Parikh's Theorem and Descriptional Complexity

Giovanna J. Lavado and Giovanni Pighizzini

Dipartimento di Informatica e Comunicazione
Università degli Studi di Milano

SOFSEM 2012
Špindlerův Mlýn, Czech Republic
January 21–27, 2012

UNIVERSITÀ DEGLI STUDI
DI MILANO

Parikh's Image

- $\Sigma = \{a_1, \ldots, a_m\}$ alphabet of m symbols
- *Parikh's map* $\psi : \Sigma^* \to \mathbb{N}^m$:

$$\psi(w) = (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_m})$$

  for each string $w \in \Sigma^*$
- w' and w'' are *Parikh equivalent* iff $\psi(w') = \psi(w'')$ (in symbols $w' =_\pi w''$)
- *Parikh's image* of a language $L \subseteq \Sigma^*$:

$$\psi(L) = \{\psi(w) \mid w \in L\}$$

- L' and L'' are *Parikh equivalent* iff $\psi(L') = \psi(L'')$ (in symbols $L' =_\pi L''$)

Parikh's Image

- $\Sigma = \{a_1, \ldots, a_m\}$ alphabet of m symbols
- *Parikh's map* $\psi : \Sigma^* \to \mathbb{N}^m$:

$$\psi(w) = (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_m})$$

for each string $w \in \Sigma^*$

- w' and w'' are *Parikh equivalent* iff $\psi(w') = \psi(w'')$ (in symbols $w' =_\pi w''$)
- *Parikh's image* of a language $L \subseteq \Sigma^*$:

$$\psi(L) = \{\psi(w) \mid w \in L\}$$

- L' and L'' are *Parikh equivalent* iff $\psi(L') = \psi(L'')$ (in symbols $L' =_\pi L''$)

Parikh's Image

- $\Sigma = \{a_1, \ldots, a_m\}$ alphabet of m symbols
- *Parikh's map* $\psi : \Sigma^* \to \mathbb{N}^m$:

$$\psi(w) = (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_m})$$

  for each string $w \in \Sigma^*$
- w' and w'' are *Parikh equivalent* iff $\psi(w') = \psi(w'')$ (in symbols $w' =_\pi w''$)
- *Parikh's image* of a language $L \subseteq \Sigma^*$:

$$\psi(L) = \{\psi(w) \mid w \in L\}$$

- L' and L'' are *Parikh equivalent* iff $\psi(L') = \psi(L'')$ (in symbols $L' =_\pi L''$)

Parikh's Image

- $\Sigma = \{a_1, \ldots, a_m\}$ alphabet of m symbols
- *Parikh's map* $\psi : \Sigma^* \to \mathbb{N}^m$:

$$\psi(w) = (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_m})$$

  for each string $w \in \Sigma^*$
- w' and w'' are *Parikh equivalent* iff $\psi(w') = \psi(w'')$ (in symbols $w' =_\pi w''$)
- *Parikh's image* of a language $L \subseteq \Sigma^*$:

$$\psi(L) = \{\psi(w) \mid w \in L\}$$

- L' and L'' are *Parikh equivalent* iff $\psi(L') = \psi(L'')$ (in symbols $L' =_\pi L''$)

Parikh's Image

- $\Sigma = \{a_1, \ldots, a_m\}$ alphabet of m symbols
- *Parikh's map* $\psi : \Sigma^* \to \mathbb{N}^m$:

$$\psi(w) = (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_m})$$

  for each string $w \in \Sigma^*$
- w' and w'' are *Parikh equivalent* iff $\psi(w') = \psi(w'')$ (in symbols $w' =_\pi w''$)
- *Parikh's image* of a language $L \subseteq \Sigma^*$:

$$\psi(L) = \{\psi(w) \mid w \in L\}$$

- L' and L'' are *Parikh equivalent* iff $\psi(L') = \psi(L'')$ (in symbols $L' =_\pi L''$)

Parikh's Theorem

Theorem ([Parikh '66])

The Parikh image of a context-free language is a semilinear set, i.e,
each context-free language is Parikh equivalent
to a regular language

Example:

- $L = \{a^n b^n \mid n \geq 0\}$
- $R = (ab)^*$

$$\psi(L) = \psi(R) = \{(n, n) \mid n \geq 0\}$$

Different proofs after the original one of Parikh, e.g.

- [Goldstine '77]: a simplified proof
- [Aceto&Ésik&Ingólfsdóttir '02]: an equational proof
- ...

Parikh's Theorem

Theorem ([Parikh '66])

The Parikh image of a context-free language is a semilinear set, i.e,
each context-free language is Parikh equivalent
to a regular language

Example:

- $L = \{a^n b^n \mid n \geq 0\}$
- $R = (ab)^*$

$$\psi(L) = \psi(R) = \{(n, n) \mid n \geq 0\}$$

Different proofs after the original one of Parikh, e.g.

- [Goldstine '77]: a simplified proof
- [Aceto&Ésik&Ingólfsdóttir '02]: an equational proof
- ...

Parikh's Theorem

Theorem ([Parikh '66])

The Parikh image of a context-free language is a semilinear set, i.e,
each context-free language is Parikh equivalent
to a regular language

Example:

- $L = \{a^n b^n \mid n \geq 0\}$
- $R = (ab)^*$

$$\psi(L) = \psi(R) = \{(n, n) \mid n \geq 0\}$$

Different proofs after the original one of Parikh, e.g.

- [Goldstine '77]: a simplified proof
- [Aceto&Ésik&Ingólfsdóttir '02]: an equational proof
- ...

Purpose of the Work

Recent works investigating *complexity aspects* of Parikh's Theorem:

- [Kopczyński&To '10]:
  size of the "semilinear descriptions" of Parikh images of languages defined by NFAs and by CFGs
- [Esparza&Ganty&Kiefer&Luttenberger '11]:
  - new proof of Parikh's Theorem
  - solution to the problem below in the case of nondeterministic automata

Problem

Given a CFG G compare the size of G
with the sizes of finite automata accepting languages
that are Parikh equivalent to $L(G)$

Our aim is to study the same problem for *deterministic automata*

Purpose of the Work

Recent works investigating *complexity aspects* of Parikh's Theorem:

- [Kopczyński&To '10]:
  size of the "semilinear descriptions" of Parikh images of languages defined by NFAs and by CFGs
- [Esparza&Ganty&Kiefer&Luttenberger '11]:
  - new proof of Parikh's Theorem
  - solution to the problem below in the case of nondeterministic automata

Problem

Given a CFG G compare the size of G
with the sizes of finite automata accepting languages
that are Parikh equivalent to $L(G)$

Our aim is to study the same problem for *deterministic automata*

Purpose of the Work

Recent works investigating *complexity aspects* of Parikh's Theorem:

- [Kopczyński&To '10]:
  size of the "semilinear descriptions" of Parikh images of languages defined by NFAs and by CFGs
- [Esparza&Ganty&Kiefer&Luttenberger '11]:
  - new proof of Parikh's Theorem
  - solution to the problem below in the case of nondeterministic automata

Problem

Given a CFG G compare the size of G
with the sizes of finite automata accepting languages
that are Parikh equivalent to $L(G)$

Our aim is to study the same problem for *deterministic automata*

Purpose of the Work

Recent works investigating *complexity aspects* of Parikh's Theorem:

- [Kopczyński&To '10]:
  size of the "semilinear descriptions" of Parikh images of languages defined by NFAs and by CFGs
- [Esparza&Ganty&Kiefer&Luttenberger '11]:
  - new proof of Parikh's Theorem
  - solution to the problem below in the case of nondeterministic automata

Problem

Given a CFG G compare the size of G
with the sizes of finite automata accepting languages
that are Parikh equivalent to $L(G)$

Our aim is to study the same problem for *deterministic automata*

Purpose of the Work

Recent works investigating *complexity aspects* of Parikh's Theorem:

- [Kopczyński&To '10]:
  size of the "semilinear descriptions" of Parikh images of languages defined by NFAs and by CFGs
- [Esparza&Ganty&Kiefer&Luttenberger '11]:
  - new proof of Parikh's Theorem
  - solution to the problem below in the case of nondeterministic automata

Problem

Given a CFG G compare the size of G
with the sizes of finite automata accepting languages
that are Parikh equivalent to $L(G)$

Our aim is to study the same problem for *deterministic automata*

Why this Problem?

- We came to this problem from the investigation of automata over *a one letter alphabet*
- Costs in states of optimal simulations between different variant unary automata (one-way/two-way, deterministic/nondeterministic) [Chrobak '86, Mereghetti&Pighizzini '01]
- Context-free languages over a unary terminal alphabet are regular [Ginsburg&Rice '62]
- The regularity of unary CFLs is also a corollary of Parikh's Theorem
- Hence, unary PDAs and unary CFGs can be transformed into finite automata

Why this Problem?

- We came to this problem from the investigation of automata over *a one letter alphabet*
- Costs in states of optimal simulations between different variant unary automata (one-way/two-way, deterministic/nondeterministic) [Chrobak '86, Mereghetti&Pighizzini '01]
- Context-free languages over a unary terminal alphabet are regular [Ginsburg&Rice '62]
- The regularity of unary CFLs is also a corollary of Parikh's Theorem
- Hence, unary PDAs and unary CFGs can be transformed into finite automata

Why this Problem?

- We came to this problem from the investigation of automata over *a one letter alphabet*
- Costs in states of optimal simulations between different variant unary automata (one-way/two-way, deterministic/nondeterministic) [Chrobak '86, Mereghetti&Pighizzini '01]
- Context-free languages over a unary terminal alphabet are regular [Ginsburg&Rice '62]
- The regularity of unary CFLs is also a corollary of Parikh's Theorem
- Hence, unary PDAs and unary CFGs can be transformed into finite automata

Why this Problem?

- We came to this problem from the investigation of automata over *a one letter alphabet*
- Costs in states of optimal simulations between different variant unary automata (one-way/two-way, deterministic/nondeterministic) [Chrobak '86, Mereghetti&Pighizzini '01]
- Context-free languages over a unary terminal alphabet are regular [Ginsburg&Rice '62]
- The regularity of unary CFLs is also a corollary of Parikh's Theorem
- Hence, unary PDAs and unary CFGs can be transformed into finite automata

Why this Problem?

- We came to this problem from the investigation of automata over *a one letter alphabet*
- Costs in states of optimal simulations between different variant unary automata (one-way/two-way, deterministic/nondeterministic) [Chrobak '86, Mereghetti&Pighizzini '01]
- Context-free languages over a unary terminal alphabet are regular [Ginsburg&Rice '62]
- The regularity of unary CFLs is also a corollary of Parikh's Theorem
- Hence, unary PDAs and unary CFGs can be transformed into finite automata

Size: Descriptional Complexity Measures

- Finite Automata
  number of states
- Context-Free Grammars
  number of variables after converting
  into *Chomsky Normal Form*
  [Gruska '73]

Size: Descriptional Complexity Measures

- Finite Automata
  number of states
- Context-Free Grammars
  number of variables after converting
  into *Chomsky Normal Form*
  [Gruska '73]

Unary Context-Free Languages

Theorem ([Pighizzini&Shallit&Wang '02])

For each unary CFG in Chomsky normal form with h variables there are

- *an equivalent NFA with at most $2^{2h-1}+1$ states*
- *an equivalent DFA with less than 2^{h^2} states*

Both bounds are tight

Can we extend this result to larger alphabets?

- The class of CLFs is larger than the class of regular: we cannot have a result of *exactly* the same form!
- However, we can ask about the number of states of DFAs or NFAs *Parikh equivalent* to the given grammar

Unary Context-Free Languages

Theorem ([Pighizzini&Shallit&Wang '02])

For each unary CFG in Chomsky normal form with h variables there are

- *an equivalent NFA with at most $2^{2h-1}+1$ states*
- *an equivalent DFA with less than 2^{h^2} states*

Both bounds are tight

Can we extend this result to larger alphabets?

- The class of CLFs is larger than the class of regular: we cannot have a result of *exactly* the same form!
- However, we can ask about the number of states of DFAs or NFAs *Parikh equivalent* to the given grammar

Unary Context-Free Languages

Theorem ([Pighizzini&Shallit&Wang '02])

For each unary CFG in Chomsky normal form with h variables there are

- *an equivalent NFA with at most $2^{2h-1}+1$ states*
- *an equivalent DFA with less than 2^{h^2} states*

Both bounds are tight

Can we extend this result to larger alphabets?

- The class of CLFs is larger than the class of regular: we cannot have a result of *exactly* the same form!
- However, we can ask about the number of states of DFAs or NFAs *Parikh equivalent* to the given grammar

Upper and Lower Bounds

Problem

Given a CFG G compare the size of G
with the sizes of finite automata accepting languages
that are Parikh equivalent to $L(G)$

Nondeterministic automata (number of states wrt s, size of G)

Upper bound:

- $2^{2^{O(s^2)}}$
- $2^{O(s^2)}$

Lower bound: $2^{\Omega(2^s)}$

Upper and Lower Bounds

Problem

Given a CFG G compare the size of G
with the sizes of finite automata accepting languages
that are Parikh equivalent to $L(G)$

Nondeterministic automata (number of states wrt s, size of G)

Upper bound:

- $2^{2^{O(s^2)}}$ (implicit construction from classical proof of Parikh's Th.)
- $O(4^s)$ [Esparza&Ganty&Kiefer&Luttenberger '11]

Lower bound: $\Omega(2^s)$

Upper and Lower Bounds

Problem

Given a CFG G compare the size of G
with the sizes of finite automata accepting languages
that are Parikh equivalent to $L(G)$

Nondeterministic automata (number of states wrt s, size of G)

Upper bound:

- $2^{2^{O(s^2)}}$ (implicit construction from classical proof of Parikh's Th.)
- $O(4^s)$ [Esparza&Ganty&Kiefer&Luttenberger'11]

Lower bound: $\Omega(2^s)$

Upper and Lower Bounds

Problem

Given a CFG G compare the size of G
with the sizes of finite automata accepting languages
that are Parikh equivalent to $L(G)$

Nondeterministic automata (number of states wrt s, size of G)

Upper bound:

- $2^{2^{O(s^2)}}$ (implicit construction from classical proof of Parikh's Th.)
- $O(4^s)$ [Esparza&Ganty&Kiefer&Luttenberger '11]

Lower bound: $\Omega(2^s)$

Upper and Lower Bounds

Problem

Given a CFG G compare the size of G
with the sizes of finite automata accepting languages
that are Parikh equivalent to $L(G)$

Nondeterministic automata (number of states wrt s, size of G)

Upper bound:

- $2^{2^{O(s^2)}}$ (implicit construction from classical proof of Parikh's Th.)
- $O(4^s)$ [Esparza&Ganty&Kiefer&Luttenberger '11]

Lower bound: $\Omega(2^s)$

Upper and Lower Bounds

Problem

Given a CFG G compare the size of G
with the sizes of finite automata accepting languages
that are Parikh equivalent to $L(G)$

Deterministic automata (number of states wrt s, size of G)

Upper bound: $2^{O(4^s)}$ (subset construction)

Lower bound: 2^{s^2} (from the unary case)

Upper and Lower Bounds

Problem

Given a CFG G compare the size of G
with the sizes of finite automata accepting languages
that are Parikh equivalent to $L(G)$

Deterministic automata (number of states wrt s, size of G)

Upper bound: $2^{O(4^s)}$ (subset construction)

Lower bound: 2^{s^2} (from the unary case)

Upper and Lower Bounds

Problem

Given a CFG G compare the size of G
with the sizes of finite automata accepting languages
that are Parikh equivalent to $L(G)$

Deterministic automata (number of states wrt s, size of G)

Upper bound: $2^{O(4^s)}$ (subset construction)

Lower bound: 2^{s^2} (from the unary case)

Upper and Lower Bounds

Problem

Given a CFG G compare the size of G
with the sizes of finite automata accepting languages
that are Parikh equivalent to $L(G)$

Deterministic automata (number of states wrt s, size of G)

Upper bound: $2^{O(4^s)}$ (subset construction)

Lower bound: 2^{s^2} (from the unary case)

Is it possible to reduce the gap between
the upper and the lower bound?

Upper and Lower Bounds

Problem

Given a CFG G compare the size of G
with the sizes of finite automata accepting languages
that are Parikh equivalent to $L(G)$

Deterministic automata (number of states wrt s, size of G)

Upper bound: $2^{O(4^s)}$ (subset construction)

Lower bound: 2^{s^2} (from the unary case)

We reduced the upper bound to $2^{s^{O(1)}}$ in the following cases:

- bounded context-free languages
  i.e, context-free subsets of $a_1^* a_2^* \dots a_m^*$ $(m \geq 2)$
- context-free languages over two-letter alphabets

Upper and Lower Bounds

Problem

Given a CFG G compare the size of G
with the sizes of finite automata accepting languages
that are Parikh equivalent to $L(G)$

Deterministic automata (number of states wrt s, size of G)

Upper bound: $2^{O(4^s)}$ (subset construction)

Lower bound: 2^{s^2} (from the unary case)

We reduced the upper bound to $2^{s^{O(1)}}$ in the following cases:

- bounded context-free languages
  i.e, context-free subsets of $a_1^* a_2^* \dots a_m^*$ $(m \geq 2)$
- context-free languages over two-letter alphabets

Upper and Lower Bounds

Problem

Given a CFG G compare the size of G
with the sizes of finite automata accepting languages
that are Parikh equivalent to $L(G)$

Deterministic automata (number of states wrt s, size of G)

Upper bound: $2^{O(4^s)}$ (subset construction)

Lower bound: 2^{s^2} (from the unary case)

We reduced the upper bound to $2^{s^{O(1)}}$ in the following cases:

- bounded context-free languages
  i.e, context-free subsets of $a_1^* a_2^* \dots a_m^*$ $(m \geq 2)$
- context-free languages over two-letter alphabets

First Contribution: Bounded Context-Free Languages

Theorem

- $\Sigma = \{a_1, a_2, \ldots, a_m\}$ *fixed alphabet*
- *G grammar in Chomsky normal form with h variables s.t.* $L(G) \subseteq a_1^* a_2^* \ldots a_m^*$

There exists a DFA A with at most $2^{h^{O(1)}}$ states s.t. $L(G) =_\pi L(A)$

First Contribution: Proof Outline

$\Sigma = \{a_1, a_2, \ldots, a_m\}$

- Restriction to *strongly bounded grammars*

  $G = (V, \Sigma, P, S)$ *is strongly bounded iff*
  for all $A \in V$, *there are* $i \leq j$ *s.t.*
  $L_A = \{x \in \Sigma^* \mid A \stackrel{\star}{\Rightarrow} x\} \subseteq a_i^+ a_{i+1}^* \cdots a_{j-1}^* a_j^+$

- $A \in V$ is said to be *unary* iff $L_A \subseteq a_i^+$ for some i

  in this case L_A *is accepted by a DFA with* $< 2^{h^2}$ *states*
  [Pighizzini&Shallit&Wang '02]

- The use of nonunary variables is very restricted:

  If $S \stackrel{\star}{\Rightarrow} \alpha$ then α contains $\leq m - 1$ nonunary variables

  Hence a finite control of size $O(h^{m-1})$ can keep track of them

First Contribution: Proof Outline

$\Sigma = \{a_1, a_2, \ldots, a_m\}$

- Restriction to *strongly bounded grammars*

  $G = (V, \Sigma, P, S)$ *is strongly bounded iff*
  for all $A \in V$, *there are* $i \leq j$ *s.t.*
  $L_A = \{x \in \Sigma^* \mid A \stackrel{\star}{\Rightarrow} x\} \subseteq a_i^+ a_{i+1}^* \cdots a_{j-1}^* a_j^+$

- $A \in V$ is said to be *unary* iff $L_A \subseteq a_i^+$ for some i

  in this case L_A *is accepted by a DFA with* $< 2^{h^2}$ *states*
  [Pighizzini&Shallit&Wang '02]

- The use of nonunary variables is very restricted:

  If $S \stackrel{\star}{\Rightarrow} \alpha$ then α contains $\leq m - 1$ nonunary variables

  Hence a finite control of size $O(h^{m-1})$ can keep track of them

First Contribution: Proof Outline

$\Sigma = \{a_1, a_2, \ldots, a_m\}$

- Restriction to *strongly bounded grammars*

  $G = (V, \Sigma, P, S)$ *is strongly bounded iff*
  for all $A \in V$, *there are* $i \leq j$ *s.t.*
  $L_A = \{x \in \Sigma^* \mid A \stackrel{\star}{\Rightarrow} x\} \subseteq a_i^+ a_{i+1}^* \cdots a_{j-1}^* a_j^+$

- $A \in V$ is said to be *unary* iff $L_A \subseteq a_i^+$ for some i

  in this case L_A *is accepted by a DFA with* $< 2^{h^2}$ *states*
  [Pighizzini&Shallit&Wang '02]

- The use of nonunary variables is very restricted:

  If $S \stackrel{\star}{\Rightarrow} \alpha$ then α contains $\leq m - 1$ nonunary variables

  Hence a finite control of size $O(h^{m-1})$ can keep track of them

First Contribution: Proof Outline

$\Sigma = \{a_1, a_2, \ldots, a_m\}$

- Restriction to *strongly bounded grammars*

  $G = (V, \Sigma, P, S)$ *is strongly bounded iff*
  for all $A \in V$, *there are* $i \leq j$ *s.t.*
  $L_A = \{x \in \Sigma^* \mid A \stackrel{\star}{\Rightarrow} x\} \subseteq a_i^+ a_{i+1}^* \cdots a_{j-1}^* a_j^+$

- $A \in V$ is said to be *unary* iff $L_A \subseteq a_i^+$ for some i

  in this case L_A *is accepted by a DFA with* $< 2^{h^2}$ *states*
  [Pighizzini&Shallit&Wang '02]

- The use of nonunary variables is very restricted:

  If $S \stackrel{\star}{\Rightarrow} \alpha$ then α contains $\leq m - 1$ nonunary variables

  Hence a finite control of size $O(h^{m-1})$ can keep track of them

First Contribution: Proof Outline

$\Sigma = \{a_1, a_2, \ldots, a_m\}$

- Restriction to *strongly bounded grammars*

  $G = (V, \Sigma, P, S)$ *is strongly bounded iff*
  for all $A \in V$, *there are* $i \leq j$ *s.t.*
  $L_A = \{x \in \Sigma^* \mid A \stackrel{\star}{\Rightarrow} x\} \subseteq a_i^+ a_{i+1}^* \cdots a_{j-1}^* a_j^+$

- $A \in V$ is said to be *unary* iff $L_A \subseteq a_i^+$ for some i

  in this case L_A *is accepted by a DFA with* $< 2^{h^2}$ *states*
  [Pighizzini&Shallit&Wang '02]

- The use of nonunary variables is very restricted:

  If $S \stackrel{\star}{\Rightarrow} \alpha$ then α contains $\leq m - 1$ nonunary variables

  Hence a finite control of size $O(h^{m-1})$ can keep track of them

First Contribution: Proof Outline

$\Sigma = \{a_1, a_2, \ldots, a_m\}$

- Restriction to *strongly bounded grammars*

  $G = (V, \Sigma, P, S)$ *is strongly bounded iff*
  for all $A \in V$, *there are* $i \leq j$ *s.t.*
  $L_A = \{x \in \Sigma^* \mid A \stackrel{\star}{\Rightarrow} x\} \subseteq a_i^+ a_{i+1}^* \cdots a_{j-1}^* a_j^+$

- $A \in V$ is said to be *unary* iff $L_A \subseteq a_i^+$ for some i

  in this case L_A *is accepted by a DFA with* $< 2^{h^2}$ *states*
  [Pighizzini&Shallit&Wang '02]

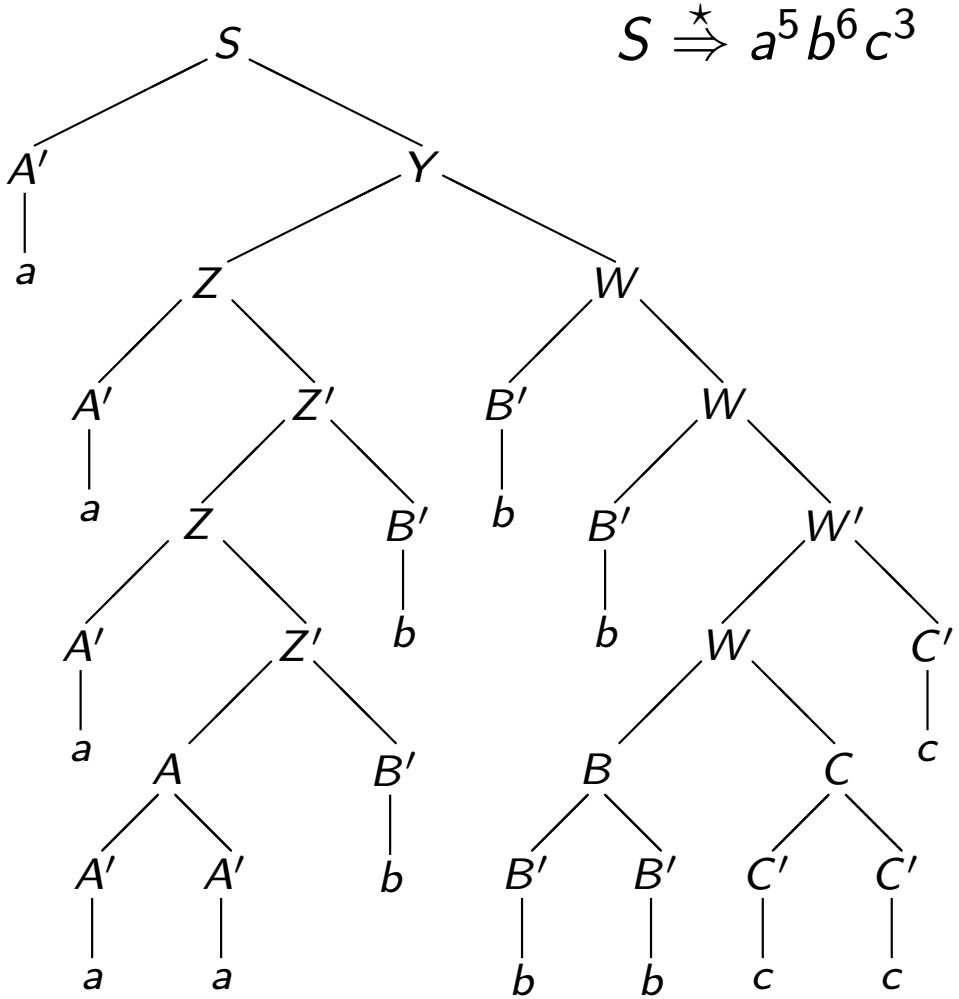
- The use of nonunary variables is very restricted:

  If $S \stackrel{\star}{\Rightarrow} \alpha$ then α contains $\leq m - 1$ nonunary variables

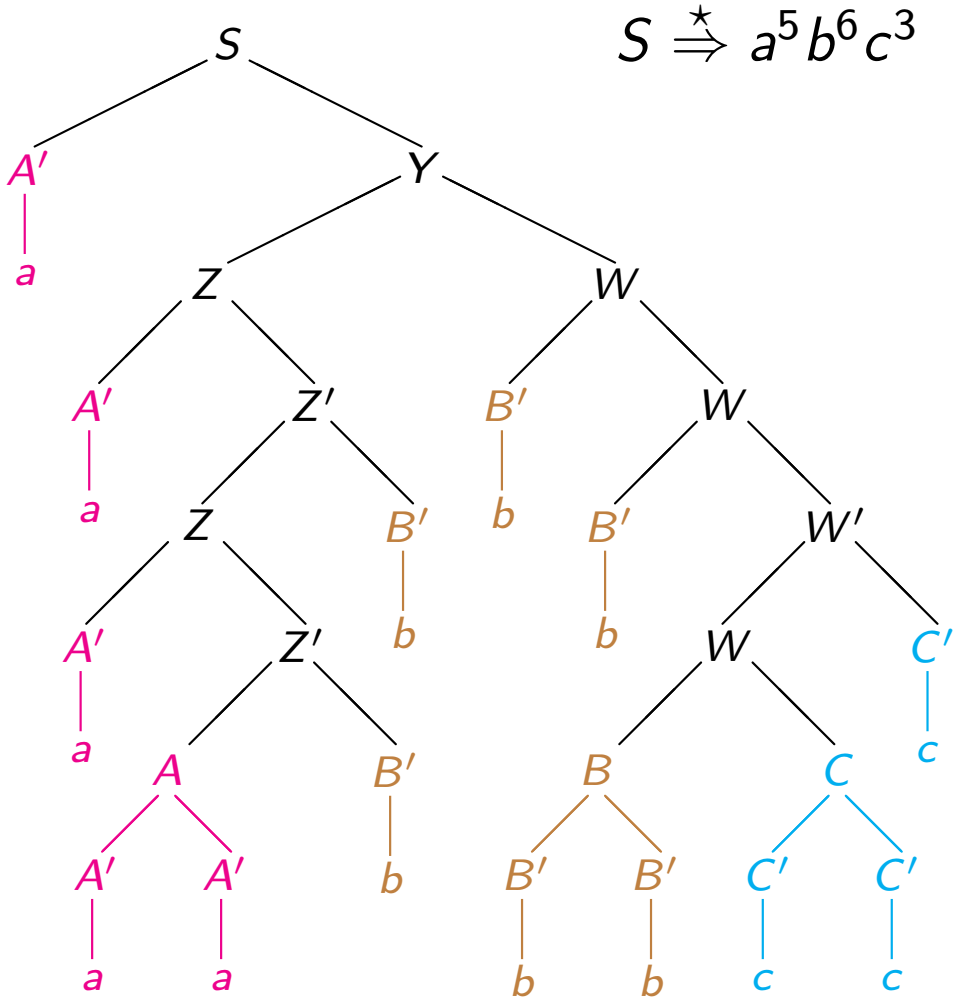
Hence a finite control of size $O(h^{m-1})$ can keep track of them

Example $\Sigma = \{a, b, c\}$

$$S \stackrel{\star}{\Rightarrow} a^5 b^6 c^3$$



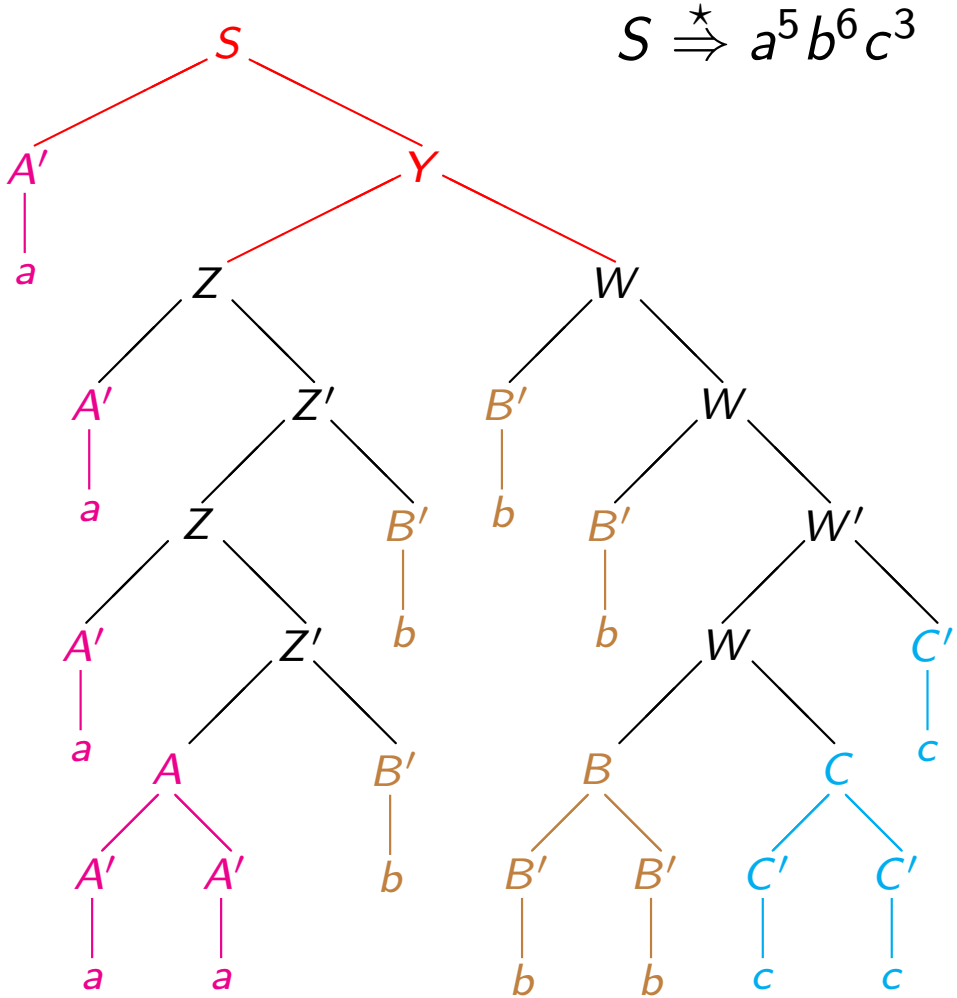
Example $\Sigma = \{a, b, c\}$



- Unary variables: A, A', B, B', C, C'
- $L_S, L_Y \subseteq a^+ b^* c^+$
- $L_Z, L_{Z'} \subseteq a^+ b^+$
- $L_W, L_{W'} \subseteq b^+ c^+$

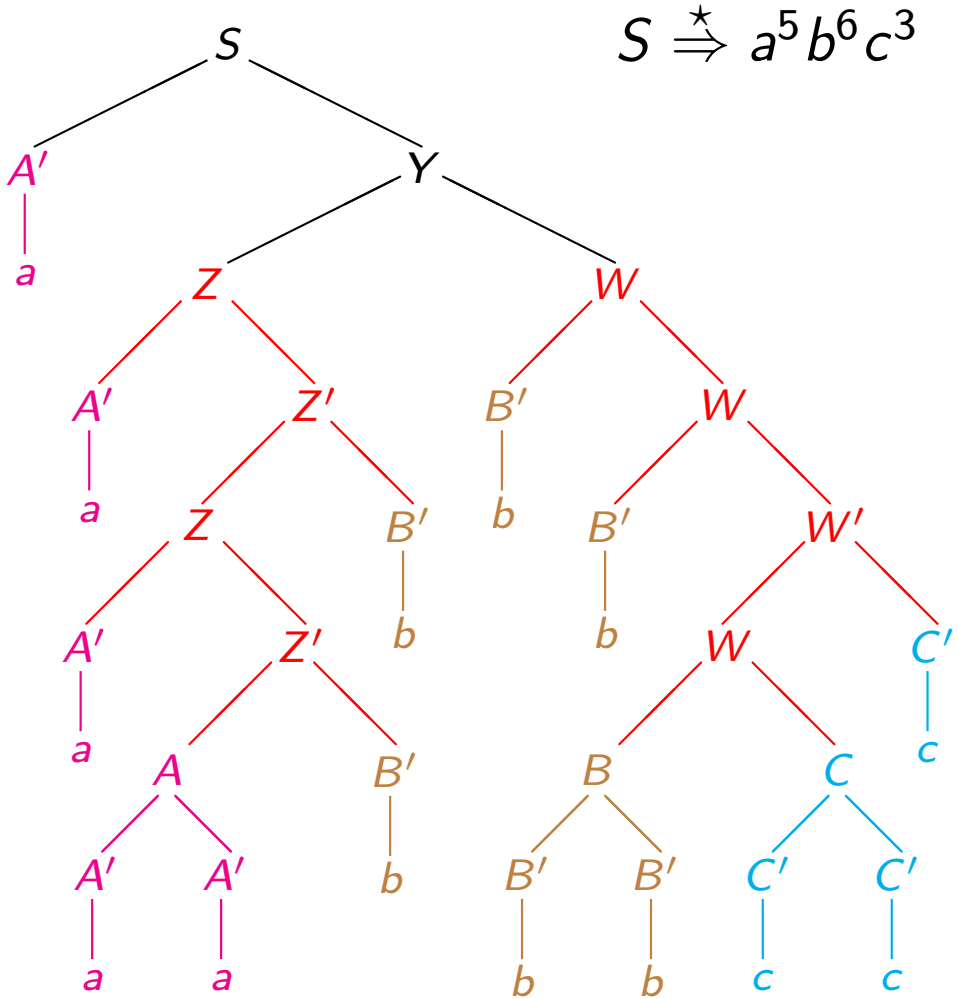
Example $\Sigma = \{a, b, c\}$

$$S \stackrel{\star}{\Rightarrow} a^5 b^6 c^3$$



- Unary variables: A, A', B, B', C, C'
- $L_S, L_Y \subseteq a^+ b^* c^+$
- $L_Z, L_{Z'} \subseteq a^+ b^+$
- $L_W, L_{W'} \subseteq b^+ c^+$

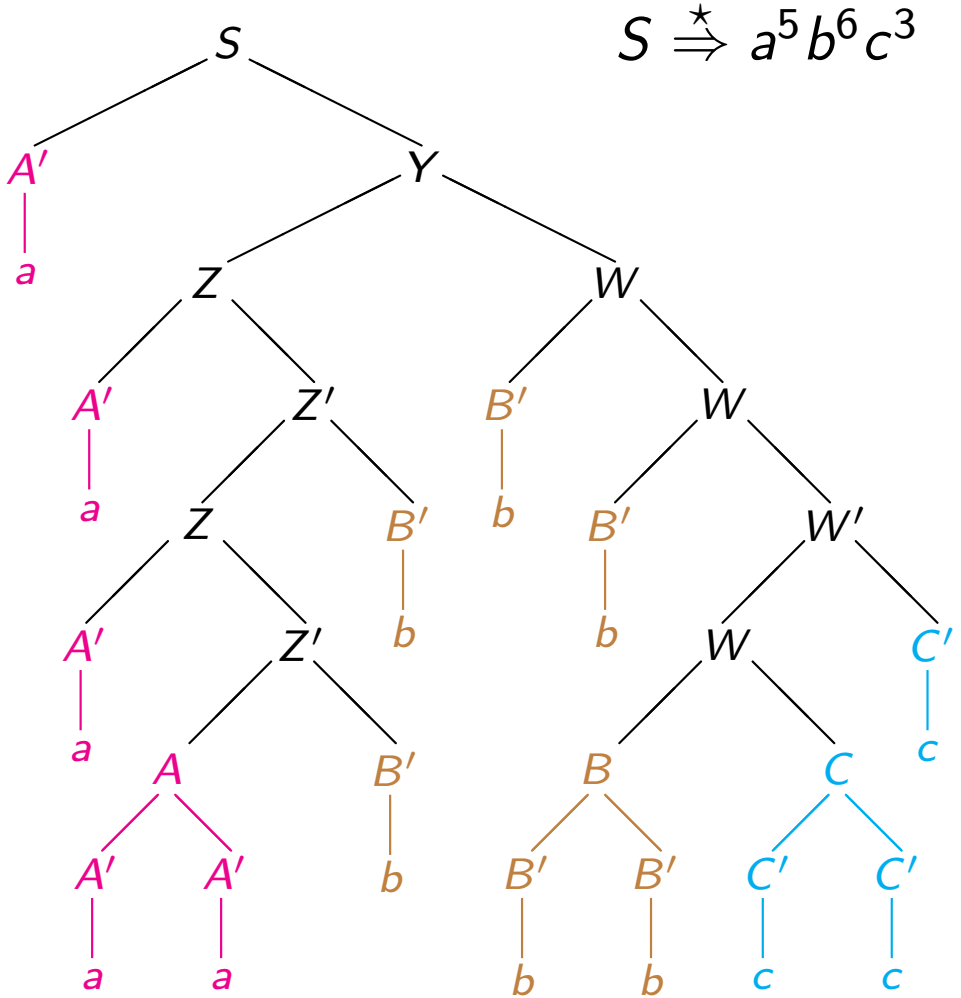
Example $\Sigma = \{a, b, c\}$



$S \stackrel{\star}{\Rightarrow} a^5 b^6 c^3$

- Unary variables: A, A', B, B', C, C'
- $L_S, L_Y \subseteq a^+ b^* c^+$
- $L_Z, L_{Z'} \subseteq a^+ b^+$
- $L_W, L_{W'} \subseteq b^+ c^+$

Example $\Sigma = \{a, b, c\}$



$S \stackrel{\star}{\Rightarrow} a^5 b^6 c^3$

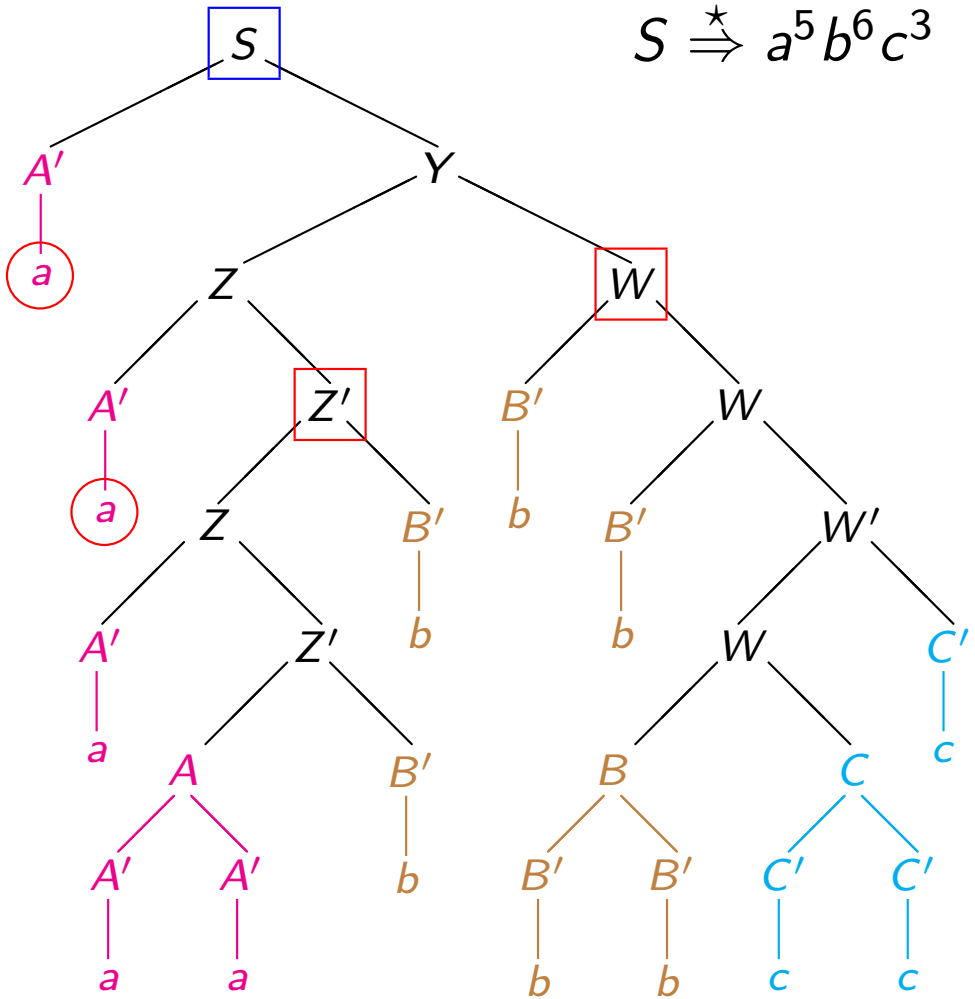
Our automaton recognizes

$$a^2 baba^2 b^2 c^3 b^2$$

by simulating a particular derivation from S

$$
\begin{aligned}
S &\stackrel{\star}{\Rightarrow} a^2 Z' W \\
&\stackrel{\star}{\Rightarrow} a^2 Z b W \\
&\stackrel{\star}{\Rightarrow} a^2 a Z' b W \\
&\stackrel{\star}{\Rightarrow} a^3 A b W \\
&\stackrel{\star}{\Rightarrow} a^3 a^2 b^2 W \\
&\stackrel{\star}{\Rightarrow} a^5 b^2 b^2 W' \\
&\stackrel{\star}{\Rightarrow} a^5 b^4 B c^3 \\
&\stackrel{\star}{\Rightarrow} a^5 b^4 b^2 c^3 \\
&= a^5 b^6 c^3 \\
&=_\pi a^2 baba^2 b^2 c^3 b^2
\end{aligned}
$$

Example $\Sigma = \{a, b, c\}$



$S \stackrel{\star}{\Rightarrow} a^5 b^6 c^3$

Our automaton recognizes

$$a^2 baba^2 b^2 c^3 b^2$$

by simulating a particular derivation from S

$S \stackrel{\star}{\Rightarrow} a^2 Z' W$

$\stackrel{\star}{\Rightarrow} a^2 Z b W$

$\stackrel{\star}{\Rightarrow} a^2 a Z' b W$

$\stackrel{\star}{\Rightarrow} a^3 A b W$

$\stackrel{\star}{\Rightarrow} a^3 a^2 b^2 W$

$\stackrel{\star}{\Rightarrow} a^5 b^2 b^2 W'$

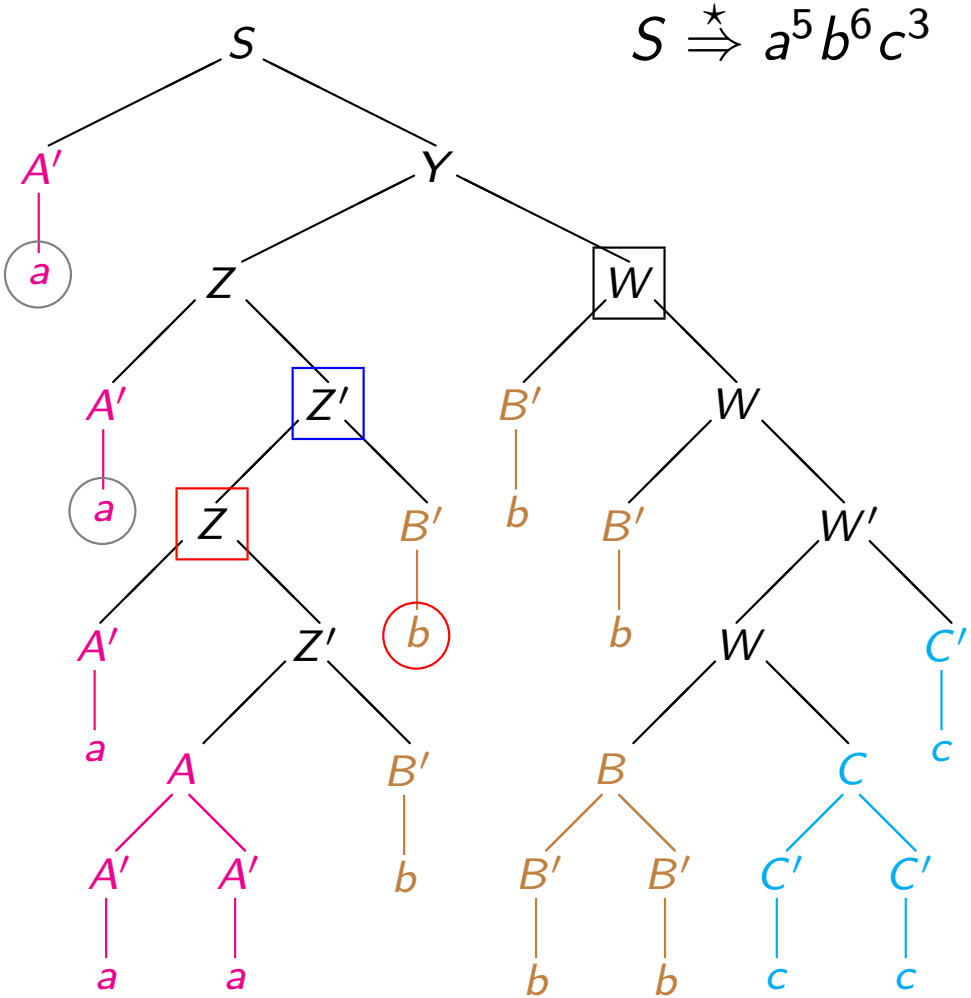
$\stackrel{\star}{\Rightarrow} a^5 b^4 B c^3$

$\stackrel{\star}{\Rightarrow} a^5 b^4 b^2 c^3$

$= a^5 b^6 c^3$

$=_\pi a^2 baba^2 b^2 c^3 b^2$

Example $\Sigma = \{a, b, c\}$



$S \stackrel{\star}{\Rightarrow} a^5 b^6 c^3$

Our automaton recognizes

$$a^2 baba^2 b^2 c^3 b^2$$

by simulating a particular derivation from S

$S \stackrel{\star}{\Rightarrow} a^2 Z' W$

$\stackrel{\star}{\Rightarrow} a^2 Z b W$

$\stackrel{\star}{\Rightarrow} a^2 a Z' b W$

$\stackrel{\star}{\Rightarrow} a^3 A b W$

$\stackrel{\star}{\Rightarrow} a^3 a^2 b^2 W$

$\stackrel{\star}{\Rightarrow} a^5 b^2 b^2 W'$

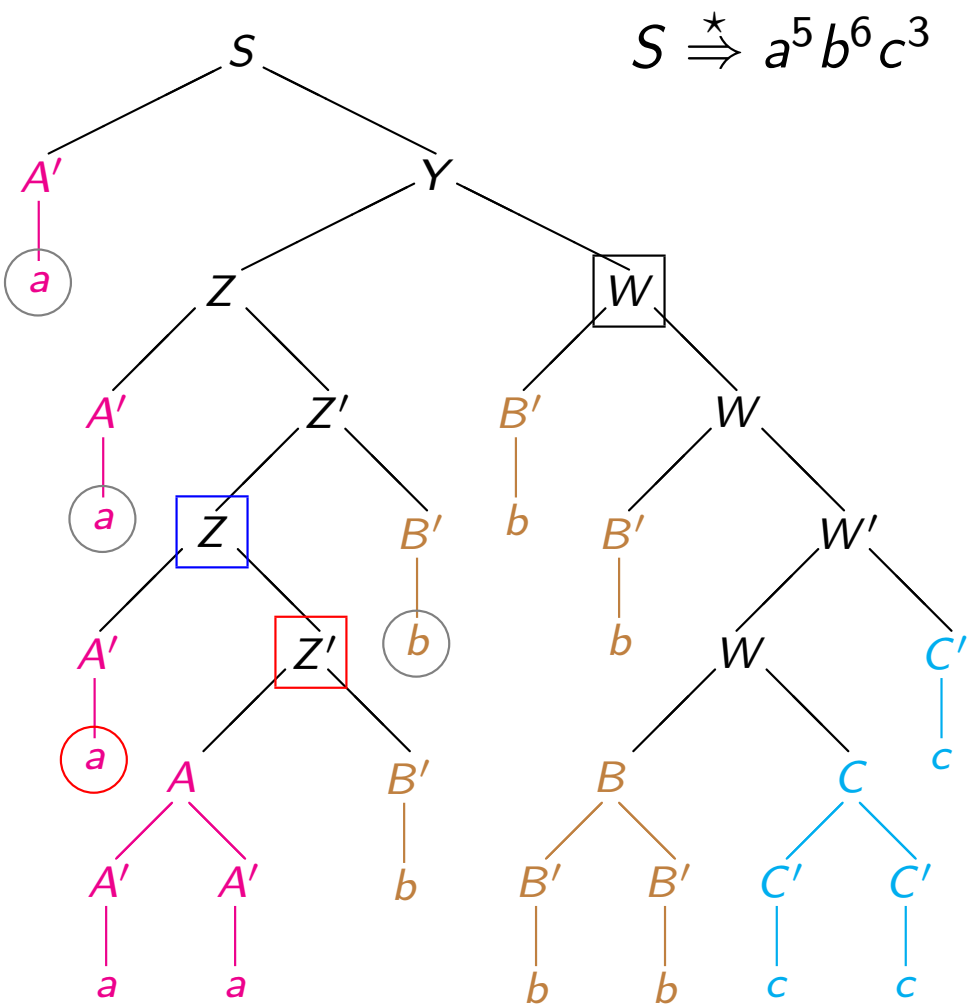
$\stackrel{\star}{\Rightarrow} a^5 b^4 B c^3$

$\stackrel{\star}{\Rightarrow} a^5 b^4 b^2 c^3$

$= a^5 b^6 c^3$

$=_\pi a^2 baba^2 b^2 c^3 b^2$

Example $\Sigma = \{a, b, c\}$



$S \stackrel{\star}{\Rightarrow} a^5 b^6 c^3$

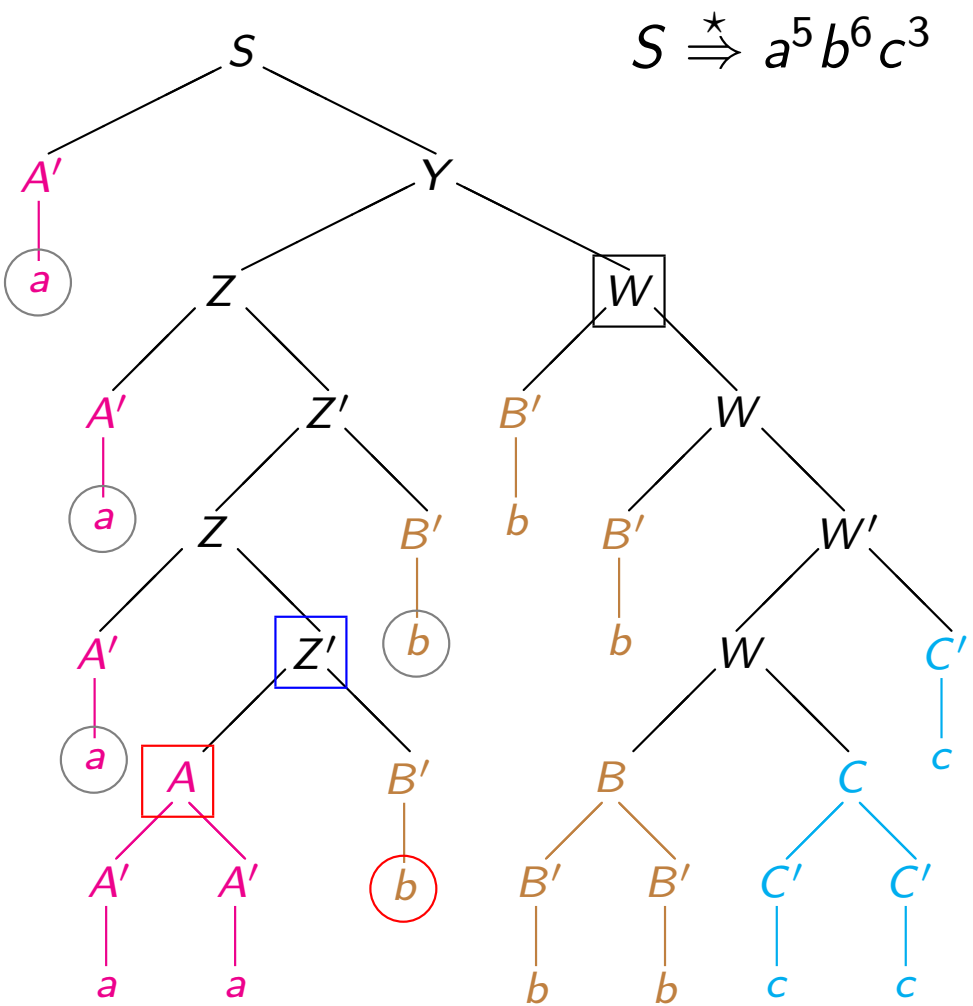
Our automaton recognizes

$$a^2 baba^2 b^2 c^3 b^2$$

by simulating a particular derivation from S

$$\begin{aligned}
S &\stackrel{\star}{\Rightarrow} a^2 Z' W \\
&\stackrel{\star}{\Rightarrow} a^2 Z b W \\
&\stackrel{\star}{\Rightarrow} a^2 a Z' b W \\
&\stackrel{\star}{\Rightarrow} a^3 A b W \\
&\stackrel{\star}{\Rightarrow} a^3 a^2 b^2 W \\
&\stackrel{\star}{\Rightarrow} a^5 b^2 b^2 W' \\
&\stackrel{\star}{\Rightarrow} a^5 b^4 B c^3 \\
&\stackrel{\star}{\Rightarrow} a^5 b^4 b^2 c^3 \\
&= a^5 b^6 c^3 \\
&=_\pi a^2 baba^2 b^2 c^3 b^2
\end{aligned}$$

Example $\Sigma = \{a, b, c\}$



$S \stackrel{\star}{\Rightarrow} a^5 b^6 c^3$

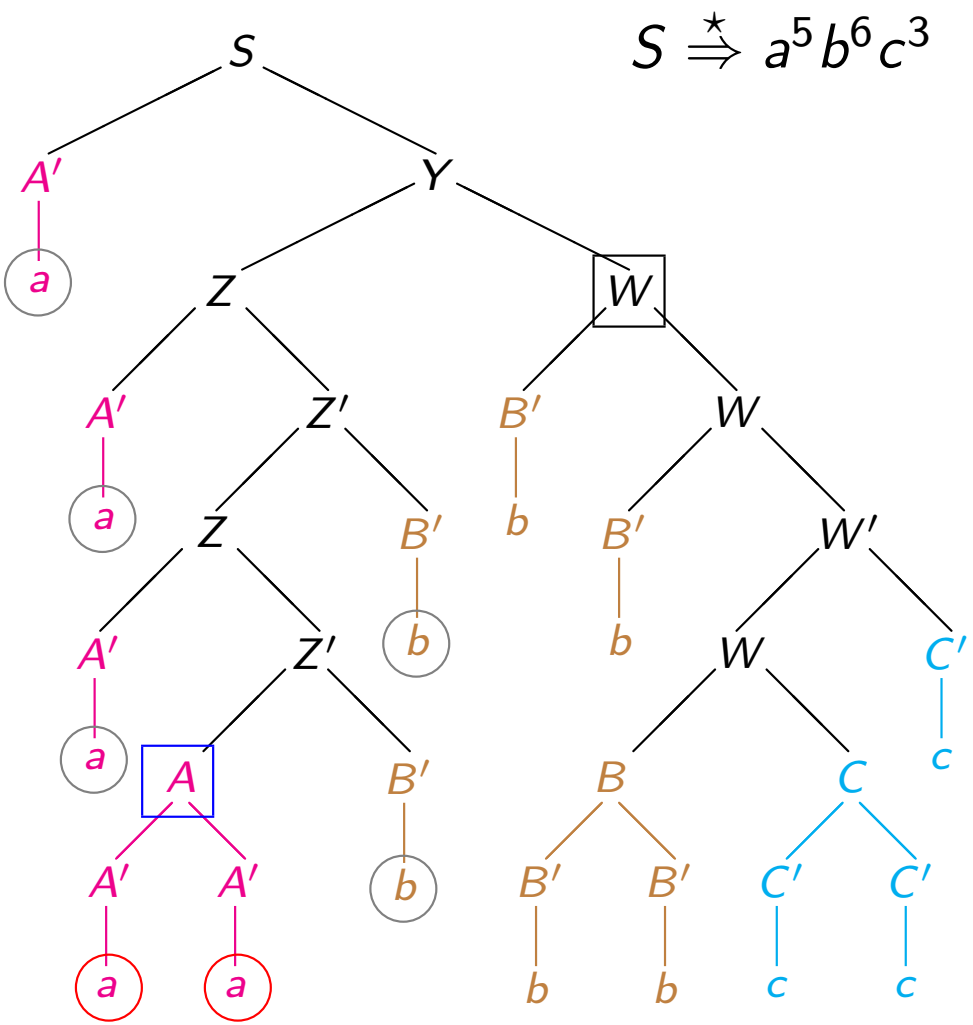
Our automaton recognizes

$$a^2 baba^2 b^2 c^3 b^2$$

by simulating a particular derivation from S

$$\begin{aligned}
S &\stackrel{\star}{\Rightarrow} a^2 Z' W \\
&\stackrel{\star}{\Rightarrow} a^2 Z b W \\
&\stackrel{\star}{\Rightarrow} a^2 a Z' b W \\
&\stackrel{\star}{\Rightarrow} a^3 A b b W \\
&\stackrel{\star}{\Rightarrow} a^3 a^2 b^2 W \\
&\stackrel{\star}{\Rightarrow} a^5 b^2 b^2 W' \\
&\stackrel{\star}{\Rightarrow} a^5 b^4 B c^3 \\
&\stackrel{\star}{\Rightarrow} a^5 b^4 b^2 c^3 \\
&= a^5 b^6 c^3 \\
&=_\pi a^2 baba^2 b^2 c^3 b^2
\end{aligned}$$

Example $\Sigma = \{a, b, c\}$



$S \stackrel{\star}{\Rightarrow} a^5 b^6 c^3$

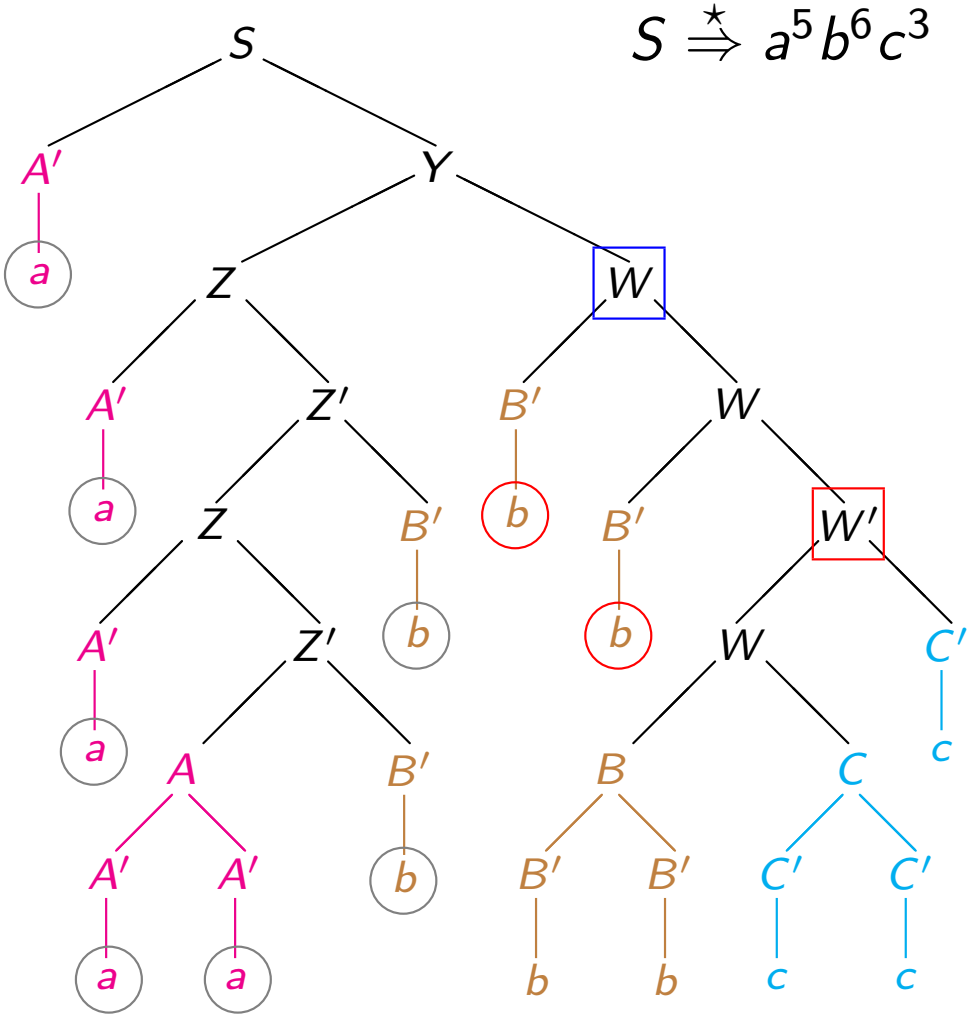
Our automaton recognizes

$$a^2 baba^2 b^2 c^3 b^2$$

by simulating a particular derivation from S

$$\begin{aligned}
S &\stackrel{\star}{\Rightarrow} a^2 Z' W \\
&\stackrel{\star}{\Rightarrow} a^2 Z b W \\
&\stackrel{\star}{\Rightarrow} a^2 a Z' b W \\
&\stackrel{\star}{\Rightarrow} a^3 A b W \\
&\stackrel{\star}{\Rightarrow} a^3 a^2 b^2 W \\
&\stackrel{\star}{\Rightarrow} a^5 b^2 b^2 W' \\
&\stackrel{\star}{\Rightarrow} a^5 b^4 B c^3 \\
&\stackrel{\star}{\Rightarrow} a^5 b^4 b^2 c^3 \\
&= a^5 b^6 c^3 \\
&=_\pi a^2 baba^2 b^2 c^3 b^2
\end{aligned}$$

Example $\Sigma = \{a, b, c\}$



$S \stackrel{\star}{\Rightarrow} a^5 b^6 c^3$

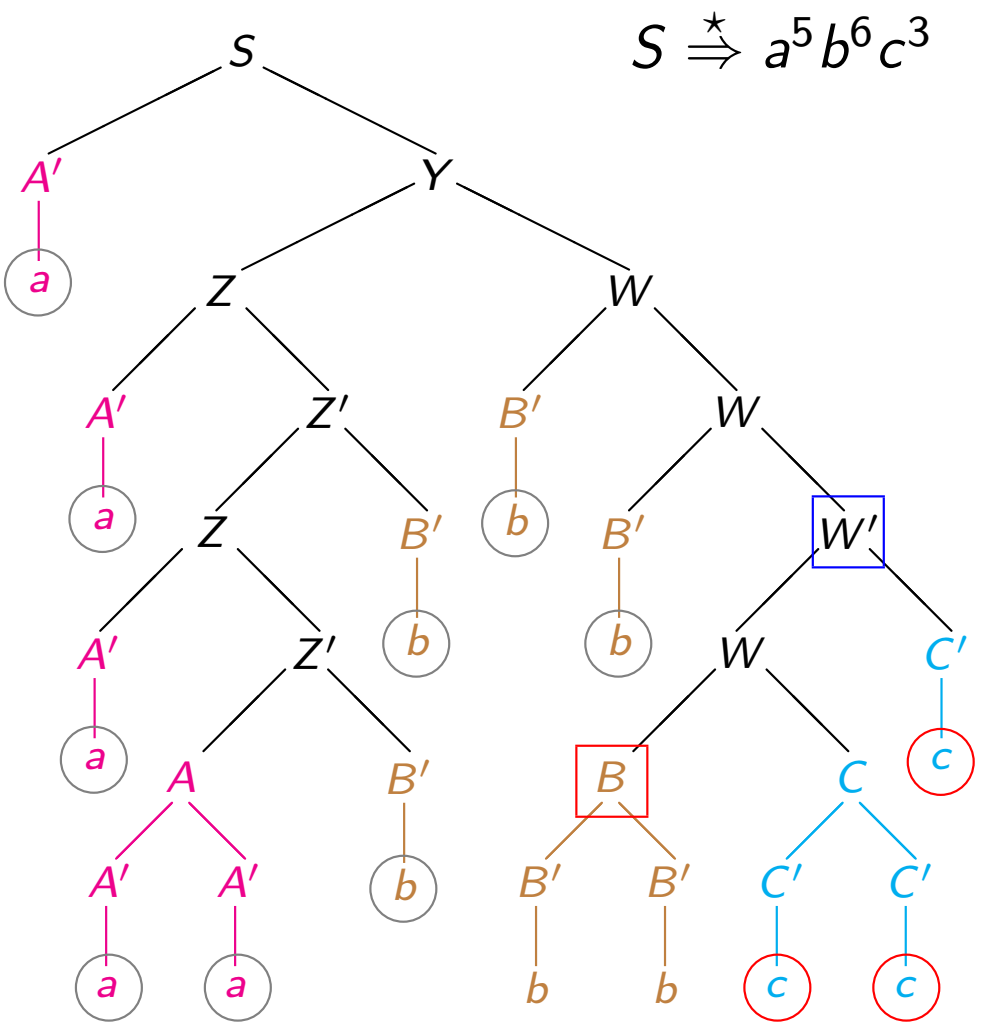
Our automaton recognizes

$$a^2 baba^2 b^2 c^3 b^2$$

by simulating a particular derivation from S

$$
\begin{aligned}
S &\stackrel{\star}{\Rightarrow} a^2 Z' W \\
&\stackrel{\star}{\Rightarrow} a^2 Z b W \\
&\stackrel{\star}{\Rightarrow} a^2 a Z' b W \\
&\stackrel{\star}{\Rightarrow} a^3 A b W \\
&\stackrel{\star}{\Rightarrow} a^3 a^2 b^2 W \\
&\stackrel{\star}{\Rightarrow} a^5 b^2 b^2 W' \\
&\stackrel{\star}{\Rightarrow} a^5 b^4 B c^3 \\
&\stackrel{\star}{\Rightarrow} a^5 b^4 b^2 c^3 \\
&= a^5 b^6 c^3 \\
&=_{\pi} a^2 baba^2 b^2 c^3 b^2
\end{aligned}
$$

Example $\Sigma = \{a, b, c\}$



$S \stackrel{\star}{\Rightarrow} a^5 b^6 c^3$

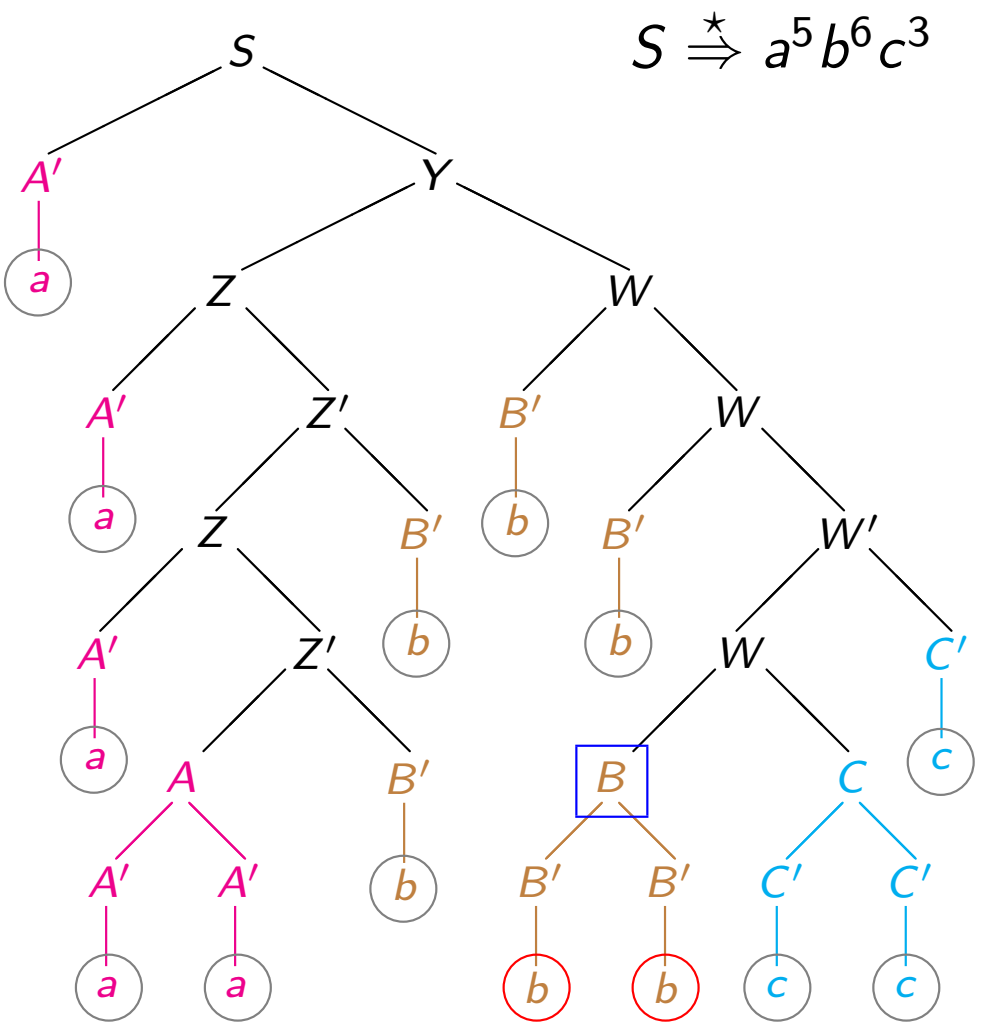
Our automaton recognizes

$$a^2 baba^2 b^2 c^3 b^2$$

by simulating a particular derivation from S

$$\begin{aligned}
S &\stackrel{\star}{\Rightarrow} a^2 Z' W \\
&\stackrel{\star}{\Rightarrow} a^2 Z b W \\
&\stackrel{\star}{\Rightarrow} a^2 a Z' b W \\
&\stackrel{\star}{\Rightarrow} a^3 A b W \\
&\stackrel{\star}{\Rightarrow} a^3 a^2 b^2 W \\
&\stackrel{\star}{\Rightarrow} a^5 b^2 b^2 W' \\
&\stackrel{\star}{\Rightarrow} a^5 b^4 B c^3 \\
&\stackrel{\star}{\Rightarrow} a^5 b^4 b^2 c^3 \\
&= a^5 b^6 c^3 \\
&=_\pi a^2 baba^2 b^2 c^3 b^2
\end{aligned}$$

Example $\Sigma = \{a, b, c\}$



$S \stackrel{\star}{\Rightarrow} a^5 b^6 c^3$

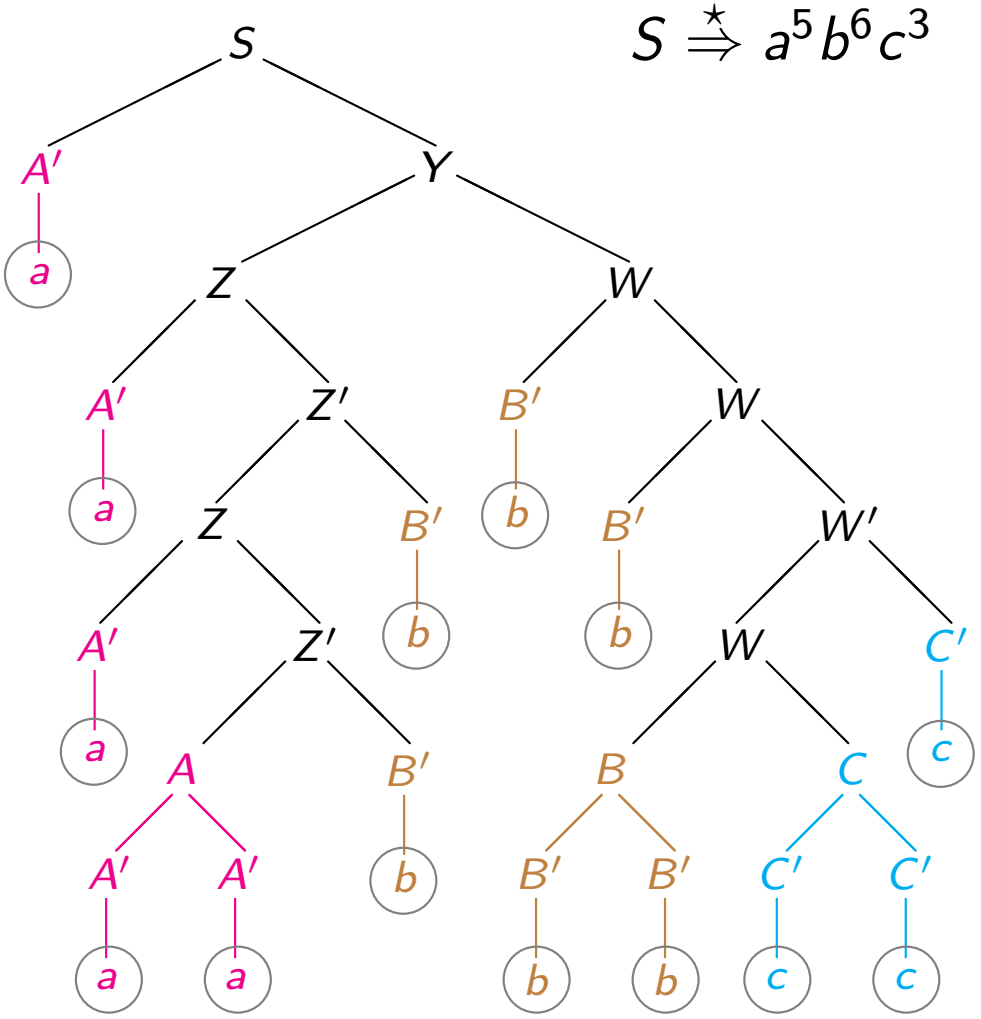
Our automaton recognizes

$$a^2 baba^2 b^2 c^3 b^2$$

by simulating a particular derivation from S

$$\begin{aligned}
S &\stackrel{\star}{\Rightarrow} a^2 Z' W \\
&\stackrel{\star}{\Rightarrow} a^2 Z b W \\
&\stackrel{\star}{\Rightarrow} a^2 a Z' b W \\
&\stackrel{\star}{\Rightarrow} a^3 A b W \\
&\stackrel{\star}{\Rightarrow} a^3 a^2 b^2 W \\
&\stackrel{\star}{\Rightarrow} a^5 b^2 b^2 W' \\
&\stackrel{\star}{\Rightarrow} a^5 b^4 B c^3 \\
&\stackrel{\star}{\Rightarrow} a^5 b^4 b^2 c^3 \\
&= a^5 b^6 c^3 \\
&=_\pi a^2 baba^2 b^2 c^3 b^2
\end{aligned}$$

Example $\Sigma = \{a, b, c\}$



$S \stackrel{\star}{\Rightarrow} a^5 b^6 c^3$

Our automaton recognizes

$$a^2 baba^2 b^2 c^3 b^2$$

by simulating a particular derivation from S

$$\begin{aligned}
S &\stackrel{\star}{\Rightarrow} a^2 Z' W \\
&\stackrel{\star}{\Rightarrow} a^2 Z b W \\
&\stackrel{\star}{\Rightarrow} a^2 a Z' b W \\
&\stackrel{\star}{\Rightarrow} a^3 A b W \\
&\stackrel{\star}{\Rightarrow} a^3 a^2 b^2 W \\
&\stackrel{\star}{\Rightarrow} a^5 b^2 b^2 W' \\
&\stackrel{\star}{\Rightarrow} a^5 b^4 B c^3 \\
&\stackrel{\star}{\Rightarrow} a^5 b^4 b^2 c^3 \\
&= a^5 b^6 c^3 \\
&=_\pi a^2 baba^2 b^2 c^3 b^2
\end{aligned}$$

First Contribution: Proof Outline

- This derivation process is simulated by an automaton which tests the matching between generated terminals and input symbols
- At each step the automaton needs to remember at most $\#\Sigma - 1$ variables
- The process is nondeterministic
- It can be implemented using $O(h^{\#\Sigma-1})$ states
- Hence, a deterministic control can be implemented with $2^{poly(h)}$ states
- The "unary parts" can be simulated within the same state bound

First Contribution: Proof Outline

- This derivation process is simulated by an automaton which tests the matching between generated terminals and input symbols
- At each step the automaton needs to remember at most $\#\Sigma - 1$ variables
- The process is nondeterministic
- It can be implemented using $O(h^{\#\Sigma-1})$ states
- Hence, a deterministic control can be implemented with $2^{poly(h)}$ states
- The "unary parts" can be simulated within the same state bound

First Contribution: Proof Outline

- This derivation process is simulated by an automaton which tests the matching between generated terminals and input symbols
- At each step the automaton needs to remember at most $\#\Sigma - 1$ variables
- The process is nondeterministic
- It can be implemented using $O(h^{\#\Sigma-1})$ states
- Hence, a deterministic control can be implemented with $2^{poly(h)}$ states
- The "unary parts" can be simulated within the same state bound

First Contribution: Proof Outline

- This derivation process is simulated by an automaton which tests the matching between generated terminals and input symbols
- At each step the automaton needs to remember at most $\#\Sigma - 1$ variables
- The process is nondeterministic
- It can be implemented using $O(h^{\#\Sigma-1})$ states
- Hence, a deterministic control can be implemented with $2^{poly(h)}$ states
- The "unary parts" can be simulated within the same state bound

First Contribution: Proof Outline

- This derivation process is simulated by an automaton which tests the matching between generated terminals and input symbols
- At each step the automaton needs to remember at most $\#\Sigma - 1$ variables
- The process is nondeterministic
- It can be implemented using $O(h^{\#\Sigma-1})$ states
- Hence, a deterministic control can be implemented with $2^{poly(h)}$ states
- The "unary parts" can be simulated within the same state bound

First Contribution: Proof Outline

- This derivation process is simulated by an automaton which tests the matching between generated terminals and input symbols
- At each step the automaton needs to remember at most $\#\Sigma - 1$ variables
- The process is nondeterministic
- It can be implemented using $O(h^{\#\Sigma-1})$ states
- Hence, a deterministic control can be implemented with $2^{poly(h)}$ states
- The "unary parts" can be simulated within the same state bound

Second Contribution: Binary Context-Free Languages

Theorem

Let G grammar in Chomsky normal form with h variables with a binary terminal alphabet.
Then there is a DFA A with at most $2^{h^{O(1)}}$ states s.t. $L(A) =_\pi L(G)$

The proof relies the following results:

Lemma ([Kopczyński&To '10])

For G as in the theorem, it holds that $\psi(L(G)) = \bigcup_{i \in I} Z_i$ where:

- *I is a set of indices with $\#I = O(h^2)$*
- *$Z_i = \bigcup_{\alpha_0 \in W_i} \{\alpha_0 + \alpha_{1,i} n + \alpha_{2,i} m \mid n, m \geq 0\}$*
- *$W_i \subseteq \mathbb{N}^2$ is finite*
- *integers in W_i, $\alpha_{1,i}$, $\alpha_{2,i}$ do not exceed 2^{h^c}, where $c > 0$*

From sets Z_i it is possible to derive "small" DFAs and, by standard constructions, the DFA A s.t. $L(A) =_\pi L(G)$

Second Contribution: Binary Context-Free Languages

Theorem

Let G grammar in Chomsky normal form with h variables with a binary terminal alphabet.
Then there is a DFA A with at most $2^{h^{O(1)}}$ states s.t. $L(A) =_\pi L(G)$

The proof relies the following results:

Lemma ([Kopczyński&To '10])

For G as in the theorem, it holds that $\psi(L(G)) = \bigcup_{i \in I} Z_i$ where:

- *I is a set of indices with $\#I = O(h^2)$*
- *$Z_i = \bigcup_{\alpha_0 \in W_i} \{\alpha_0 + \alpha_{1,i} n + \alpha_{2,i} m \mid n, m \geq 0\}$*
- *$W_i \subseteq \mathbb{N}^2$ is finite*
- *integers in W_i, $\alpha_{1,i}$, $\alpha_{2,i}$ do not exceed 2^{h^c}, where $c > 0$*

From sets Z_i it is possible to derive "small" DFAs and, by standard constructions, the DFA A s.t. $L(A) =_\pi L(G)$

Second Contribution: Binary Context-Free Languages

Theorem

Let G grammar in Chomsky normal form with h variables with a binary terminal alphabet.
Then there is a DFA A with at most $2^{h^{O(1)}}$ states s.t. $L(A) =_\pi L(G)$

The proof relies the following results:

Lemma ([Kopczyński&To '10])

For G as in the theorem, it holds that $\psi(L(G)) = \bigcup_{i \in I} Z_i$ where:

- *I is a set of indices with $\#I = O(h^2)$*
- *$Z_i = \bigcup_{\alpha_0 \in W_i} \{\alpha_0 + \alpha_{1,i} n + \alpha_{2,i} m \mid n, m \geq 0\}$*
- *$W_i \subseteq \mathbb{N}^2$ is finite*
- *integers in W_i, $\alpha_{1,i}$, $\alpha_{2,i}$ do not exceed 2^{h^c}, where $c > 0$*

From sets Z_i it is possible to derive "small" DFAs and, by standard constructions, the DFA A s.t. $L(A) =_\pi L(G)$

Optimality

- For each CFG in Chomsky normal form with h variables we provided a Parikh equivalent DFA with $2^{h^{O(1)}}$ states in the following cases:
  - bounded languages
  - binary languages
- This upper bound cannot be reduced (consequence of the unary case)

Optimality

- For each CFG in Chomsky normal form with h variables we provided a Parikh equivalent DFA with $2^{h^{O(1)}}$ states in the following cases:
  - bounded languages
  - binary languages
- This upper bound cannot be reduced
  (consequence of the unary case)

Open Questions

Is it possible to extend these results to
all context-free languages?

- Bounded case
  crucial argument: it is enough to remember $\#\Sigma - 1$ variables
- Binary case
  the main lemma does not hold for alphabets with ≥ 3 letters

Other questions:

- What about *word bounded* CFLs?
  i.e., subsets of $w_1^* w_2^* \cdots w_p^*$ where each w_i is a string
- In our construction, the cost is double exponential in the size of the alphabet. State whether or not this is optimal.

Open Questions

Is it possible to extend these results to
all context-free languages?

- Bounded case
  crucial argument: it is enough to remember $\#\Sigma - 1$ variables
- Binary case
  the main lemma does not hold for alphabets with ≥ 3 letters

Other questions:

- What about *word bounded* CFLs?
  i.e., subsets of $w_1^* w_2^* \ldots w_m^*$, where each w_i is a string
- In our construction the cost is double exponential in the size of the alphabet: state whether or not this is optimal

Open Questions

Is it possible to extend these results to
all context-free languages?

- Bounded case
  crucial argument: it is enough to remember $\#\Sigma - 1$ variables
- Binary case
  the main lemma does not hold for alphabets with ≥ 3 letters

Other questions:

- What about *word bounded* CFLs?
  i.e., subsets of $w_1^* w_2^* \ldots w_m^*$, where each w_i is a string
- In our construction the cost is double exponential in the size of the alphabet: state whether or not this is optimal

Open Questions

Is it possible to extend these results to
all context-free languages?

- Bounded case
  crucial argument: it is enough to remember $\#\Sigma - 1$ variables
- Binary case
  the main lemma does not hold for alphabets with ≥ 3 letters

Other questions:

- What about *word bounded* CFLs?
  i.e., subsets of $w_1^* w_2^* \dots w_m^*$, where each w_i is a string
- In our construction the cost is double exponential in the size of the alphabet: state whether or not this is optimal

Open Questions

Is it possible to extend these results to
all context-free languages?

- Bounded case
  crucial argument: it is enough to remember $\#\Sigma - 1$ variables
- Binary case
  the main lemma does not hold for alphabets with ≥ 3 letters

Other questions:

- What about *word bounded* CFLs?
  i.e., subsets of $w_1^* w_2^* \dots w_m^*$, where each w_i is a string
- In our construction the cost is double exponential in the size of the alphabet: state whether or not this is optimal