Deterministic Pushdown Automata and Unary Languages

Giovanni Pighizzini

Dipartimento di Informatica e Comunicazione
Università degli Studi di Milano
ITALY

CIAA 2008

Outline of the talk

- Context-free grammars and pda's vs regular languages: some descriptional complexity results
- Exponential simulation of unary dpda's by dfa's
- Optimality of the simulation

Outline of the talk

- Context-free grammars and pda's vs regular languages: some descriptional complexity results
- Exponential simulation of unary dpda's by dfa's
- Optimality of the simulation
- Simulation of unary dfa's by dpda's

Outline of the talk

- Context-free grammars and pda's vs regular languages: some descriptional complexity results
- Exponential simulation of unary dpda's by dfa's
- Optimality of the simulation
- Simulation of unary dfa's by dpda's
- Unary dpda's vs context-free grammars

Outline of the talk

- Context-free grammars and pda's vs regular languages: some descriptional complexity results
- Exponential simulation of unary dpda's by dfa's
- Optimality of the simulation
- Simulation of unary dfa's by dpda's
- Unary dpda's vs context-free grammars

Outline of the talk

- Context-free grammars and pda's vs regular languages: some descriptional complexity results
- Exponential simulation of unary dpda's by dfa's
- Optimality of the simulation
- Simulation of unary dfa's by dpda's
- Unary dpda's vs context-free grammars

Outline of the talk

- Context-free grammars and pda's vs regular languages: some descriptional complexity results
- Exponential simulation of unary dpda's by dfa's
- Optimality of the simulation
- Simulation of unary dfa's by dpda's
- Unary dpda's vs context-free grammars

Context-free vs regular: descriptional complexity

Given a context-free grammar (or a pushdown automaton) of size n, generating a regular language, how much is big an equivalent finite automaton, wrt n ?

Theorem ([Meyer and Fischer, 1971])

For any recursive function f and arbitrarily large integers n, there exists a cfg G of size n generating a regular language L, s.t. any dfa accepting L must have at least $f(n)$ states.

As a consequence, the trade-off between context-grammars and finite automata is not recursive.

However... The witness language L is defined over a binary alphabet

Context-free vs regular: descriptional complexity

Given a context-free grammar (or a pushdown automaton) of size *n*, generating a regular language, how much is big an equivalent finite automaton, wrt *n* ?

Theorem ([Meyer and Fischer, 1971])

For any recursive function f and arbitrarily large integers n, there exists a cfg G of size n generating a regular language L, s.t. any dfa accepting L must have at least $f(n)$ states.

As a consequence, the trade-off between context-grammars and finite automata is not recursive.

However... The witness language *L* is defined over a binary alphabet.

What about languages over a one-letter alphabet?

Context-free vs regular: descriptional complexity

Given a context-free grammar (or a pushdown automaton) of size n, generating a regular language, how much is big an equivalent finite automaton, wrt n ?

Theorem ([Meyer and Fischer, 1971])

For any recursive function f and arbitrarily large integers n, there exists a cfg G of size n generating a regular language L, s.t. any dfa accepting L must have at least $f(n)$ states.

As a consequence, the trade-off between context-grammars and finite automata is not recursive.
However... The witness language L is defined over a binary alphabet.

What about languages over a one letter alphabet?

Context-free vs regular: descriptional complexity

Given a context-free grammar (or a pushdown automaton) of size n, generating a regular language, how much is big an equivalent finite automaton, wrt n ?

Theorem ([Meyer and Fischer, 1971])

For any recursive function f and arbitrarily large integers n, there exists a cfg G of size n generating a regular language L, s.t. any dfa accepting L must have at least $f(n)$ states.

As a consequence, the trade-off between context-grammars and finite automata is not recursive.
However... The witness language L is defined over a binary alphabet.

What about languages over a one letter alphabet?

Context-free vs regular: descriptional complexity

Given a context-free grammar (or a pushdown automaton) of size n, generating a regular language, how much is big an equivalent finite automaton, wrt n ?

Theorem ([Meyer and Fischer, 1971])

For any recursive function f and arbitrarily large integers n, there exists a cfg G of size n generating a regular language L, s.t. any dfa accepting L must have at least $f(n)$ states.

As a consequence, the trade-off between context-grammars and finite automata is not recursive.
However... The witness language L is defined over a binary alphabet.

What about languages over a one letter alphabet?

$\Sigma = \{a\}$

Theorem ([Ginsurg and Rice, 1962])

Every unary context-free language is regular.

Hence the classes of unary regular languages and unary *context–free* languages coincide!

Problem

Study the equivalence between unary context-free and regular languages from the descriptional complexity point of view.

Unary languages

$\Sigma = \{a\}$

Theorem ([Ginsurg and Rice, 1962])

Every unary context-free language is regular.

Hence the classes of unary regular languages and unary *context–free* languages coincide!

Problem

Study the equivalence between unary context-free and regular languages from the descriptional complexity point of view.

Context-free vs regular

Unary case [Pighizzini, Shallit, Wang, 2002]

Theorem

For any cfg in Chomsky normal form with h variables, generating a unary language, there exists an equivalent dfa with 2^{h^2} states. Furthermore, this bound is tight.

Corollary

Each unary pda with n states and m stack symbols, s.t. each push adds exactly one symbol, can be simulated by a dfa with $2^{O(n^4 m^2)}$ states.

What about the deterministic case?

Context-free vs regular

Unary case [Pighizzini, Shallit, Wang, 2002]

Theorem

For any cfg in Chomsky normal form with h variables, generating a unary language, there exists an equivalent dfa with 2^{h^2} states. Furthermore, this bound is tight.

Corollary

Each unary pda with n states and m stack symbols, s.t. each push adds exactly one symbol, can be simulated by a dfa with $2^{O(n^4 m^2)}$ states.

What about the deterministic case?

Context-free vs regular

Unary case [Pighizzini, Shallit, Wang, 2002]

Theorem

For any cfg in Chomsky normal form with h variables, generating a unary language, there exists an equivalent dfa with 2^{h^2} states. Furthermore, this bound is tight.

Corollary

Each unary pda with n states and m stack symbols, s.t. each push adds exactly one symbol, can be simulated by a dfa with $2^{O(n^4 m^2)}$ states.

What about the deterministic case?

Dpda's vs finite automata (general case)

- Each dpda of size s accepting a regular language can be simulated by a dfa with $2^{2^{2^s}}$ states. [Stearns, 1967]
- This upper bound was reduced to 2^{2^s} in [Valiant, 1975].
- It cannot be further reduced because a matching lower bound [Meyer and Fischer, 1971].
- However, in the *unary* case, it can be reduced to 2^s (tight bound).

Dpda's vs finite automata (general case)

- Each dpda of size s accepting a regular language can be simulated by a dfa with $2^{2^{2^s}}$ states. [Stearns, 1967]
- This upper bound was reduced to 2^{2^s} in [Valiant, 1975].
- It cannot be further reduced because a matching lower bound [Meyer and Fischer, 1971].
- However, in the *unary* case, it can be reduced to 2^s [this work] (tight bound).

Dpda's vs finite automata (general case)

- Each dpda of size s accepting a regular language can be simulated by a dfa with $2^{2^{2^s}}$ states. [Stearns, 1967]
- This upper bound was reduced to 2^{2^s} in [Valiant, 1975].
- It cannot be further reduced because a matching lower bound [Meyer and Fischer, 1971].
- However, in the *unary* case, it can be reduced to 2^s [this work] (tight bound).

Dpda's vs finite automata (general case)

- Each dpda of size s accepting a regular language can be simulated by a dfa with $2^{2^{2^s}}$ states. [Stearns, 1967]
- This upper bound was reduced to 2^{2^s} in [Valiant, 1975].
- It cannot be further reduced because a matching lower bound [Meyer and Fischer, 1971].
- However, in the *unary* case, it can be reduced to 2^s [this work] (tight bound).

Size

- Size of a finite automaton:
  Number of its states
- Size of a pushdown automaton:
  Total number of symbols needed to write down its description.
  We have to keep into account:
  - the number of the states
  - the cardinality of the pushdown alphabet
  - the length of the strings that can be pushed in one move on the stack
  - the number of transitions

Size

- Size of a finite automaton:
  Number of its states
- Size of a pushdown automaton:
  Total number of symbols needed to write down its description.
  We have to keep into account:
  - the number of the states
  - the cardinality of the pushdown alphabet
  - the length of the strings that can be pushed in one move on the stack
  - the number of transitions

Size

- Size of a finite automaton:
  Number of its states
- Size of a pushdown automaton:
  Total number of symbols needed to write down its description.
  We have to keep into account:
  - the number of the states
  - the cardinality of the pushdown alphabet
  - the length of the strings that can be pushed in one move on the stack
  - the number of transitions

Normal form for pda's (some restrictions on the transitions)

- We can prove that each dpda of size s can be converted into an equivalent dpda in normal form such that the product of
  - the number of states
  - the cardinality of the pushdown alphabet

  is $O(s)$.

Hence, we can restrict our attention to:

- dpda's in normal form with
- size = # states × # pushdown alphabet

Size of dpda's

Normal form for pda's (some restrictions on the transitions)

- We can prove that each dpda of size s can be converted into an equivalent dpda in normal form such that the product of
  - the number of states
  - the cardinality of the pushdown alphabet

  is $O(s)$.

Hence, we can restrict our attention to:

- dpda's in normal form with
- size = # states × # pushdown alphabet

Size of dpda's

Normal form for pda's (some restrictions on the transitions)

- We can prove that each dpda of size s can be converted into an equivalent dpda in normal form such that the product of
  - the number of states
  - the cardinality of the pushdown alphabet

  is $O(s)$.

Hence, we can restrict our attention to:

- dpda's in normal form with
- size = $\#$ states $\times$ $\#$ pushdown alphabet

Unary dfa's

Input alphabet $\Sigma = \{a\}$

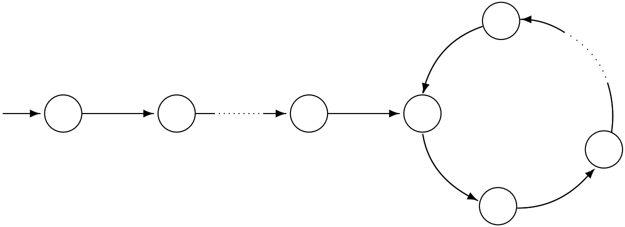
Theorem

$L \subseteq \{a\}^*$ is regular iff $\exists \mu \geq 0, \lambda \geq 1$ *s.t.*

$$\forall n \geq \mu : a^n \in L \text{ iff } a^{n+\lambda} \in L.$$

Unary dfa's

Input alphabet $\Sigma = \{a\}$



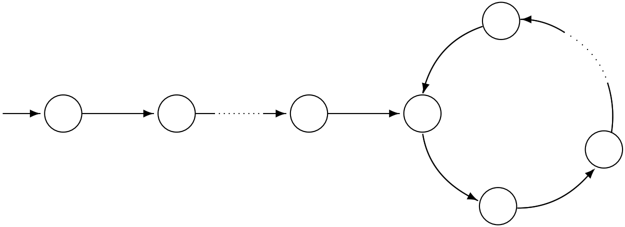
Theorem

$L \subseteq \{a\}^*$ is regular iff $\exists \mu \geq 0, \lambda \geq 1$ s.t.

$$\forall n \geq \mu : a^n \in L \text{ iff } a^{n+\lambda} \in L.$$

Unary dfa's

Input alphabet $\Sigma = \{a\}$

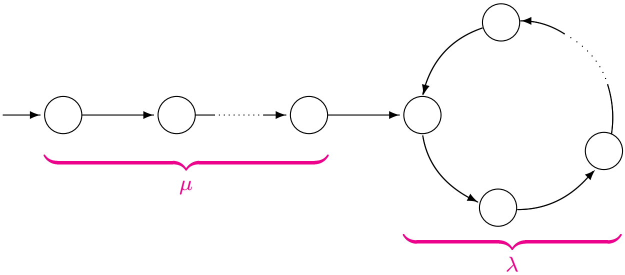


Theorem

$L \subseteq \{a\}^*$ is regular iff $\exists \mu \geq 0, \lambda \geq 1$ *s.t.*

$$\forall n \geq \mu : a^n \in L \text{ iff } a^{n+\lambda} \in L.$$

Input alphabet $\Sigma = \{a\}$



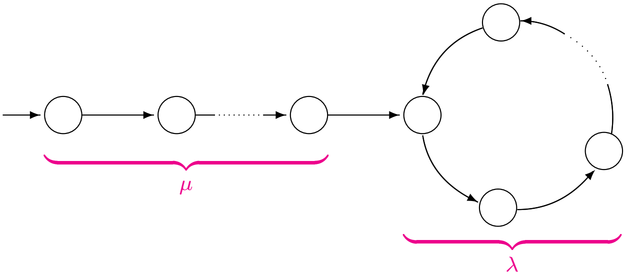
Theorem

$L \subseteq \{a\}^*$ is regular iff $\exists \mu \geq 0, \lambda \geq 1$ s.t.

$$\forall n \geq \mu : a^n \in L \text{ iff } a^{n+\lambda} \in L.$$

Unary dfa's

Input alphabet $\Sigma = \{a\}$



Theorem

$L \subseteq \{a\}^*$ is regular iff $\exists \mu \geq 0, \lambda \geq 1$ s.t.

$$\forall n \geq \mu : a^n \in L \text{ iff } a^{n+\lambda} \in L.$$

Unary automata

The costs of the optimal simulations between automata are different in the unary and in the general case!

Costs in the unary case:

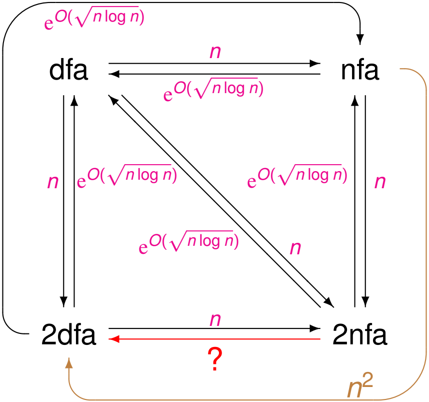
[Chrobak 1986, Mereghetti and Pighizzini 2001]

Unary automata

The costs of the optimal simulations between automata are different in the unary and in the general case!

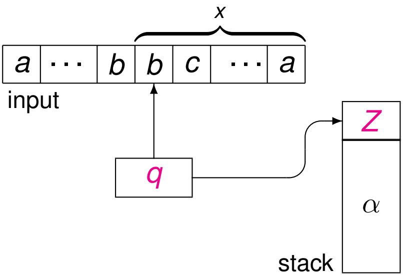
Costs in the unary case:

[Chrobak 1986, Mereghetti and Pighizzini 2001]



Pushdown automata

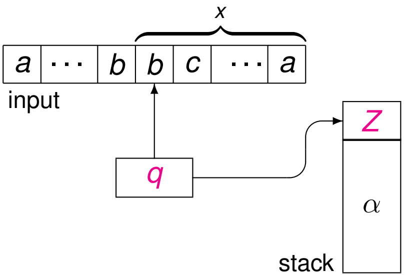
$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$



- $(q, x, Z\alpha)$ configuration
- $[qZ]$ mode

Pushdown automata

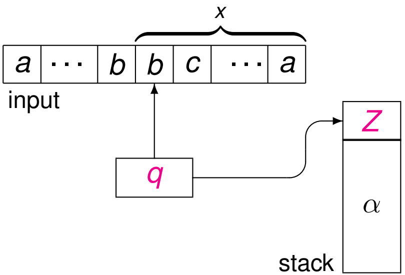
$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$



- $(q, x, Z\alpha)$ configuration
- $[qZ]$ mode

Pushdown automata

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$



- $(q, x, Z\alpha)$ configuration
- $[qZ]$ mode

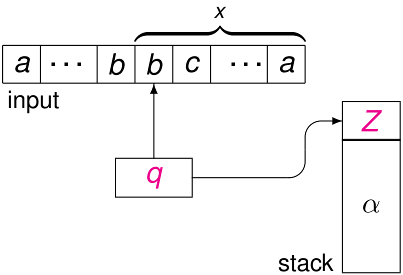
- M is *deterministic* iff $\forall q \in Q, Z \in \Gamma$:
  - if $\delta(q, \epsilon, Z) \neq \emptyset$ then $\delta(q, a, Z) = \emptyset$, for each $a \in \Sigma$
  - $\#\delta(q, \sigma, Z) \leq 1$, for each $\sigma \in \Sigma \cup \{\epsilon\}$.

- Deterministic cfl's: acceptance by *final states*

$$L(M) = \{x \in \Sigma^* \mid (q_0, x, Z_0) \overset{*}{\vdash} (q, \epsilon, \gamma), q \in F, \gamma \in \Gamma^*\}$$

Pushdown automata

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$



- $(q, x, Z\alpha)$ configuration
- $[qZ]$ mode

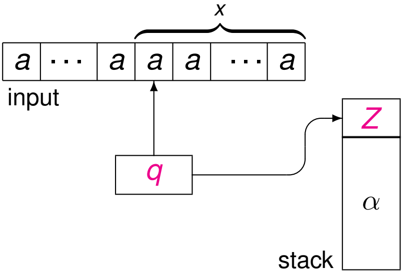
- M is *deterministic* iff $\forall q \in Q, Z \in \Gamma$:
  - if $\delta(q, \epsilon, Z) \neq \emptyset$ then $\delta(q, a, Z) = \emptyset$, for each $a \in \Sigma$
  - $\#\delta(q, \sigma, Z) \leq 1$, for each $\sigma \in \Sigma \cup \{\epsilon\}$.

- Deterministic cfl's: acceptance by *final states*

$$L(M) = \{x \in \Sigma^* \mid (q_0, x, Z_0) \vdash^{\star} (q, \epsilon, \gamma), q \in F, \gamma \in \Gamma^*\}$$

Pushdown automata

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$



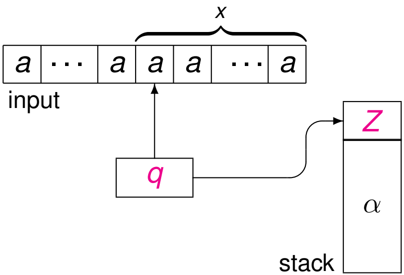
- $(q, x, Z\alpha)$ configuration
- $[qZ]$ mode

Unary deterministic pda's:

For each integer $t \geq 0$:

- if the computation does not stop before t steps then the configuration reach at the step t *does not depend* on the input length

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$



- $(q, x, Z\alpha)$ configuration
- $[qZ]$ mode

Unary deterministic pda's:

For each integer $t \geq 0$:

- if the computation does not stop before t steps then the configuration reach at the step t *does not depend* on the input length

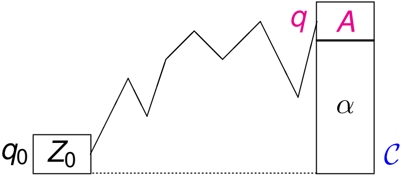
Modes

$[qA] \leq [pB]$ iff all the following conditions hold:

1. A configuration C with mode $[qA]$ is reachable from the initial configuration
2. A configuration with mode $[pB]$ is reachable from the configuration with mode $[qA]$ and pushdown store containing only A
3. If a configuration C' with mode $[pB]$ is reachable before C then

Modes

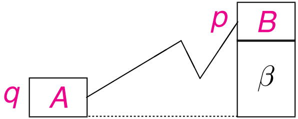
$[qA] \le [pB]$ iff all the following conditions hold:



1. A configuration $\mathcal{C}$ with mode $[qA]$ is reachable from the initial configuration
2. A configuration with mode $[pB]$ is reachable from the configuration with mode $[qA]$ and pushdown store containing only A
3. If a configuration $\mathcal{C}'$ with mode $[pB]$ is reachable before $\mathcal{C}$, then the stack height in some configuration between $\mathcal{C}'$ and $\mathcal{C}$ must be less than in $\mathcal{C}'$.

Modes

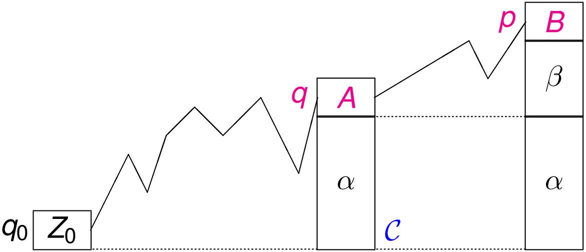
$[qA] \leq [pB]$ iff all the following conditions hold:



1. A configuration $\mathcal{C}$ with mode $[qA]$ is reachable from the initial configuration
2. A configuration with mode $[pB]$ is reachable from the configuration with mode $[qA]$ and pushdown store containing only A
3. If a configuration $\mathcal{C}'$ with mode $[pB]$ is reachable before $\mathcal{C}$, then the stack height in some configuration between $\mathcal{C}'$ and $\mathcal{C}$ must be less than in $\mathcal{C}'$.

Modes

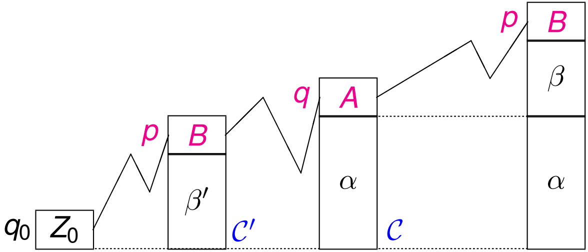
$[qA] \leq [pB]$ iff all the following conditions hold:



1. A configuration $\mathcal{C}$ with mode $[qA]$ is reachable from the initial configuration
2. A configuration with mode $[pB]$ is reachable from the configuration with mode $[qA]$ and pushdown store containing only A
3. If a configuration $\mathcal{C}'$ with mode $[pB]$ is reachable before $\mathcal{C}$, then the stack height in some configuration between $\mathcal{C}'$ and $\mathcal{C}$ must be less than in $\mathcal{C}'$.

Modes

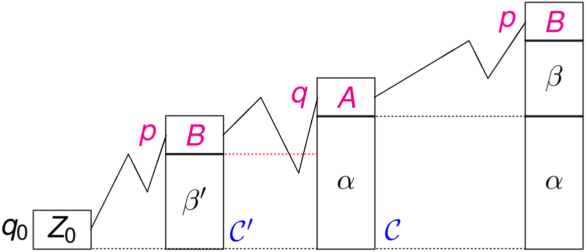
$[qA] \leq [pB]$ iff all the following conditions hold:



1. A configuration $\mathcal{C}$ with mode $[qA]$ is reachable from the initial configuration
2. A configuration with mode $[pB]$ is reachable from the configuration with mode $[qA]$ and pushdown store containing only A
3. If a configuration $\mathcal{C}'$ with mode $[pB]$ is reachable before $\mathcal{C}$, then the stack height in some configuration between $\mathcal{C}'$ and $\mathcal{C}$ must be less than in $\mathcal{C}'$.

Modes

$[qA] \leq [pB]$ iff all the following conditions hold:



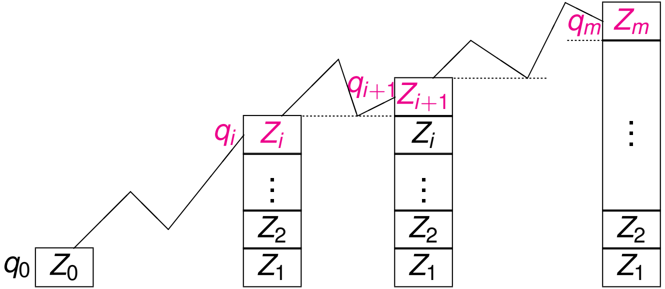
1. A configuration $\mathcal{C}$ with mode $[qA]$ is reachable from the initial configuration
2. A configuration with mode $[pB]$ is reachable from the configuration with mode $[qA]$ and pushdown store containing only A
3. If a configuration $\mathcal{C}'$ with mode $[pB]$ is reachable before $\mathcal{C}$, then the stack height in some configuration between $\mathcal{C}'$ and $\mathcal{C}$ must be less than in $\mathcal{C}'$.

Modes

Lemma

The relation $\leq$ defines a partial order on the set of the modes.

Stack content + state information



History h_t at the time t:
sequence of modes $[q_m Z_m] \ldots [q_1 Z_1]$ s.t.:

- $Z_m \ldots Z_1$ is the stack content after t computation steps
- for $i = 1, \ldots, m$, $[q_i Z_i]$ is the mode of the last visited configuration with stack height i

Histories and modes

For each step $t \geq 0$ we consider:

- h_t history
- m_t mode (leftmost element of h_t)

For the given dpda M we consider:

- $H = \{h_t \mid t \geq 0\}$, the set all *reachable histories*
- $(m_t)_{t \geq 0}$, the sequence of *reached modes*

Two possibilities:

1. Every history belonging to H does not contain a repeated mode
2. At least one history belonging to H contains a repetition

Histories and modes

For each step $t \geq 0$ we consider:

- h_t history
- m_t mode (leftmost element of h_t)

For the given dpda M we consider:

- $H = \{h_t \mid t \geq 0\}$, the set all *reachable histories*
- $(m_t)_{t \geq 0}$, the sequence of *reached modes*

Two possibilities:

1. Every history belonging to H does not contain a repeated mode
2. At least one history belonging to H contains a repetition

Histories and modes

For each step $t \geq 0$ we consider:

- h_t history
- m_t mode (leftmost element of h_t)

For the given dpda M we consider:

- $H = \{h_t \mid t \geq 0\}$, the set all *reachable histories*
- $(m_t)_{t \geq 0}$, the sequence of *reached modes*

Two possibilities:

1. Every history belonging to H does not contain a repeated mode
2. At least one history belonging to H contains a repetition

Case 1: Every history belonging to H does not contain a repeated mode

- H is finite
- The given dpda can be simulated by a deterministic automaton A whose set of states is H
- The number of the states of A is bounded by the number of histories without repetitions
- If an history $[q_m Z_m] \ldots [q_1 Z_1]$ does not contain any repetition, then $[q_1 Z_1] \leq [q_2 Z_2] \leq \ldots \leq [q_m Z_m]$
- Hence:

the number of states of the deterministic automaton A is bounded by $2^{\#Q \cdot \#\Gamma}$

Case 1: Every history belonging to H does not contain a repeated mode

- H is finite
- The given dpda can be simulated by a deterministic automaton A whose set of states is H
- The number of the states of A is bounded by the number of histories without repetitions
- If an history $[q_m Z_m] \ldots [q_1 Z_1]$ does not contain any repetition, then $[q_1 Z_1] \leq [q_2 Z_2] \leq \ldots \leq [q_m Z_m]$
- Hence:

the number of states of the deterministic automaton A is bounded by $2^{\#Q \cdot \#\Gamma}$

Case 1: Every history belonging to H does not contain a repeated mode

- H is finite
- The given dpda can be simulated by a deterministic automaton A whose set of states is H
- The number of the states of A is bounded by the number of histories without repetitions
- If an history $[q_m Z_m] \ldots [q_1 Z_1]$ does not contain any repetition, then $[q_1 Z_1] \le [q_2 Z_2] \le \ldots \le [q_m Z_m]$
- Hence:

the number of states of the deterministic automaton A is bounded by $2^{\#Q \cdot \#\Gamma}$

Case 1: Every history belonging to H does not contain a repeated mode

- H is finite
- The given dpda can be simulated by a deterministic automaton A whose set of states is H
- The number of the states of A is bounded by the number of histories without repetitions
- If an history $[q_m Z_m] \ldots [q_1 Z_1]$ does not contain any repetition, then $[q_1 Z_1] \leq [q_2 Z_2] \leq \ldots \leq [q_m Z_m]$
- Hence:

the number of states of the deterministic automaton A is bounded by $2^{\#Q \cdot \#\Gamma}$

Case 1: Every history belonging to H does not contain a repeated mode

- H is finite
- The given dpda can be simulated by a deterministic automaton A whose set of states is H
- The number of the states of A is bounded by the number of histories without repetitions
- If an history $[q_m Z_m] \ldots [q_1 Z_1]$ does not contain any repetition, then $[q_1 Z_1] \leq [q_2 Z_2] \leq \ldots \leq [q_m Z_m]$
- Hence:

the number of states of the deterministic automaton A is bounded by $2^{\#Q \cdot \#\Gamma}$

Case 1: Every history belonging to H does not contain a repeated mode

- H is finite
- The given dpda can be simulated by a deterministic automaton A whose set of states is H
- The number of the states of A is bounded by the number of histories without repetitions
- If an history $[q_m Z_m] \ldots [q_1 Z_1]$ does not contain any repetition, then $[q_1 Z_1] \leq [q_2 Z_2] \leq \ldots \leq [q_m Z_m]$
- Hence:

  the number of states of the deterministic automaton A is bounded by $2^{\#Q \cdot \#\Gamma}$

Case 2: At least one history in H contains a repetition

- The histories in H grow in a periodic way, i.e.:
  $\exists \mu \geq 0, \lambda \geq 1$, $\exists$ sequences of modes $\tilde{h}_0, \tilde{h}_1, \ldots, \tilde{h}_\lambda$ s.t. for $t \geq \mu$, the history at the step t is:

$$h_t = \tilde{h}_{t \bmod \lambda} (\tilde{h}_\lambda)^{\lfloor \frac{t-\mu}{\lambda} \rfloor} h_\mu$$

- The sequence $(m_t)_{t \geq 0}$ is ultimately periodic (period λ, from $t \geq \mu$)
- The language can be accepted by a deterministic automaton A with at most $\lambda + \mu$ states
- $\lambda + \mu \leq 2^{\#Q \cdot \#\Gamma}$
- Hence:

the given dpda can be simulated by a deterministic automaton A with at most $2^{\#Q \cdot \#\Gamma}$ states

Histories

Case 2: At least one history in H contains a repetition

- The histories in H grow in a periodic way, i.e.:
  $\exists \mu \geq 0, \lambda \geq 1$, $\exists$ sequences of modes $\tilde{h}_0, \tilde{h}_1, \ldots, \tilde{h}_\lambda$ s.t. for $t \geq \mu$, the history at the step t is:

$$h_t = \tilde{h}_{t \,\mathrm{MOD}\, \lambda} (\tilde{h}_\lambda)^{\lfloor \frac{t-\mu}{\lambda} \rfloor} h_\mu$$

- The sequence $(m_t)_{t \geq 0}$ is ultimately periodic (period λ, from $t \geq \mu$)
- The language can be accepted by a deterministic automaton A with at most $\lambda + \mu$ states
- $\lambda + \mu \leq 2^{\#Q \cdot \#\Gamma}$
- Hence:

the given dpda can be simulated by a deterministic automaton A with at most $2^{\#Q \cdot \#\Gamma}$ states

Histories

Case 2: At least one history in H contains a repetition

- The histories in H grow in a periodic way, i.e.:
  $\exists \mu \geq 0, \lambda \geq 1$, $\exists$ sequences of modes $\tilde{h}_0, \tilde{h}_1, \ldots, \tilde{h}_\lambda$ s.t. for $t \geq \mu$, the history at the step t is:

$$h_t = \tilde{h}_{t \,\mathrm{MOD}\, \lambda} (\tilde{h}_\lambda)^{\lfloor \frac{t-\mu}{\lambda} \rfloor} h_\mu$$

- The sequence $(m_t)_{t \geq 0}$ is ultimately periodic (period λ, from $t \geq \mu$)
- The language can be accepted by a deterministic automaton A with at most $\lambda + \mu$ states
- $\lambda + \mu \leq 2^{\#Q \cdot \#\Gamma}$
- Hence:

the given dpda can be simulated by a deterministic automaton A with at most $2^{\#Q \cdot \#\Gamma}$ states

Case 2: At least one history in H contains a repetition

- The histories in H grow in a periodic way, i.e.:
  $\exists \mu \geq 0, \lambda \geq 1$, $\exists$ sequences of modes $\tilde{h}_0, \tilde{h}_1, \ldots, \tilde{h}_\lambda$ s.t. for $t \geq \mu$, the history at the step t is:

$$h_t = \tilde{h}_{t \,\mathrm{MOD}\, \lambda} (\tilde{h}_\lambda)^{\lfloor \frac{t-\mu}{\lambda} \rfloor} h_\mu$$

- The sequence $(m_t)_{t \geq 0}$ is ultimately periodic (period λ, from $t \geq \mu$)
- The language can be accepted by a deterministic automaton A with at most $\lambda + \mu$ states
- $\lambda + \mu \leq 2^{\#Q \cdot \#\Gamma}$
- Hence:

the given dpda can be simulated by a deterministic automaton A with at most $2^{\#Q \cdot \#\Gamma}$ states

Case 2: At least one history in H contains a repetition

- The histories in H grow in a periodic way, i.e.: $\exists \mu \geq 0, \lambda \geq 1$, $\exists$ sequences of modes $\tilde{h}_0, \tilde{h}_1, \ldots, \tilde{h}_\lambda$ s.t. for $t \geq \mu$, the history at the step t is:

$$h_t = \tilde{h}_{t \text{ MOD } \lambda} (\tilde{h}_\lambda)^{\lfloor \frac{t-\mu}{\lambda} \rfloor} h_\mu$$

- The sequence $(m_t)_{t \geq 0}$ is ultimately periodic (period λ, from $t \geq \mu$)
- The language can be accepted by a deterministic automaton A with at most $\lambda + \mu$ states
- $\lambda + \mu \leq 2^{\#Q \cdot \#\Gamma}$
- Hence:

the given dpda can be simulated by a deterministic automaton A with at most $2^{\#Q \cdot \#\Gamma}$ states

Histories

Case 2: At least one history in H contains a repetition

- The histories in H grow in a periodic way, i.e.: $\exists \mu \geq 0, \lambda \geq 1$, $\exists$ sequences of modes $\tilde{h}_0, \tilde{h}_1, \ldots, \tilde{h}_\lambda$ s.t. for $t \geq \mu$, the history at the step t is:

$$h_t = \tilde{h}_{t \,\mathrm{MOD}\, \lambda} (\tilde{h}_\lambda)^{\lfloor \frac{t-\mu}{\lambda} \rfloor} h_\mu$$

- The sequence $(m_t)_{t \geq 0}$ is ultimately periodic (period λ, from $t \geq \mu$)
- The language can be accepted by a deterministic automaton A with at most $\lambda + \mu$ states
- $\lambda + \mu \leq 2^{\#Q \cdot \#\Gamma}$
- Hence:

the given dpda can be simulated by a deterministic automaton A with at most $2^{\#Q \cdot \#\Gamma}$ states

Histories

Case 2: At least one history in H contains a repetition

- The histories in H grow in a periodic way, i.e.: $\exists \mu \geq 0, \lambda \geq 1$, $\exists$ sequences of modes $\tilde{h}_0, \tilde{h}_1, \ldots, \tilde{h}_\lambda$ s.t. for $t \geq \mu$, the history at the step t is:

$$h_t = \tilde{h}_{t \bmod \lambda} (\tilde{h}_\lambda)^{\lfloor \frac{t-\mu}{\lambda} \rfloor} h_\mu$$

- The sequence $(m_t)_{t \geq 0}$ is ultimately periodic (period λ, from $t \geq \mu$)
- The language can be accepted by a deterministic automaton A with at most $\lambda + \mu$ states
- $\lambda + \mu \leq 2^{\#Q \cdot \#\Gamma}$
- Hence:

the given dpda can be simulated by a deterministic automaton A with at most $2^{\#Q \cdot \#\Gamma}$ states

Unary dpda's vs dfa's

As a consequence:

Theorem

Each unary dpda of size s can be simulated by a dfa with $2^{O(s)}$ states.

What about the optimality of this simulation?

As a consequence:

Theorem

Each unary dpda of size s can be simulated by a dfa with $2^{O(s)}$ states.

What about the optimality of this simulation?

Unary dpda's vs dfa's: lower bound

Given $s > 0$ consider $L_s = (a^{2^s})^*$.

We can prove that:

- There exists a dpda of size $8s + 4$ accepting L_s.
- Each dfa accepting L_s must have at least 2^s states.

Hence our simulation is optimal!

Problem: Does it is possible to reduce the cost of the simulation of unary dpda's, by using *nondeterministic* or *two-way* finite automata?

Unary dpda's vs dfa's: lower bound

Given $s > 0$ consider $L_s = (a^{2^s})^*$.

We can prove that:

- There exists a dpda of size $8s + 4$ accepting L_s.
- Each dfa accepting L_s must have at least 2^s states.

Hence our simulation is optimal!

Problem: Does it is possible to reduce the cost of the simulation of unary dpda's, by using *nondeterministic* or *two-way* finite automata?

Unary dpda's vs dfa's: lower bound

Given $s > 0$ consider $L_s = (a^{2^s})^*$.

We can prove that:

- There exists a dpda of size $8s + 4$ accepting L_s.
- Each dfa accepting L_s must have at least 2^s states.

Hence our simulation is optimal!

Problem: Does it is possible to reduce the cost of the simulation of unary dpda's, by using *nondeterministic* or *two-way* finite automata?

Unary dpda's vs dfa's: lower bound

Given $s > 0$ consider $L_s = (a^{2^s})^*$.

We can prove that:

- There exists a dpda of size $8s + 4$ accepting L_s.
- Each dfa accepting L_s must have at least 2^s states.

Hence our simulation is optimal!

Problem: Does it is possible to reduce the cost of the simulation of unary dpda's, by using *nondeterministic* or *two-way* finite automata?

Unary dpda's vs dfa's: lower bound

Given $s > 0$ consider $L_s = (a^{2^s})^*$.

We can prove that:

- There exists a dpda of size $8s + 4$ accepting L_s.
- Each dfa accepting L_s must have at least 2^s states.

Hence our simulation is optimal!

Problem: Does it is possible to reduce the cost of the simulation of unary dpda's, by using *nondeterministic* or *two-way* finite automata?

Unary dpda's vs 2nfa's

- We consider again $L_s = (a^{2^s})^*$, $s > 0$
- L_s is accepted by a dpda of size $8s + 4$
- Furthermore, even each two-way nondeterministic automaton accepting L_s needs 2^s states [Mereghetti, Pighizzini, 2000]

Hence:

Even the cost of the optimal simulation of unary dpda's by 2nfa's is exponential!

Unary dpda's vs 2nfa's

- We consider again $L_s = (a^{2^s})^*$, $s > 0$
- L_s is accepted by a dpda of size $8s + 4$
- Furthermore, even each two-way nondeterministic automaton accepting L_s needs 2^s states [Mereghetti, Pighizzini, 2000]

Hence:

Even the cost of the optimal simulation of unary dpda's by 2nfa's is exponential!

Unary dpda's vs 2nfa's

- We consider again $L_s = (a^{2^s})^*$, $s > 0$
- L_s is accepted by a dpda of size $8s + 4$
- Furthermore, even each two-way nondeterministic automaton accepting L_s needs 2^s states [Mereghetti, Pighizzini, 2000]

Hence:

Even the cost of the optimal simulation of unary dpda's by 2nfa's is exponential!

Unary dpda's vs 2nfa's

- We consider again $L_s = (a^{2^s})^*$, $s > 0$
- L_s is accepted by a dpda of size $8s + 4$
- Furthermore, even each two-way nondeterministic automaton accepting L_s needs 2^s states [Mereghetti, Pighizzini, 2000]

Hence:

Even the cost of the optimal simulation of unary dpda's by 2nfa's is exponential!

Unary dpda's vs 2nfa's

- We consider again $L_s = (a^{2^s})^*$, $s > 0$
- L_s is accepted by a dpda of size $8s + 4$
- Furthermore, even each two-way nondeterministic automaton accepting L_s needs 2^s states [Mereghetti, Pighizzini, 2000]

Hence:

Even the cost of the optimal simulation of unary dpda's by 2nfa's is exponential!

Languages with “complex” dpda’s

Unary dpda’s can be exponentially more succinct than dfa’s.
Does this is true for *each* unary regular language?

Problem

For $m \geq 0$, let $L_m \subseteq a^*$ be a language accepted by a dfa with 2^m states.
Does there exists an equivalent dpda with $O(m)$ states?

The answer to this question is negative:

For each $m > 0$ there exists a language $L_m \subseteq a^*$ s.t.:

- L_m is accepted by a dfa with 2^m states.
- The size of any dpda accepting L_m is at least $d\frac{2^m}{m^2}$, for a constant d.

Unary dpda’s can be exponentially more succinct than dfa’s.
Does this is true for *each* unary regular language?

Problem

For $m \geq 0$, let $L_m \subseteq a^*$ be a language accepted by a dfa with 2^m states.
Does there exists an equivalent dpda with $O(m)$ states?

The answer to this question is negative:

For each $m > 0$ there exists a language $L_m \subseteq a^*$ s.t.:

- L_m is accepted by a dfa with 2^m states.
- The size of any dpda accepting L_m is at least $d\frac{2^m}{m^2}$, for a constant d.

Unary dpda’s can be exponentially more succinct than dfa’s.
Does this is true for *each* unary regular language?

Problem

For $m \geq 0$, let $L_m \subseteq a^*$ be a language accepted by a dfa with 2^m states.
Does there exists an equivalent dpda with $O(m)$ states?

The answer to this question is negative:

For each $m > 0$ there exists a language $L_m \subseteq a^*$ s.t.:

- L_m is accepted by a dfa with 2^m states.
- The size of any dpda accepting L_m is at least $d\frac{2^m}{m^2}$, for a constant d.

Languages with “complex” dpda’s

w_m **de Bruijn word** of order m on $\{0, 1\}$:

- $|w_m| = 2^m + m - 1$
- each string of length m is a factor of w_m, occurring in w_m exactly one time
- the suffix and the prefix of length $m - 1$ of w_m coincide.

Example: $w_3 = 0001011100$

$L_m = \{a^k \mid \text{the letter of } w_m \text{ in position } k \text{ MOD}' 2^m \text{ is } 1\}$,
where $x \text{ MOD}' y = x \text{ MOD } y$, if $x \text{ MOD } y > 0$, y otherwise

Example: $L_3 = \{a^0, a^4, a^6, a^7\}\{a^8\}^*$.

L_m is accepted by a dfa with 2^m states

Languages with “complex” dpda’s

w_m de Bruijn word of order m on $\{0, 1\}$:

- $|w_m| = 2^m + m - 1$
- each string of length m is a factor of w_m, occurring in w_m exactly one time
- the suffix and the prefix of length $m - 1$ of w_m coincide.

Example: $w_3 = 0001011100$

$L_m = \{a^k \mid$ the letter of w_m in position $k \text{ MOD}' 2^m$ is $1\}$,
where $x \text{ MOD}' y = x \text{ MOD } y$, if $x \text{ MOD } y > 0$, y otherwise

Example: $L_3 = \{a^0, a^4, a^6, a^7\}\{a^8\}^*$.

L_m is accepted by a dfa with 2^m states

Languages with “complex” dpda’s

- M: a dpda of size s accepting L_m
- M': M extended with an output tape to generate the de Bruijn word
- A: a dfa with $m+1$ states, input alphabet $\{0,1\}$, ending and accepting when the last m input symbols coincide with the suffix of length m of w_m
- M'': a dpda of size $O(ms)$, composition of M' and A, accepting $\{a^{2^m+m-1}\}$
- G: cfg grammar of size $O(ms)$, obtained from M'', generating $\{w_m\}$

Lemma ([Domaratzki, Pighizzini, Shallit, 2002])

The size of each grammar G generating $\{w_m\}$ must be at least $c\frac{2^m}{m}$, for some constant c.

Languages with “complex” dpda’s

- M: a dpda of size s accepting L_m
- M': M extended with an output tape to generate the de Bruijn word
- A: a dfa with $m+1$ states, input alphabet $\{0,1\}$, ending and accepting when the last m input symbols coincide with the suffix of length m of w_m
- M'': a dpda of size $O(ms)$, composition of M' and A, accepting $\{a^{2^m+m-1}\}$
- G: cfg grammar of size $O(ms)$, obtained from M'', generating $\{w_m\}$

Lemma ([Domaratzki, Pighizzini, Shallit, 2002])

The size of each grammar G generating $\{w_m\}$ must be at least $c\frac{2^m}{m}$, for some constant c.

Hence $s \geq d\frac{2^m}{m^2}$, for some $d > 0$.

Languages with “complex” dpda’s

- *M*: a dpda of size s accepting L_m
- *M′*: M extended with an output tape to generate the de Bruijn word
- *A*: a dfa with $m+1$ states, input alphabet $\{0,1\}$, ending and accepting when the last m input symbols coincide with the suffix of length m of w_m
- *M″*: a dpda of size $O(ms)$, composition of M' and A, accepting $\{a^{2^m+m-1}\}$
- *G*: cfg grammar of size $O(ms)$, obtained from M'', generating $\{w_m\}$

Lemma ([Domaratzki, Pighizzini, Shallit, 2002])

The size of each grammar G generating $\{w_m\}$ must be at least $c\frac{2^m}{m}$, for some constant c.

Hence $s \geq d\frac{2^m}{m^2}$, for some $d > 0$

Languages with “complex” dpda’s

- M: a dpda of size s accepting L_m
- M': M extended with an output tape to generate the de Bruijn word
- A: a dfa with $m+1$ states, input alphabet $\{0,1\}$, ending and accepting when the last m input symbols coincide with the suffix of length m of w_m
- M'': a dpda of size $O(ms)$, composition of M' and A, accepting $\{a^{2^m+m-1}\}$
- G: cfg grammar of size $O(ms)$, obtained from M'', generating $\{w_m\}$

Lemma ([Domaratzki, Pighizzini, Shallit, 2002])

The size of each grammar G generating $\{w_m\}$ must be at least $c\frac{2^m}{m}$, for some constant c.

Hence $s \geq d\frac{2^m}{m^2}$, for some $d > 0$.

Languages with “complex” dpda’s

- *M*: a dpda of size s accepting L_m
- *M′*: M extended with an output tape to generate the de Bruijn word
- *A*: a dfa with $m+1$ states, input alphabet $\{0,1\}$, ending and accepting when the last m input symbols coincide with the suffix of length m of w_m
- *M″*: a dpda of size $O(ms)$, composition of M' and A, accepting $\{a^{2^m+m-1}\}$
- *G*: cfg grammar of size $O(ms)$, obtained from M'', generating $\{w_m\}$

Lemma ([Domaratzki, Pighizzini, Shallit, 2002])

The size of each grammar G generating $\{w_m\}$ must be at least $c\frac{2^m}{m}$, for some constant c.

Hence $s \geq d^{\frac{2^m}{m^2}}$, for some $d > 0$.

Languages with "complex" dpda's

- M: a dpda of size s accepting L_m
- M': M extended with an output tape to generate the de Bruijn word
- A: a dfa with $m+1$ states, input alphabet $\{0,1\}$, ending and accepting when the last m input symbols coincide with the suffix of length m of w_m
- M'': a dpda of size $O(ms)$, composition of M' and A, accepting $\{a^{2^m+m-1}\}$
- G: cfg grammar of size $O(ms)$, obtained from M'', generating $\{w_m\}$

Lemma ([Domaratzki, Pighizzini, Shallit, 2002])

The size of each grammar G generating $\{w_m\}$ must be at least $c\frac{2^m}{m}$, for some constant c.

Hence $s \geq d\frac{2^m}{m^2}$, for some $d > 0$.

- M: a dpda of size s accepting L_m
- M': M extended with an output tape to generate the de Bruijn word
- A: a dfa with $m+1$ states, input alphabet $\{0,1\}$, ending and accepting when the last m input symbols coincide with the suffix of length m of w_m
- M'': a dpda of size $O(ms)$, composition of M' and A, accepting $\{a^{2^m+m-1}\}$
- G: cfg grammar of size $O(ms)$, obtained from M'', generating $\{w_m\}$

Lemma ([Domaratzki, Pighizzini, Shallit, 2002])

The size of each grammar G generating $\{w_m\}$ must be at least $c^{\frac{2^m}{m}}$, for some constant c.

Hence $s \geq d^{\frac{2^m}{m^2}}$, for some $d > 0$.

Simulation of unary dfa's by dpda's

As a consequence we get the following *lower bound*:

Corollary

There exists a constant $K > 0$ such that the conversion of unary n-state dfa's into equivalent dpda's produces dpda's of size at least $K\frac{n}{\log^2 n}$, for infinitely many n's.

Pda's vs cfg's

To prove the last result we have investigated the transformation of unary dpda's into context-free grammars.

- Each pda can be transformed into an equivalent cfg with $(\#Q)^2 \cdot \#\Gamma + 1$ variables.
- This number cannot be reduced, even if the given pda is deterministic [Goldstine, Price, Wotschke, 1982].

However, we proved that:

Theorem

Each unary dpda can be transformed into an equivalent cfg grammar with $\#Q \cdot \#\Gamma$ variables.

Pda's vs cfg's

To prove the last result we have investigated the transformation of unary dpda's into context-free grammars.

- Each pda can be transformed into an equivalent cfg with $(\#Q)^2 \cdot \#\Gamma + 1$ variables.
- This number cannot be reduced, even if the given pda is deterministic [Goldstine, Price, Wotschke, 1982].

However, we proved that:

Theorem

Each unary dpda can be transformed into an equivalent cfg grammar with $\#Q \cdot \#\Gamma$ variables.

Pda's vs cfg's

To prove the last result we have investigated the transformation of unary dpda's into context-free grammars.

- Each pda can be transformed into an equivalent cfg with $(\#Q)^2 \cdot \#\Gamma + 1$ variables.
- This number cannot be reduced, even if the given pda is deterministic [Goldstine, Price, Wotschke, 1982].

However, we proved that:

Theorem

Each unary dpda can be transformed into an equivalent cfg grammar with $\#Q \cdot \#\Gamma$ variables.

To prove the last result we have investigated the transformation of unary dpda's into context-free grammars.

- Each pda can be transformed into an equivalent cfg with $(\#Q)^2 \cdot \#\Gamma + 1$ variables.
- This number cannot be reduced, even if the given pda is deterministic [Goldstine, Price, Wotschke, 1982].

However, we proved that:

Theorem

Each unary dpda can be transformed into an equivalent cfg grammar with $\#Q \cdot \#\Gamma$ variables.

Bounded languages:

Subsets of $w_1^* w_2^* \ldots w_n^*$, for given words $w_1, \ldots, w_n$.

Extend the investigation to *bounded deterministic context-free* languages:

- Simulation of dpda's accepting *bounded regular* languages, by finite automata.
- Simulation of dpda's accepting *bounded (context-free)* languages, by finite-turn pushdown automata.

In the nondeterministic case we have the following:

Theorem ([Malcher, Pighizzini, 2007])

Each bounded context-free language generated by a cfg with h variables in Chomsky normal form is accepted by a finite-turn pda with 2^h and $O(1)$ stack symbols.

Bounded languages:
Subsets of $w_1^* w_2^* \ldots w_n^*$, for given words $w_1, \ldots, w_n$.

Extend the investigation to *bounded deterministic context-free* languages:

- Simulation of dpda's accepting *bounded regular* languages, by finite automata.
- Simulation of dpda's accepting *bounded (context-free)* languages, by finite-turn pushdown automata.

In the nondeterministic case we have the following:

Theorem ([Malcher, Pighizzini, 2007])

Each bounded context-free language generated by a cfg with h variables in Chomsky normal form is accepted by a finite-turn pda with 2^h and $O(1)$ stack symbols.

Related questions and results

Bounded languages:
Subsets of $w_1^* w_2^* \ldots w_n^*$, for given words $w_1, \ldots, w_n$.

Extend the investigation to *bounded deterministic context-free* languages:

- Simulation of dpda's accepting *bounded regular* languages, by finite automata.
- Simulation of dpda's accepting *bounded (context-free)* languages, by finite-turn pushdown automata.

In the nondeterministic case we have the following:

Theorem ([Malcher, Pighizzini, 2007])

Each bounded context-free language generated by a cfg with h variables in Chomsky normal form is accepted by a finite-turn pda with 2^h and $O(1)$ stack symbols.

Related questions and results

Bounded languages:
Subsets of $w_1^* w_2^* \ldots w_n^*$, for given words $w_1, \ldots, w_n$.

Extend the investigation to *bounded deterministic context-free* languages:

- Simulation of dpda's accepting *bounded regular* languages, by finite automata.
- Simulation of dpda's accepting *bounded (context-free)* languages, by finite-turn pushdown automata.

In the nondeterministic case we have the following:

Theorem ([Malcher, Pighizzini, 2007])

Each bounded context-free language generated by a cfg with h variables in Chomsky normal form is accepted by a finite-turn pda with 2^h and $O(1)$ stack symbols.

Related questions and results

Bounded languages:
Subsets of $w_1^* w_2^* \dots w_n^*$, for given words $w_1, \dots, w_n$.

Extend the investigation to *bounded deterministic context-free* languages:

- Simulation of dpda's accepting *bounded regular* languages, by finite automata.
- Simulation of dpda's accepting *bounded (context-free)* languages, by finite-turn pushdown automata.

In the nondeterministic case we have the following:

Theorem ([Malcher, Pighizzini, 2007])

Each bounded context-free language generated by a cfg with h variables in Chomsky normal form is accepted by a finite-turn pda with 2^h and $O(1)$ stack symbols.