An improved construction of deterministic ω-automaton from derivatives

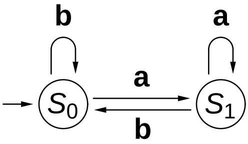
Roman Redziejowski

CS&P 2011

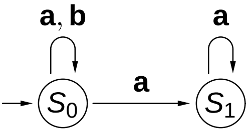
What is ω-automaton?

Automaton: states, transitions

deterministic



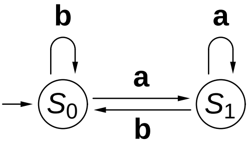
nondeterministic



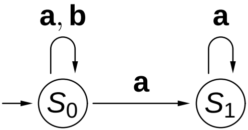
What is ω-automaton?

Automaton: states, transitions

deterministic



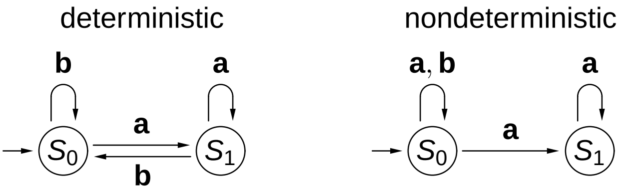
nondeterministic



Omega-automaton:
recognizes ω-languages (sets of infinite words).

What is ω-automaton?

Automaton: states, transitions

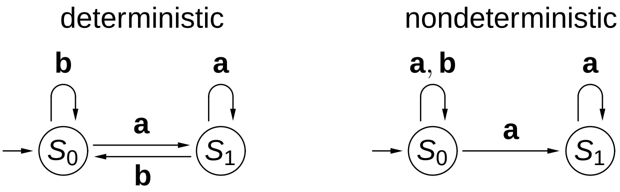


Omega-automaton:
recognizes ω-languages (sets of infinite words).

How: infinite word w accepted $\Leftrightarrow$ exists an accepting run on w.

What is ω-automaton?

Automaton: states, transitions

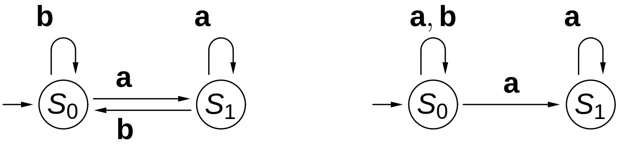


Omega-automaton:
recognizes ω-languages (sets of infinite words).

How: infinite word w accepted $\Leftrightarrow$ exists an accepting run on w.

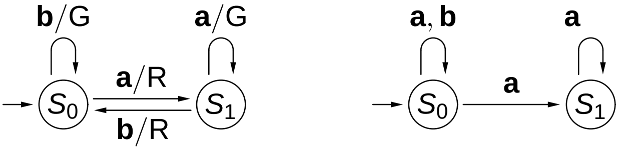
Accepting run defined via set of states visited infinitely often (Büchi, Muller, Rabin, Streett, parity...)

Alternative acceptance



Accepting run can also be defined in terms of transitions.

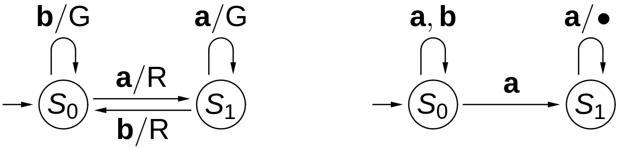
Alternative acceptance



Accepting run can also be defined in terms of transitions.

G inifinitely often & R finitely often recognizes $(\mathbf{a} \cup \mathbf{b})^*(\mathbf{a}^\omega \cup \mathbf{b}^\omega)$.

Alternative acceptance



Accepting run can also be defined in terms of transitions.

G inifinitely often & R finitely often recognizes $(\mathbf{a} \cup \mathbf{b})^*(\mathbf{a}^\omega \cup \mathbf{b}^\omega)$.

Blob • inifinitely often recognizes $(\mathbf{a} \cup \mathbf{b})^*\mathbf{a}^\omega$.

Each ω-automaton recognizes an ω-*regular language* described by an ω-*regular expression* such as $(\mathbf{a} \cup \mathbf{b})^*(\mathbf{a}^\omega \cup \mathbf{b}^\omega)$ or $(\mathbf{a} \cup \mathbf{b})^*\mathbf{a}^\omega$.

Each ω-automaton recognizes an ω-*regular language* described by an ω-*regular expression* such as
$(\mathbf{a} \cup \mathbf{b})^*(\mathbf{a}^\omega \cup \mathbf{b}^\omega)$ or $(\mathbf{a} \cup \mathbf{b})^*\mathbf{a}^\omega$.

Recalling: ω-regular language
is constructed from $\varnothing$, $\{\varepsilon\}$, and $\{a\}$ for $a \in \Sigma$
by a finite number of applications of
union, product, star, omega.

ω-regular language

Each ω-automaton recognizes an ω-*regular language* described by an ω-*regular expression* such as
$(\mathbf{a} \cup \mathbf{b})^*(\mathbf{a}^\omega \cup \mathbf{b}^\omega)$ or $(\mathbf{a} \cup \mathbf{b})^*\mathbf{a}^\omega$.

Recalling: ω-regular language
is constructed from $\varnothing$, $\{\varepsilon\}$, and $\{a\}$ for $a \in \Sigma$
by a finite number of applications of
union, product, star, omega.

(*Regular* language is constructed using only
union, product, and star.)

The problem

Given an an ω-regular expression

construct deterministic ω-automaton

recognizing the language

defined by that expression.

What is derivative?

(Brzozowski 1964)

Derivative of $X \subseteq \Sigma^\infty$ with respect to $w \in \Sigma^*$:
set of words obtained by stripping the initial w
from words in X starting with w.

$$\partial_w X = \{z \in \Sigma^\infty \,|\, wz \in X\}$$

What is derivative?

(Brzozowski 1964)

Derivative of $X \subseteq \Sigma^\infty$ with respect to $w \in \Sigma^*$:
set of words obtained by stripping the initial w
from words in X starting with w.

$$\partial_w X = \{z \in \Sigma^\infty \,|\, wz \in X\}$$

Use: suppose you check if input is in X.
After reading w, remains to check if the rest is in $\partial_w X$.

Derivatives of ω-regular language

Results from Brzozowski 1964, extended to ω-languages.

(1) An (ω-)regular language has finitely many distinct derivatives.

Derivatives of ω-regular language

Results from Brzozowski 1964, extended to ω-languages.

(1) An (ω-)regular language has finitely many distinct derivatives.

(2) These derivatives are also (ω-)regular and can be effectively computed using rules such as these:

$$\partial_a \varnothing = \partial_a \{\varepsilon\} = \varnothing\,, \qquad \partial_a (X \cup Y) = \partial_a X \cup \partial_a Y\,,$$
$$\partial_a \{a\} = \varepsilon\,, \qquad \partial_a (XY) = (\partial_a X) Y \cup \nu(X)(\partial_a Y)\,,$$
$$\partial_{wa} X = \partial_a (\partial_w X)\,, \qquad \text{etc..}$$

Using derivatives to recognize regular language

Identify states with languages they recognize.

Using derivatives to recognize regular language

Identify states with languages they recognize.

Suppose you start in state $D_0 = (\mathbf{a} \cup \mathbf{b})^* \mathbf{a}$.
If you read **a**, go to state $\partial_{\mathbf{a}} X = (\mathbf{a} \cup \mathbf{b})^* \mathbf{a} \cup \varepsilon = D_1$.
If you read **b**, go to state $\partial_{\mathbf{b}} X = (\mathbf{a} \cup \mathbf{b})^* \mathbf{a} = D_0$.

Identify states with languages they recognize.

Suppose you start in state $D_0 = (\mathbf{a} \cup \mathbf{b})^* \mathbf{a}$.
If you read **a**, go to state $\partial_{\mathbf{a}} X = (\mathbf{a} \cup \mathbf{b})^* \mathbf{a} \cup \varepsilon = D_1$.
If you read **b**, go to state $\partial_{\mathbf{b}} X = (\mathbf{a} \cup \mathbf{b})^* \mathbf{a} = D_0$.

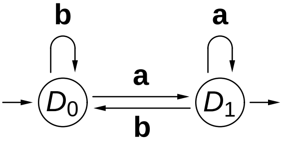
From state D_1:
If there is no more input, you are done because $\varepsilon \in D_1$.
If you read **a**, go to state $\partial_{\mathbf{a}} D_1 = D_1$.
If you read **b**, go to state $\partial_{\mathbf{b}} D_1 = D_0$.

Using derivatives to recognize regular language

Identify states with languages they recognize.

Suppose you start in state $D_0 = (\mathbf{a} \cup \mathbf{b})^*\mathbf{a}$.
If you read **a**, go to state $\partial_\mathbf{a} X = (\mathbf{a} \cup \mathbf{b})^*\mathbf{a} \cup \varepsilon = D_1$.
If you read **b**, go to state $\partial_\mathbf{b} X = (\mathbf{a} \cup \mathbf{b})^*\mathbf{a} = D_0$.

From state D_1:
If there is no more input, you are done because $\varepsilon \in D_1$.
If you read **a**, go to state $\partial_\mathbf{a} D_1 = D_1$.
If you read **b**, go to state $\partial_\mathbf{b} D_1 = D_0$.



Brzozowski's derivative automaton

Automaton recognizing a **regular** language X.

- **States:** distinct derivatives of X.
- **Initial state:** $\partial_\varepsilon X$.
- **Transitions:** $D \xrightarrow{a} \partial_a D$.
- **Final state:** any derivative containing ε.

Does not work for ω-regular language

$$X = (\mathbf{a} \cup \mathbf{b})^*(\mathbf{a}^\omega \cup (\mathbf{ab})^\omega)$$

Does not work for ω-regular language

$$X = (\mathbf{a} \cup \mathbf{b})^*(\mathbf{a}^\omega \cup (\mathbf{ab})^\omega)$$

$$\partial_{\mathbf{a}} X = (\mathbf{a} \cup \mathbf{b})^*(\mathbf{a}^\omega \cup (\mathbf{ab})^\omega) = X$$

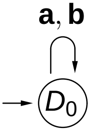
$$\partial_{\mathbf{b}} X = (\mathbf{a} \cup \mathbf{b})^*(\mathbf{a}^\omega \cup (\mathbf{ab})^\omega) = X$$

Does not work for ω-regular language

$$X = (\mathbf{a} \cup \mathbf{b})^*(\mathbf{a}^\omega \cup (\mathbf{ab})^\omega)$$

$$\partial_{\mathbf{a}} X = (\mathbf{a} \cup \mathbf{b})^*(\mathbf{a}^\omega \cup (\mathbf{ab})^\omega) = X$$

$$\partial_{\mathbf{b}} X = (\mathbf{a} \cup \mathbf{b})^*(\mathbf{a}^\omega \cup (\mathbf{ab})^\omega) = X$$



Too few transitions to recognize X.

A bright idea

Distinguish derivatives that bite the omega part:

Insert "marker" $\sharp$ before the operand of each $^\omega$.
Take derivatives with respect to a and $\sharp a$.

A bright idea

Distinguish derivatives that bite the omega part:

Insert "marker" $\sharp$ before the operand of each $\,^{\omega}$.
Take derivatives with respect to a and $\sharp a$.

For example:

$$X = (\mathbf{a} \cup \mathbf{b})^*(\mathbf{a}^\omega \cup (\mathbf{ab})^\omega)$$

$$X' = (\mathbf{a} \cup \mathbf{b})^*((\sharp\mathbf{a})^\omega \cup (\sharp\mathbf{ab})^\omega)$$

A bright idea

Distinguish derivatives that bite the omega part:

Insert "marker" $\sharp$ before the operand of each ${}^{\omega}$.
Take derivatives with respect to a and $\sharp a$.

For example:

$$X = (\mathbf{a} \cup \mathbf{b})^*(\mathbf{a}^\omega \cup (\mathbf{ab})^\omega)$$

$$X' = (\mathbf{a} \cup \mathbf{b})^*((\sharp\mathbf{a})^\omega \cup (\sharp\mathbf{ab})^\omega)$$

$$\partial_{\mathbf{a}} X' = X'$$

$$\partial_{\sharp\mathbf{a}} X' = (\sharp\mathbf{a})^\omega \cup \mathbf{b}\,(\sharp\mathbf{ab})^\omega$$

A bright idea

New derivative automaton:

- **States:** nonempty derivatives of X'.
- **Initial state:** $\partial_\varepsilon X'$.
- **Transitions:**

$$D \xrightarrow{a} \partial_a D,$$
$$D \xrightarrow{a/\bullet} \partial_{\sharp a} D \qquad \text{(enters } \omega\text{-iteration).}$$

- **Accepting run:** infinitely many transitions with $\bullet$.

Derivative automaton

$$X' = (\mathbf{a} \cup \mathbf{b})^*((\sharp\mathbf{a})^\omega \cup (\sharp\mathbf{ab})^\omega)$$

Derivative automaton

$$X' = (\mathbf{a} \cup \mathbf{b})^*((\sharp\mathbf{a})^\omega \cup (\sharp\mathbf{ab})^\omega)$$

$$D_0 = \partial_\varepsilon X' = X';$$
$$D_1 = \partial_{\sharp\mathbf{a}} X' = (\sharp\mathbf{a})^\omega \cup \mathbf{b}\,(\sharp\mathbf{ab})^\omega;$$
$$D_2 = \partial_{\sharp\mathbf{ab}} X' = (\sharp\mathbf{ab})^\omega;$$
$$D_3 = \partial_{\sharp\mathbf{a}\sharp\mathbf{a}} X' = (\sharp\mathbf{a})^\omega;$$
$$D_4 = \partial_{\sharp\mathbf{ab}\sharp\mathbf{a}} X' = \mathbf{b}\,(\sharp\mathbf{ab})^\omega.$$

$$\partial_{\mathbf{a}} D_0 = \partial_{\mathbf{b}} D_0 = D_0;$$
$$\partial_{\sharp\mathbf{a}} D_0 = D_1;$$
$$\partial_{\mathbf{b}} D_1 = \partial_{\mathbf{b}} D_4 = D_2;$$
$$\partial_{\sharp\mathbf{a}} D_1 = D_3;$$
$$\partial_{\sharp\mathbf{a}} D_2 = D_4.$$

Derivative automaton

$$X' = (\mathbf{a} \cup \mathbf{b})^*((\sharp\mathbf{a})^\omega \cup (\sharp\mathbf{ab})^\omega)$$

$D_0 = \partial_\varepsilon X' = X'$;

$D_1 = \partial_{\sharp\mathbf{a}} X' = (\sharp\mathbf{a})^\omega \cup \mathbf{b}\,(\sharp\mathbf{ab})^\omega$;

$D_2 = \partial_{\sharp\mathbf{ab}} X' = (\sharp\mathbf{ab})^\omega$;

$D_3 = \partial_{\sharp\mathbf{a}\sharp\mathbf{a}} X' = (\sharp\mathbf{a})^\omega$;

$D_4 = \partial_{\sharp\mathbf{ab}\sharp\mathbf{a}} X' = \mathbf{b}\,(\sharp\mathbf{ab})^\omega$.

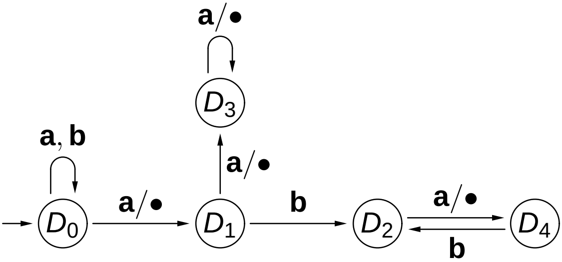
$\partial_{\mathbf{a}} D_0 = \partial_{\mathbf{b}} D_0 = D_0$;

$\partial_{\sharp\mathbf{a}} D_0 = D_1$;

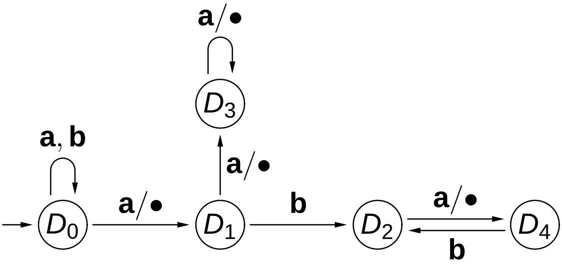
$\partial_{\mathbf{b}} D_1 = \partial_{\mathbf{b}} D_4 = D_2$;

$\partial_{\sharp\mathbf{a}} D_1 = D_3$;

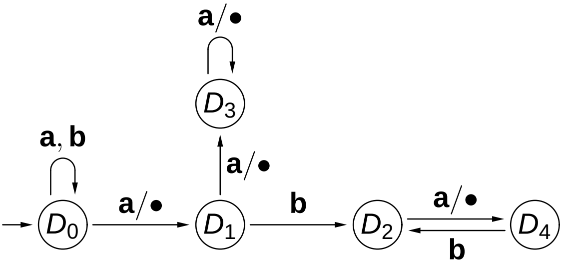
$\partial_{\sharp\mathbf{a}} D_2 = D_4$.



Derivative automaton

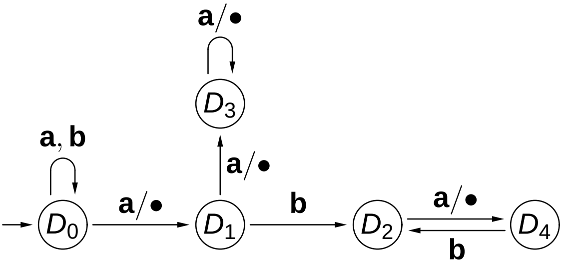


Derivative automaton



Has run with infinitely many • $\Leftrightarrow$ input is in $(\mathbf{a} \cup \mathbf{b})^*(\mathbf{a}^\omega \cup (\mathbf{ab})^\omega)$.

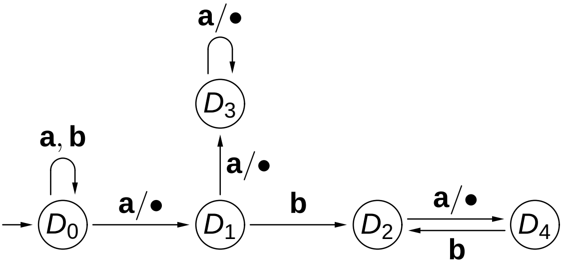
Derivative automaton



Has run with infinitely many • $\Leftrightarrow$ input is in $(\mathbf{a} \cup \mathbf{b})^*(\mathbf{a}^\omega \cup (\mathbf{ab})^\omega)$.

But, it is nondeterministic.

Derivative automaton



Has run with infinitely many • ⇔ input is in $(\mathbf{a} \cup \mathbf{b})^*(\mathbf{a}^\omega \cup (\mathbf{ab})^\omega)$.

But, it is nondeterministic.

There exist determinization methods.

Determinization

Different ways to obtain states of deterministic automaton.

Determinization

Different ways to obtain states of deterministic automaton.

Safra 1988 used trees built from the original states.

Determinization

Different ways to obtain states of deterministic automaton.

Safra 1988 used trees built from the original states.

RR 1999 used annotations to run tree.

Determinization

Different ways to obtain states of deterministic automaton.

Safra 1988 used trees built from the original states.

RR 1999 used annotations to run tree.

Piterman 2007 used a numbering trick to improve Safra's trees.

Determinization

Different ways to obtain states of deterministic automaton.

Safra 1988 used trees built from the original states.

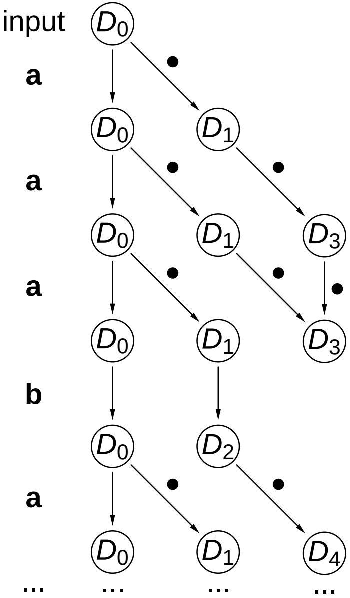
RR 1999 used annotations to run tree.

Piterman 2007 used a numbering trick to improve Safra's trees.

We are going to improve RR 1999 by using annotations to run DAG [1] enhanced with Piterman's trick.

[1] Directed Acyclic Graph

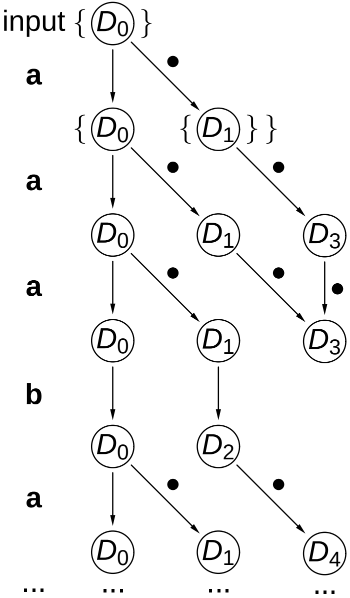
Run DAG



All possible runs on given input.

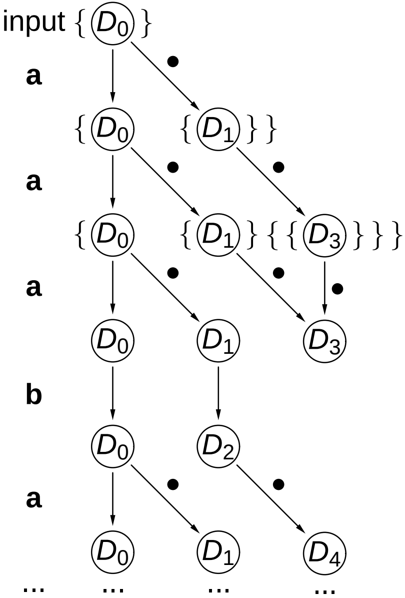
Input is in X if and only if the DAG contains a *live path*: path with infinitely many •.

Annotating the run DAG



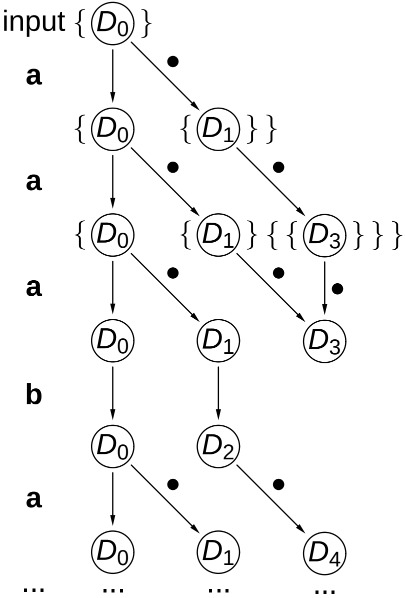
Brackets enclose descendants
+ any node reached via •

Annotating the run DAG



Brackets enclose descendants
+ any node reached via •

Easier to do it on the side...



$\{ D_0 \}$

$\{ D_0 \{ D_1 \} \}$

$\{ D_0 \{ D_1 \} \{ \{ D_3 \} \} \}$

Green event

Watch for this situation:

$\{ D_{i_1} \ldots D_{i_n} \}$ level i

... ...

$\{ \{ D_{j_1} \} \{ D_{j_2} \} \ldots \{ D_{j_m} \} \}$ level j

Green event

Watch for this situation:

$\{ D_{i_1} \dots D_{i_n} \}$ level i

... ...

$\{ \{ D_{j_1} \} \{ D_{j_2} \} \dots \{ D_{j_m} \} \}$ level j

All paths from level i to level j are marked with $\bullet$.

Green event

Watch for this situation:

$$\{\, D_{i_1} \;\ldots\; D_{i_n} \,\} \qquad \text{level } i$$

$$\ldots \;\; \ldots$$

$$\{\, \{\, D_{j_1} \,\}\{\, D_{j_2} \,\}\ldots\{\, D_{j_m} \,\} \,\} \qquad \text{level } j$$

All paths from level i to level j are marked with $\bullet$.
We call this "green event" for the enclosing brackets, remove inner brackets, and emit green light.

$$\{\, D_{i_1} \;\ldots\; D_{i_n} \,\}$$

$$\ldots \;\; \ldots$$

$$\{\; D_{j_1} \;\; D_{j_2} \;\ldots\; D_{j_m} \;\} \qquad \Rightarrow \text{G}$$

Green event

Watch for this situation:

$$\{\, D_{i_1} \;\ldots\; D_{i_n} \,\} \qquad \text{level } i$$

$$\ldots\;\ldots$$

$$\{\, \{\, D_{j_1} \,\}\{\, D_{j_2} \,\}\ldots\{\, D_{j_m} \,\} \,\} \qquad \text{level } j$$

All paths from level i to level j are marked with $\bullet$.
We call this "green event" for the enclosing brackets, remove inner brackets, and emit green light.

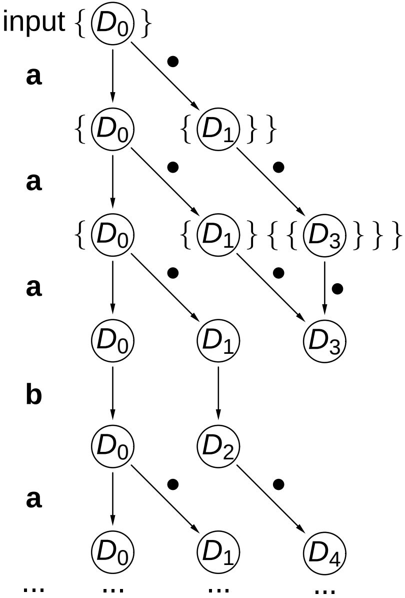
$$\{\, D_{i_1} \;\ldots\; D_{i_n} \,\}$$

$$\ldots\;\ldots$$

$$\{\; D_{j_1} \;\; D_{j_2} \;\ldots\; D_{j_m} \;\} \qquad \Rightarrow \text{G}$$

Repeated green events $\Rightarrow$ live path exists.

Green event

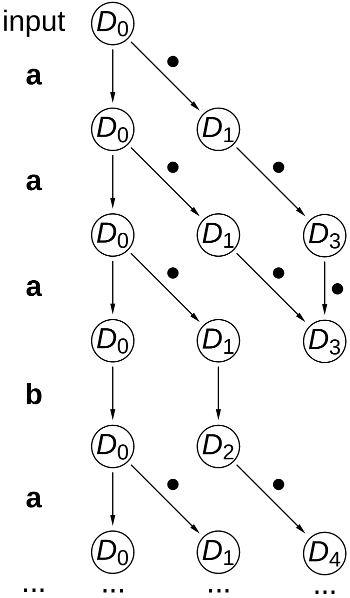


$\{ D_0 \}$

$\{ D_0 \{ D_1 \} \}$

$\{ D_0 \{ D_1 \} \{ D_3 \} \}$ $\Rightarrow$ G

It must be the same pair of brackets all the time!



$\{ D_0 \}$

$\{ D_0 \{ D_1 \} \}$

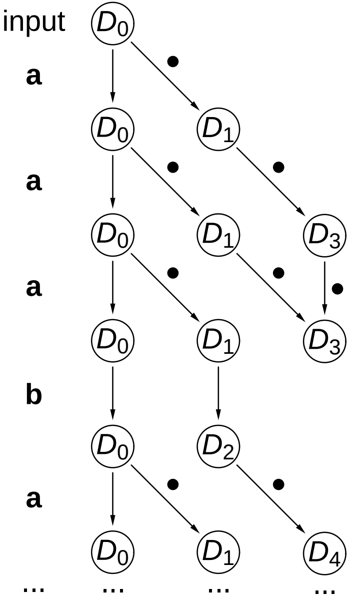
$\{ D_0 \{ D_1 \} \{ D_3 \} \} \quad \Rightarrow G$

$\{ D_0 \{ D_1 \} \{ D_3 \} \} \quad \Rightarrow G$

$\{ D_0 \{ D_2 \} \}$

$\{ D_0 \{ D_1 \} \{ D_4 \} \} \quad \Rightarrow G$

Solution: numbering



$\{\ D_0\ \}$
1 1

$\{\ D_0\ \{\ D_1\ \}\ \}$
1 2 2 1

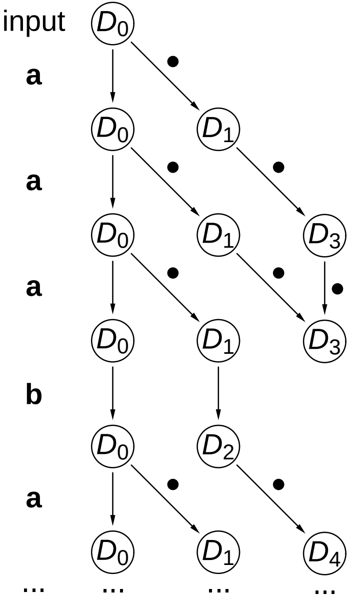
$\{\ D_0\ \{\ D_1\ \}\ \{\ D_3\ \}\ \}\ \Rightarrow$ G 2
1 3 3 2 2 1

$\{\ D_0\ \{\ D_1\ \}\ \{\ D_3\ \}\ \}\ \Rightarrow$ G 2
1 3 3 2 2 1

$\{\ D_0\ \{\ D_2\ \}\ \}$
1 3 3 1

$\{\ D_0\ \{\ D_1\ \}\ \{\ D_4\ \}\ \}\ \Rightarrow$ G 3
1 4 4 3 3 1

But numbers cannot grow to ∞ - must be reused



$\{ D_0 \}$
1 1

$\{ D_0 \{ D_1 \} \}$
1 2 2 1

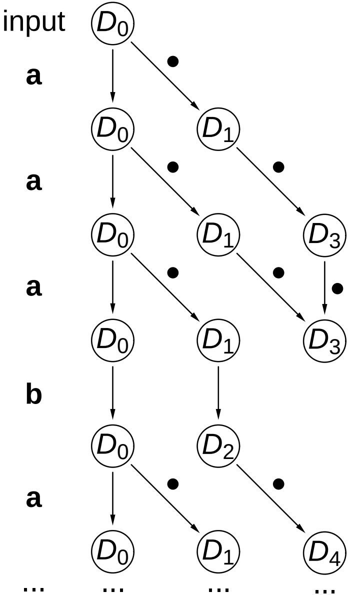
$\{ D_0 \{ D_1 \} \{ D_3 \} \}$ $\Rightarrow$ G 2
1 3 3 2 2 1

$\{ D_0 \{ D_1 \} \{ D_3 \} \}$ $\Rightarrow$ G 2
1 3 3 2 2 1

$\{ D_0 \{ D_2 \} \}$ oops! 2 reused
1 2 2 1

$\{ D_0 \{ D_1 \} \{ D_4 \} \}$ $\Rightarrow$ G 2
1 3 3 2 2 1

Red event to signal reuse: next G2 is another path



$\{ D_0 \}$
1 1

$\{ D_0 \{ D_1 \} \}$
1 2 2 1

$\{ D_0 \{ D_1 \} \{ D_3 \} \}$ $\Rightarrow$ G 2
1 3 3 2 2 1

$\{ D_0 \{ D_1 \} \{ D_3 \} \}$ $\Rightarrow$ G 2
1 3 3 2 2 1

$\{ D_0 \{ D_2 \} \}$ $\Rightarrow$ R 2
1 2 2 1

$\{ D_0 \{ D_1 \} \{ D_4 \} \}$ $\Rightarrow$ G 2
1 3 3 2 2 1

Acceptance condition

Live path exists - that is, input is in X - if and only if

$\Rightarrow$ G 2 occurs infinitely often and

$\Rightarrow$ R 2 occurs finitely often.

Acceptance condition

Live path exists - that is, input is in X - if and only if

$\Rightarrow$ G 2 occurs infinitely often and

$\Rightarrow$ R 2 occurs finitely often.

(But just wait, it will be more complicated.)

Meanwhile, note this:

No pictures needed!

We can produce annotations without ever constructing the derivative automaton or drawing the DAG!

Meanwhile, note this:

No pictures needed!

We can produce annotations without ever constructing the derivative automaton or drawing the DAG!

Start with $\underset{1}{\{} D_0 \underset{1}{\}}$.

Meanwhile, note this:

No pictures needed!

We can produce annotations without ever constructing the derivative automaton or drawing the DAG!

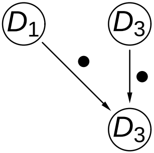
Start with $\underset{1}{\{}\, D_0 \,\underset{1}{\}}$.

For input letter a, just replace every occurrence of D_i by

$\partial_a D_i \{\, \partial_{(\sharp a)} D_i \,\}$,

then remove empty derivatives, remove empty brackets, add numbers (indicating reuse), and handle green events.

But there is a snag...

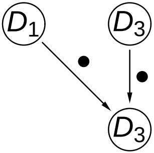


$$\underset{3}{\{}\, D_1 \,\underset{3}{\}}\ \underset{2}{\{}\, D_3 \,\underset{2}{\}}$$

becomes

$$\underset{3}{\{}\ \underset{5}{\{}\, D_3 \,\underset{5}{\}}\ \underset{3}{\}}\ \underset{2}{\{}\ \underset{6}{\{}\, D_3 \,\underset{6}{\}}\ \underset{2}{\}}$$

But there is a snag...



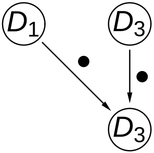
$\{_3\, D_1 \,\}_3 \; \{_2\, D_3 \,\}_2$

becomes

$\{_3\, \{_5\, D_3 \,\}_5 \,\}_3 \; \{_2\, \{_6\, D_3 \,\}_6 \,\}_2$

What to do here? Have to delete one of D_3's.
Which one? We may miss live path.

But there is a snag...



$\underset{3}{\{} D_1 \underset{3}{\}} \; \underset{2}{\{} D_3 \underset{2}{\}}$

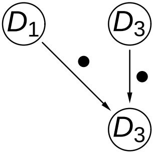
becomes

$\underset{3}{\{} \underset{5}{\{} D_3 \underset{5}{\}} \underset{3}{\}} \; \underset{2}{\{} \underset{6}{\{} D_3 \underset{6}{\}} \underset{2}{\}}$

What to do here? Have to delete one of D_3's.
Which one? We may miss live path.

Safra 1988 orders nodes by "age"
and retains the "oldest" predecessor.

But there is a snag...



$\{_3\, D_1 \,\}_3 \; \{_2\, D_3 \,\}_2$

becomes

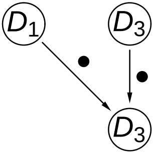
$\{_3\, \{_5\, D_3 \,\}_5 \,\}_3 \; \{_2\, \{_6\, D_3 \,\}_6 \,\}_2$

What to do here? Have to delete one of D_3's.
Which one? We may miss live path.

Safra 1988 orders nodes by "age"
and retains the "oldest" predecessor.

RR 1999 uses left-to right ordering and retains the rightmost.

But there is a snag...



$$\{_3\, D_1\, \}_3\ \ \{_2\, D_3\, \}_2$$

becomes

$$\{_3\, \{_5\, D_3\, \}_5\, \}_3\ \ \{_2\, \{_6\, D_3\, \}_6\, \}_2$$

What to do here? Have to delete one of D_3's.
Which one? We may miss live path.

Safra 1988 orders nodes by "age"
and retains the "oldest" predecessor.

RR 1999 uses left-to right ordering and retains the rightmost.

Piterman 2007 exploits the numbering.
We are going to use his trick.

Numbering and renumbering

Part 1 of the trick is numbering and renumbering of brackets.

New brackets get a number higher than those present.
Removal of empty brackets may leave gaps in the numbering:

1 2 4 6

We close the gaps by reducing numbers above the gap:

Number 4 is changed to 3
Number 6 is changed to 4

$$\begin{array}{cccc} 1 & 2 & 4 & 6 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 1 & 2 & 3 & 4 \end{array}$$

Removing duplicates

Part 2 of the trick is: from multiple occurrences of D_i retain one with the lowest nesting pattern.

$$\underset{1}{\{} D_0 \underset{4}{\{} D_1 \underset{4}{\}} \underset{3}{\{} \underset{5}{\{} D_3 \underset{5}{\}} \underset{3}{\}} \underset{2}{\{} \underset{6}{\{} D_3 \underset{6}{\}} \underset{2}{\}} \underset{1}{\}}$$

Nesting patterns for D_3 are (1 - 3 - 5) and (1 - 2 - 6).
The second is lexicografically lower.
We remove the first occurrence of D_3:

$$\underset{1}{\{} D_0 \underset{4}{\{} D_1 \underset{4}{\}} \underset{3}{\{} \underset{5}{\{} \underset{5}{\}} \underset{3}{\}} \underset{2}{\{} \underset{6}{\{} D_3 \underset{6}{\}} \underset{2}{\}} \underset{1}{\}}$$

Summing up...

How to get the next annotation:

(A1) Replace each D_i as described. Each time assign the lowest unused number to new brackets.

(A2) Remove duplicates, leaving one with lowest nesting pattern.

(A3) Remove all empty pairs of brackets.
Set $r =$ the lowest number on removed pair
or $n+1$ if none removed ($n =$ number of derivatives).

(A4) Handle green events.
Set $g =$ the lowest number on green pair or $n+1$ if none.

(A5) Renumber brackets to fill the gaps.

(A6) If $g < r$, append $\Rightarrow \mathrm{G}\,g$ on the right.
If $r \leq g$ and $r \neq n+1$, append $\Rightarrow \mathrm{R}\,r$.

Example for input **a**

before:	$\underset{1}{\{} D_0 \underset{3}{\{} D_1 \underset{3}{\}} \underset{2}{\{} D_3 \underset{2}{\}} \underset{1}{\}}$
replace D_i's:	$\underset{1}{\{} D_0 \underset{4}{\{} D_1 \underset{4}{\}} \underset{3}{\{} \underset{5}{\{} D_3 \underset{5}{\}} \underset{3}{\}} \underset{2}{\{} \underset{6}{\{} D_3 \underset{6}{\}} \underset{2}{\}} \underset{1}{\}}$
remove duplicates:	$\underset{1}{\{} D_0 \underset{4}{\{} D_1 \underset{4}{\}} \underset{3}{\{} \underset{5}{\{} \underset{5}{\}} \underset{3}{\}} \underset{2}{\{} \underset{6}{\{} D_3 \underset{6}{\}} \underset{2}{\}} \underset{1}{\}}$
remove empty brackets:	$\underset{1}{\{} D_0 \underset{4}{\{} D_1 \underset{4}{\}} \underset{2}{\{} \underset{6}{\{} D_3 \underset{6}{\}} \underset{2}{\}} \underset{1}{\}} \quad r = 3$
handle green events:	$\underset{1}{\{} D_0 \underset{4}{\{} D_1 \underset{4}{\}} \underset{2}{\{} D_3 \underset{2}{\}} \underset{1}{\}} \quad g = 2$
renumber:	$\underset{1}{\{} D_0 \underset{3}{\{} D_1 \underset{3}{\}} \underset{2}{\{} D_3 \underset{2}{\}} \underset{1}{\}}$
add output:	$\underset{1}{\{} D_0 \underset{3}{\{} D_1 \underset{3}{\}} \underset{2}{\{} D_3 \underset{2}{\}} \underset{1}{\}} \quad \Rightarrow$ G 2

Deterministic automaton

Only finitely many distinct annotations exist, so the following automaton will be finite:

- **States:** Annotations reachable from the initial state by transitions defined below.
- **Initial state:** $\underset{1}{\{}\,\partial_\varepsilon X'\,\underset{1}{\}}$.
- **Transitions:** For a state s and an input letter $a \in \Sigma$, apply (A1)–(A6) to s. The part of the result between, and including, the brackets numbered 1 is the next state. The output is to the right of $\Rightarrow$ (if any).
- **Acceptance condition:** A word $w \in \Sigma^\omega$ is accepted if and only if exists g such that the automaton applied to w emits Gg infinitely many times, and emits any Rr with $r \leq g$ only finitely many times.

States & transitions for $X = (\mathbf{a} \cup \mathbf{b})^*(\mathbf{a}^\omega \cup (\mathbf{ab})^\omega)$

$$A = \underset{1}{\{} D_0 \underset{1}{\}} \quad \xrightarrow{\mathbf{a}} B \quad \xrightarrow{\mathbf{b}} A$$

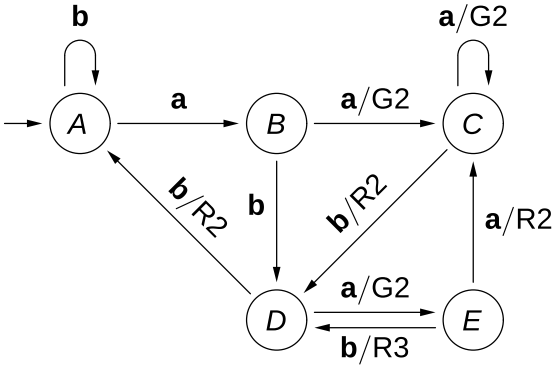
$$B = \underset{1}{\{} D_0 \underset{2}{\{} D_1 \underset{2}{\}} \underset{1}{\}} \quad \xrightarrow{\mathbf{a}} C \Rightarrow \text{G2} \quad \xrightarrow{\mathbf{b}} D$$

$$C = \underset{1}{\{} D_0 \underset{3}{\{} D_1 \underset{3}{\}} \underset{2}{\{} D_3 \underset{2}{\}} \underset{1}{\}} \quad \xrightarrow{\mathbf{a}} C \Rightarrow \text{G2} \quad \xrightarrow{\mathbf{b}} D \Rightarrow \text{R2}$$

$$D = \underset{1}{\{} D_0 \underset{2}{\{} D_2 \underset{2}{\}} \underset{1}{\}} \quad \xrightarrow{\mathbf{a}} E \Rightarrow \text{G2} \quad \xrightarrow{\mathbf{b}} A \Rightarrow \text{R2}$$

$$E = \underset{1}{\{} D_0 \underset{3}{\{} D_1 \underset{3}{\}} \underset{2}{\{} D_4 \underset{2}{\}} \underset{1}{\}} \quad \xrightarrow{\mathbf{a}} C \Rightarrow \text{R2} \quad \xrightarrow{\mathbf{b}} D \Rightarrow \text{R3}$$

Automaton for $X = (\mathbf{a} \cup \mathbf{b})^*(\mathbf{a}^\omega \cup (\mathbf{ab})^\omega)$



Accepting run:
G2 infinitely often, R2 finitely often.
Don't care about R3.

A good question?

Using the method of Safra / Piterman one can estimate the maximum number of possible states to $n^n(n-1)!$ where $n =$ number of states of derivative automaton.

For $n = 5$ this gives 75000.

How come we got only 5 states?

That's all folks ...

Thanks for your attention!