from the game.
- Adam wins so modified game $\mathcal{G}_R$.

A winning strategy for Adam in $\mathcal{G}_R$ is based on the following lemma.

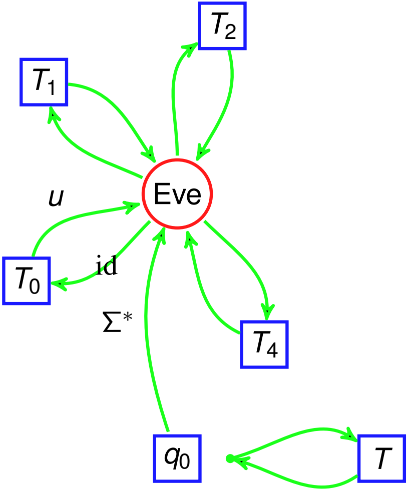
Lemma

Let $T \neq R$ be history trees. There exists a word $u = u(T, R)$ such that

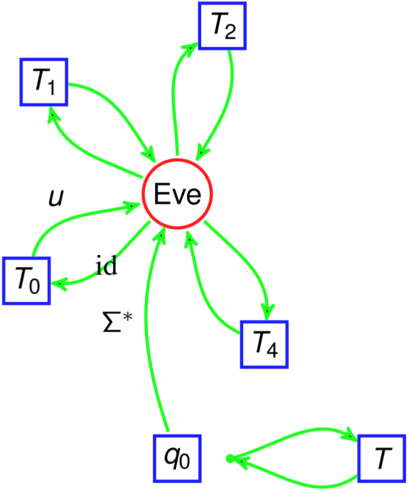
- *Adam may play u from the vertex T,*
- $R = \delta(R, u)$,
- *for all x, $(x, E) \notin \mathcal{E}(R, u)$.*



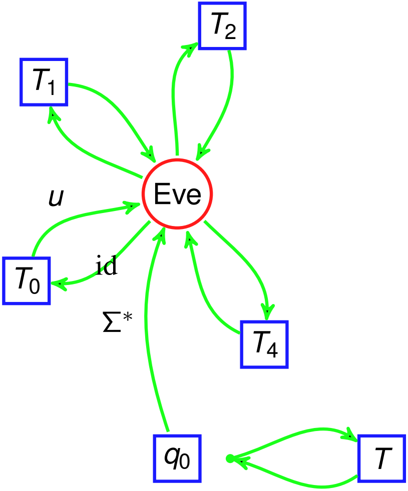
- During his first move Adam plays a word u_0 such that $\delta_{\mathcal{R}}(T_0, u_0) = R$.



- During his first move Adam plays a word u_0 such that $\delta_{\mathcal{R}}(T_0, u_0) = R$.
- Eve has to go to some $T \neq R$.



- During his first move Adam plays a word u_0 such that $\delta_{\mathcal{R}}(T_0, u_0) = R$.
- Eve has to go to some $T \neq R$.
- Then Adam may play a word $u = u(T, R)$.



- During his first move Adam plays a word u_0 such that $\delta_{\mathcal{R}}(T_0, u_0) = R$.
- Eve has to go to some $T \neq R$.
- Then Adam may play a word $u = u(T, R)$.
- After each such round no event (x, E) is in $\mathcal{E}(R, u)$. It follows that $\mathcal{R}$ does not accept produced infinite word and Adam wins!

Conclusions

- We have solved the lower bound problem for determinization of finite Büchi transducers.
- For input being a state labeled automaton the lower bound is slightly weaker, $hist(n-1)$.

Further Work

- An extension to determinizing Streett automata (determinization construction given by Safra).
- Lower bounds for operations on tree automata.
- Considering parity automata as output.
- Considering state labeled automata as output.

Further Work

- An extension to determinizing Streett automata (determinization construction given by Safra).
- Lower bounds for operations on tree automata.
- Considering parity automata as output.
- Considering state labeled automata as output. But we truly

  believe that the right notion for a finite automata is being transition labeled.

Thank you.