

Expressive power of pebble automata
or
Unary and positive transitive closure logic on
trees

Bonn
June 2006

Thomas Schwentick

Joint work with

Mikołaj Bojańczyk, Mathias Samuelides, Luc Segoufin

Contents

▷ **Introduction and Overview of Results**

The Behavior of Pebble Automata

On Strong Pebbles

Hierarchy Theorems

Conclusion

Some results...

... on pebble automata

-
- —
-
-
-

... on transitive closure logic

Some results...

... on pebble automata

- $\text{PA} \subsetneq \text{REG}$
- For each $n \geq 0$,
 - $\text{PA}_n \subsetneq \text{PA}_{n+1}$ and $\text{DPA}_n \subsetneq \text{DPA}_{n+1}$
 - $\text{TWA} \not\subseteq \text{DPA}_n$
 - $\text{sPA}_n = \text{PA}_n$ and $\text{sDPA}_n = \text{DPA}_n$
 - sDPA_n is closed under complement

... on transitive closure logic

$\text{FO+posTC}^1 \subsetneq \text{MSO}$ on binary trees

Some results...

... on pebble automata

- $\text{PA} \subsetneq \text{REG}$
- For each $n \geq 0$,
 - $\text{PA}_n \subsetneq \text{PA}_{n+1}$ and $\text{DPA}_n \subsetneq \text{DPA}_{n+1}$
 - $\text{TWA} \not\subseteq \text{DPA}_n$
 - $\text{sPA}_n = \text{PA}_n$ and $\text{sDPA}_n = \text{DPA}_n$
 - sDPA_n is closed under complement

... on transitive closure logic

$\text{FO+posTC}^1 \subsetneq \text{MSO on binary trees}$

Some results...

... on pebble automata

- $\text{PA} \subsetneq \text{REG}$
- For each $n \geq 0$,
 - $\text{PA}_n \subsetneq \text{PA}_{n+1}$ and $\text{DPA}_n \subsetneq \text{DPA}_{n+1}$
 - $\text{TWA} \not\subseteq \text{DPA}_n$
 - $\text{sPA}_n = \text{PA}_n$ and $\text{sDPA}_n = \text{DPA}_n$
 - sDPA_n is closed under complement

... on transitive closure logic

$\text{FO+posTC}^1 \subsetneq \text{MSO}$ on binary trees

Proof sketch

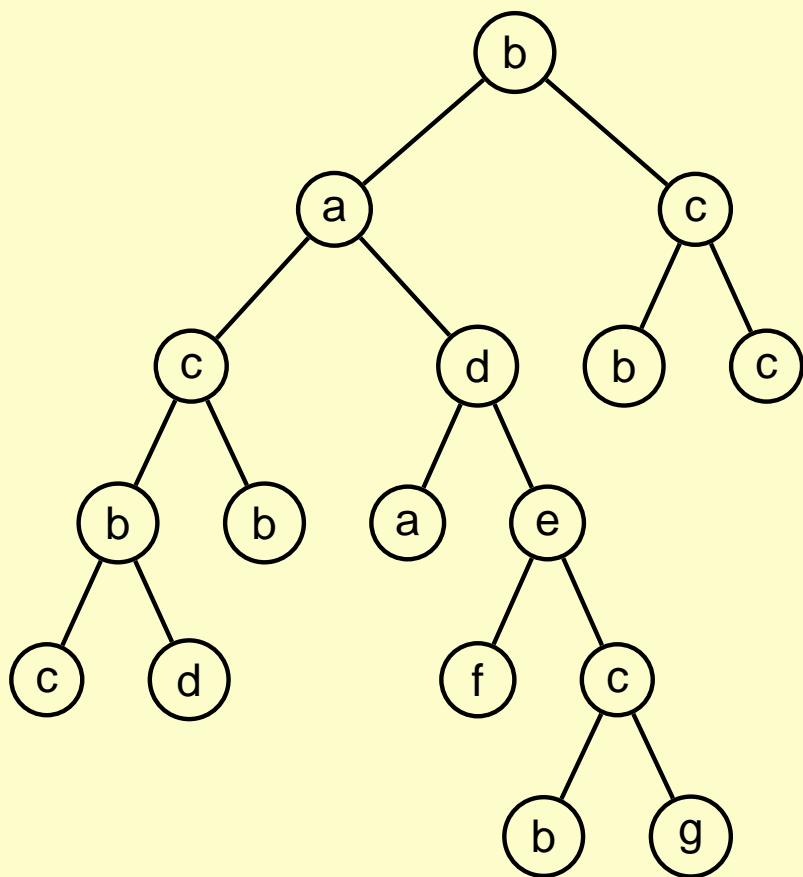
... in 9 steps

Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

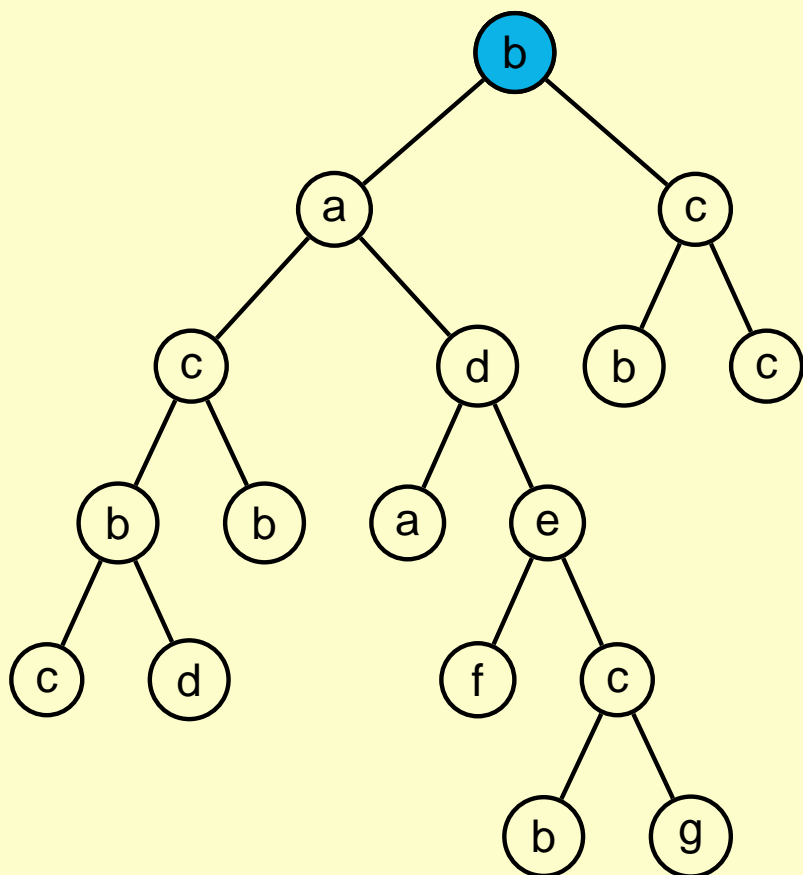


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

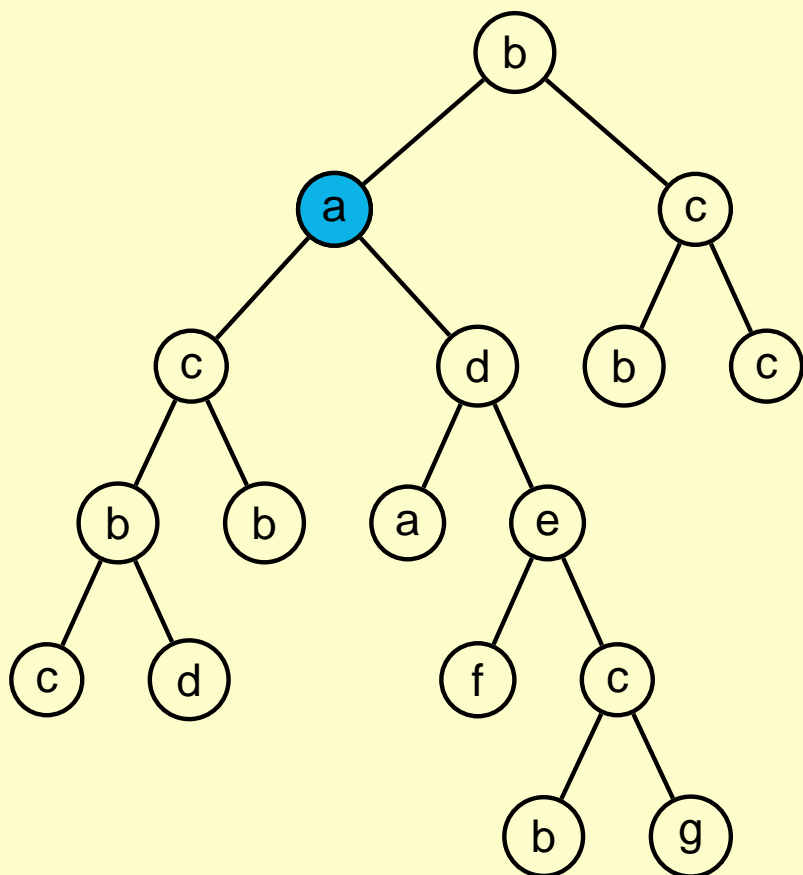


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

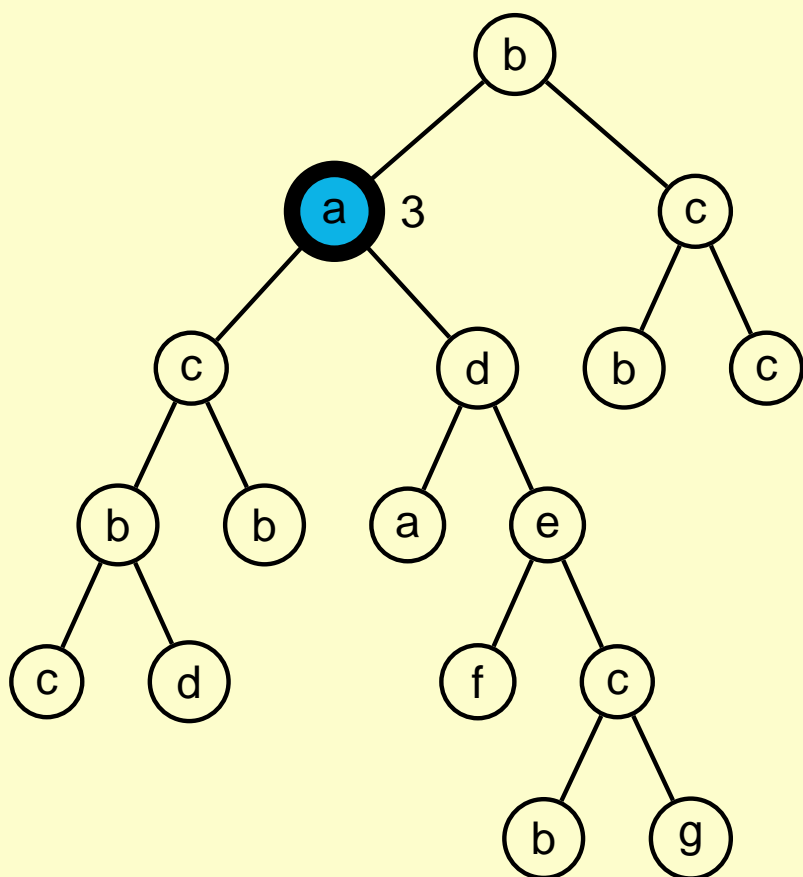


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

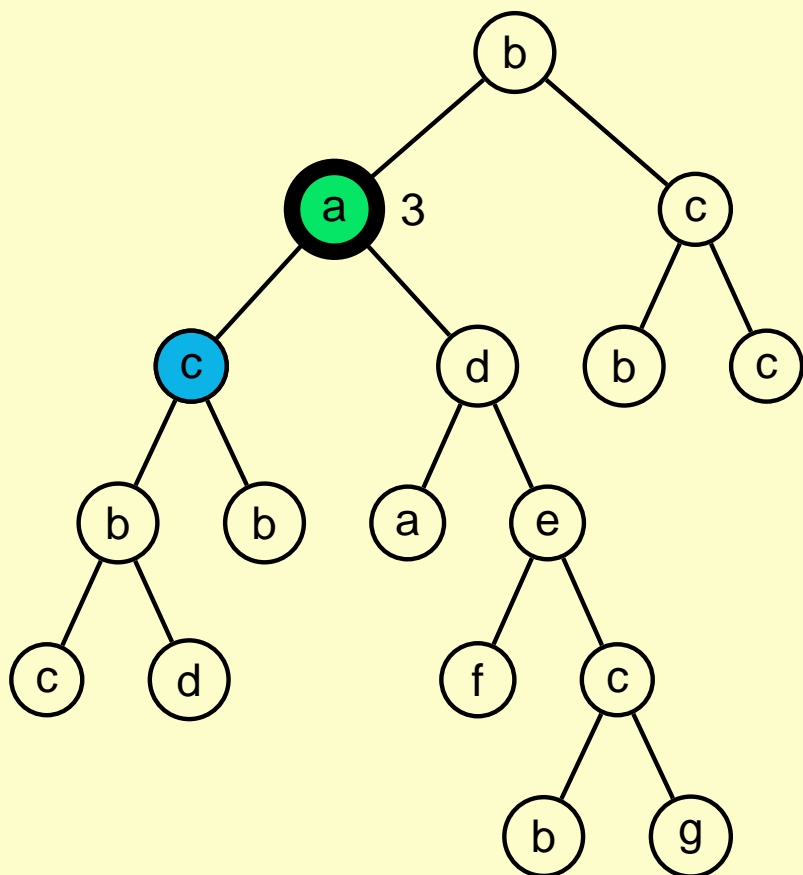


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

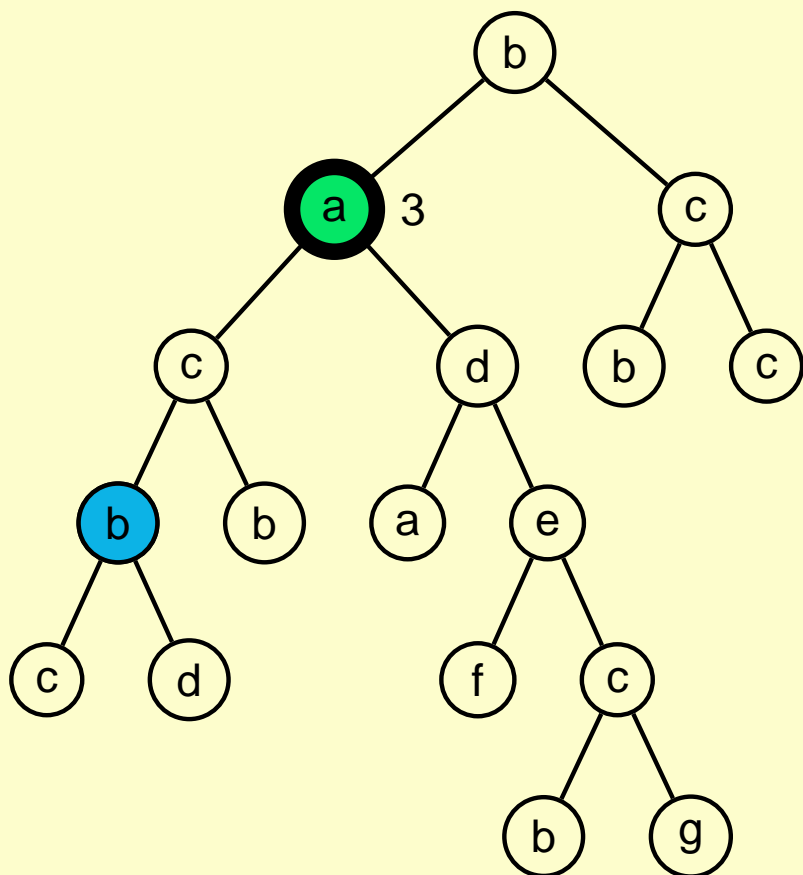


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

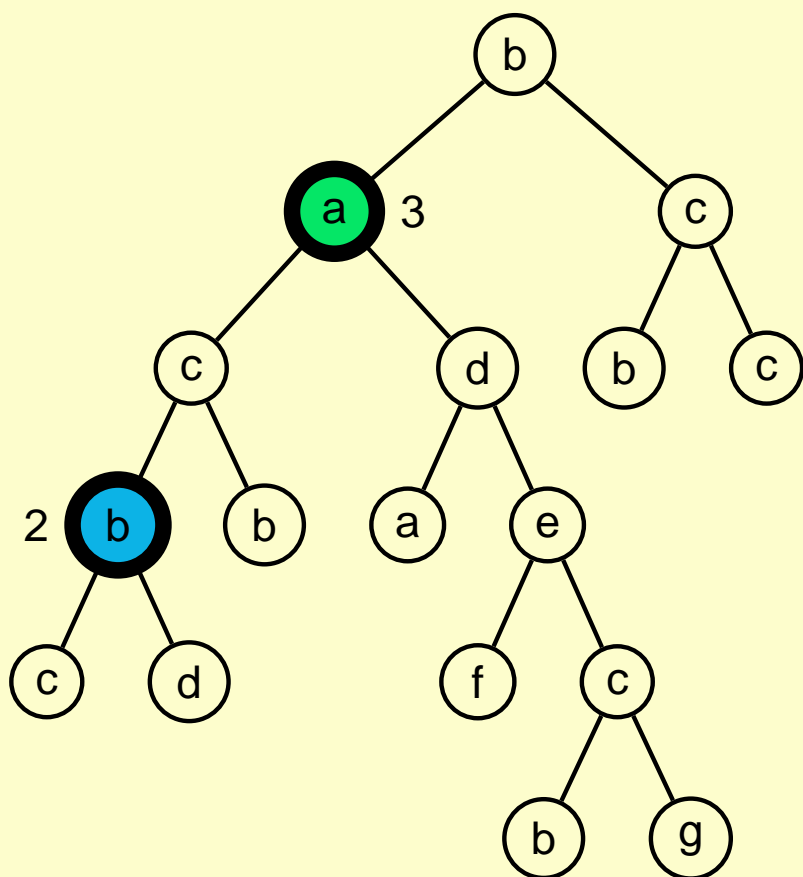


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

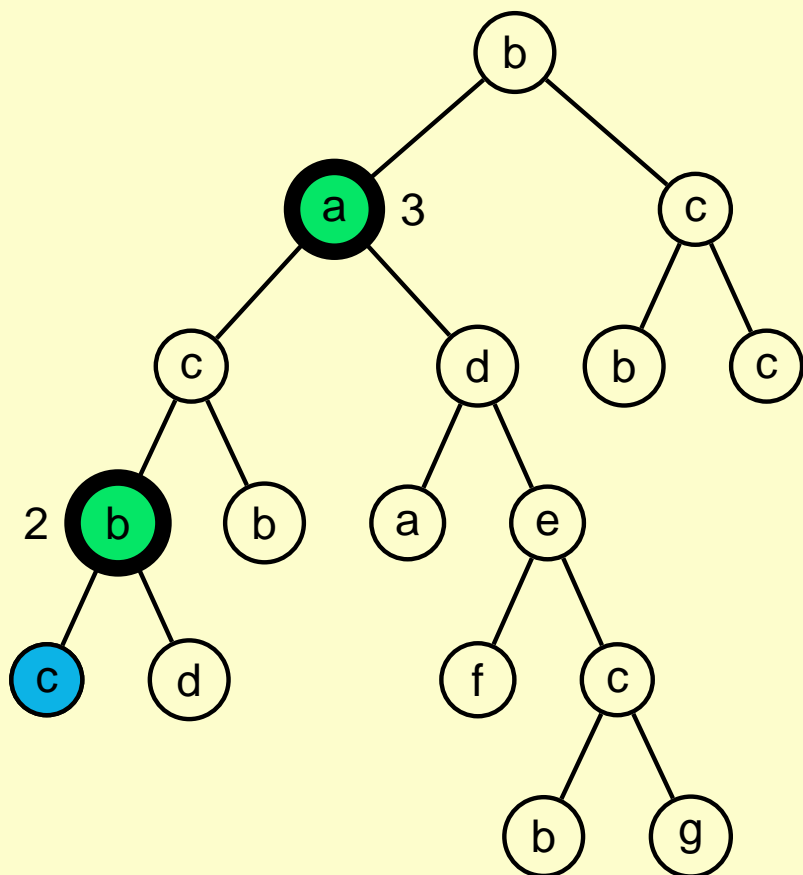


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

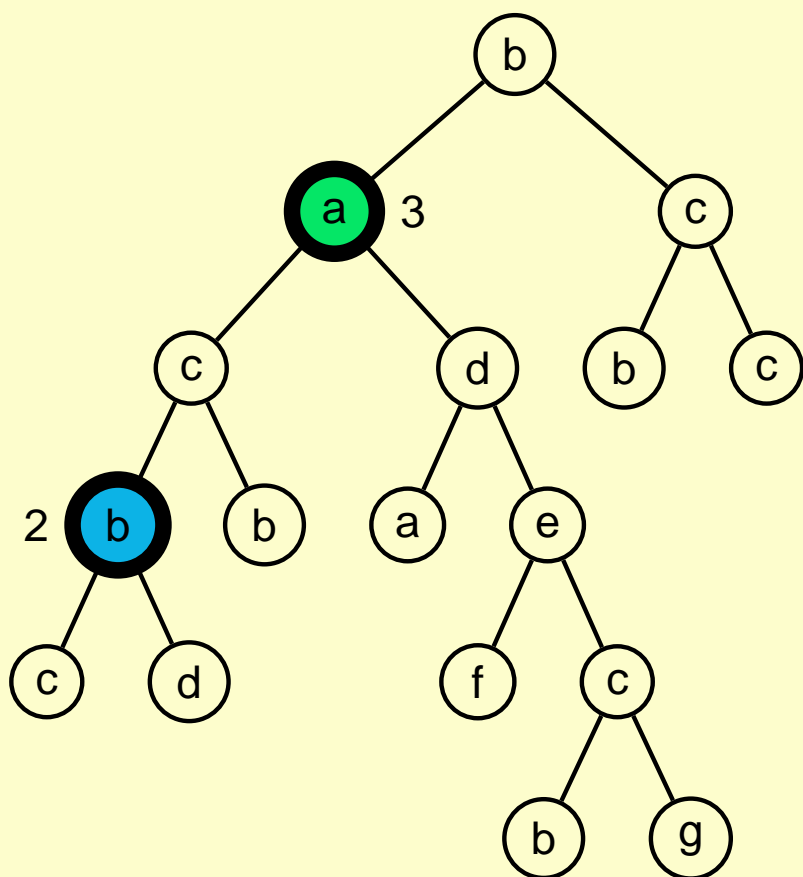


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

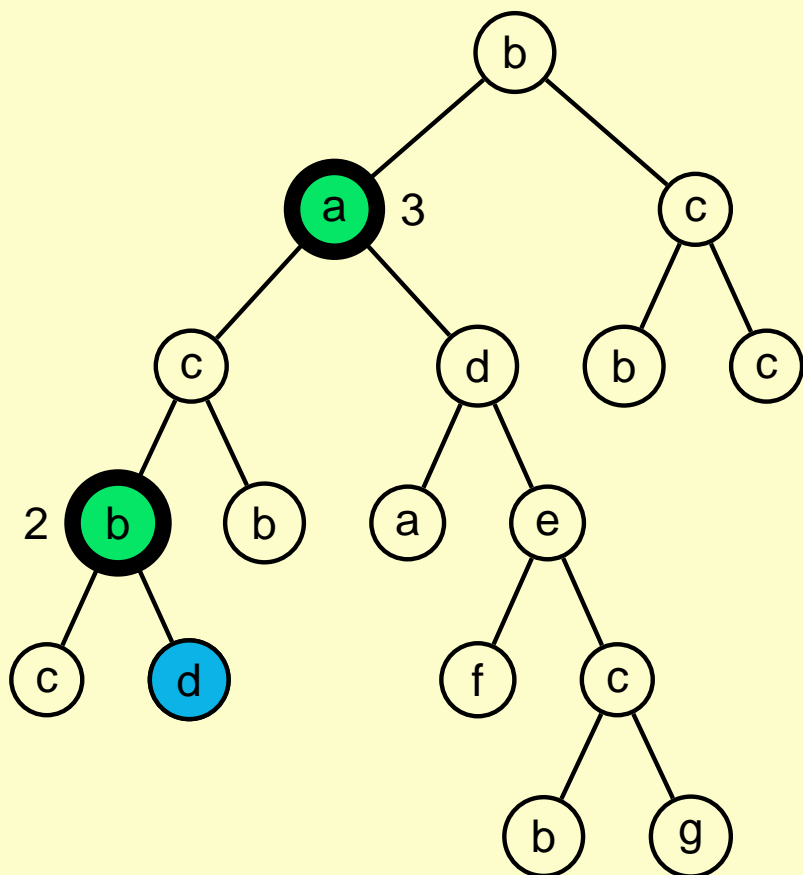


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

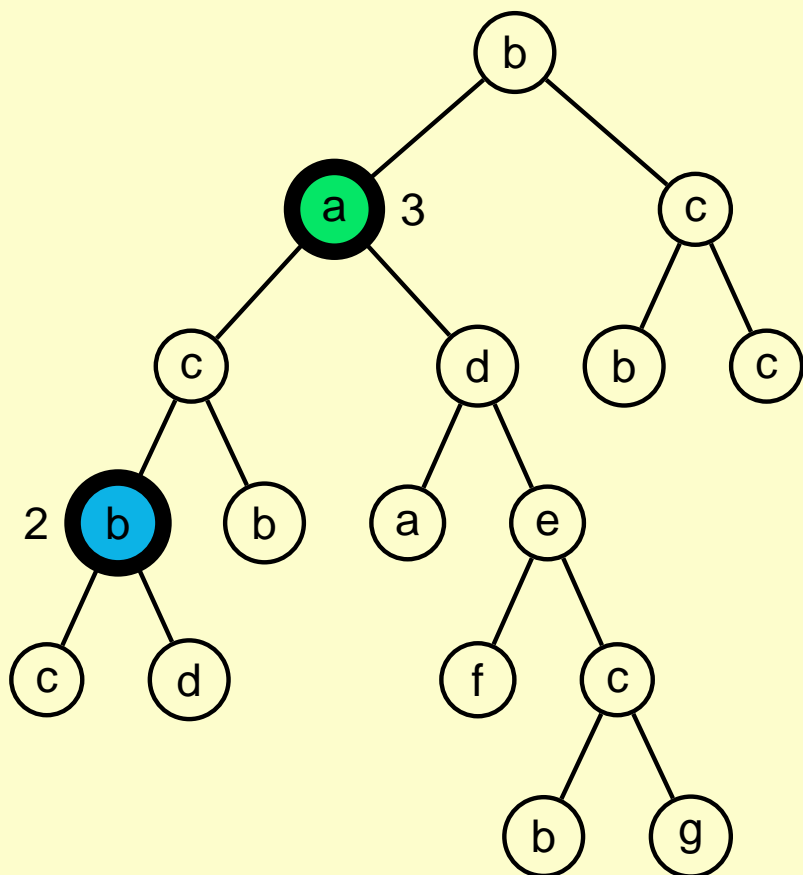


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

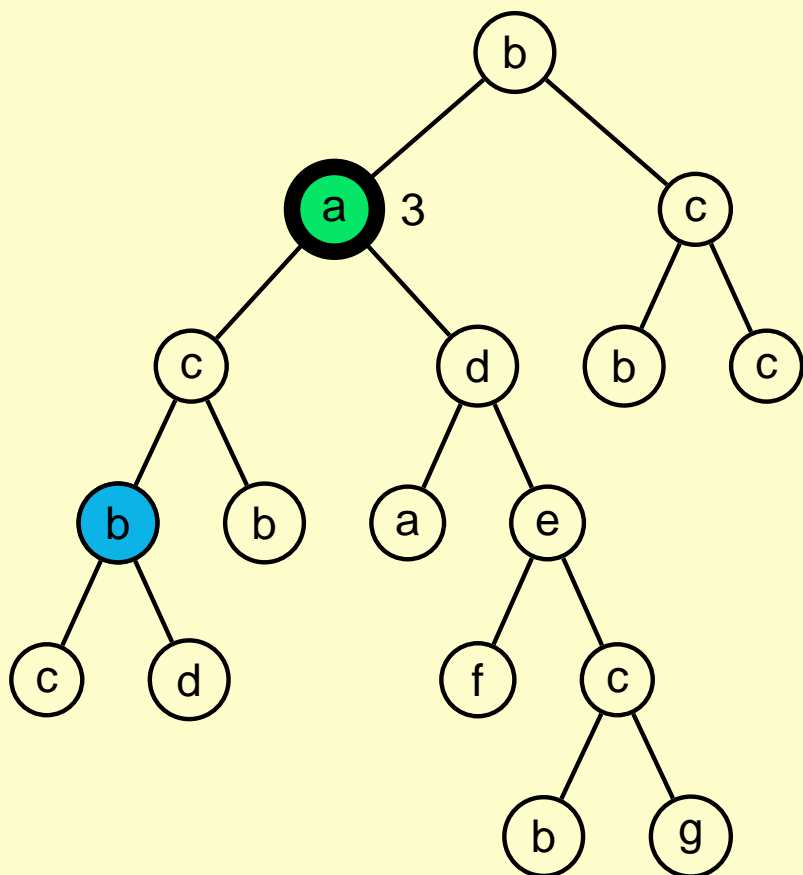


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

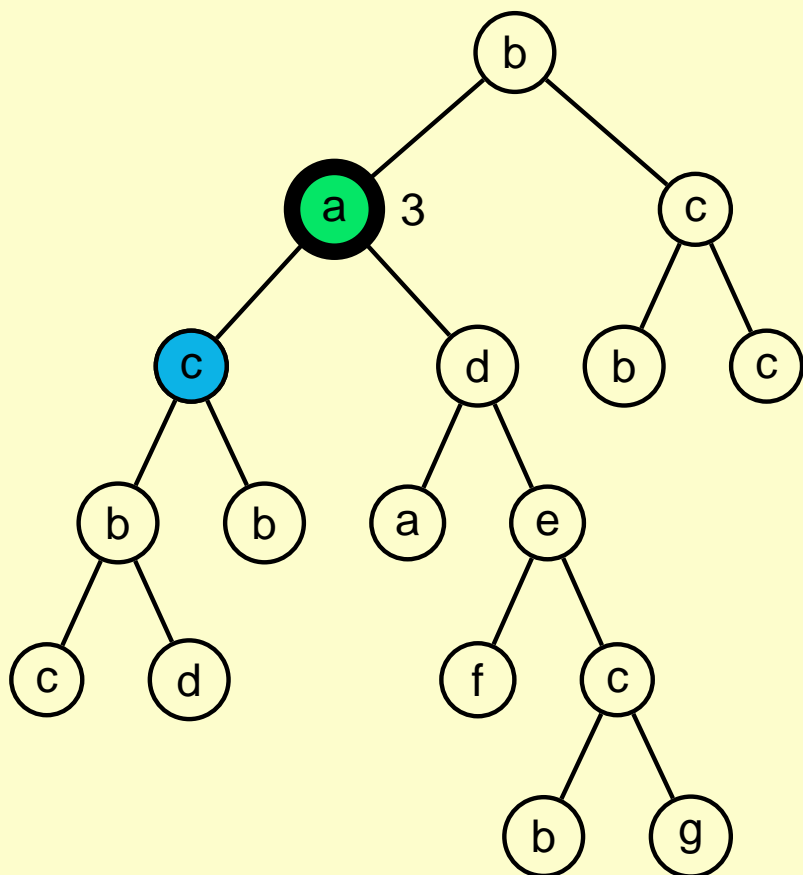


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

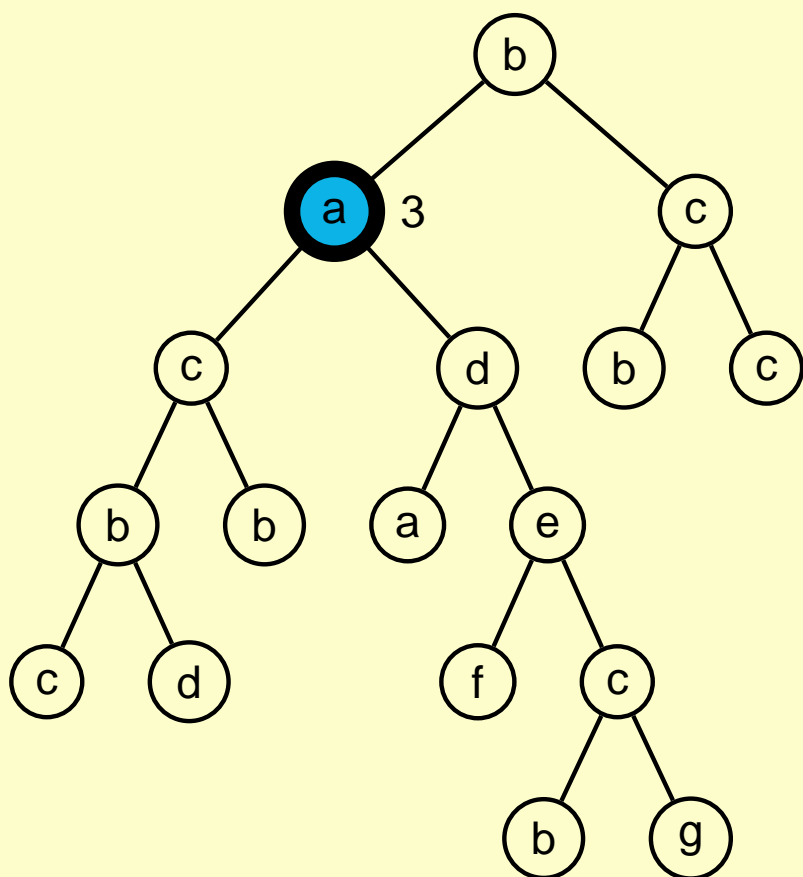


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

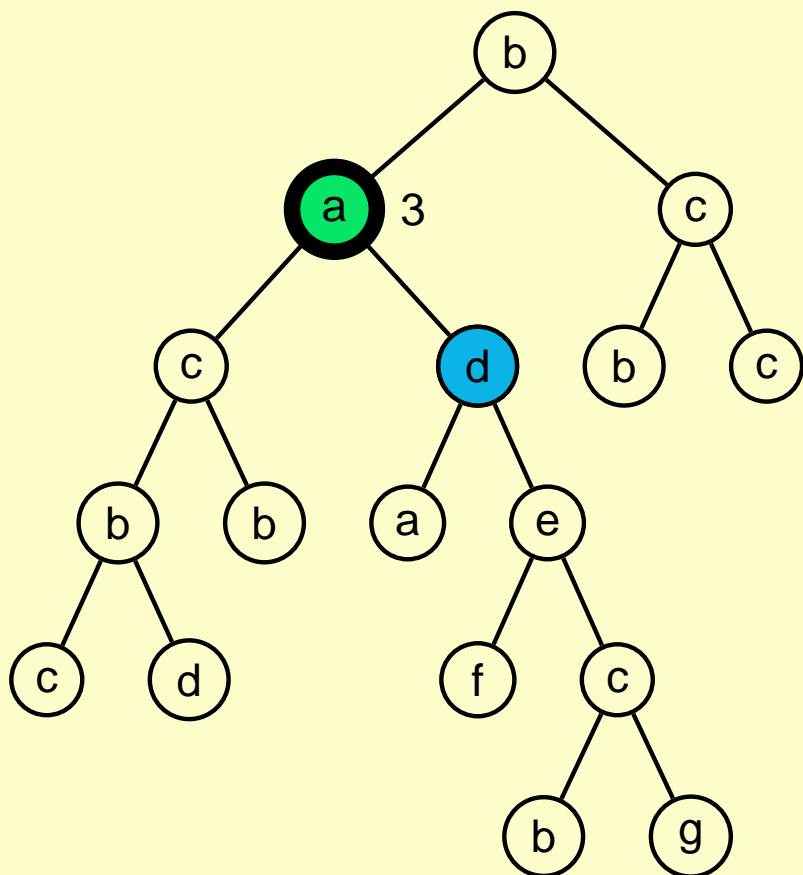


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

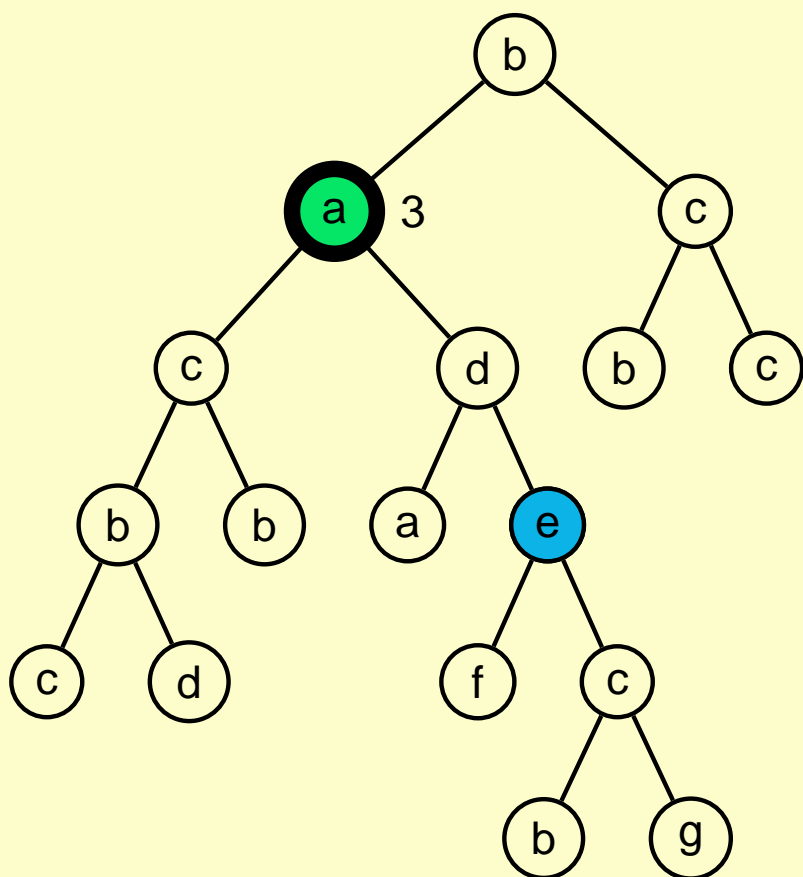


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

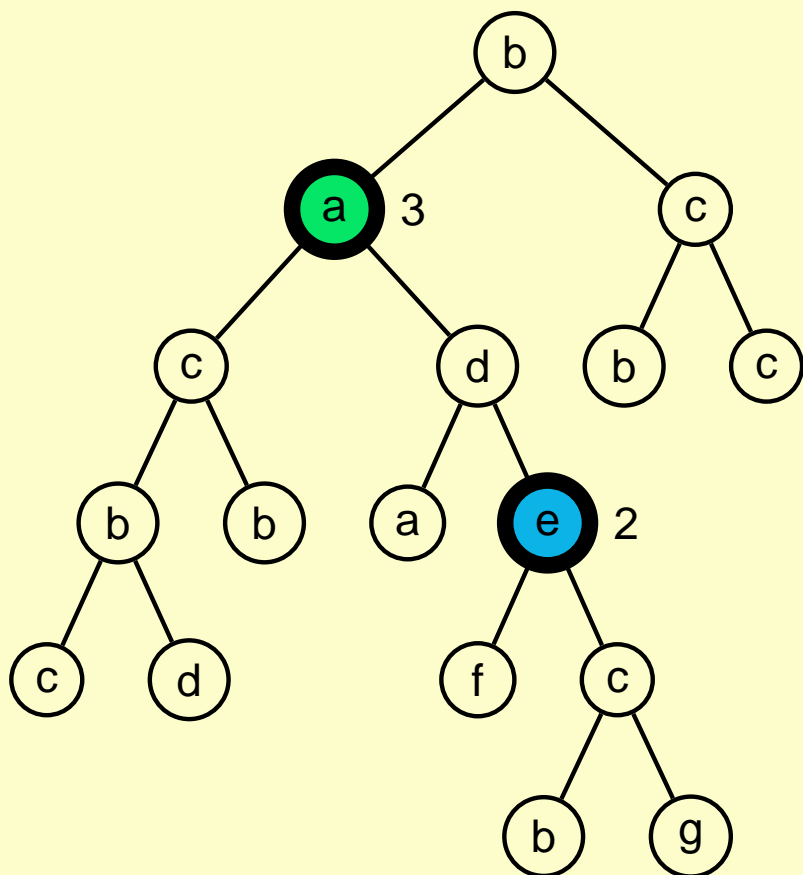


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

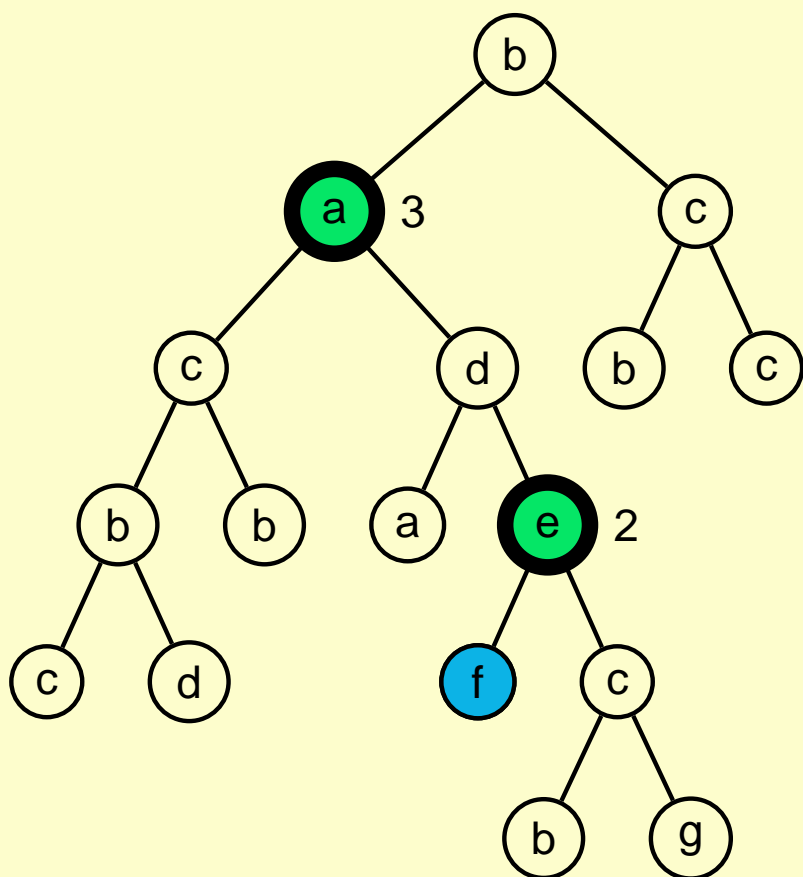


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

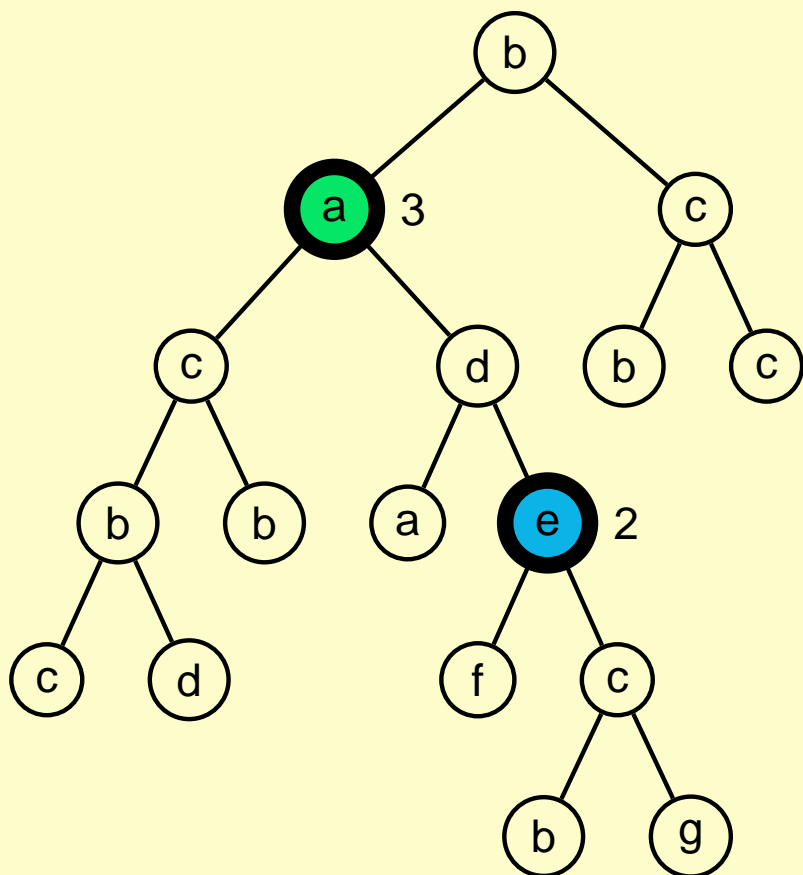


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

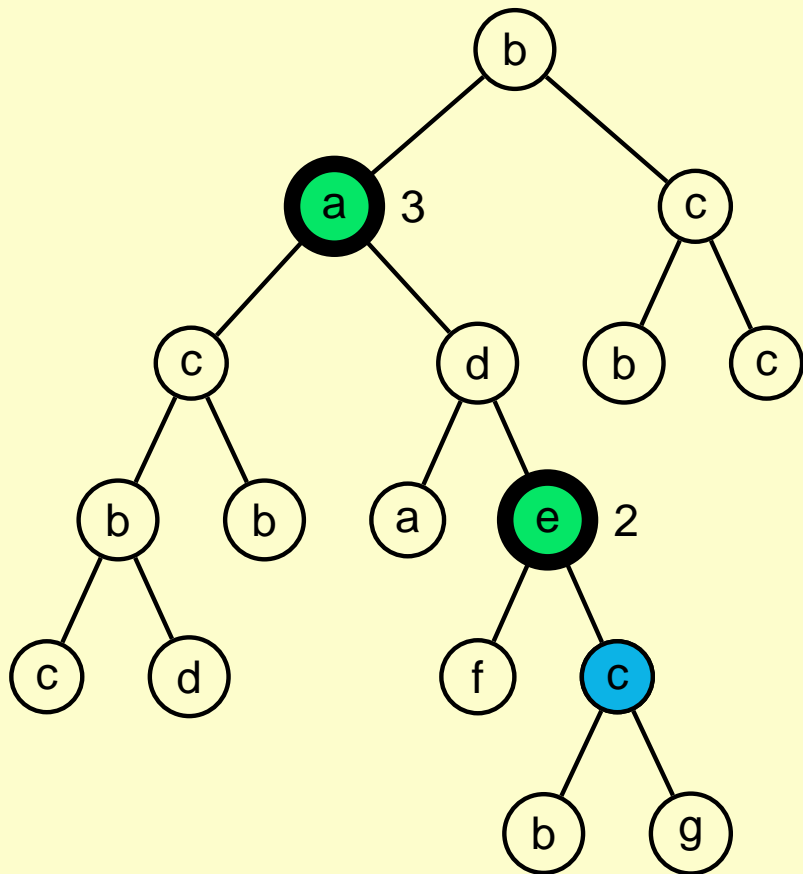


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

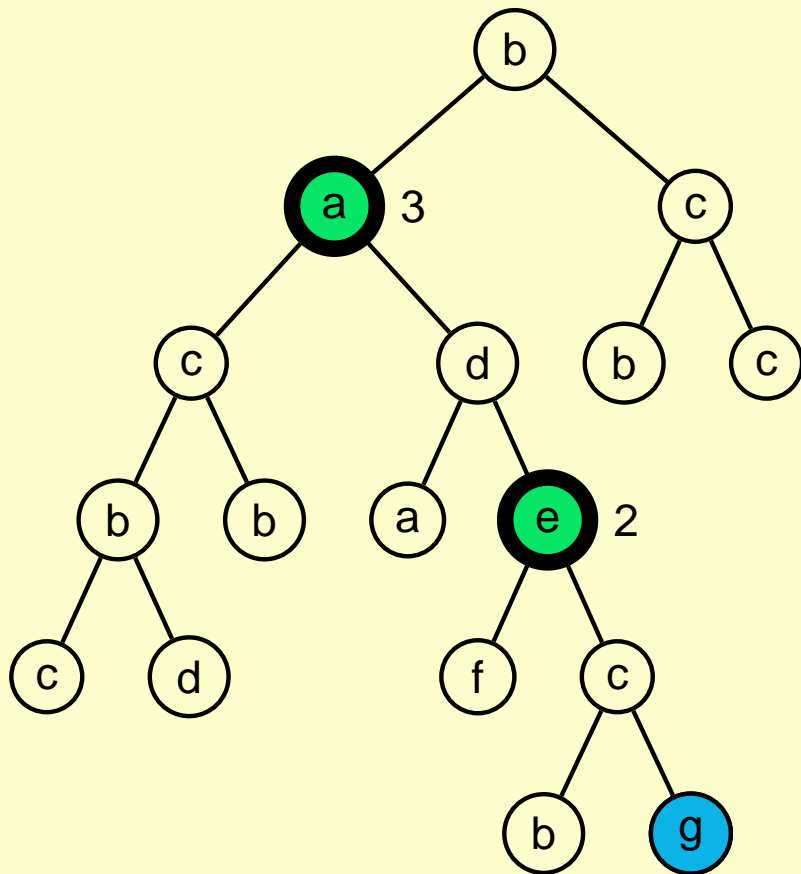


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

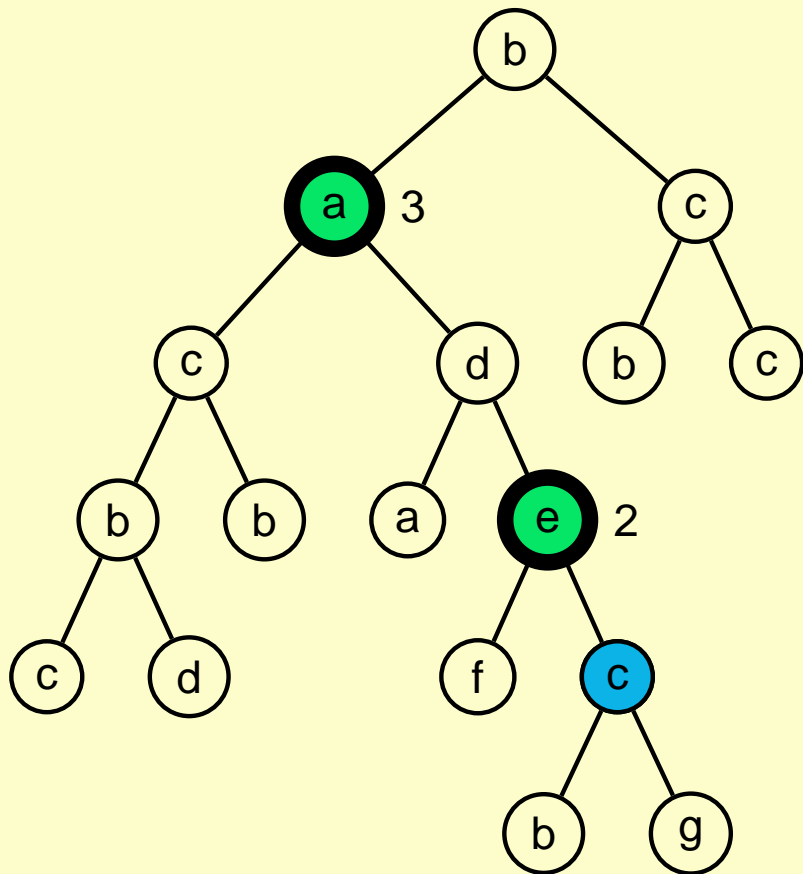


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

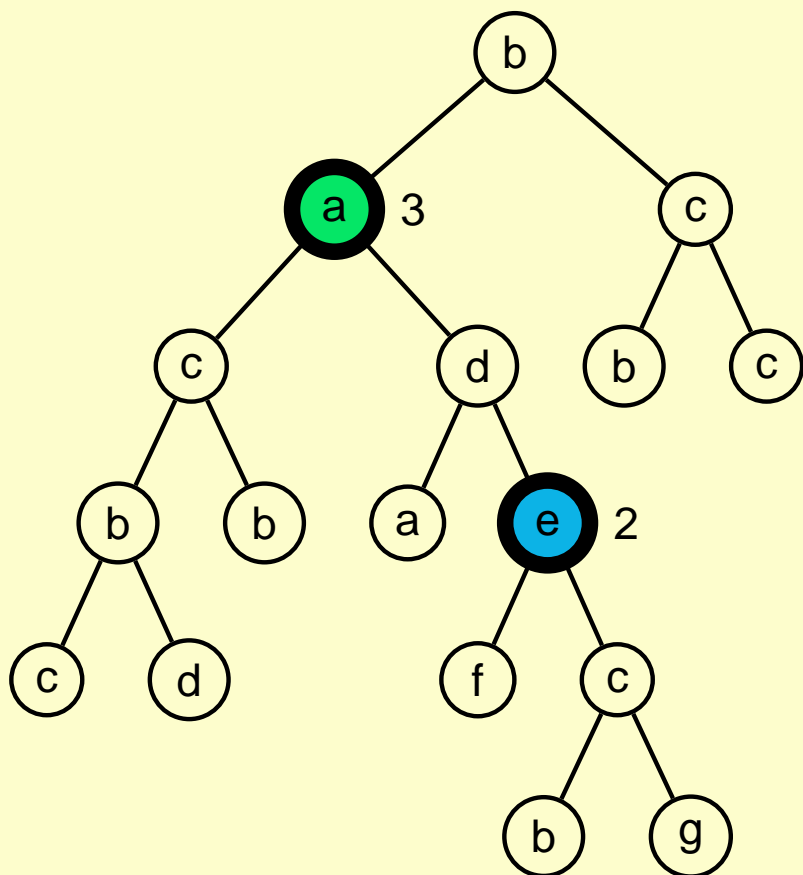


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

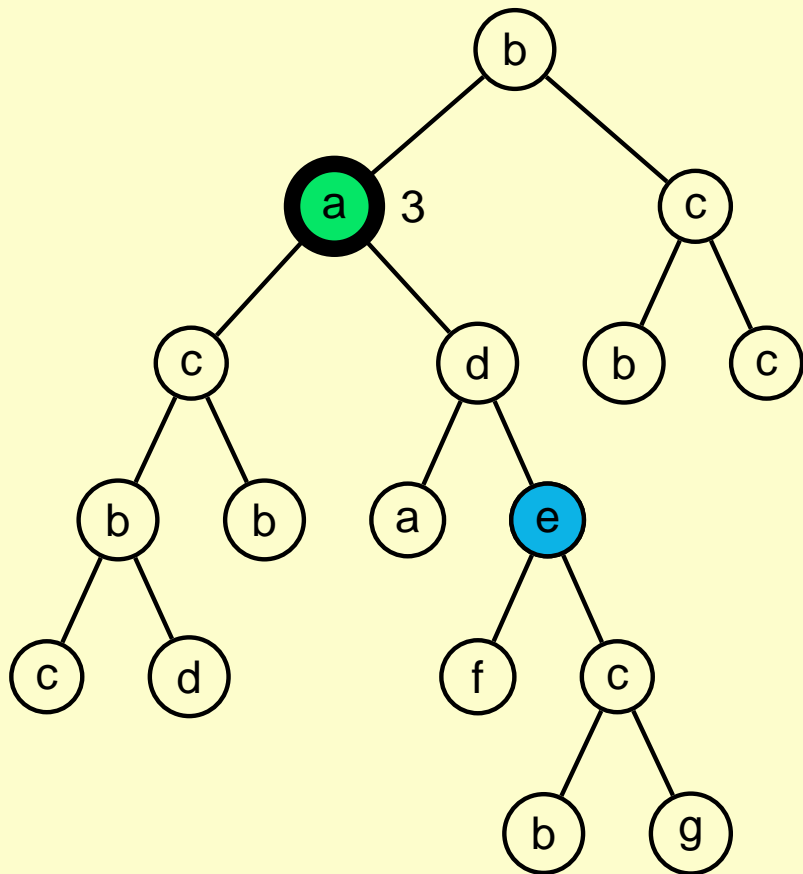


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

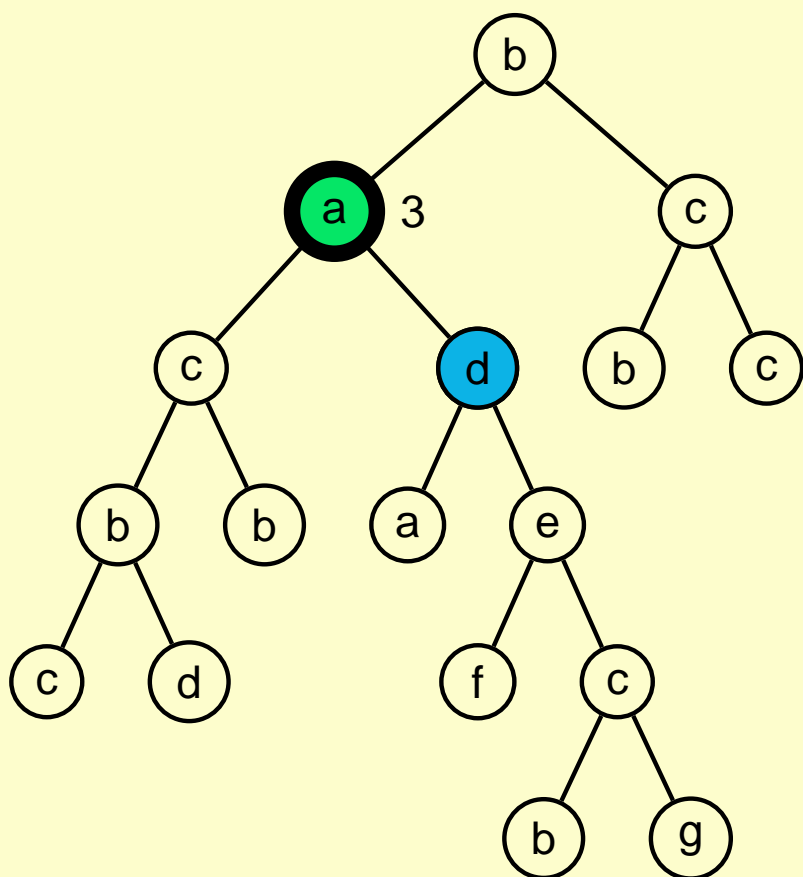


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

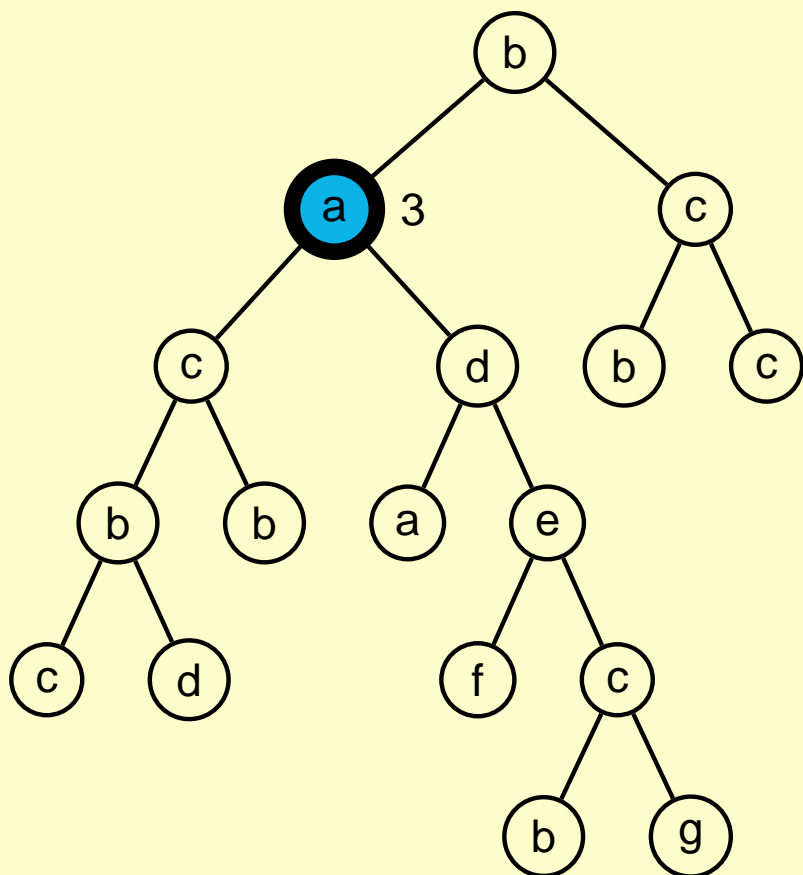


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

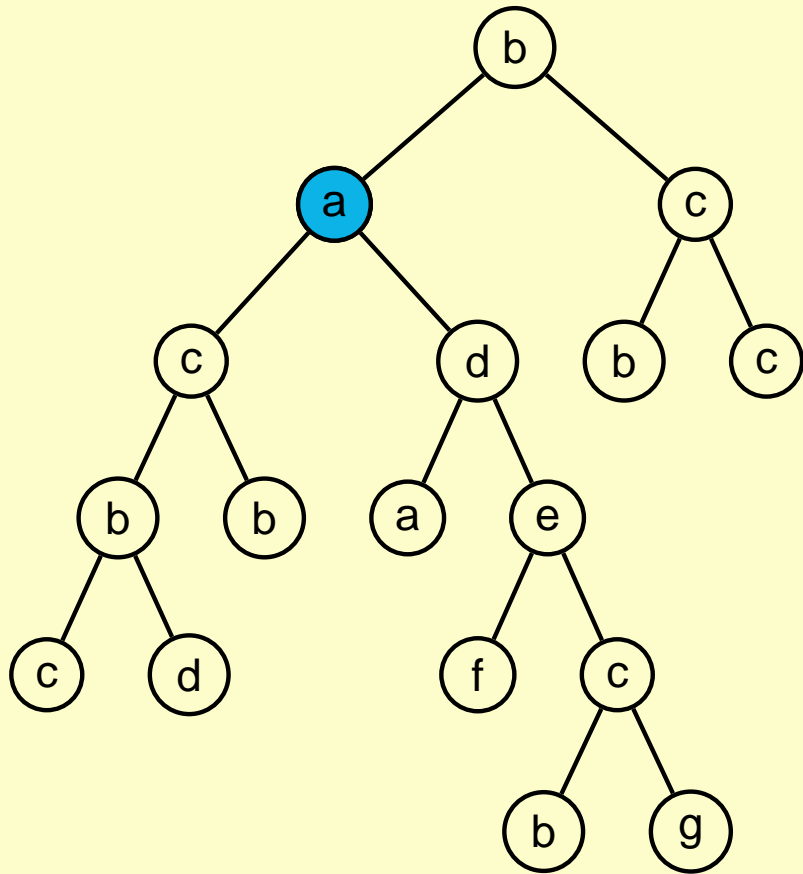


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

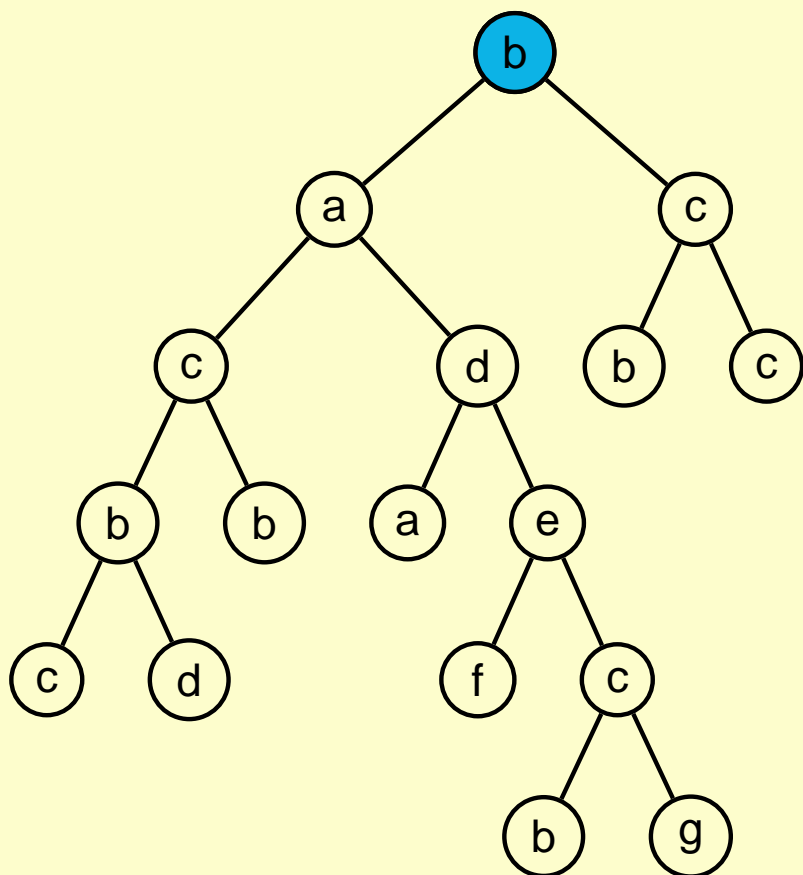


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$

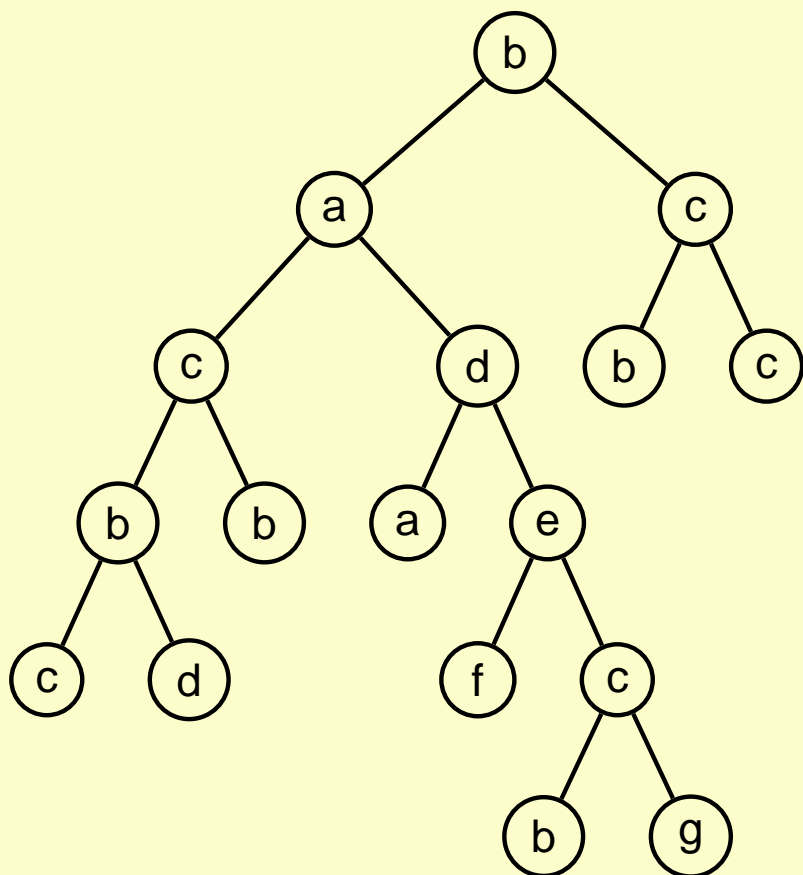


Pebble automata

Definition

Pebble automata :

- states Q
- pebbles $n, \dots, 1$
- Possible moves:
 $D = \{\uparrow, \searrow, \swarrow, \text{stay}, \text{drop}, \text{lift}\}$
- Depending on
 - state,
 - minimum pebble number,
 - set of pebbles at current node,
 - label,
 - type of node:
 $\{\text{root}, \text{leftchild}, \text{rightchild}\} \times \{\text{leaf}, \text{innernode}\}$



Step 1: pebble automata and logic

Theorem [Engelfriet, Hoogeboom 06]

● $\text{FO} + \text{DTC}^1 = \text{sDPA}$

● $\text{FO} + \text{posTC}^1 = \text{sPA}$

(They prove a stronger result for multihead-automata)

Proof idea

“Logic \Rightarrow automaton” requires a liberal pebble lifting policy:

- To check $\text{TC}_{x,y}[\varphi(x, y, \vec{z})](u, v)$:
 - Pebbles n, \dots, k on \vec{z} do not move
 - Pebble $k - 1$ on v , pebble $k - 2$ on u
 - Guess u_1 by placing pebble $k - 3$ on it
 - Check $\varphi(u, u_1, \vec{z})$ recursively
 - Go back to u_1 , lift pebbles $k - 2, k - 3$
 - Put pebble $k - 2$ on u_1
 - Continue with $u_2 \dots$

→ **strong pebbles**: the minimum pebble can be lifted even if the head is somewhere else

Step 1: pebble automata and logic

Theorem [Engelfriet, Hoogeboom 06]

- $\text{FO} + \text{DTC}^1 = \text{sDPA}$
- $\text{FO} + \text{posTC}^1 = \text{sPA}$

(They prove a stronger result for multihead-automata)

Proof idea

“Logic \Rightarrow automaton” requires a liberal pebble lifting policy:

- To check $\text{TC}_{x,y}[\varphi(x, y, \vec{z})](u, v)$:
 - Pebbles n, \dots, k on \vec{z} do not move
 - Pebble $k - 1$ on v , pebble $k - 2$ on u
 - Guess u_1 by placing pebble $k - 3$ on it
 - Check $\varphi(u, u_1, \vec{z})$ recursively
 - Go back to u_1 , lift pebbles $k - 2, k - 3$
 - Put pebble $k - 2$ on u_1
 - Continue with $u_2 \dots$

→ **strong pebbles**: the minimum pebble can be lifted even if the head is somewhere else

Agenda

- Show that strong pebbles are not stronger than “normal” ones

Step 1: pebble automata and logic

Theorem [Engelfriet, Hoogeboom 06]

- $\text{FO} + \text{DTC}^1 = \text{sDPA}$
- $\text{FO} + \text{posTC}^1 = \text{sPA}$

(They prove a stronger result for multihead-automata)

Proof idea

“Logic \Rightarrow automaton” requires a liberal pebble lifting policy:

- To check $\text{TC}_{x,y}[\varphi(x, y, \vec{z})](u, v)$:
 - Pebbles n, \dots, k on \vec{z} do not move
 - Pebble $k - 1$ on v , pebble $k - 2$ on u
 - Guess u_1 by placing pebble $k - 3$ on it
 - Check $\varphi(u, u_1, \vec{z})$ recursively
 - Go back to u_1 , lift pebbles $k - 2, k - 3$
 - Put pebble $k - 2$ on u_1
 - Continue with $u_2 \dots$

→ **strong pebbles**: the minimum pebble can be lifted even if the head is somewhere else

Agenda

- Show that strong pebbles are not stronger than “normal” ones
- Show that “normal” pebble automata do not capture all regular tree languages

Step 1: pebble automata and logic

Theorem [Engelfriet, Hoogeboom 06]

- $\text{FO} + \text{DTC}^1 = \text{sDPA}$
- $\text{FO} + \text{posTC}^1 = \text{sPA}$

(They prove a stronger result for multihead-automata)

Proof idea

“Logic \Rightarrow automaton” requires a liberal pebble lifting policy:

- To check $\text{TC}_{x,y}[\varphi(x, y, \vec{z})](u, v)$:
 - Pebbles n, \dots, k on \vec{z} do not move
 - Pebble $k - 1$ on v , pebble $k - 2$ on u
 - Guess u_1 by placing pebble $k - 3$ on it
 - Check $\varphi(u, u_1, \vec{z})$ recursively
 - Go back to u_1 , lift pebbles $k - 2, k - 3$
 - Put pebble $k - 2$ on u_1
 - Continue with $u_2 \dots$

\rightarrow **strong pebbles**: the minimum pebble can be lifted even if the head is somewhere else

Agenda

- Show that strong pebbles are not stronger than “normal” ones
 - Show that “normal” pebble automata do not capture all regular tree languages
- \rightarrow first learn more about pebble automata

Contents

Introduction and Overview of Results

▷ **The Behavior of Pebble Automata**

On Strong Pebbles

Hierarchy Theorems

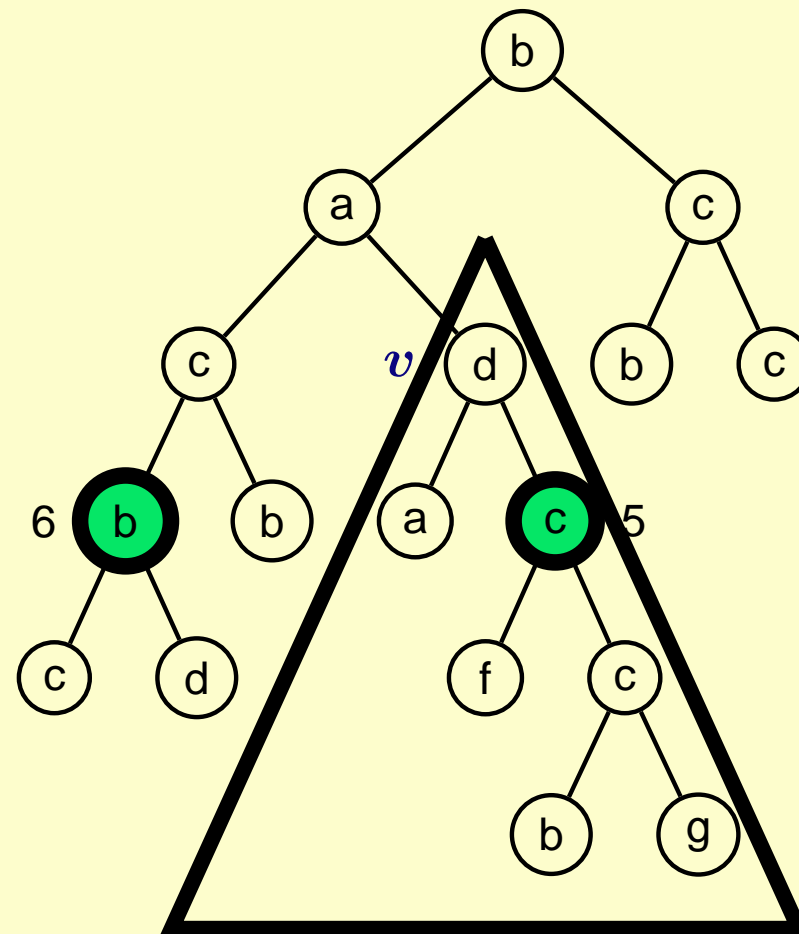
Conclusion

Step 2: tree loops...

We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?



Step 2: tree loops...

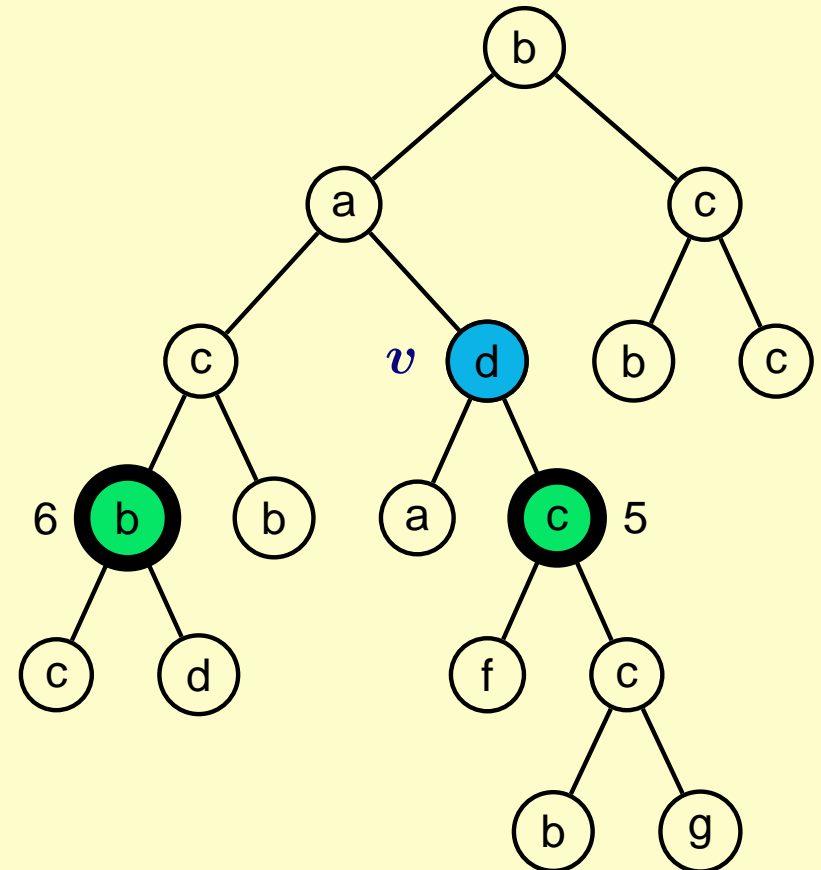
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

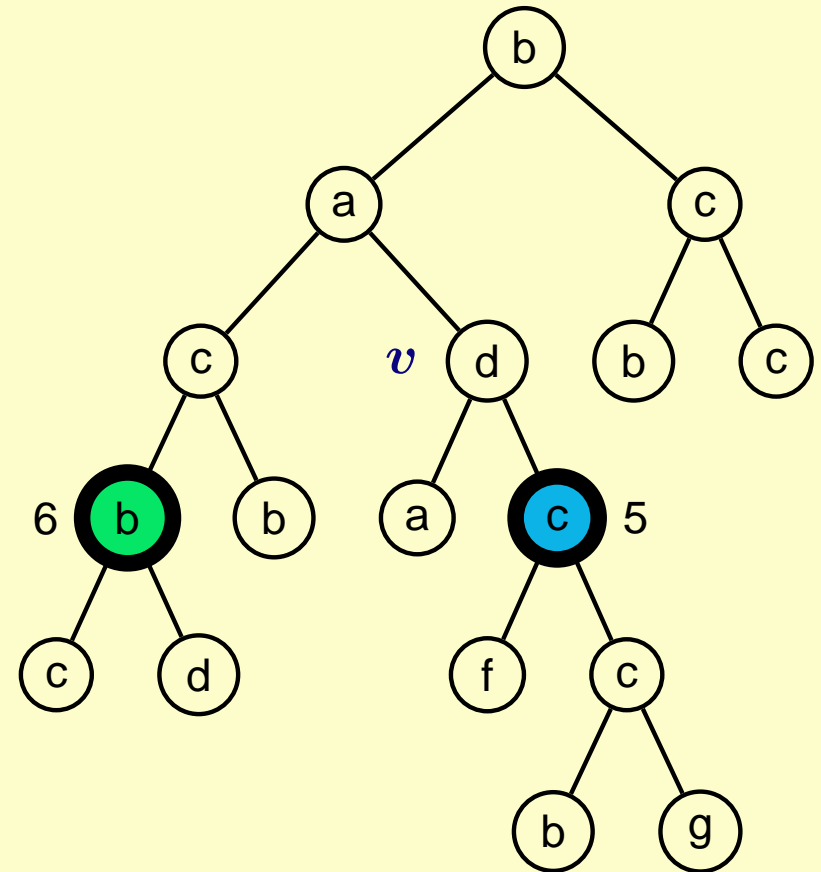
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

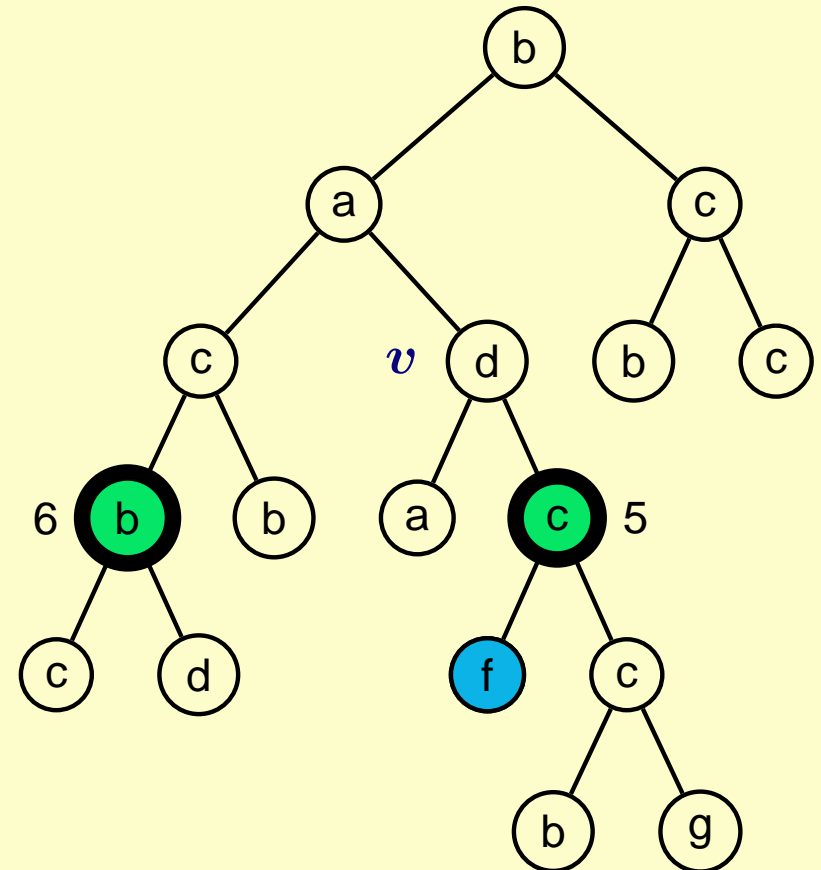
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

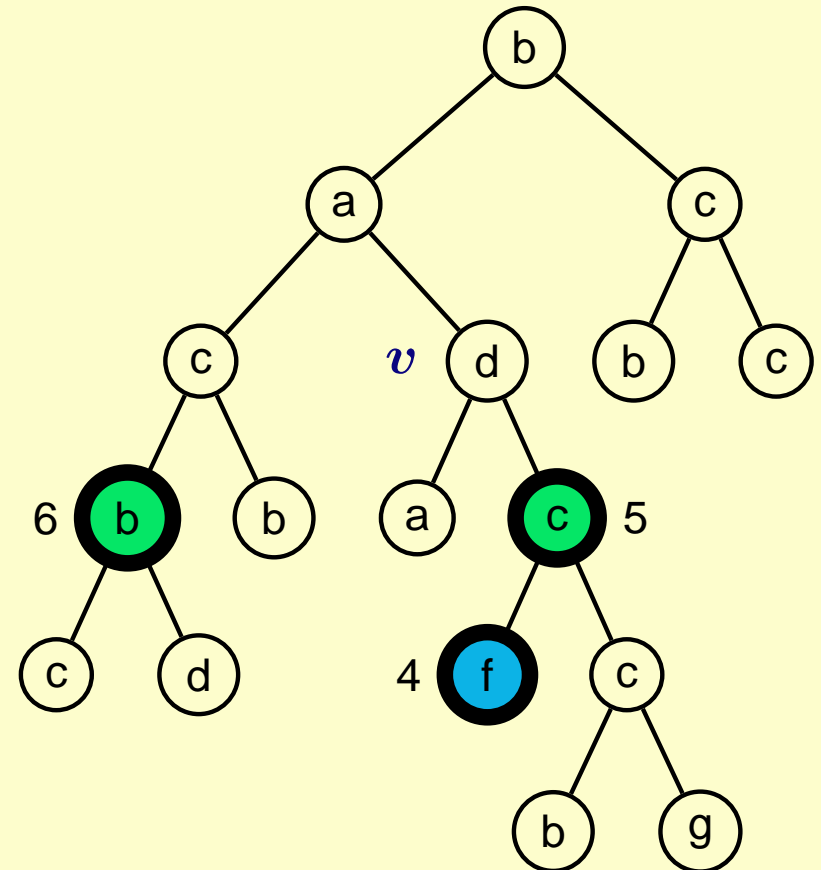
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

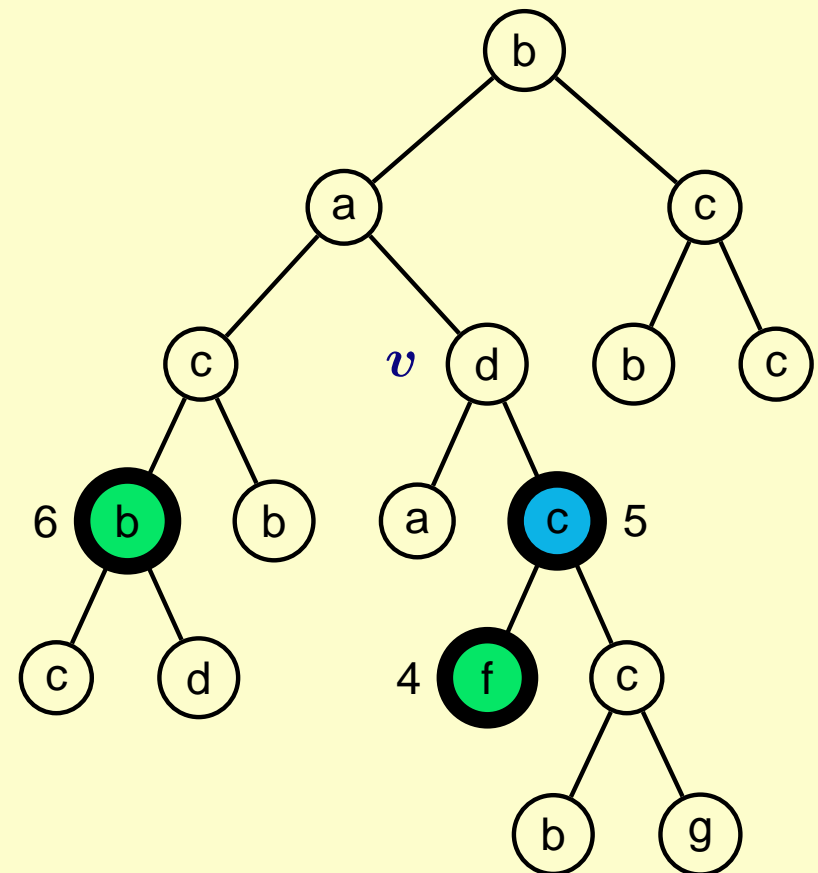
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

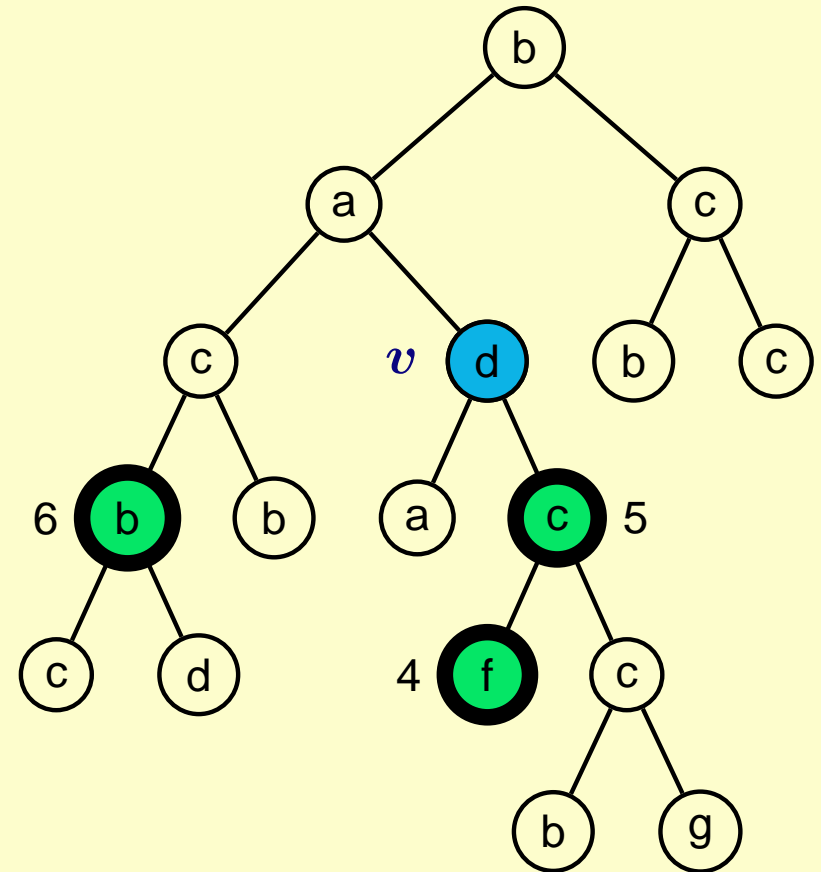
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

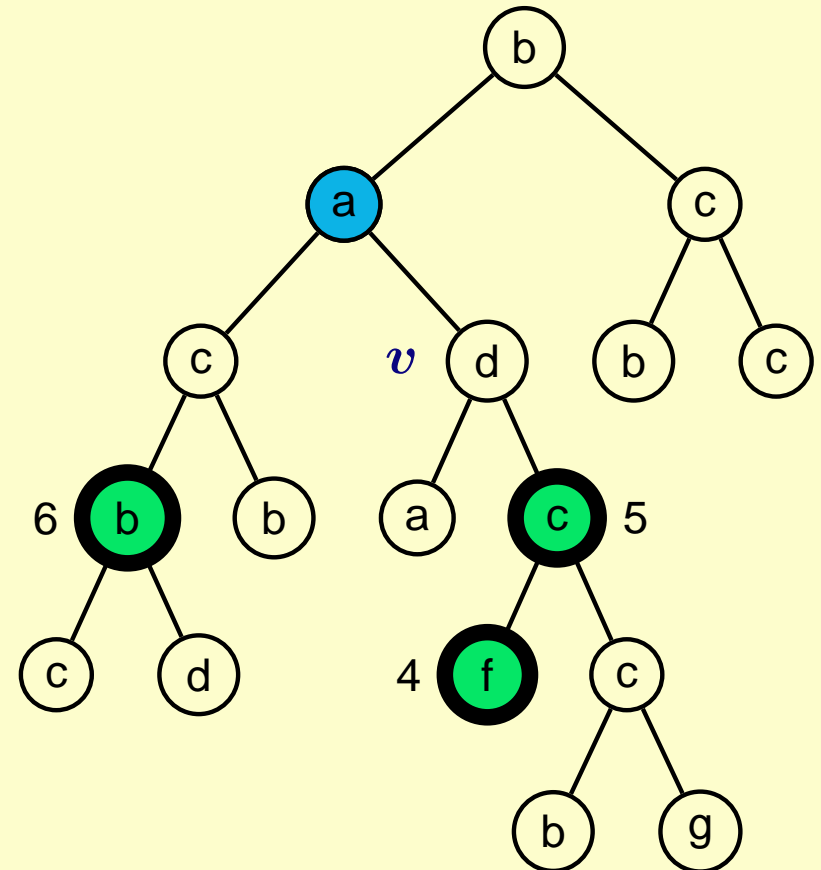
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

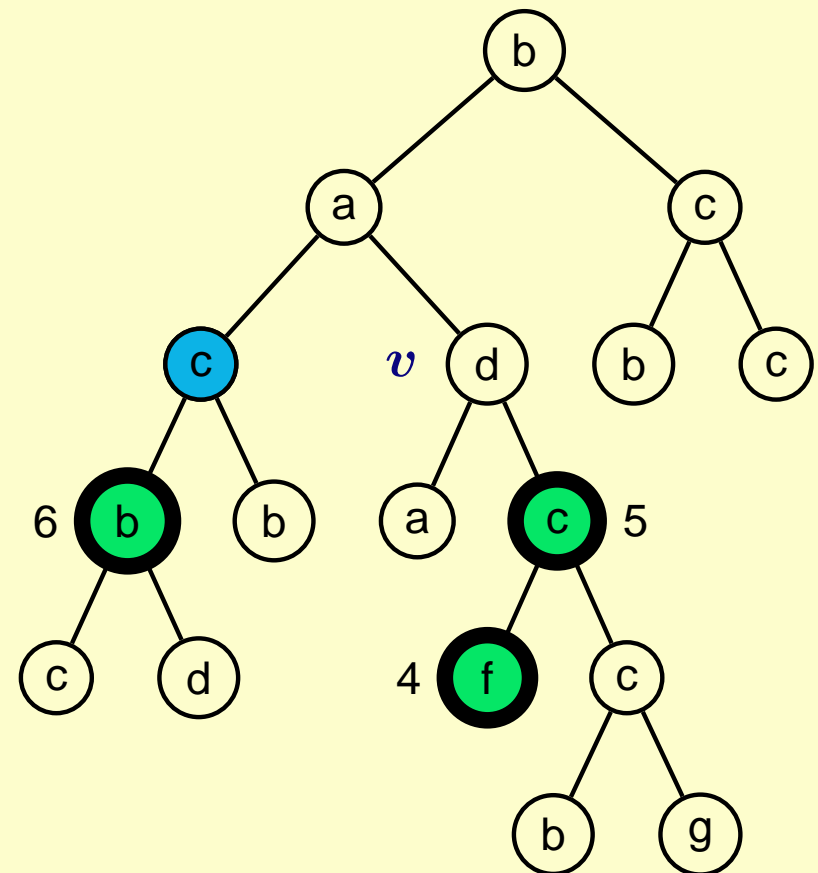
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

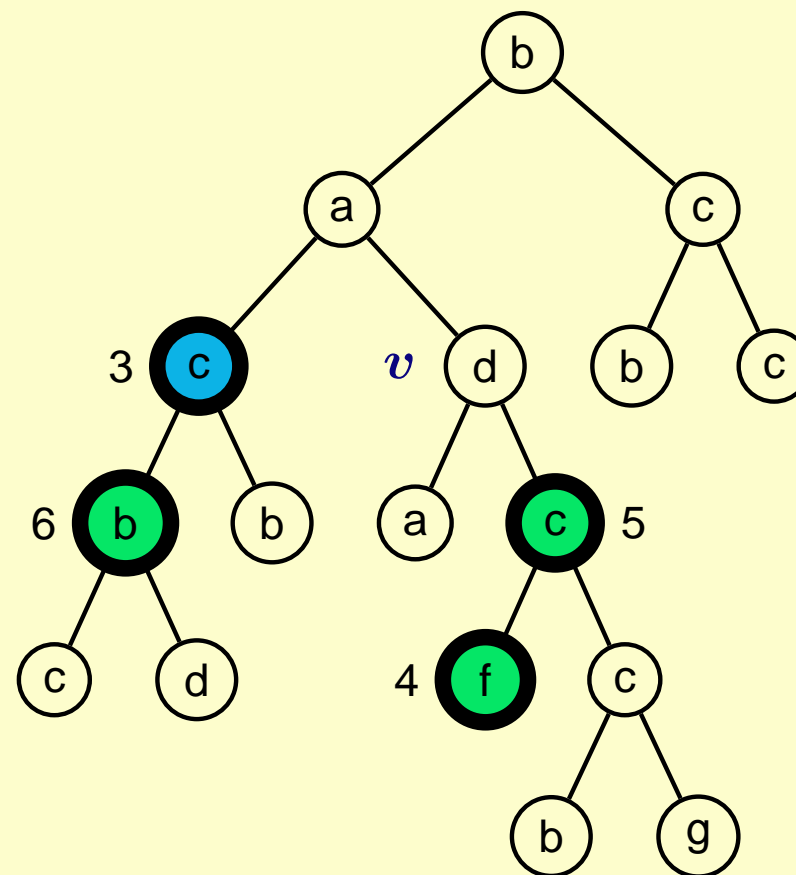
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

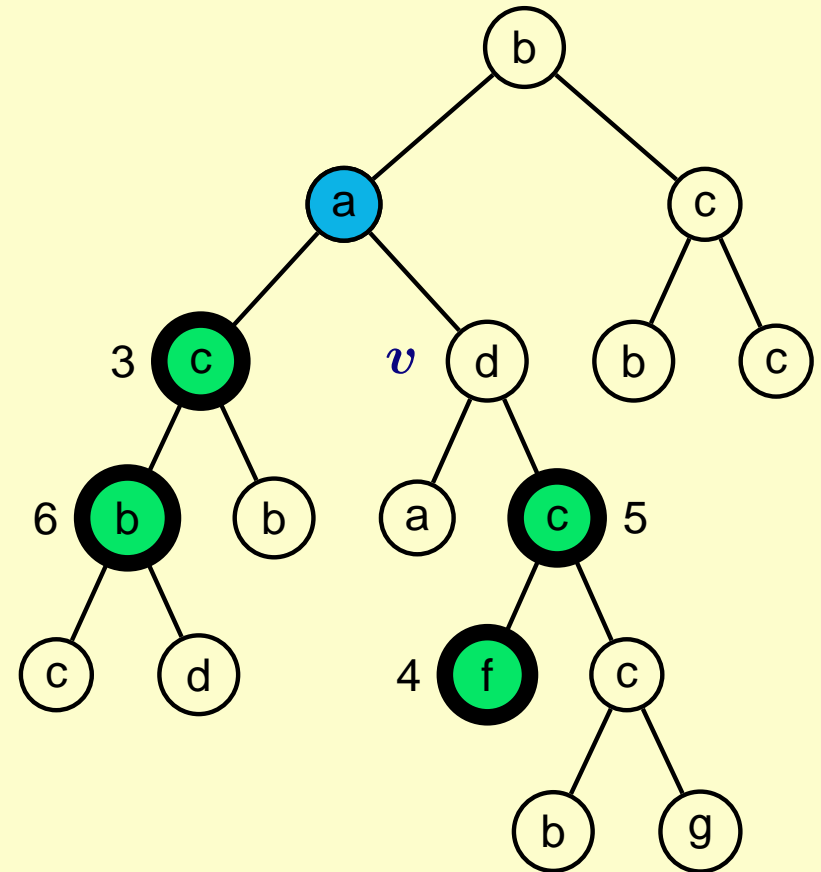
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

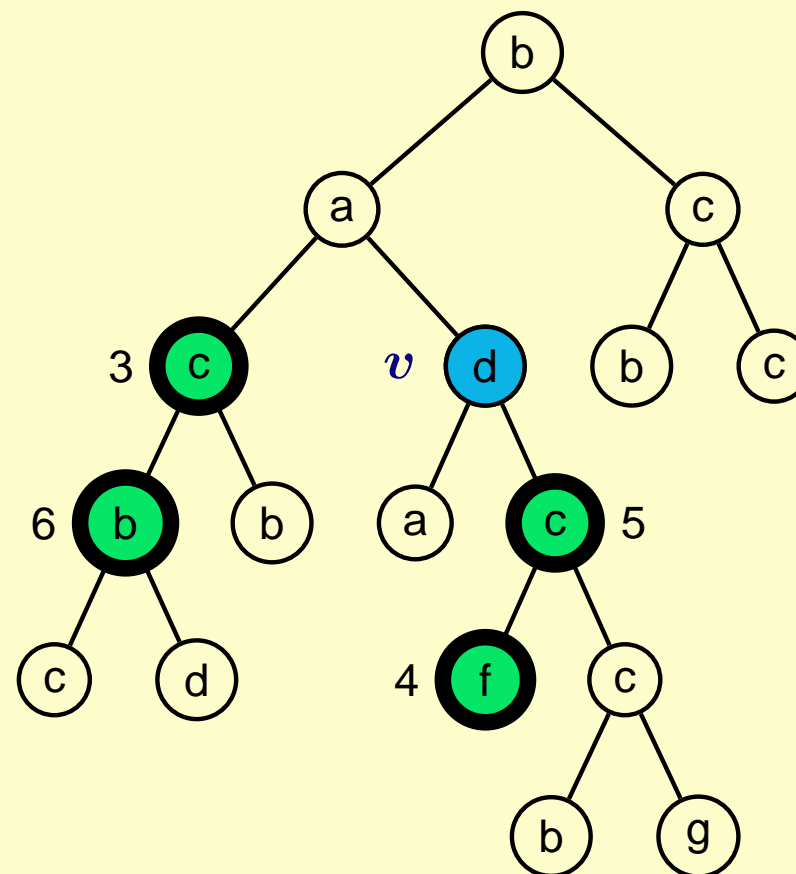
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

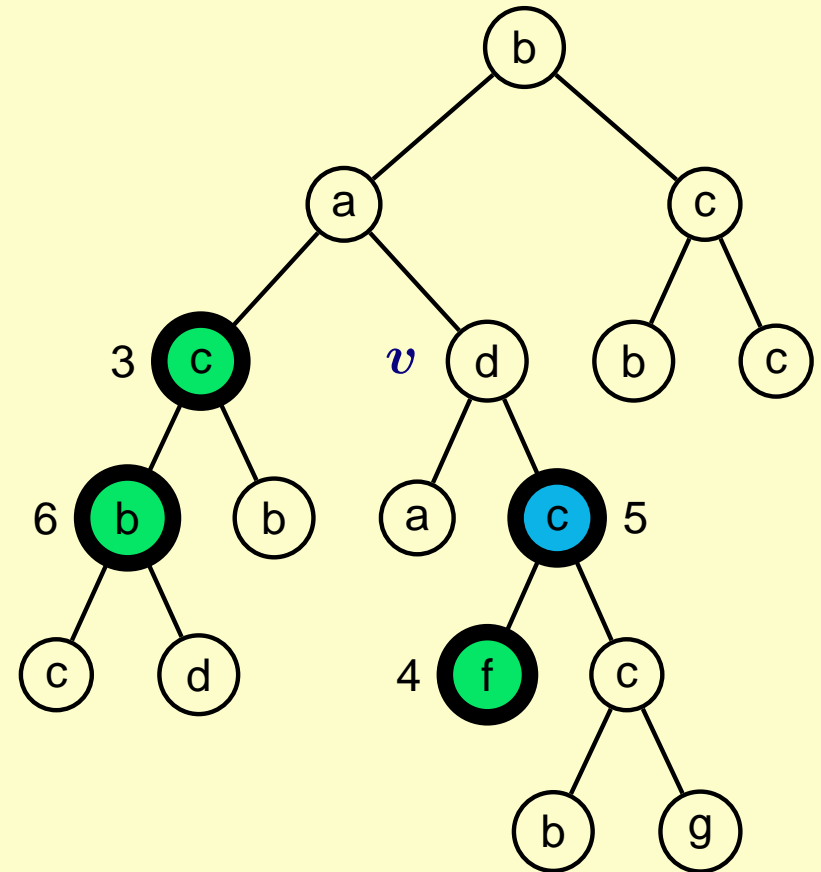
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

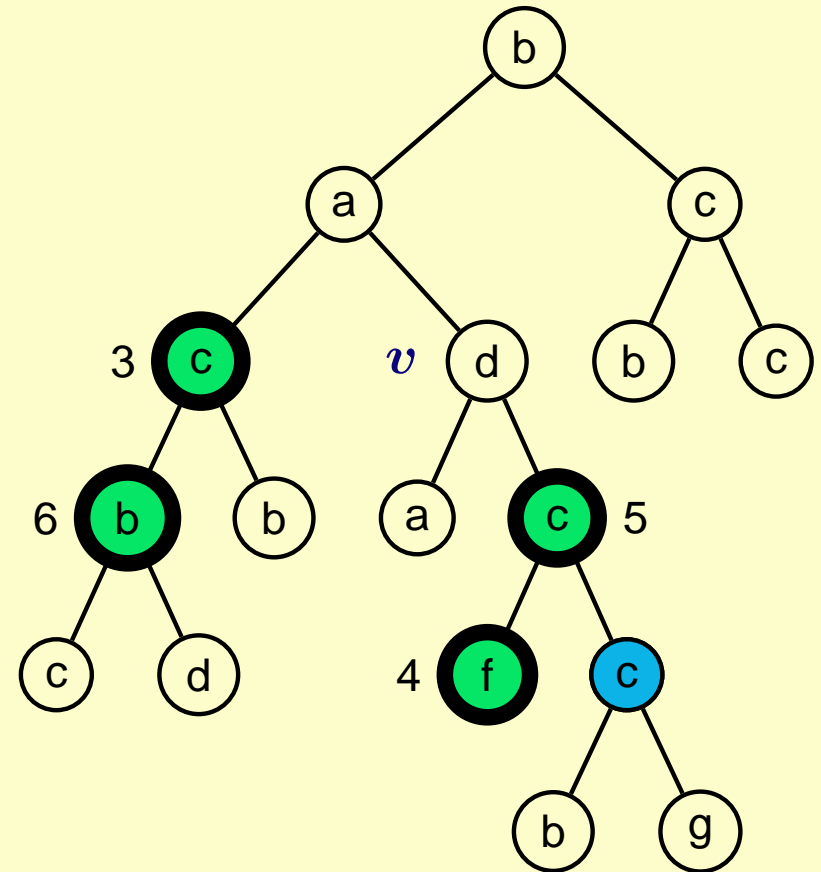
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

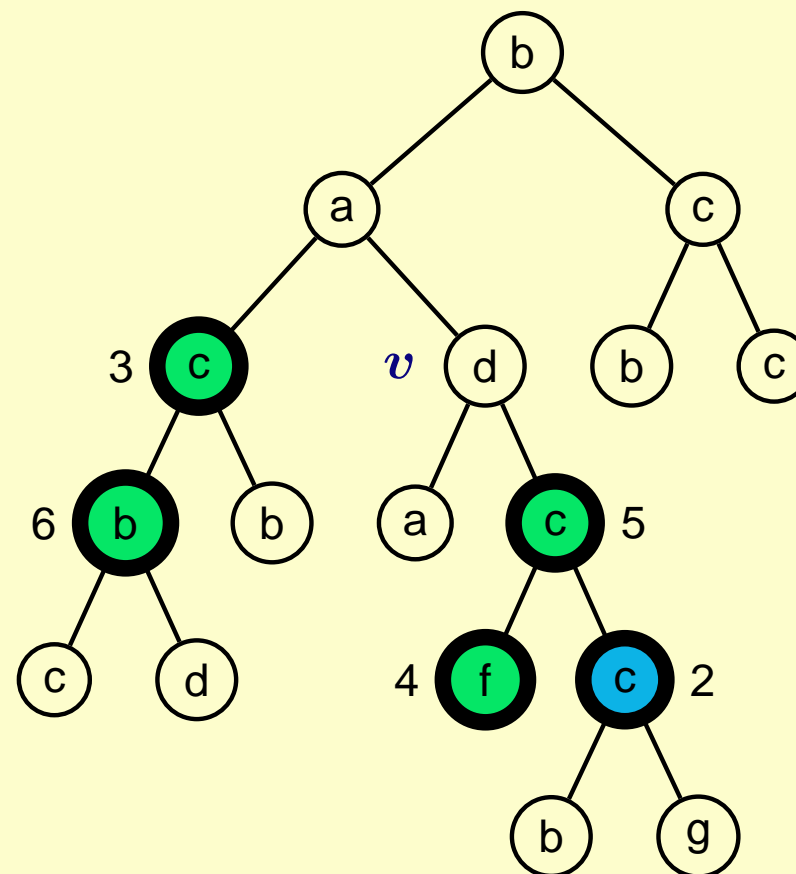
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

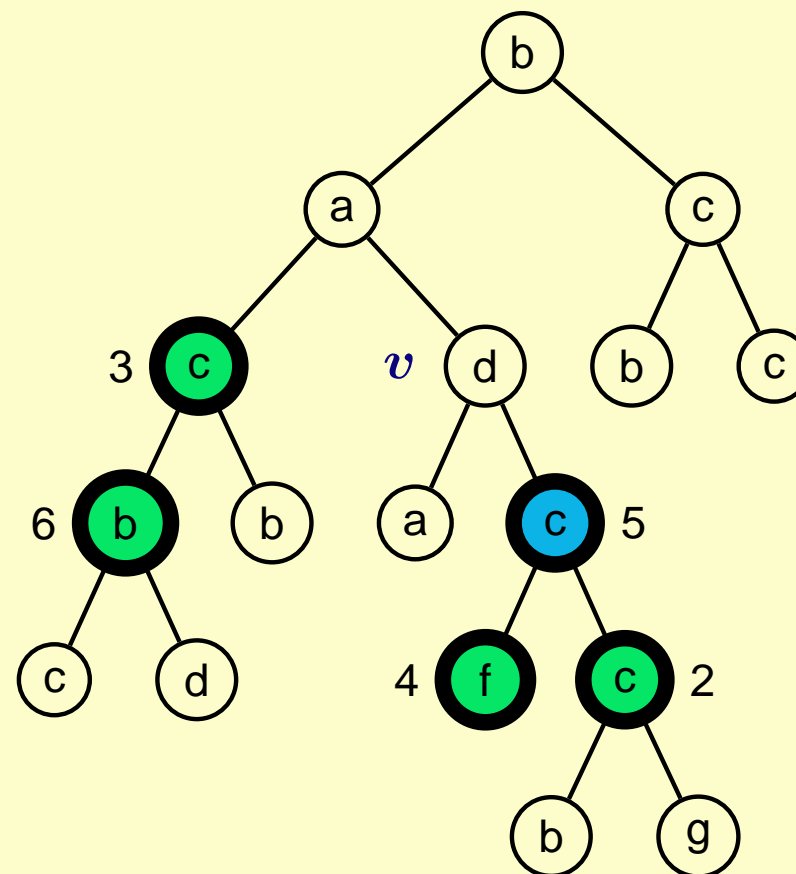
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

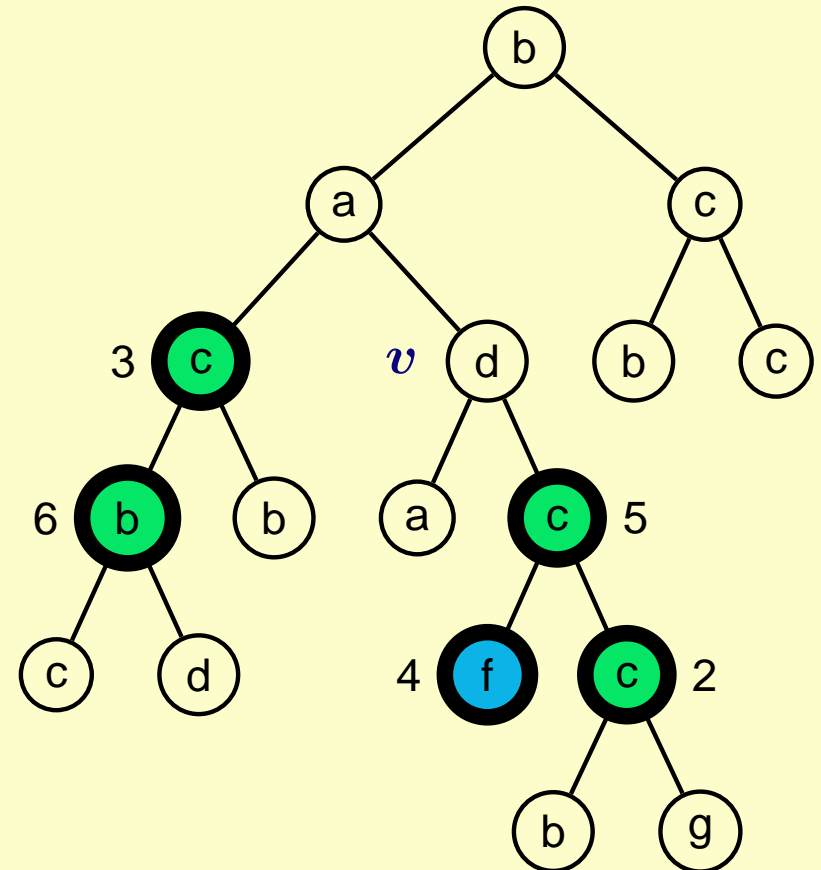
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

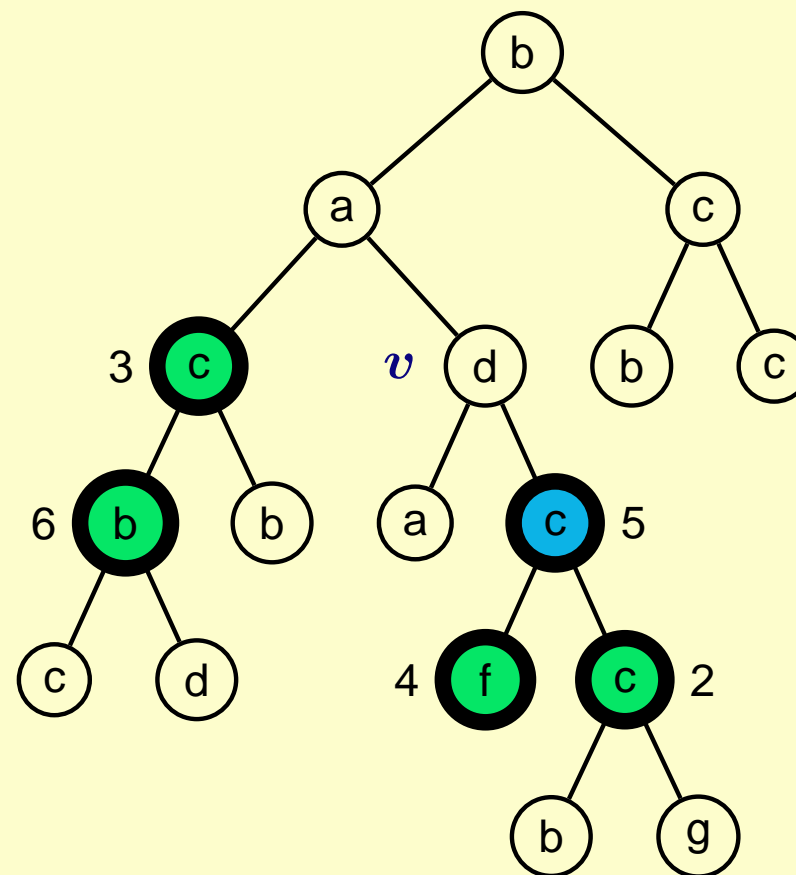
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

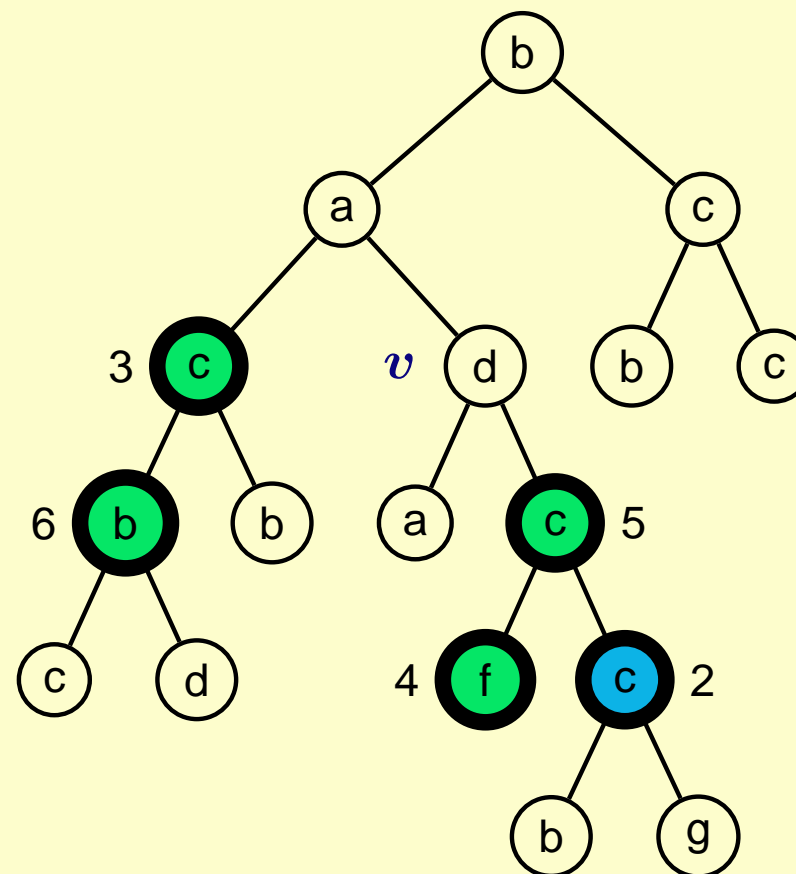
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

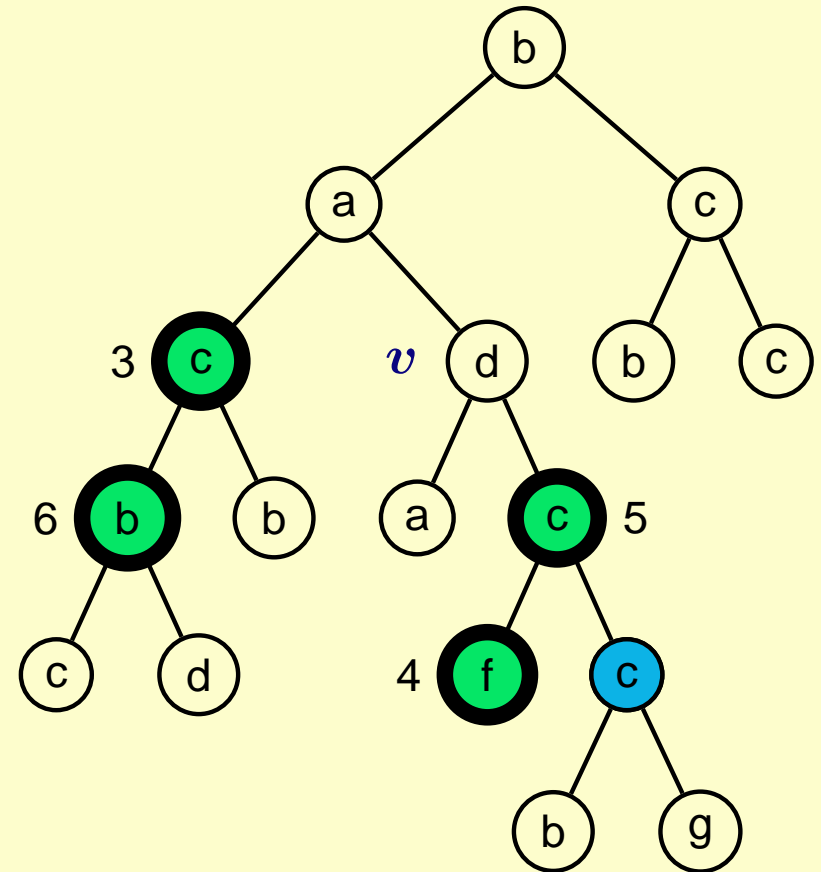
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

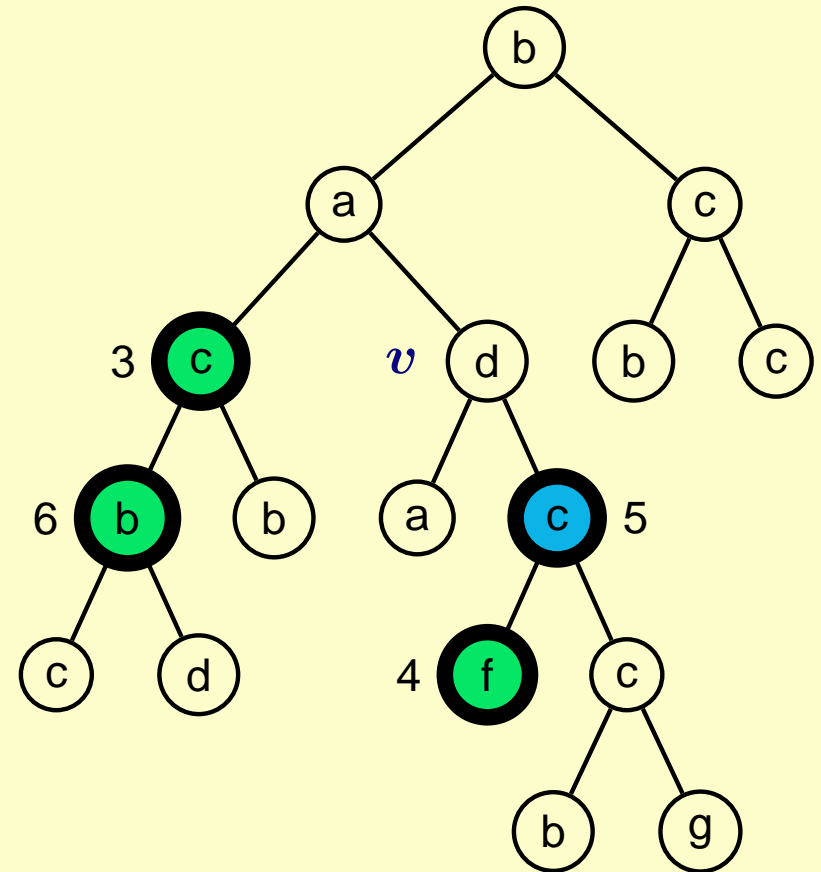
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

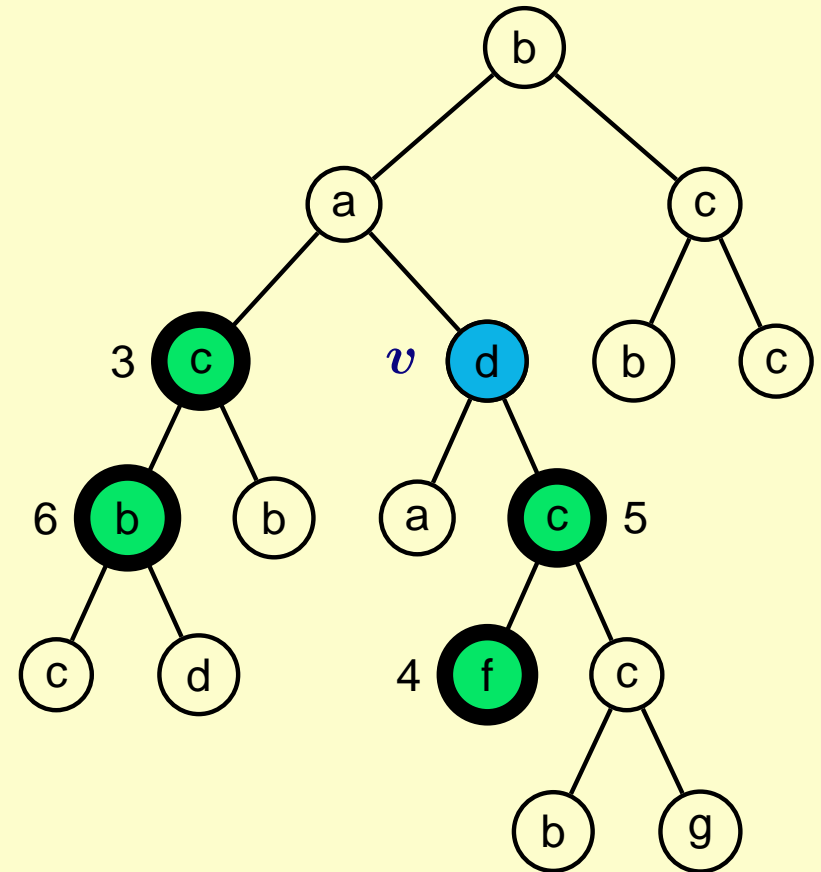
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

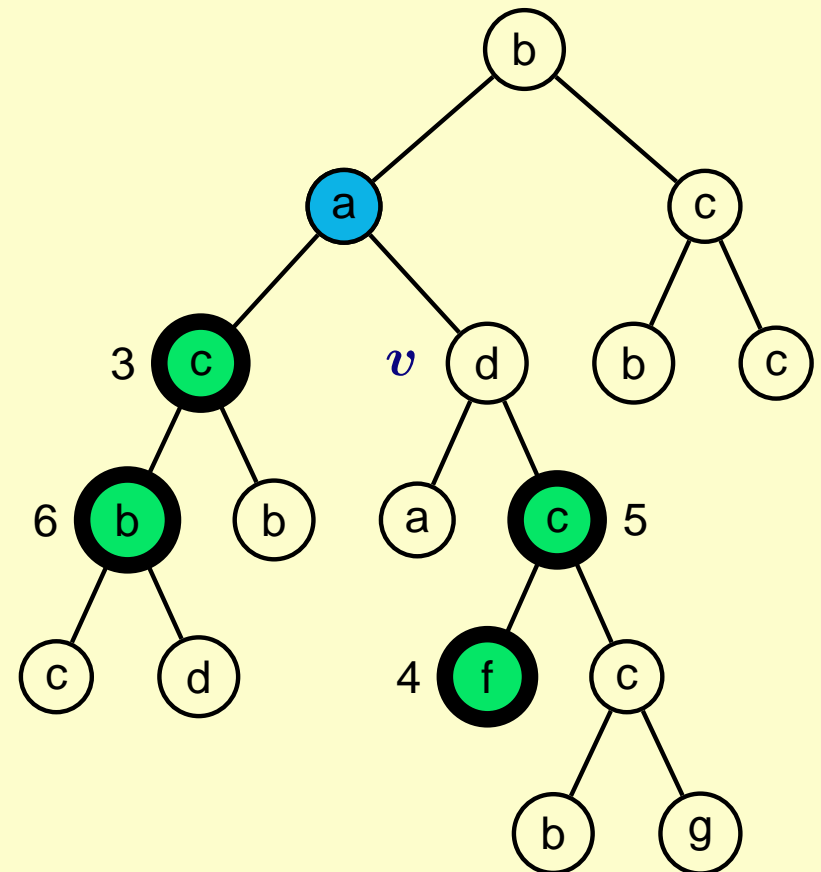
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

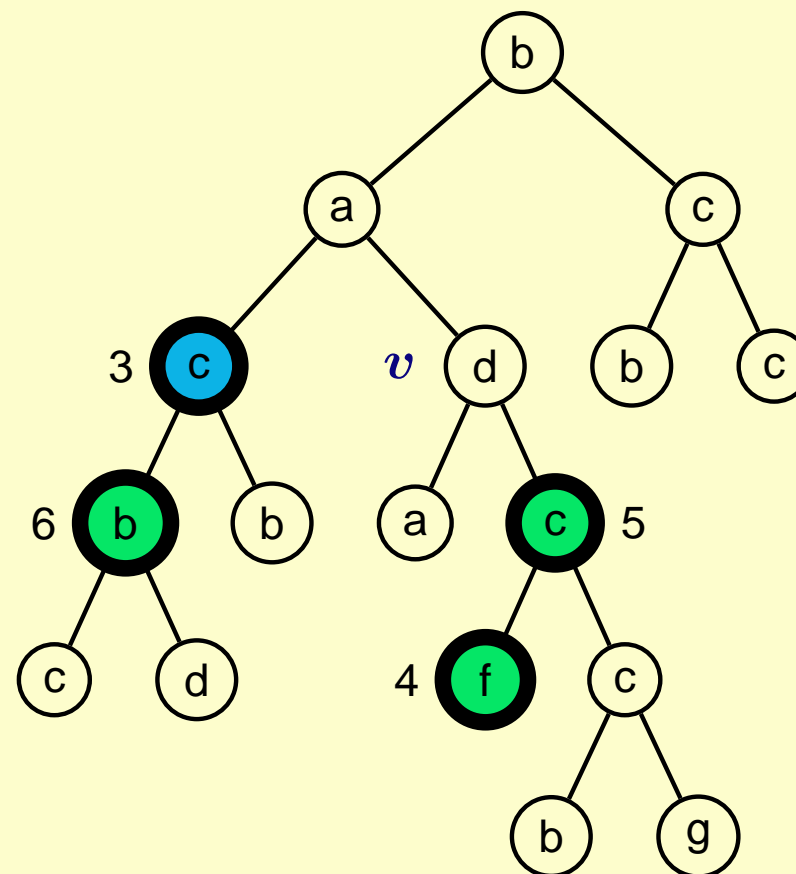
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

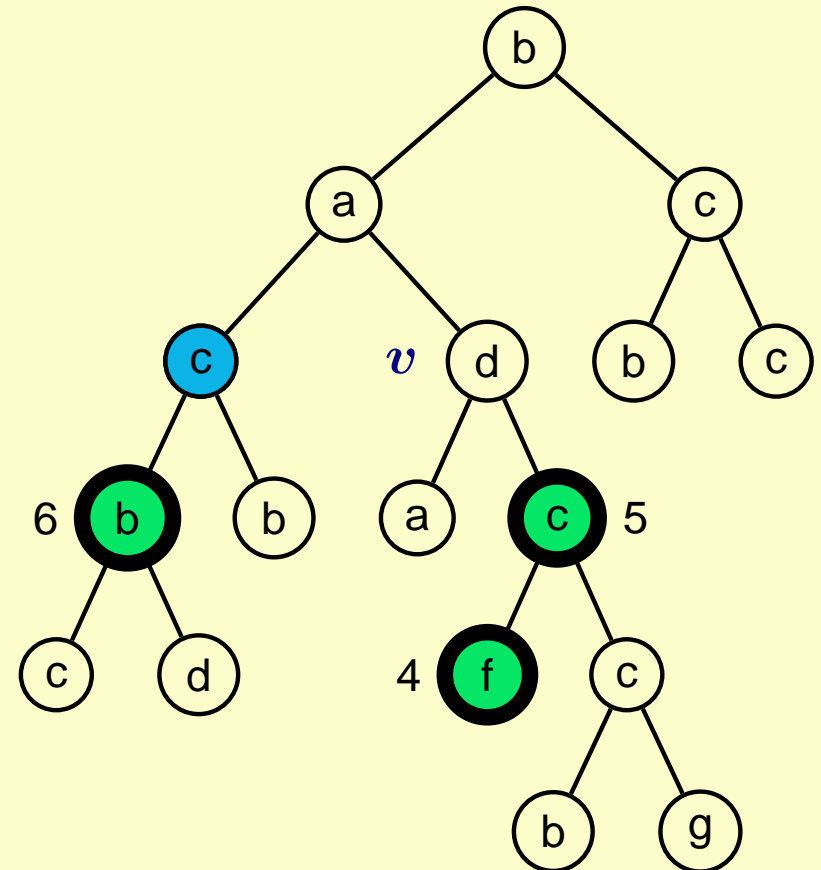
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

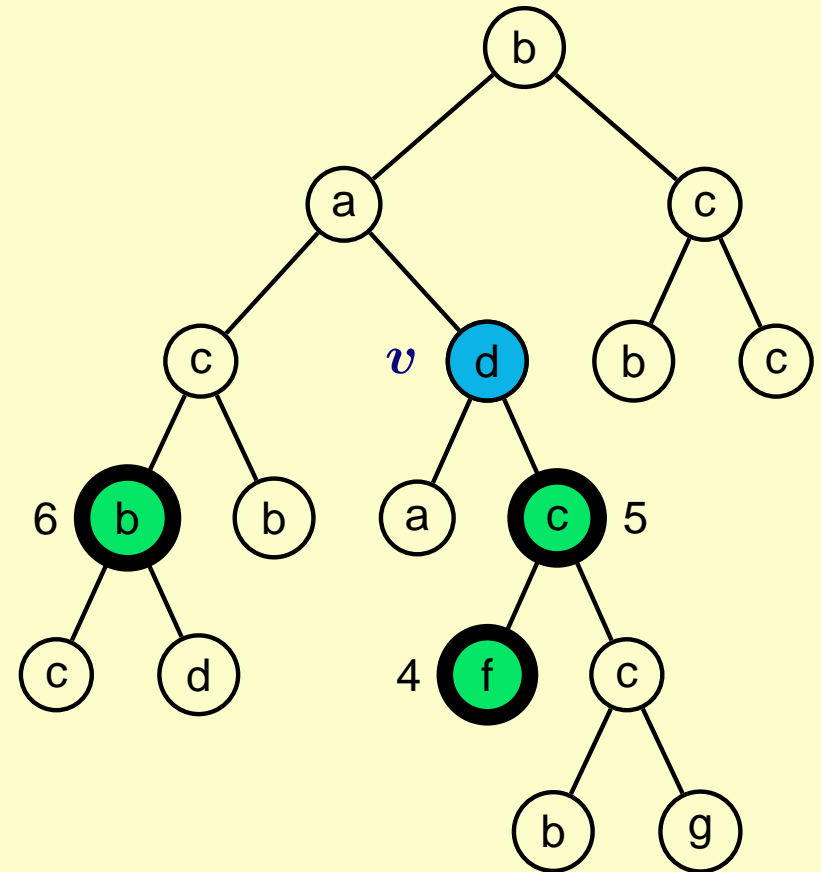
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

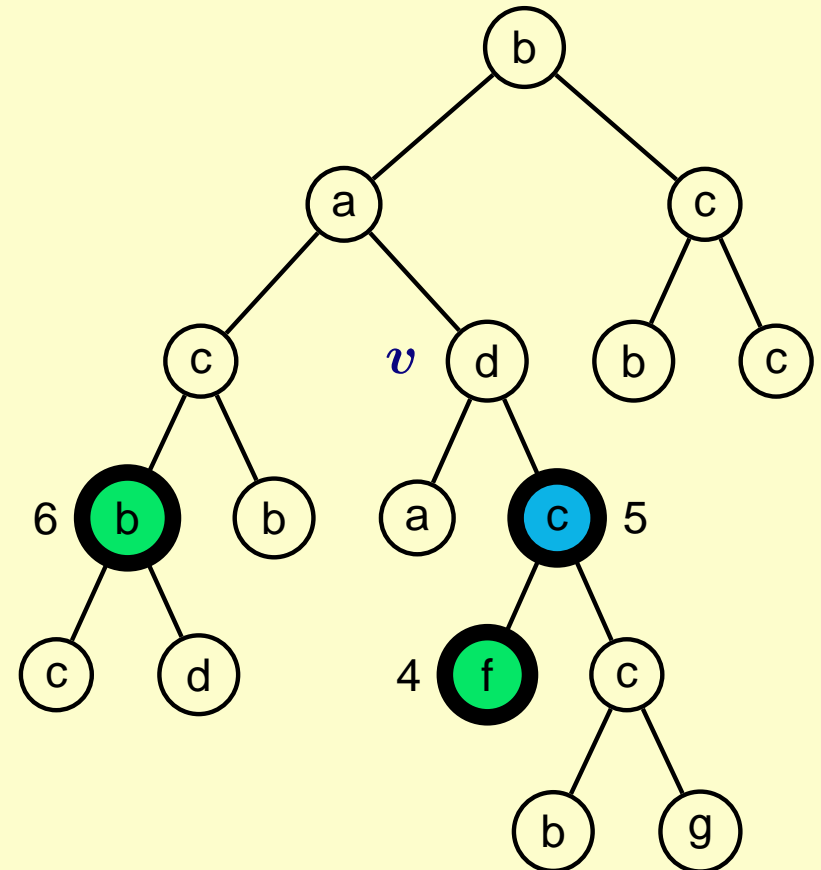
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

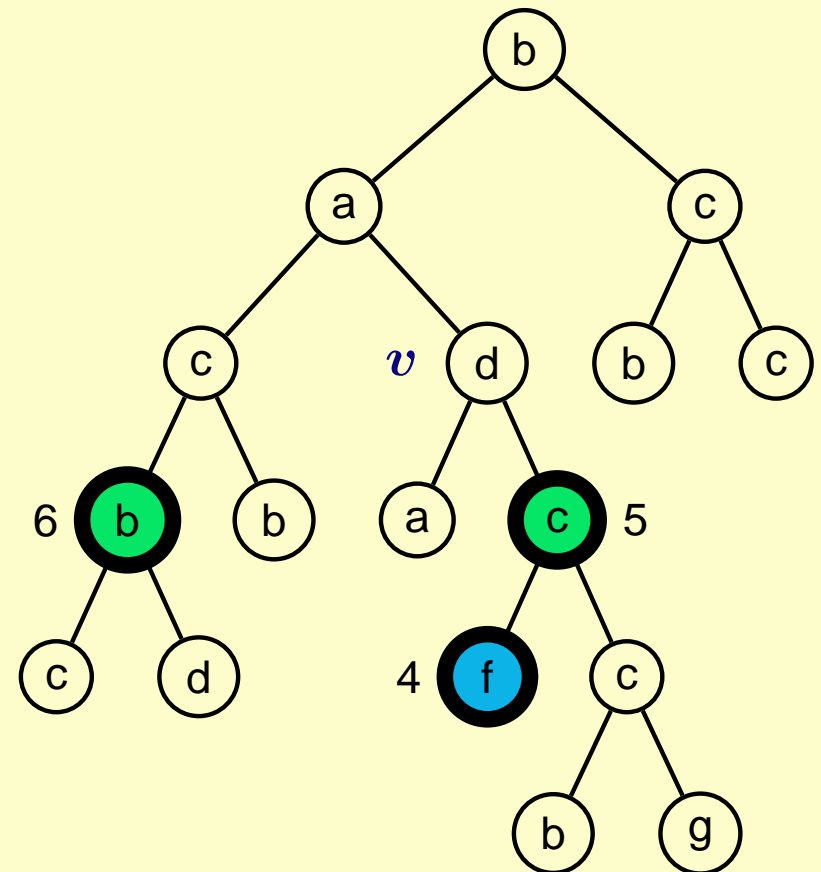
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



Step 2: tree loops...

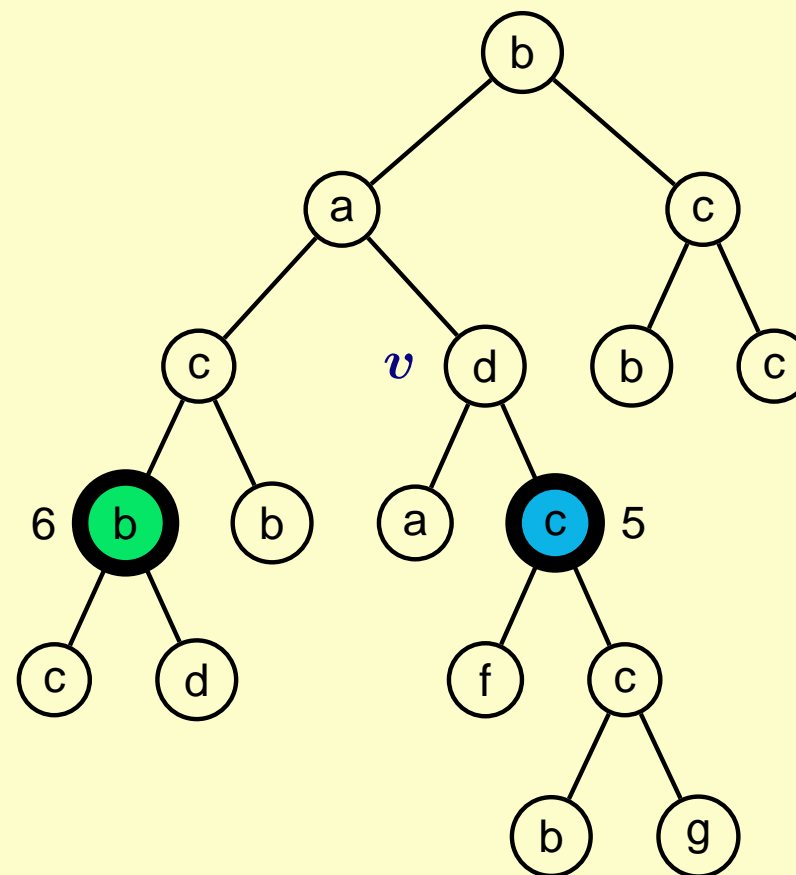
We fix some PA A

Basic question

What is the **behavior** of A on a subtree t_v ?

Some definitions

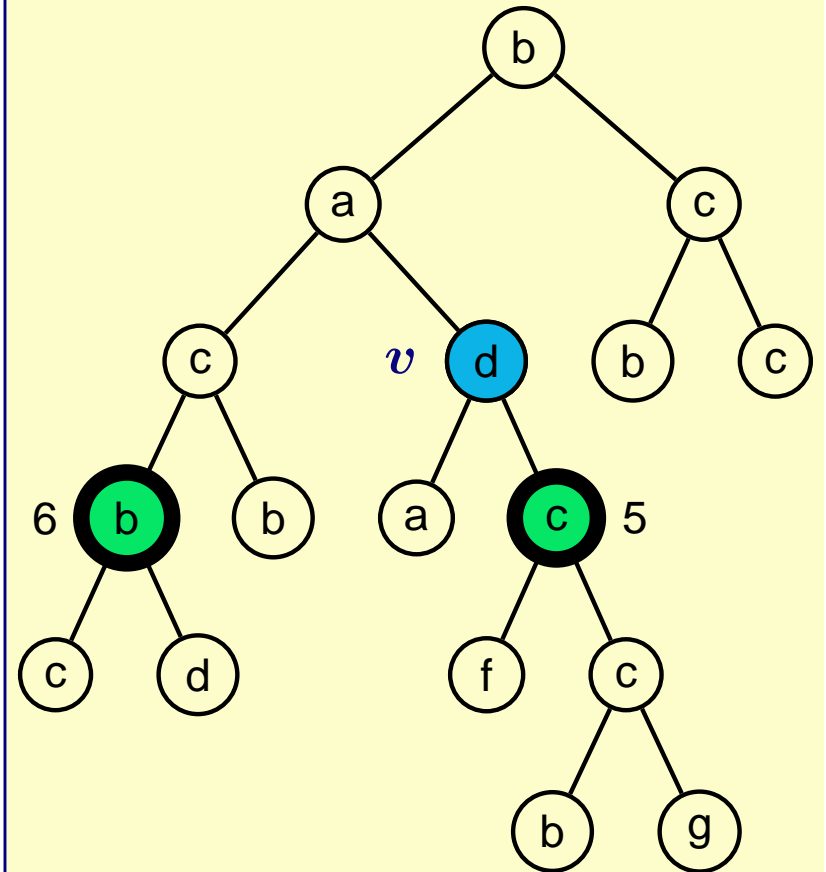
- **i -configuration** : $(v, p, v_{i+1}, \dots, v_n)$
- **tree i -loop** : subcomputation from $(v, p, v_{i+1}, \dots, v_n)$ to $(v, q, v_{i+1}, \dots, v_n)$, where for all intermediate j -configurations:
 - $j \leq i$, and
 - $j = i \Rightarrow$ head below v



... and types

Definition

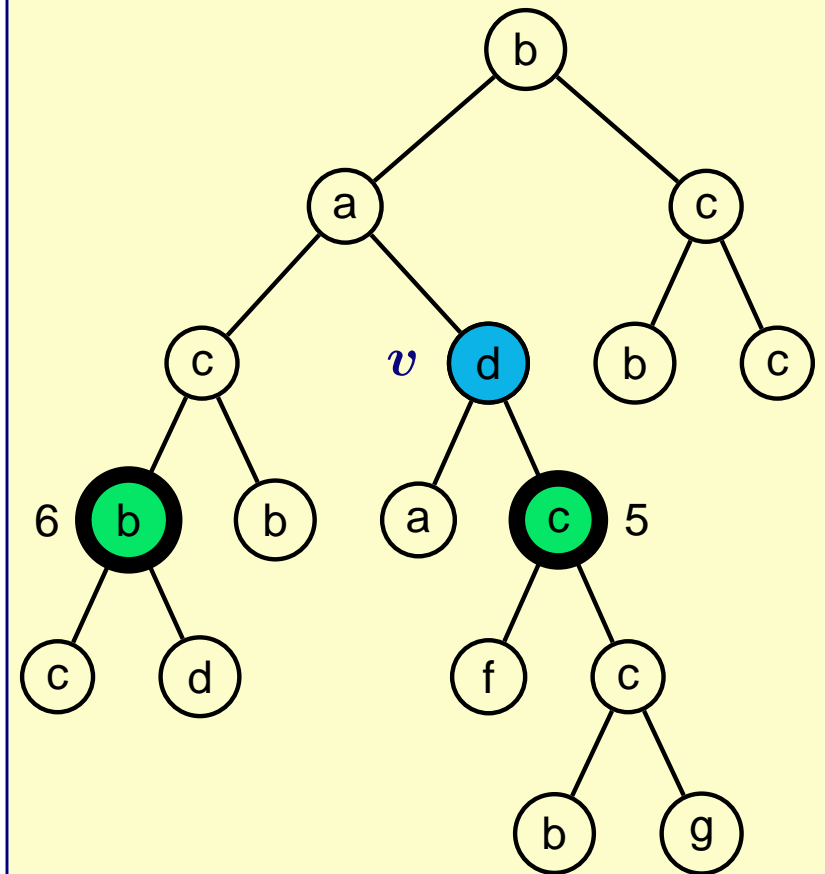
- **0-type of t_v** : all pairs (p, q) such that there is a tree **0-loop** ...



... and types

Definition

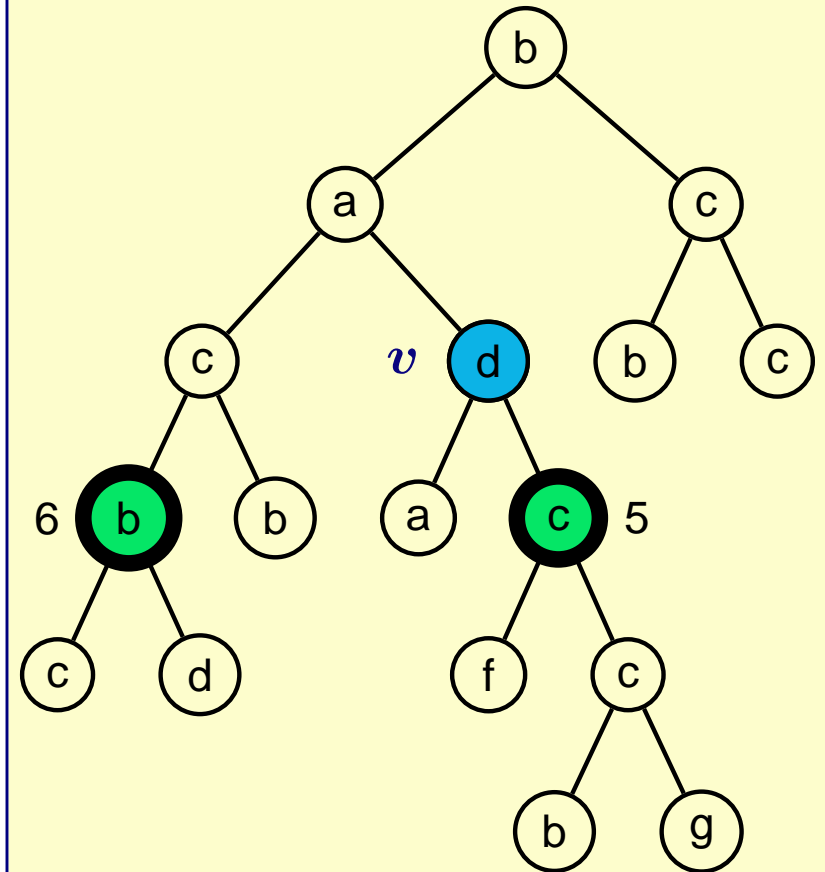
- **0-type of t_v** : all pairs (p, q) such that there is a tree **0-loop** ...
- **1-type of t_v** : for each **0-type** B_0 of a context C_v all pairs (p, q) such that there is a tree **1-loop** ...



... and types

Definition

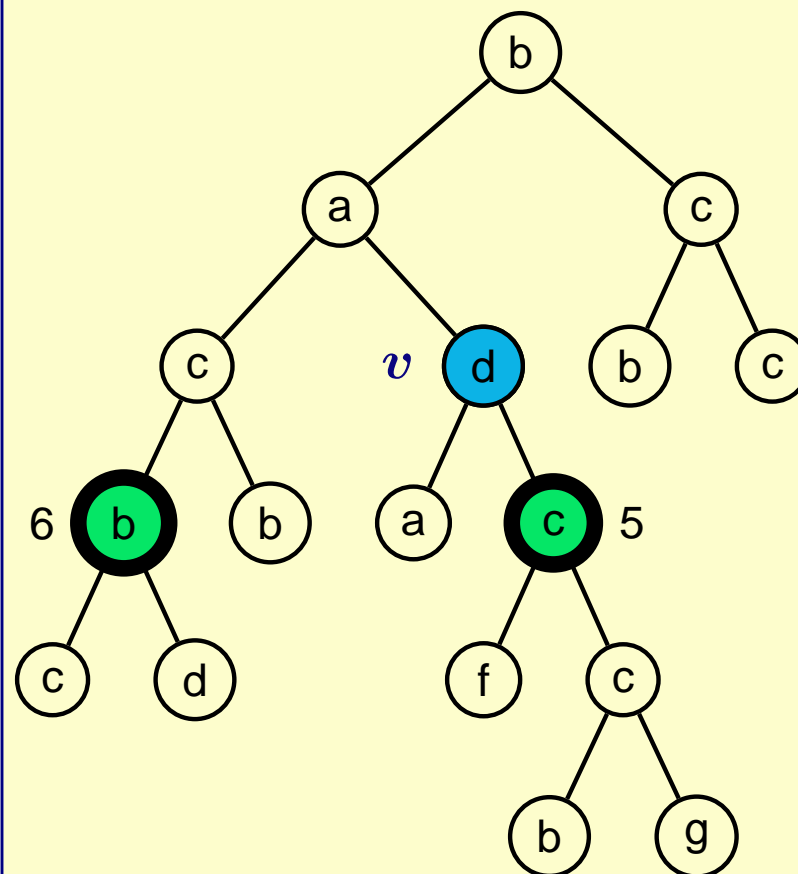
- **0-type of t_v** : all pairs (p, q) such that there is a tree **0-loop** ...
- **1-type of t_v** : for each **0-type** B_0 of a context C_v all pairs (p, q) such that there is a tree **1-loop** ...
- **2-type of t_v** : for each **0-type** B_0 and **1-type** B_1 of a context C_v all pairs (p, q) such that there is a tree **2-loop** ...



... and types

Definition

- **0-type of t_v** : all pairs (p, q) such that there is a tree **0-loop** ...
- **1-type of t_v** : for each **0-type** B_0 of a context C_v all pairs (p, q) such that there is a tree **1-loop** ...
- **2-type of t_v** : for each **0-type** B_0 and **1-type** B_1 of a context C_v all pairs (p, q) such that there is a tree **2-loop** ...
- **i -type of t_v** : for each **0-type** $B_0, \dots, (i - 1)$ -type B_{i-1} of a context C_v all pairs (p, q) such that there is a tree **i -loop** ...



Contents

Introduction and Overview of Results

The Behavior of Pebble Automata

▷ **On Strong Pebbles**

Hierarchy Theorems

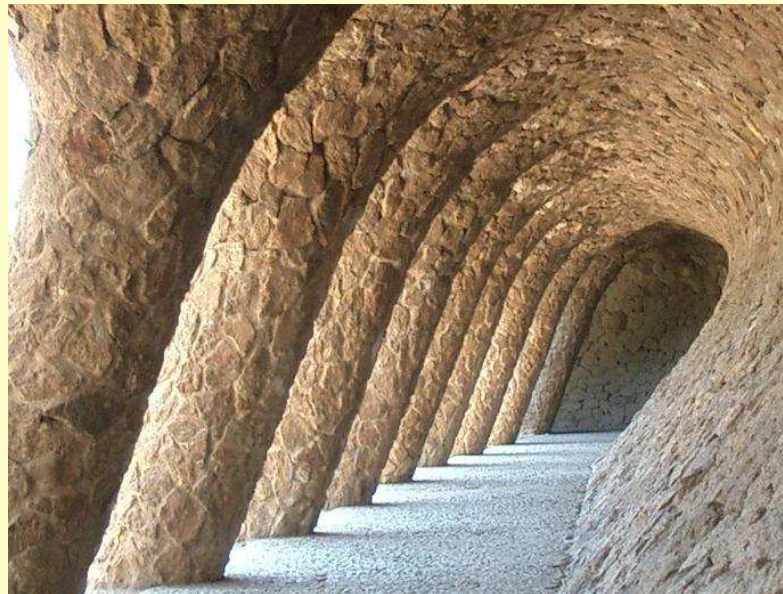
Conclusion

Coming soon...

Coming soon...

11th Int. Conf. on Database Theory (ICDT 2007)

Barcelona, January 10-12, 2007



Important dates

Title and Abstract Submission Deadline:

July 10, 2006

Paper Submission Deadline: July 17, 2006

Notification Deadline: October 3, 2006

Camera ready deadline: October 31, 2006

Invited speakers

- Jan Chomicki
- Laura Haas
- Cynthia Dwork

Step 3: strong pebbles aren't that strong

Theorem 1

For each $n \geq 0$:

(a) $\text{sPA}_n = \text{PA}_n$

(b) $\text{sDPA}_n = \text{DPA}_n$

Proof idea

- **k -weak-PA** :

- pebbles k, \dots, n are strong
- pebbles $1, \dots, k - 1$ are weak

- Induction on k .

- Each k -weak n -pebble automaton A has an equivalent $(k + 1)$ -weak automaton A'

→ we basically have to show how to simulate strong pebble k by a weak pebble k (where strong pebbles $k + 1, \dots, n$ are fixed)

Step 3: strong pebbles aren't that strong

Theorem 1

For each $n \geq 0$:

(a) $sPA_n = PA_n$

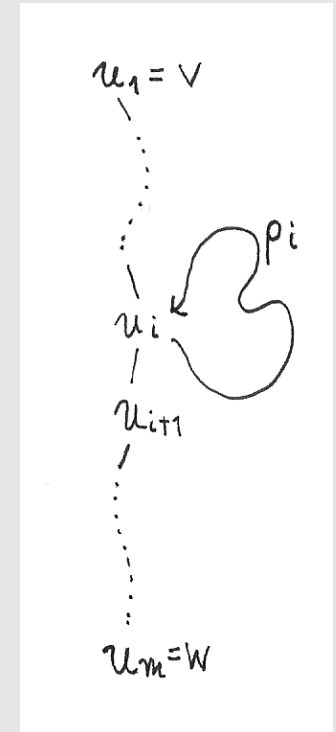
(b) $sDPA_n = DPA_n$

Proof idea

- **k -weak-PA** :
 - pebbles k, \dots, n are strong
 - pebbles $1, \dots, k - 1$ are weak
 - Induction on k .
 - Each k -weak n -pebble automaton A has an equivalent $(k + 1)$ -weak automaton A'
- we basically have to show how to simulate strong pebble k by a weak pebble k (where strong pebbles $k + 1, \dots, n$ are fixed)

Proof idea (cont.)

- Assume A drops strong pebble k at v and lifts it when its head is at w (say: below v)
- Let $u_1 = v, u_2, \dots, u_m = w$ path from v to w
- Idea: A' moves pebble k towards w
 - When k -configuration at u_i is reached, pebble k is moved to it
 - The head never moves above pebble k
 - Behavior of C_{u_i} is maintained inductively



- The deterministic case requires more care

Contents

Introduction and Overview of Results

The Behavior of Pebble Automata

On Strong Pebbles

▷ **Hierarchy Theorems**

Conclusion

Hierarchy theorems

Goal of this part:

Theorem 2

$$\text{PA} \subsetneq \text{REG}$$

We first show:

Theorem 3

$$\text{For each } n \geq 0, \text{PA}_n \subsetneq \text{PA}_{n+1}$$

We build on

Theorem [Bojańczyk, Colcombet 05]

$$\text{TWA} \subsetneq \text{REG}$$

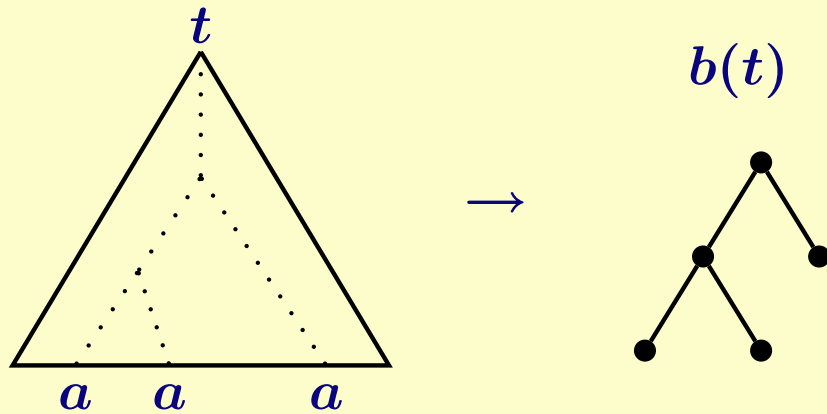
A look at “TWA $\not\subseteq$ REG”

- **Quasi-blank trees :**

Alphabet $\{a, b\}$, a appears only in leaves

A look at “TWA \subsetneq REG”

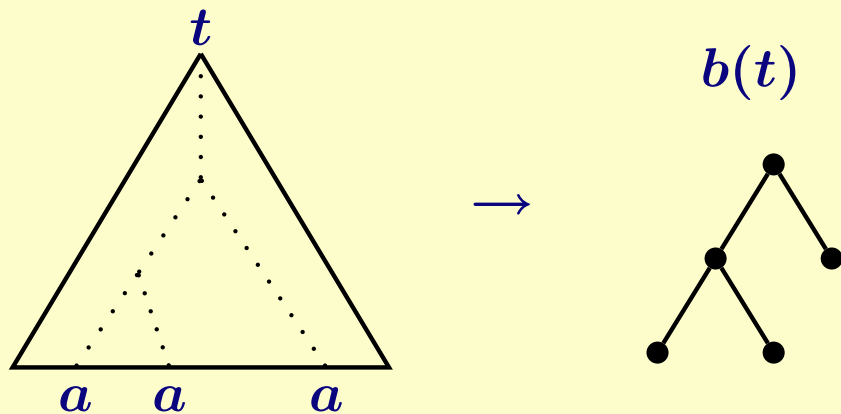
- **Quasi-blank trees** :
Alphabet $\{a, b\}$, a appears only in leaves
- **Branching structure** :



A look at “TWA $\not\subseteq$ REG”

- **Quasi-blank trees** :
Alphabet $\{a, b\}$, a appears only in leaves

- **Branching structure** :

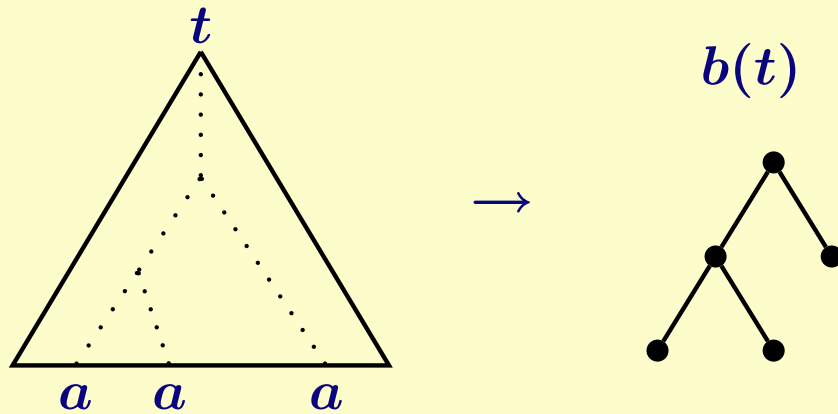


- **L_{branch}** : all trees t , where in $b(t)$ all root-to-leaf paths have even length

A look at “TWA \subsetneq REG”

- **Quasi-blank trees** :
Alphabet $\{a, b\}$, a appears only in leaves

- **Branching structure** :



- **L_{branch}** : all trees t , where in $b(t)$ all root-to-leaf paths have even length

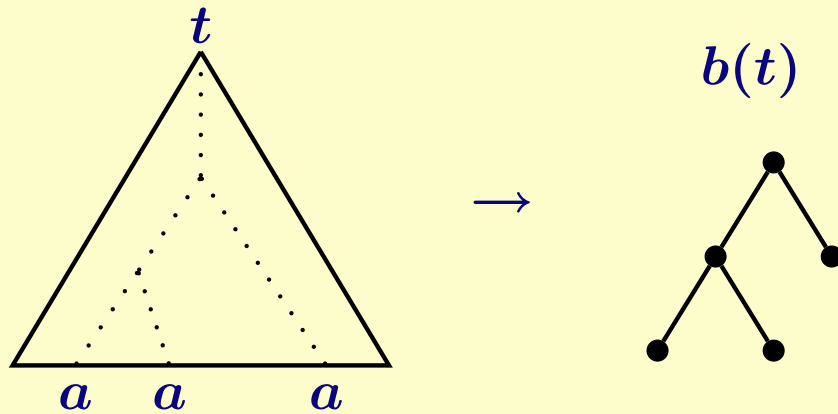
Fact [Bojańczyk, Colcombet 05]

$L_{\text{branch}} \in \text{REG} - \text{TWA}$

A look at “TWA \subsetneq REG”

- **Quasi-blank trees** :
Alphabet $\{a, b\}$, a appears only in leaves

- **Branching structure** :



- **L_{branch}** : all trees t , where in $b(t)$ all root-to-leaf paths have even length

Fact [Bojańczyk, Colcombet 05]

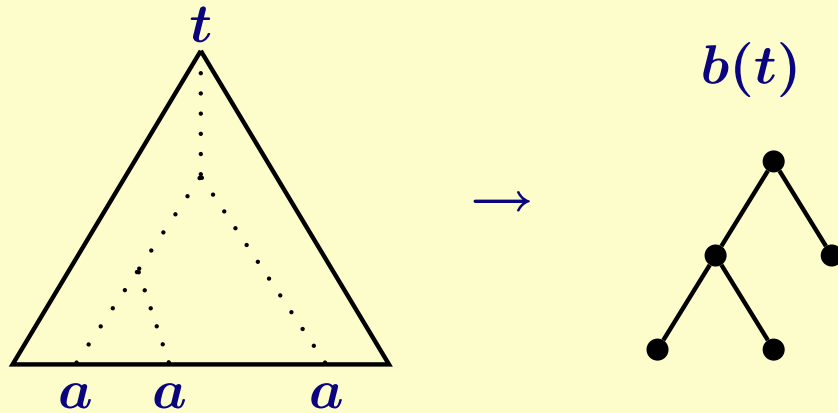
$L_{\text{branch}} \in \text{REG} - \text{TWA}$

- Even more: for each A there are $s \in L_{\text{branch}}$ and $t \notin L_{\text{branch}}$ such that each root-to-root loop of A on s also exists on t

A look at “TWA \subsetneq REG”

- **Quasi-blank trees** :
Alphabet $\{a, b\}$, a appears only in leaves

- **Branching structure** :



- **L_{branch}** : all trees t , where in $b(t)$ all root-to-leaf paths have even length

Fact [Bojańczyk, Colcombet 05]

$L_{\text{branch}} \in \text{REG} - \text{TWA}$

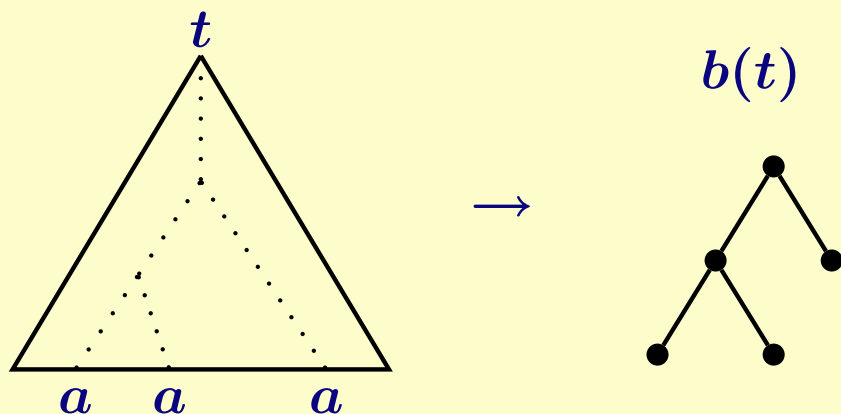
- Even more: for each A there are $s \in L_{\text{branch}}$ and $t \notin L_{\text{branch}}$ such that each root-to-root loop of A on s also exists on t

- We need a stronger statement: the root-to-root behavior of A on s and t should be exactly the same
- **L_{even}** : Number of 0^*1 -nodes v in $b(t)$ whose subtree only has even length branches is even

A look at “TWA \subsetneq REG”

- **Quasi-blank trees** :
Alphabet $\{a, b\}$, a appears only in leaves

- **Branching structure** :



- **L_{branch}** : all trees t , where in $b(t)$ all root-to-leaf paths have even length

Fact [Bojańczyk, Colcombet 05]

$L_{\text{branch}} \in \text{REG} - \text{TWA}$

- Even more: for each A there are $s \in L_{\text{branch}}$ and $t \notin L_{\text{branch}}$ such that each root-to-root loop of A on s also exists on t

- We need a stronger statement: the root-to-root behavior of A on s and t should be exactly the same
- **L_{even}** : Number of 0^*1 -nodes v in $b(t)$ whose subtree only has even length branches is even

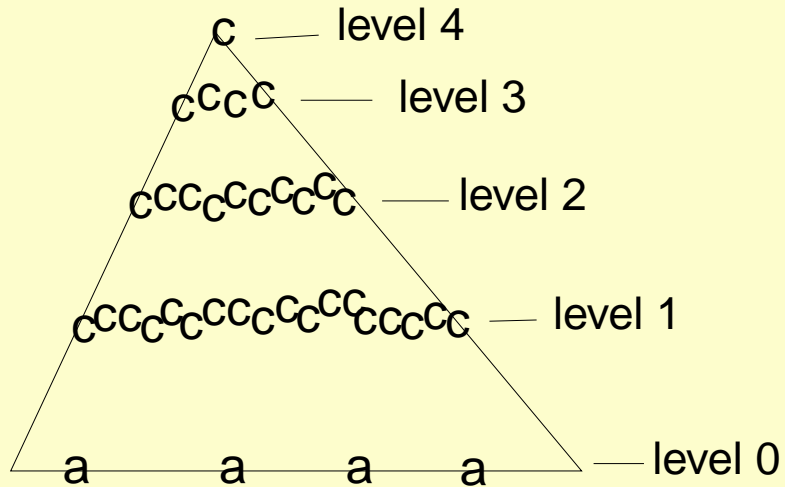
Proposition

For each A there are $s \in L_{\text{even}}$ and $t \notin L_{\text{even}}$ such that A has the same root-to-root behavior on s and t

Step 4: define a language separating PA_n from PA_{n-1}

- **n-leveled tree :**

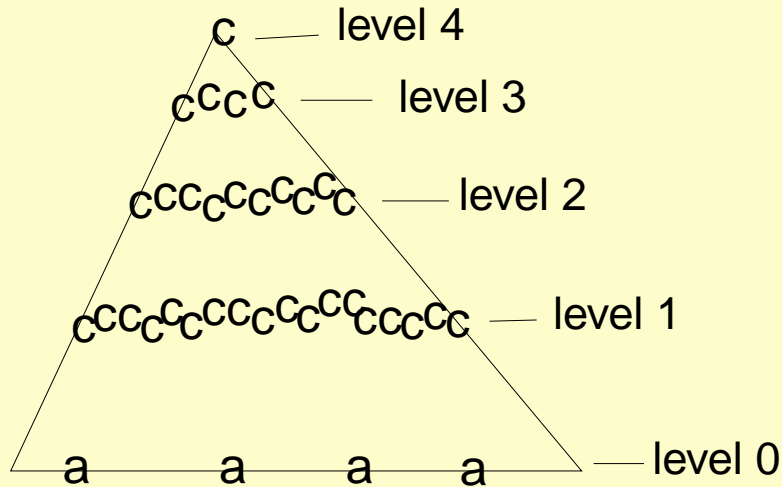
- Alphabet $\{a, b, c\}$
- All root-to-leaf paths are in $(cb^*)^n(a + b)$



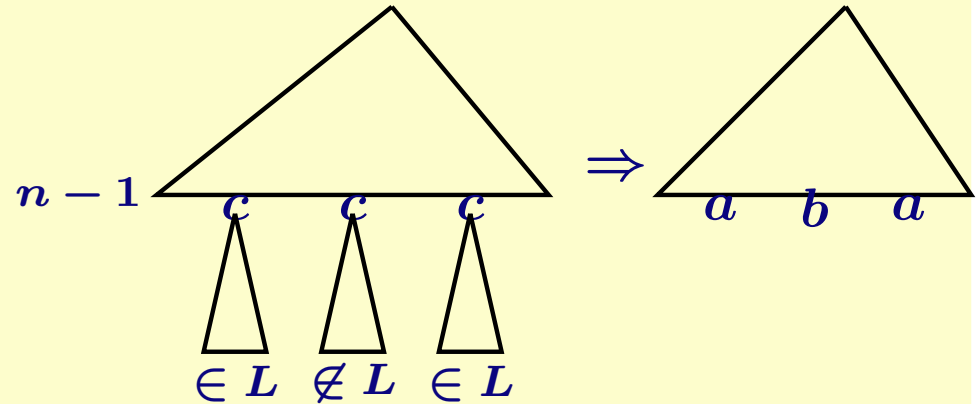
Step 4: define a language separating PA_n from PA_{n-1}

- n-leveled tree :**

- Alphabet $\{a, b, c\}$
- All root-to-leaf paths are in $(cb^*)^n(a + b)$



- L-folding :**

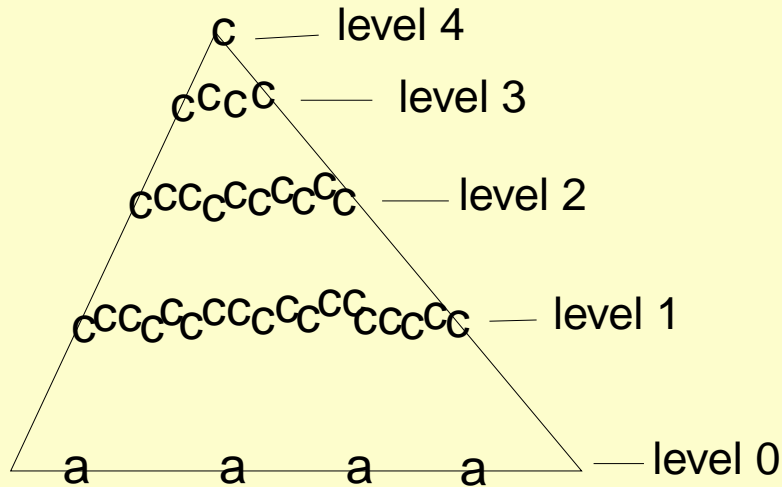


- L_n :** all n -leveled trees whose L_{n-1} -folding is in L_{even}

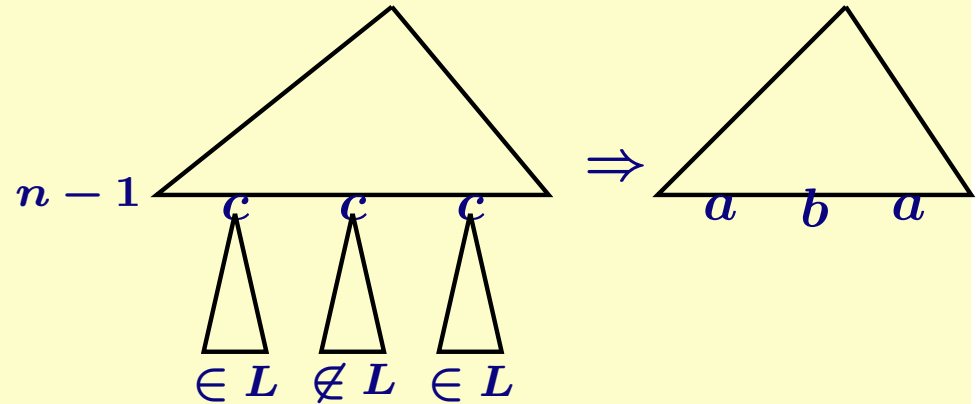
Step 4: define a language separating PA_n from PA_{n-1}

- n-leveled tree :**

- Alphabet $\{a, b, c\}$
- All root-to-leaf paths are in $(cb^*)^n(a + b)$



- L-folding :**



- **L_n** : all n -leveled trees whose L_{n-1} -folding is in L_{even}

Theorem 4

- $L_n \in \text{DPA}_n$
- $L_n \notin \text{PA}_{n-1}$

Proof of Theorem 4

Proof

- **Step 5:**

Towards a contradiction, assume

$L(A) = L_n$, for some $(n - 1)$ -pebble automaton A

- We inductively construct

- $s_1, \dots, s_n \in L_n$

- $t_1, \dots, t_n \notin L_n$

such that s_i and t_i have equivalent i -type wrt A

- At level $i - 1$, both s_i, t_i have only subtrees s_{i-1}, t_{i-1}

Proof of Theorem 4

Proof

- **Step 5:**

Towards a contradiction, assume

$L(A) = L_n$, for some $(n - 1)$ -pebble automaton A

- We inductively construct

- $s_1, \dots, s_n \in L_n$

- $t_1, \dots, t_n \notin L_n$

such that s_i and t_i have equivalent i -type wrt A

- At level $i - 1$, both s_i, t_i have only subtrees s_{i-1}, t_{i-1}

Proof (cont.)

Intuition:

- If pebble i is not in the tree, A acts like a TWA
- If pebble i is placed above level $i - 1$, A can not distinguish s_{i-1} -subtrees from t_{i-1} -subtrees
- Subcomputations taking pebble i below level $i - 1$ do not learn more than the type of the subtree

Proof of Theorem 4

Proof

- **Step 5:**

Towards a contradiction, assume

$L(A) = L_n$, for some $(n - 1)$ -pebble automaton A

- We inductively construct

- $s_1, \dots, s_n \in L_n$

- $t_1, \dots, t_n \notin L_n$

such that s_i and t_i have equivalent i -type wrt A

- At level $i - 1$, both s_i, t_i have only subtrees s_{i-1}, t_{i-1}

Proof (cont.)

Intuition:

- If pebble i is not in the tree, A acts like a TWA
 - If pebble i is placed above level $i - 1$, A can not distinguish s_{i-1} -subtrees from t_{i-1} -subtrees
 - Subcomputations taking pebble i below level $i - 1$ do not learn more than the type of the subtree
 - Not very precise
- Formalization through “oracle automata”

Oracle automata

Proof (cont.)

- **Oracle automaton** :
 - basis: tree-walking automaton
 - no pebbles!
 - works on folding
 - **structure oracle** :
 - * MSO-formulas $\varphi_1(x), \dots, \varphi_l(x)$
which do not test labels
 - * transition at node v depends on
 $(\varphi_1(v), \dots, \varphi_l(v))$

Step 6: Proposition

For each oracle automaton A there are $s \in L_1$ and $t \notin L_1$ with the same root-to-root loops of A

Proof idea for proposition

Slight extension of TWA-proof

Oracle automata

Proof (cont.)

- **Oracle automaton** :
 - basis: tree-walking automaton
 - no pebbles!
 - works on folding
 - **structure oracle** :
 - * MSO-formulas $\varphi_1(x), \dots, \varphi_l(x)$ which do not test labels
 - * transition at node v depends on $(\varphi_1(v), \dots, \varphi_l(v))$

Step 6: Proposition

For each oracle automaton A there are $s \in L_1$ and $t \notin L_1$ with the same root-to-root loops of A

Proof idea for proposition

Slight extension of TWA-proof

Proof of Theorem 4 (cont.)

- $i = 0$: follows from Proposition (even TWA-case sufficient)
 - $i - 1 \rightarrow i$: (**Step 7**)
 - Choose m large enough wrt A, i
 - Pick $s \in L_1, t \notin L_1$ such that no oracle automaton of size $\leq m$ can distinguish s from t
 - $s_{i+1} := s$, where each a -leaf is replaced by s_i and each b -leaf by t_i
 - (t_{i+1} accordingly)
-

Oracle automata

Proof (cont.)

- **Oracle automaton** :
 - basis: tree-walking automaton
 - no pebbles!
 - works on folding
 - **structure oracle** :
 - * MSO-formulas $\varphi_1(x), \dots, \varphi_l(x)$ which do not test labels
 - * transition at node v depends on $(\varphi_1(v), \dots, \varphi_l(v))$

Step 6: Proposition

For each oracle automaton A there are $s \in L_1$ and $t \notin L_1$ with the same root-to-root loops of A

Proof idea for proposition

Slight extension of TWA-proof

Proof of Theorem 4 (cont.)

- $i = 0$: follows from Proposition (even TWA-case sufficient)
- $i - 1 \rightarrow i$: (**Step 7**)
 - Choose m large enough wrt A, i
 - Pick $s \in L_1, t \notin L_1$ such that no oracle automaton of size $\leq m$ can distinguish s from t
 - $s_{i+1} := s$, where each a -leaf is replaced by s_i and each b -leaf by t_i
 - (t_{i+1} accordingly)

- Assumption: i -type of A distinguishes s_{i+1} from t_{i+1}

Oracle automata

Proof (cont.)

- **Oracle automaton** :
 - basis: tree-walking automaton
 - no pebbles!
 - works on folding
 - **structure oracle** :
 - * MSO-formulas $\varphi_1(x), \dots, \varphi_l(x)$ which do not test labels
 - * transition at node v depends on $(\varphi_1(v), \dots, \varphi_l(v))$

Step 6: Proposition

For each oracle automaton A there are $s \in L_1$ and $t \notin L_1$ with the same root-to-root loops of A

Proof idea for proposition

Slight extension of TWA-proof

Proof of Theorem 4 (cont.)

- $i = 0$: follows from Proposition (even TWA-case sufficient)
- $i - 1 \rightarrow i$: (**Step 7**)
 - Choose m large enough wrt A, i
 - Pick $s \in L_1, t \notin L_1$ such that no oracle automaton of size $\leq m$ can distinguish s from t
 - $s_{i+1} := s$, where each a -leaf is replaced by s_i and each b -leaf by t_i
 - (t_{i+1} accordingly)

 - Assumption: i -type of A distinguishes s_{i+1} from t_{i+1}

\Rightarrow there is an oracle automaton O of size $\leq m$ which distinguishes s from t

\Rightarrow contradiction

Finale

Proof of Theorem 4 (cont.)

Step 8: simulation of A by O :

● 3 cases:

– all pebbles lifted, A 's head above level i :

O just does the same as A

– all pebbles lifted, A 's head moves below level i :

→ O reads a iff subtree is s_i

→ transition corresponding to i -loops of A on s_i and t_i

– A drops pebble i (above level i):

subcomputation can be simulated by

$(i - 1)$ -pebble automaton A' which does not read labels

⇒ behavior of A' can be described by formulas

$\varphi_1(x), \dots, \varphi_l(x)$

Finale

Proof of Theorem 4 (cont.)

Step 8: simulation of A by O :

● 3 cases:

- all pebbles lifted, A 's head above level i :
 O just does the same as A
- all pebbles lifted, A 's head moves below level i :
→ O reads a iff subtree is s_i
→ transition corresponding to i -loops of A on s_i and t_i
- A drops pebble i (above level i):
subcomputation can be simulated by
 $(i - 1)$ -pebble automaton A' which does not read labels

⇒ behavior of A' can be described by formulas
 $\varphi_1(x), \dots, \varphi_l(x)$

Step 9: conclude $PA \subsetneq REG$

$$\bigcup_n L_n \notin PA \text{ but also}$$
$$\bigcup_n L_n \notin REG$$

Finale

Proof of Theorem 4 (cont.)

Step 8: simulation of A by O :

● 3 cases:

– all pebbles lifted, A 's head above level i :

O just does the same as A

– all pebbles lifted, A 's head moves below level i :

→ O reads a iff subtree is s_i

→ transition corresponding to i -loops of A on s_i and t_i

– A drops pebble i (above level i):

subcomputation can be simulated by

$(i - 1)$ -pebble automaton A' which does not read labels

⇒ behavior of A' can be described by formulas

$\varphi_1(x), \dots, \varphi_l(x)$

Step 9: conclude $\text{PA} \subsetneq \text{REG}$

$\bigcup_n L_n \notin \text{PA}$ but also

$\bigcup_n L_n \notin \text{REG}$

→ Define L inductively:

– $L_{\text{even}} \subseteq L$

– L closed under partial L -unfolding

⇒ $L \in \text{REG} - \text{PA}$ and all

$s_n \in L, t_n \notin L$

Proof summary

1. $\mathbf{FO} + \text{posTC}^1 = \mathbf{sPA}$ [Engelfriet, Hoogeboom 06]
2. Formalize behavior of an automaton on a tree t (\rightarrow type)
3. Strong pebbles do not give extra power
4. For each n , define L_n such that
 $L_n \in \mathbf{DPA}_n - \mathbf{PA}_{n-1}$
5. Towards a contradiction, fix A for L_n
6. Show that oracle automata do not accept L_1
Fix $s \in L_1, t \notin L_1$ that can not be distinguished by
oracle automaton of size $\leq m$
7. Define $s_1, \dots, s_n \in L_n, t_1, \dots, t_n \notin L_n$
8. Show: i -type of s_i and t_i are identical
(otherwise: there is oracle automaton of size $\leq m$
distinguishing s from t)
9. Construct L from L_1, L_2, L_3, \dots

Further results and conclusion

Theorem

- For each $n \geq 0$,
 - $\text{DPA}_n \subsetneq \text{DPA}_{n+1}$
 - $\text{TWA} \not\subseteq \text{DPA}_n$
 - $\text{sDPA}_n = \text{DPA}_n$
 - sDPA_n is closed under complement

Open problems

- $\text{FO} + \text{DTC} \subsetneq \text{FO} + \text{posTC}$?
- $\text{FO} + \text{posTC} \subsetneq \text{FO} + \text{TC}$?
- $\text{FO} + \text{TC} \subsetneq \text{REG}$?
- Number of Pebbles \equiv TC-depth?