Automata and Logic on Trees: Unranked Tree Automata

Wim Martens[1] Stijn Vansummeren[2]

[1]University of Dortmund, Germany
[2]Hasselt University, Belgium

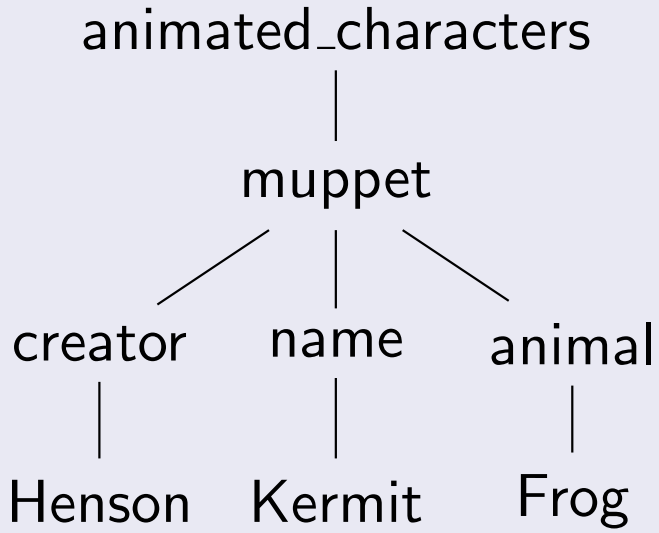
Outline

Outline

Why Unranked Tree Automata?

Data on the Web

```
<animated_characters>
  <muppet creator="Henson">
    <name> Kermit </name>
    <animal> Frog </animal>
  </muppet>
</animated_characters>
```



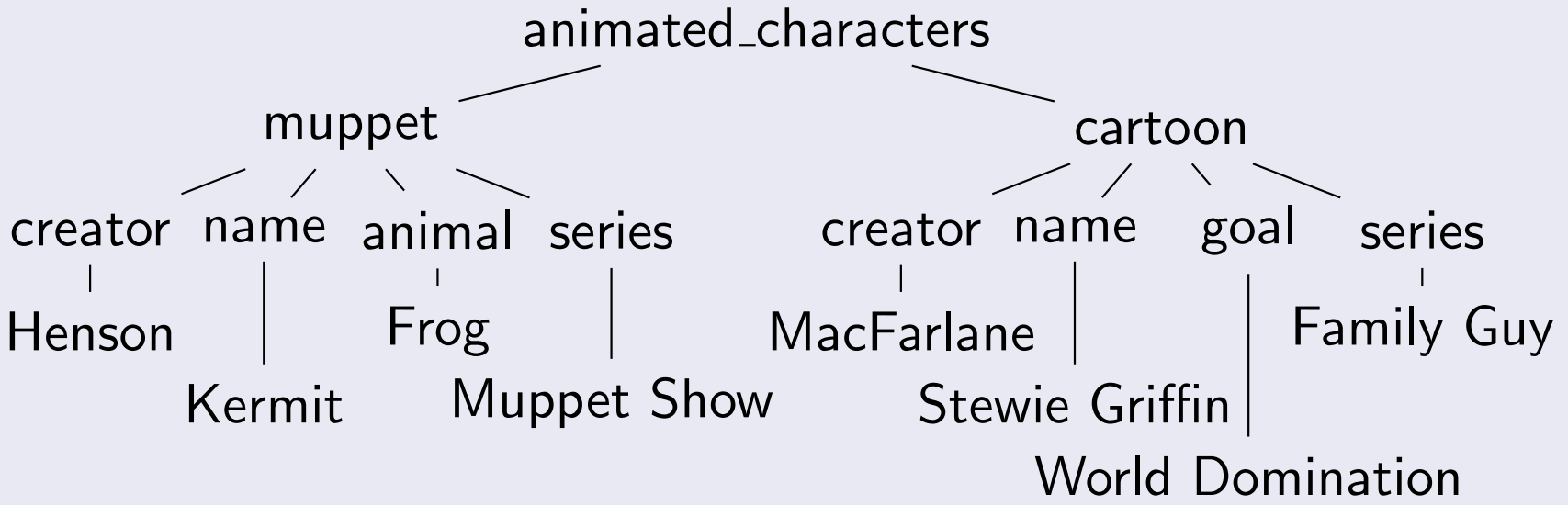
Why Unranked Tree Automata?

Data on the Web

```
<animated_characters>
  <muppet creator="Henson">
    <name> Kermit </name>
    <animal> Frog </animal>
    <series> Muppet Show </series>
  </muppet>
  <cartoon creator="MacFarlane">
    <name> Stewie Griffin </name>
    <goal> World Domination </goal>
    <series> Family Guy </series>
  </cartoon>
</animated_characters>
```

Why Unranked Tree Automata?

Data on the Web



The number of children is not predetermined by a node label
. . . and can be arbitrarily (though finitely) many

Hence, the name unranked trees

What are Unranked Tree Automata?

So, let's extend our automata the straigtforward way

(Let's take the Boolean Circuit example. That one's easy.)

Boolean Circuit Evaluation for Binary Trees

$$\varepsilon \xrightarrow{\text{true}} \mathtt{t}$$

$$\varepsilon \xrightarrow{\text{false}} \mathtt{f}$$

$$(t,t) \xrightarrow{\wedge} \mathtt{t} \qquad (t,f) \xrightarrow{\wedge} \mathtt{f} \qquad (f,t) \xrightarrow{\wedge} \mathtt{f} \qquad (f,f) \xrightarrow{\wedge} \mathtt{f}$$

$$(t,t) \xrightarrow{\vee} \mathtt{t} \qquad (t,f) \xrightarrow{\vee} \mathtt{t} \qquad (f,t) \xrightarrow{\vee} \mathtt{t} \qquad (f,f) \xrightarrow{\vee} \mathtt{f}$$

What are Unranked Tree Automata?

Boolean Circuit Evaluation for Binary Trees

$$\varepsilon \xrightarrow{\text{true}} \mathtt{t}$$

$$\varepsilon \xrightarrow{\text{false}} \mathtt{f}$$

$$(t,t) \xrightarrow{\wedge} \mathtt{t} \qquad (t,f) \xrightarrow{\wedge} \mathtt{f} \qquad (f,t) \xrightarrow{\wedge} \mathtt{f} \qquad (f,f) \xrightarrow{\wedge} \mathtt{f}$$

$$(t,t) \xrightarrow{\vee} \mathtt{t} \qquad (t,f) \xrightarrow{\vee} \mathtt{t} \qquad (f,t) \xrightarrow{\vee} \mathtt{t} \qquad (f,f) \xrightarrow{\vee} \mathtt{f}$$

Boolean Circuit Evaluation for Unranked Trees

$$\varepsilon \xrightarrow{\text{true}} \mathtt{t}$$

$$\varepsilon \xrightarrow{\text{false}} \mathtt{f}$$

$$(t) \xrightarrow{\wedge} \mathtt{t} \qquad (t,t) \xrightarrow{\wedge} \mathtt{t}$$

$$(t,t,t) \xrightarrow{\wedge} \mathtt{t} \qquad (t,t,t,t) \xrightarrow{\wedge} \mathtt{t}$$

What are Unranked Tree Automata?

An Unranked Tree Automaton A consists of:

- States(A): its set of states
- Alphabet(A): the (non-ranked) alphabet
- Rules(A): the transition rules
- Final(A): the final states

Here the rules in Rules(A) are of the form

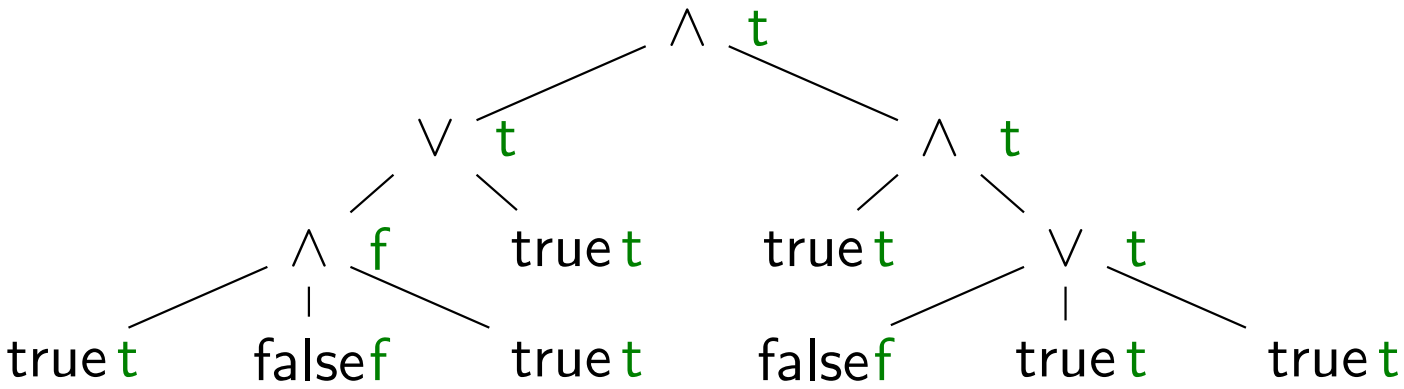
$$L \xrightarrow{a} q,$$

where L is a regular language over States(A)

We assume that no two rules $L_1 \xrightarrow{a} q$, $L_2 \xrightarrow{a} q$ occur

(Instead, $L_1 \cup L_2 \xrightarrow{a} q$)

Example



Boolean Circuit Evaluation

$$\varepsilon \xrightarrow{\text{true}} t \qquad t^+ \xrightarrow{\wedge} t \qquad (t+f)^* f (t+f)^* \xrightarrow{\wedge} f$$

$$\varepsilon \xrightarrow{\text{false}} f \qquad f^+ \xrightarrow{\vee} f \qquad (t+f)^* t (t+f)^* \xrightarrow{\vee} t$$

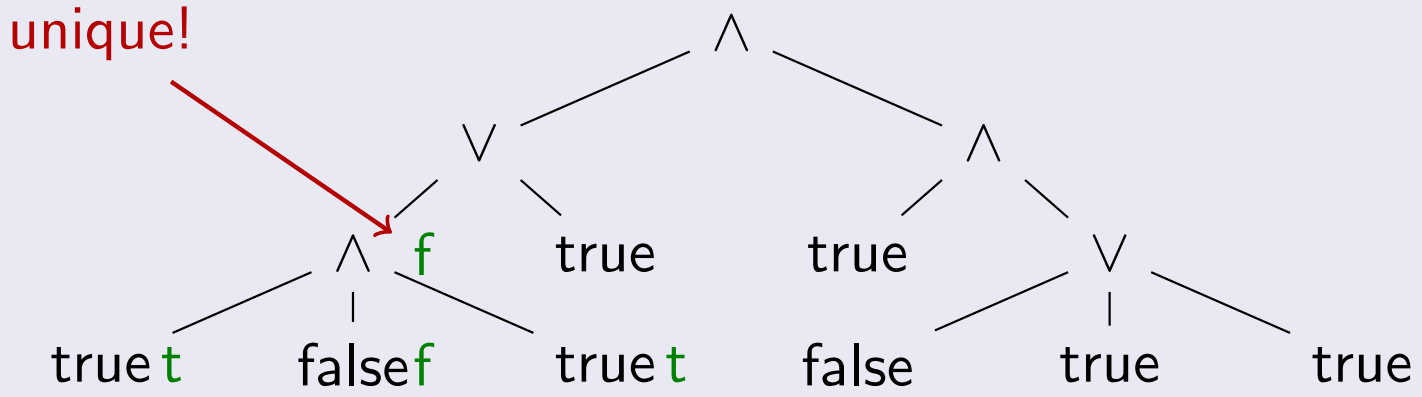
(Blockwise) Deterministic Unranked Tree Automata

Definition ((Blockwise) Deterministic Unranked Tree Automaton)

An unranked tree automaton A is blockwise deterministic if

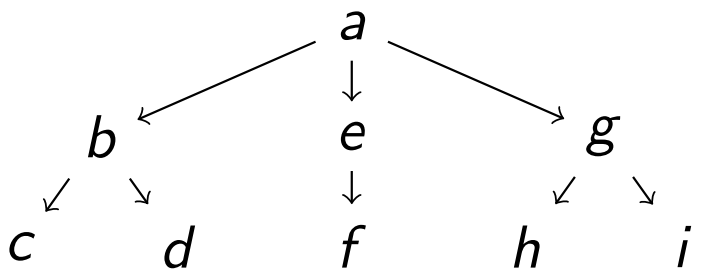
$$\text{for all rules } L_1 \xrightarrow{a} q_1,\ L_2 \xrightarrow{a} q_2,\ L_1 \cap L_2 = \emptyset$$

Why can this be seen as bottom-up determinism?

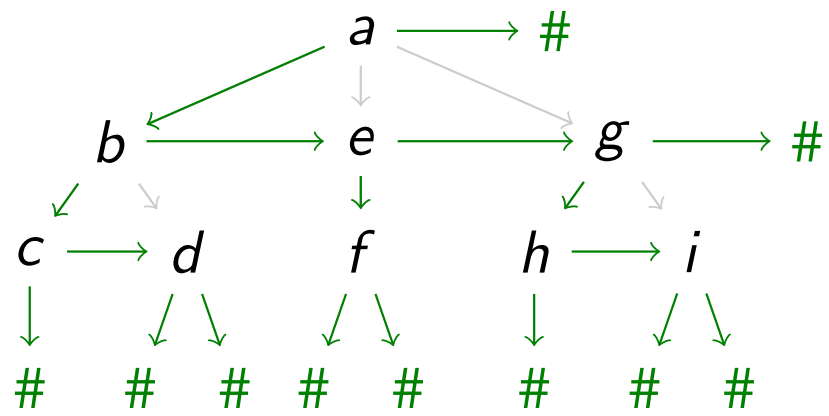


Outline

First-Child Next-Sibling Encoding



First-Child Next-Sibling Encoding



Notation

- fcns(L): first-child next-sibling encoding of (unranked) language L
- $\text{fcns}^{-1}(L)$: first-child next-sibling decoding of (ranked) language L

fcns is a bijection between

- unranked trees over alphabet Σ and
- binary trees over $\Sigma \uplus \{\#^{(0)}\}$ (every $a \in \Sigma$ is binary)

First-Child Next-Sibling Encoding: Automata

Lemma (Tree Automata Encoding Lemma)

(1) *For each UTA A, there is a*

$$\text{BTA } fcns(A) \text{ accepting } fcns(Language(A))$$

(2) *For each BTA A there is a*

$$\text{UTA } fcns^{-1}(A) \text{ accepting } fcns^{-1}(Language(A))$$

$fcns(A)$ and $fcns^{-1}(A)$ are constructible in linear time.

The Result Transfer

Corollaries of the FCNS Lemma:

Unranked Regular Tree Languages are Closed Under

- Union
- Intersection
- Complement
- ...

The Result Transfer

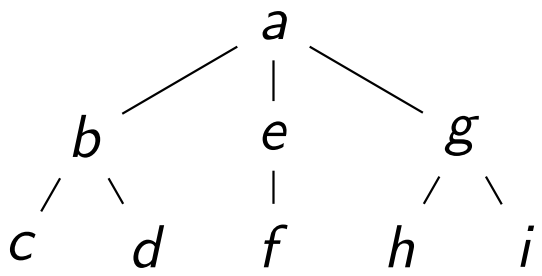
Corollaries of the FCNS Lemma:

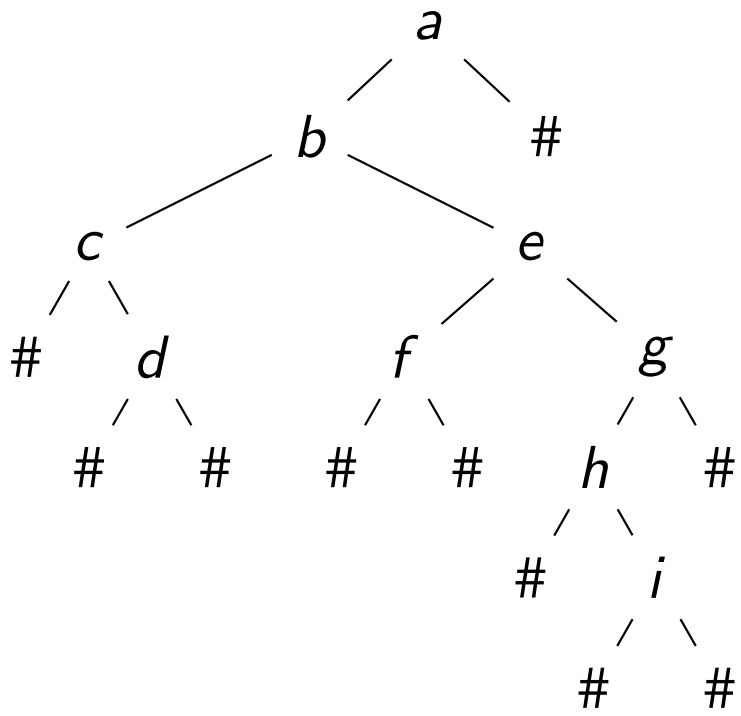
Complexities for Unranked Tree Automata

- Membership: **PTIME**
- (non)-Emptiness: **PTIME**
- Finiteness: **PTIME**
- Universality: **EXPTIME**
- Containment / Inclusion: **EXPTIME**
- Equivalence: **EXPTIME**
- Intersection (non)-Emptimess: **EXPTIME**

Outline

FCNS encoding revisited





FCNS encoding and determinism

Observation

Determinism is not preserved by FCNS encoding!

Determinism Revisited

An Unranked Tree Automaton A consists of:

- States(A): its set of states
- Alphabet(A): the (non-ranked) alphabet
- Rules(A): the transition rules
- Final(A): the final states

Here Rules(A) becomes something completely different

New Rules

Rules(A) are of the form

$$a \longrightarrow D,$$

where $a \in$ Alphabet(A) and D is a finite string automaton

Determinism Revisited

New Rules

Rules(A) are of the form

$$a \longrightarrow D,$$

where $a \in \text{Alphabet}(A)$ and D is a finite string automaton

D is a finite string automaton with

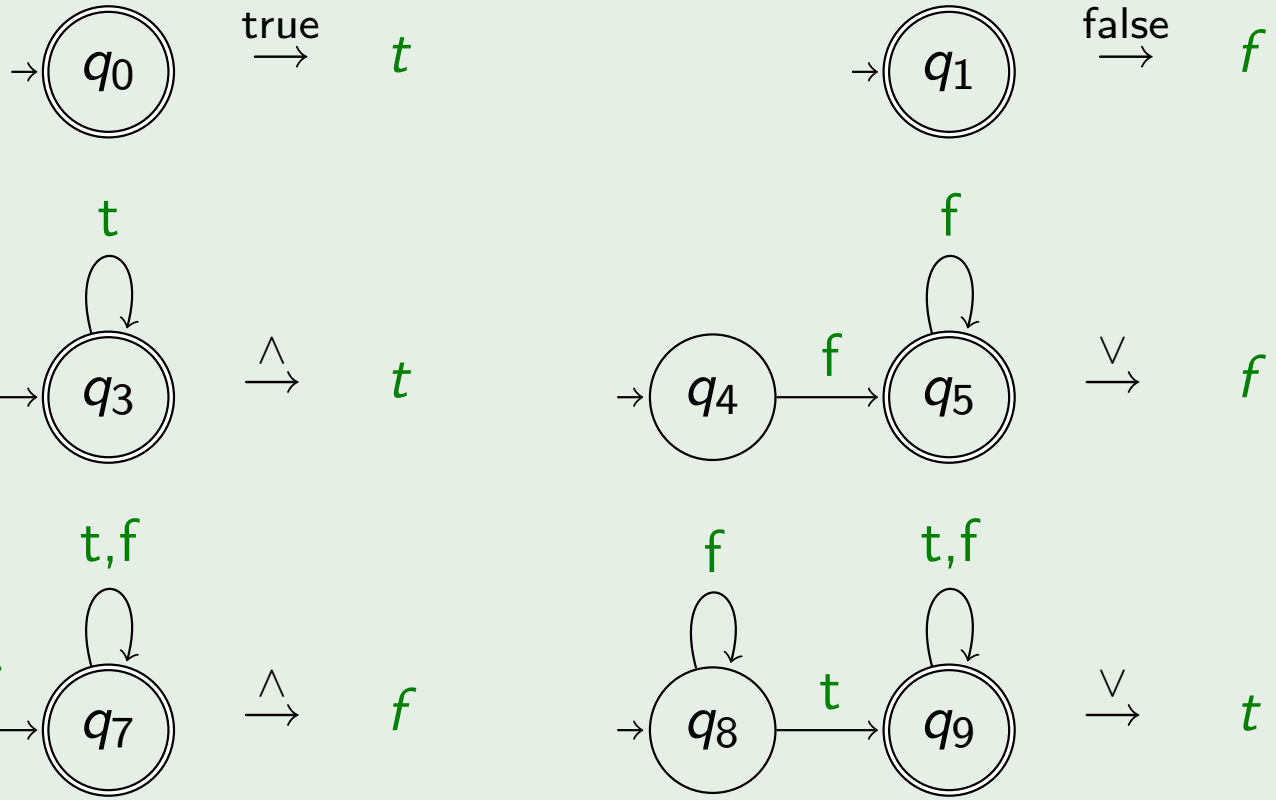
- $\text{Alphabet}(D) = \text{States}(A)$
- $\text{States}(D) = \text{States}(A)$

Intermezzo: Example

Example (Boolean Circuit Evaluation)

$$\varepsilon \xrightarrow{\text{true}} t \qquad t^+ \xrightarrow{\wedge} t \qquad (t+f)^* f (t+f)^* \xrightarrow{\wedge} f$$

$$\varepsilon \xrightarrow{\text{false}} f \qquad f^+ \xrightarrow{\vee} f \qquad (t+f)^* t (t+f)^* \xrightarrow{\vee} t$$



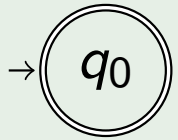
Intermezzo: Example

Example (Boolean Circuit Evaluation)

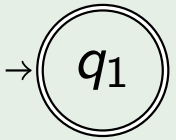
$$\varepsilon \xrightarrow{\text{true}} t \qquad t^+ \xrightarrow{\wedge} t \qquad (t+f)^* f (t+f)^* \xrightarrow{\wedge} f$$

$$\varepsilon \xrightarrow{\text{false}} f \qquad f^+ \xrightarrow{\vee} f \qquad (t+f)^* t (t+f)^* \xrightarrow{\vee} t$$

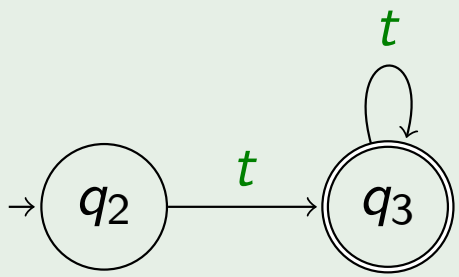
(t, true):



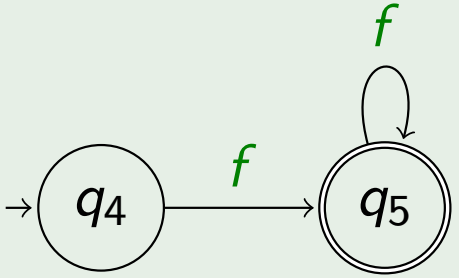
(f, false):



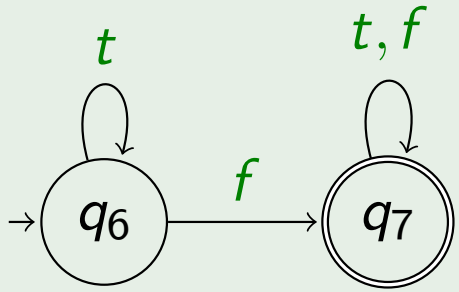
$(t, \wedge)$:



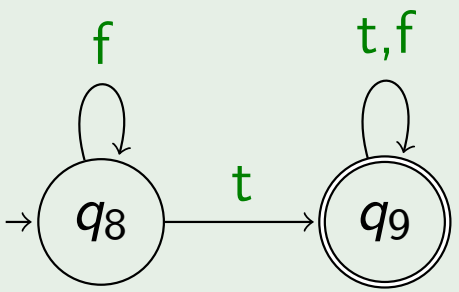
$(f, \vee)$:



$(f, \wedge)$:



$(t, \vee)$:



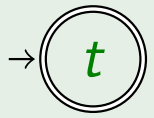
Intermezzo: Example

Example (Boolean Circuit Evaluation)

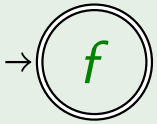
$$\varepsilon \xrightarrow{\text{true}} t \qquad t^+ \xrightarrow{\wedge} t \qquad (t+f)^* f (t+f)^* \xrightarrow{\wedge} f$$

$$\varepsilon \xrightarrow{\text{false}} f \qquad f^+ \xrightarrow{\vee} f \qquad (t+f)^* t (t+f)^* \xrightarrow{\vee} t$$

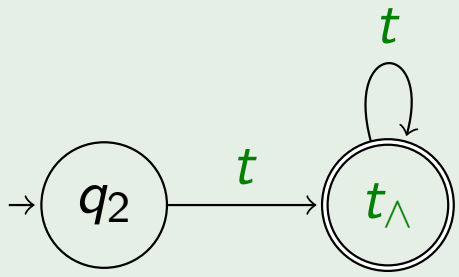
(t, true):



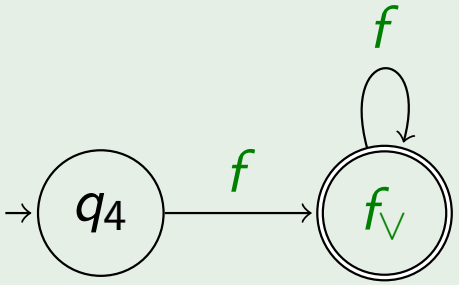
(f, false):



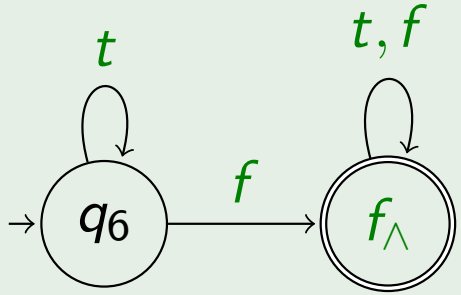
$(t, \wedge)$:



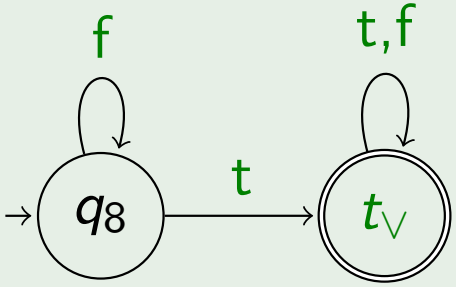
$(f, \vee)$:



$(f, \wedge)$:



$(t, \vee)$:



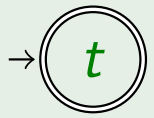
Intermezzo: Example

Example (Boolean Circuit Evaluation)

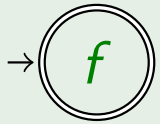
$$\varepsilon \xrightarrow{\text{true}} t \qquad t^+ \xrightarrow{\wedge} t \qquad (t+f)^* f (t+f)^* \xrightarrow{\wedge} f$$

$$\varepsilon \xrightarrow{\text{false}} f \qquad f^+ \xrightarrow{\vee} f \qquad (t+f)^* t (t+f)^* \xrightarrow{\vee} t$$

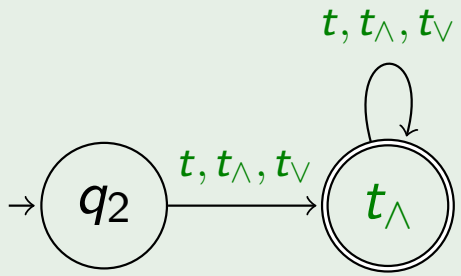
(t, true):



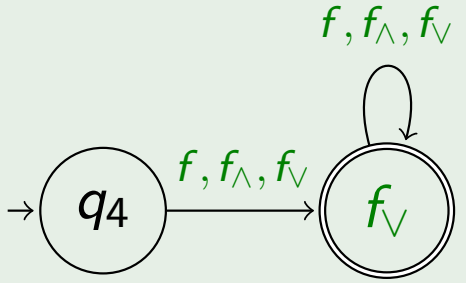
(f, false):



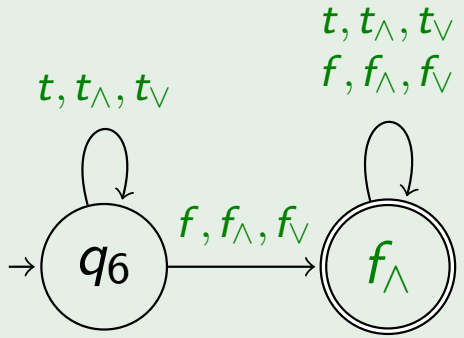
$(t, \wedge)$:



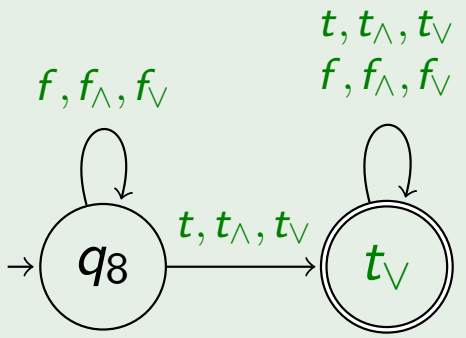
$(f, \vee)$:



$(f, \wedge)$:



$(t, \vee)$:

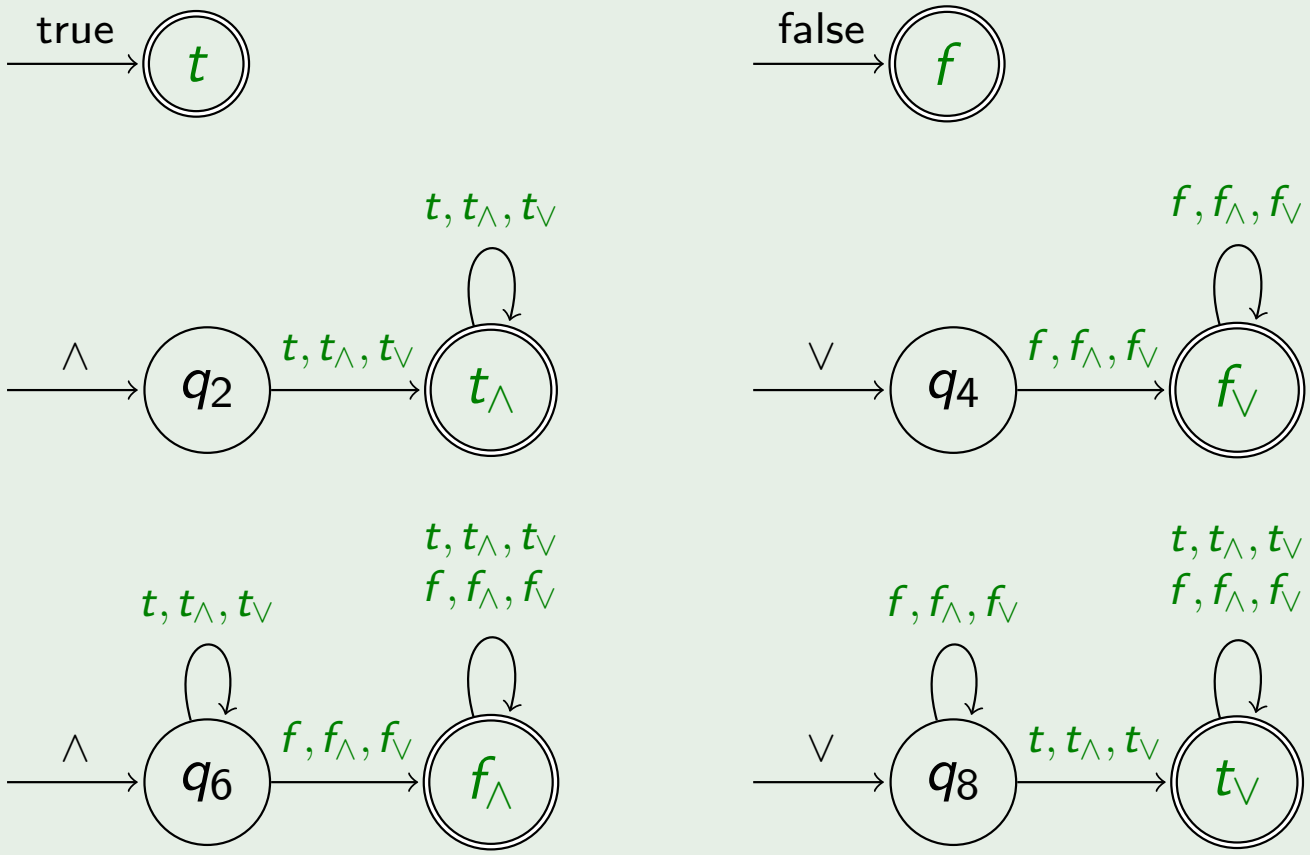


Intermezzo: Example

Example (Boolean Circuit Evaluation)

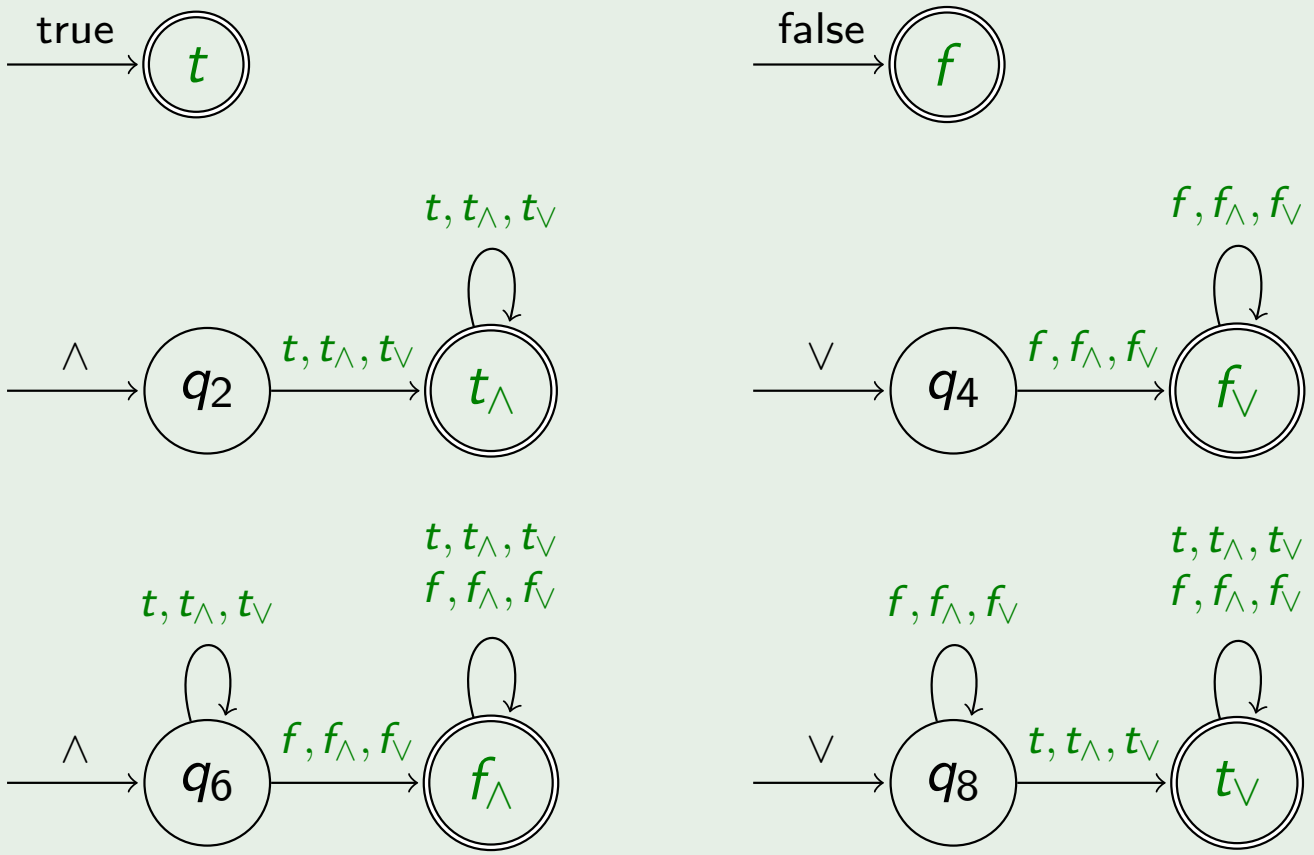
$$\varepsilon \xrightarrow{\text{true}} t \qquad t^+ \xrightarrow{\wedge} t \qquad (t+f)^* f (t+f)^* \xrightarrow{\wedge} f$$

$$\varepsilon \xrightarrow{\text{false}} f \qquad f^+ \xrightarrow{\vee} f \qquad (t+f)^* t (t+f)^* \xrightarrow{\vee} t$$



Intermezzo: Example

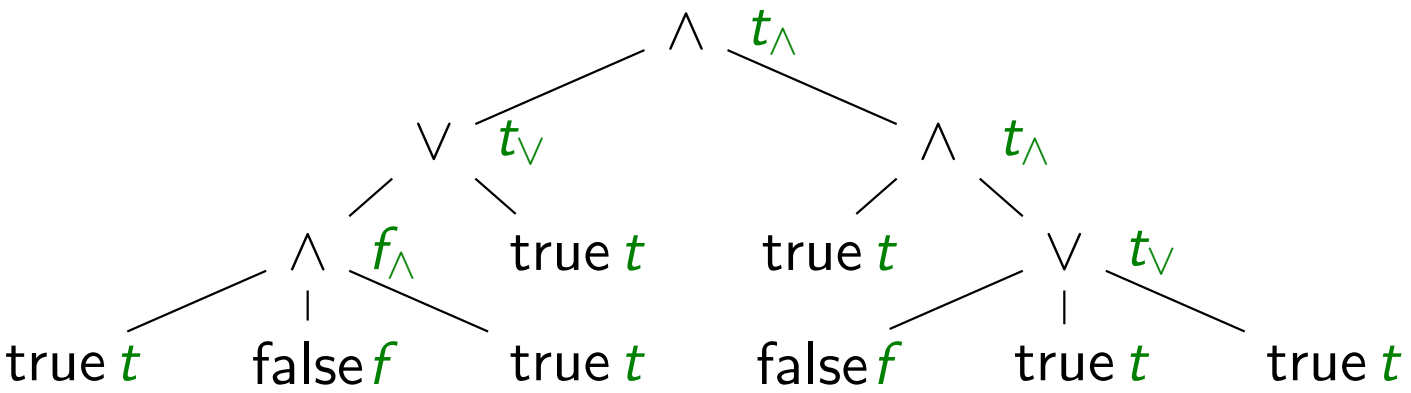
Example (Boolean Circuit Evaluation)

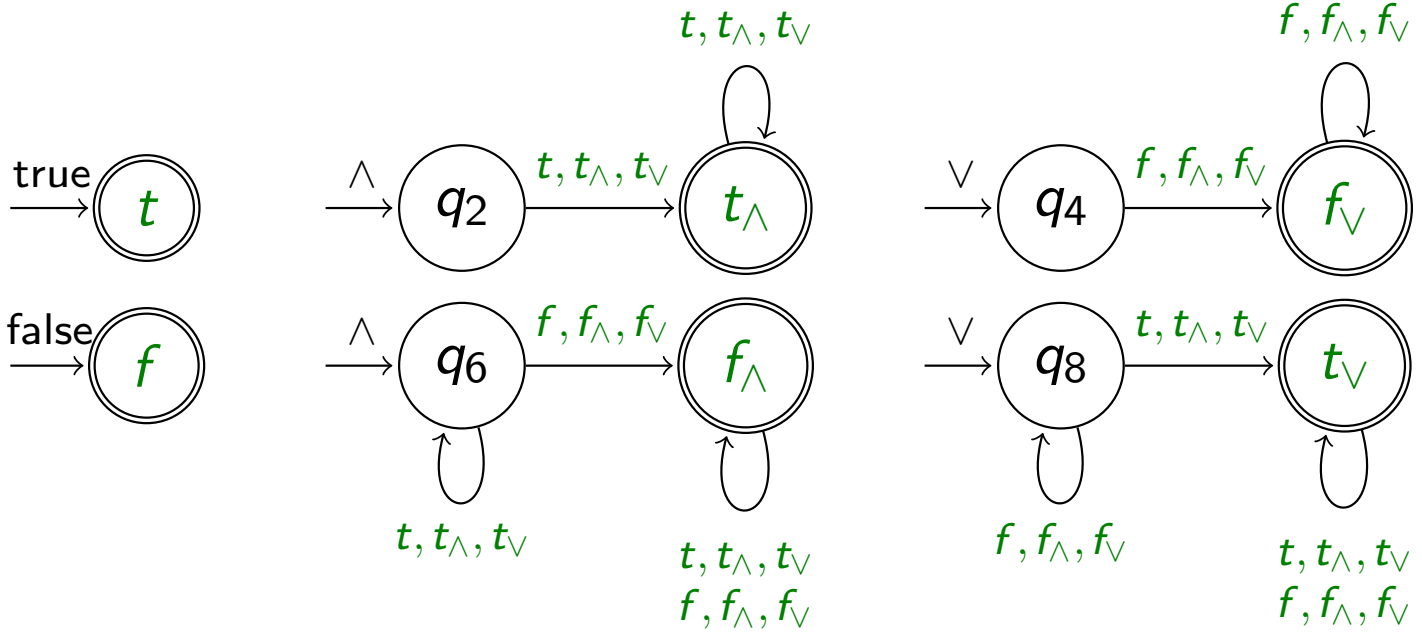


Definition (Stepwise Determinism (Intuitive))

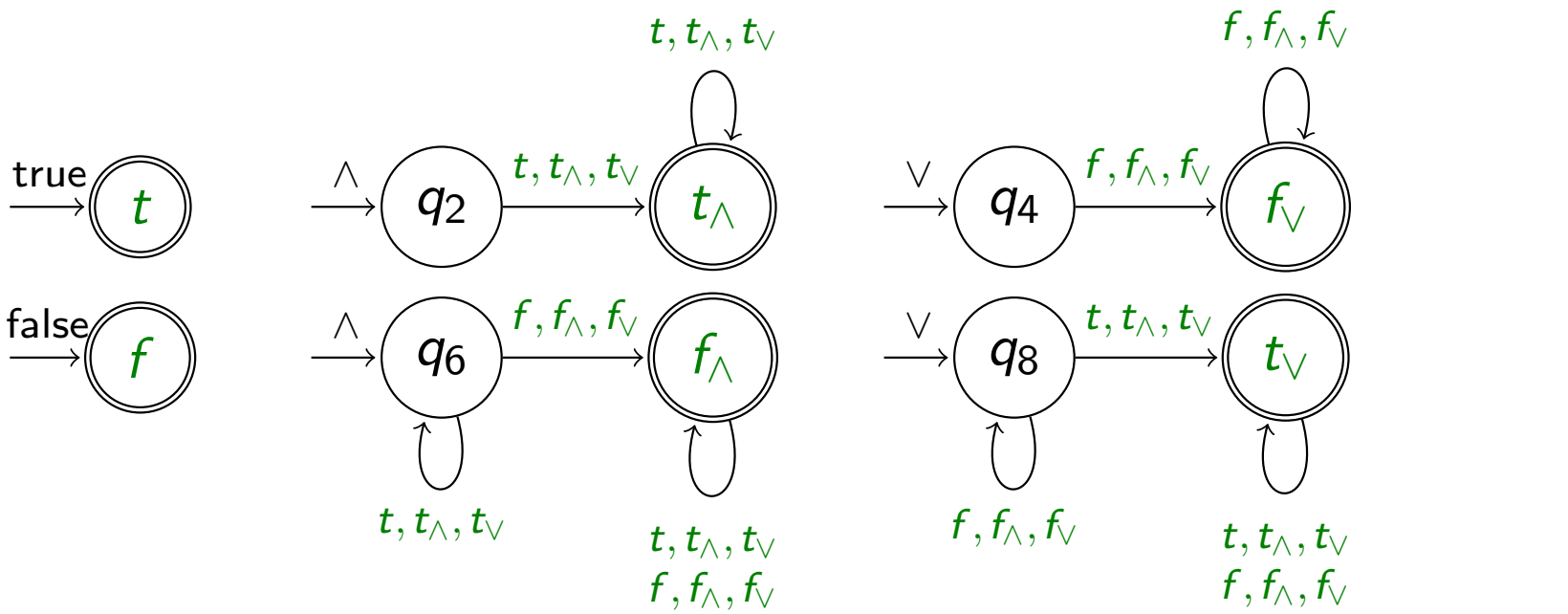
An unranked tree automaton is (stepwise) deterministic if all the above transitions are deterministic

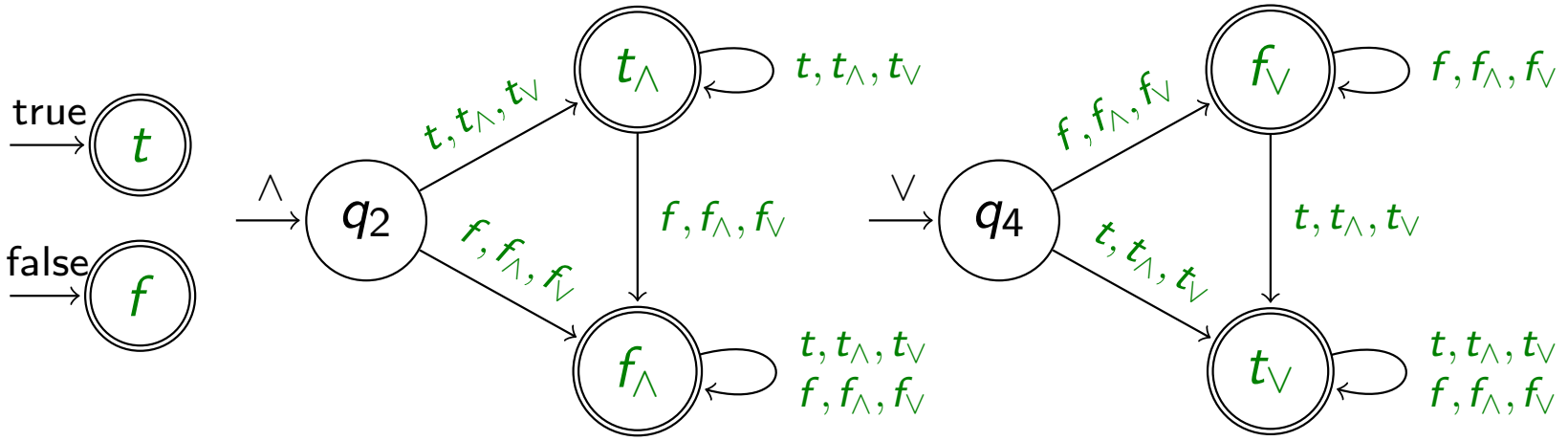
Let's run the new guy





Let's make him Deterministic

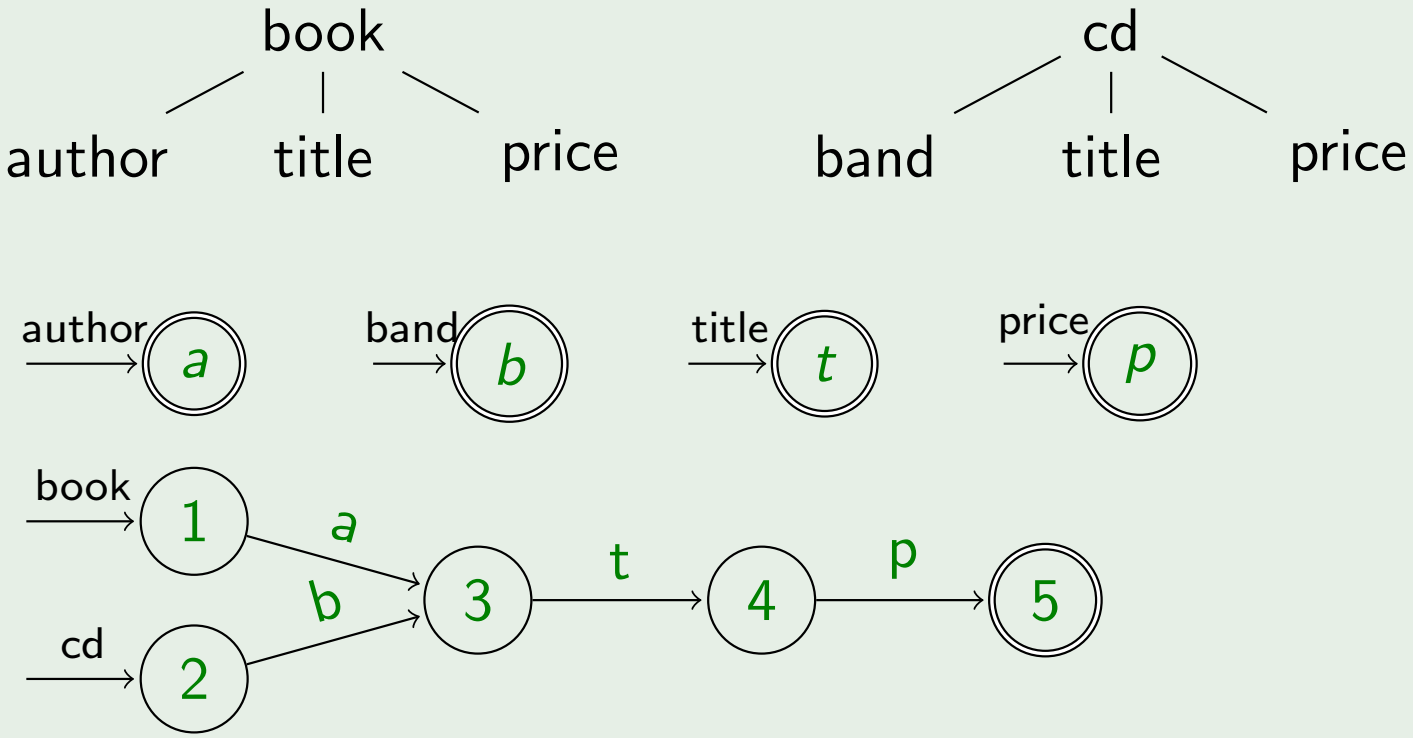




Another Example (State Sharing)

Recall our rules: $a \longrightarrow D$, in which $\text{States}(D) = \text{States}(A)$
This has advantages!

Example (State Sharing with A)

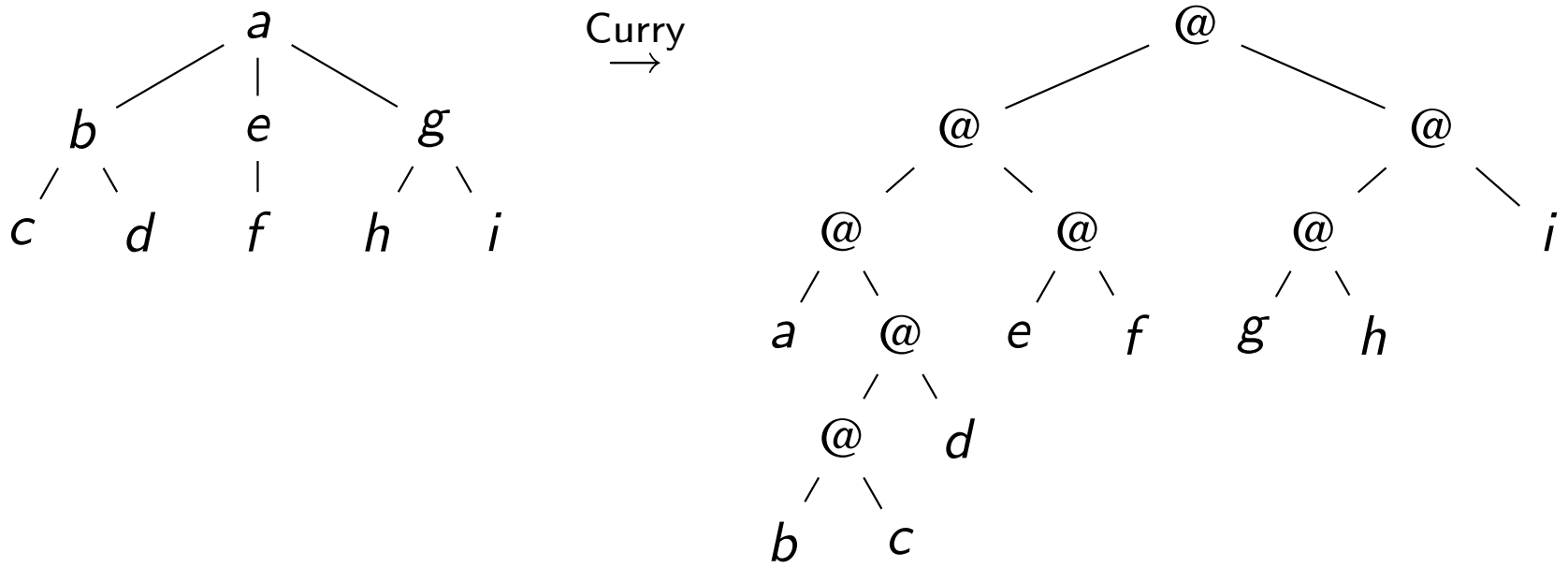


$\text{Final}(A) = \{5\}$

Currying

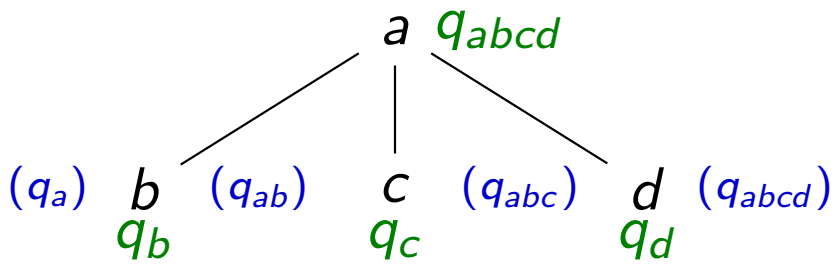
Currying

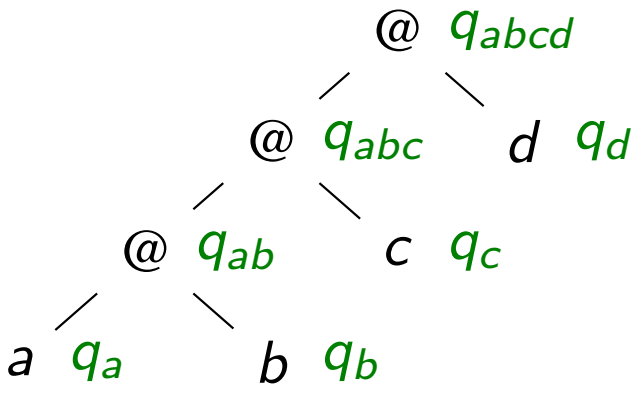
$\mathrm{Curry}\big(f(a,b,c)\big) = ((f\,@\,a)\,@\,b)\,@\,c)$
(@: pronounced "applied to")

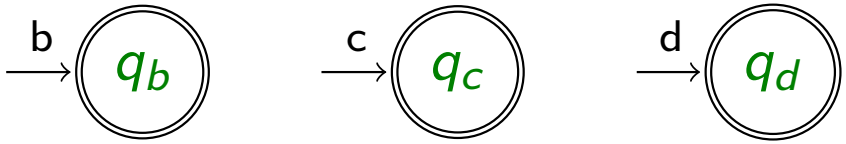


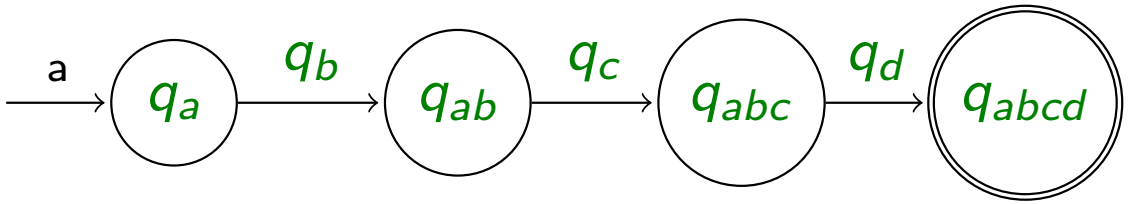
Now, a bottom-up deterministic automaton on the right tree...
does precisely what we want on the left one!

Currying Relation between Automata



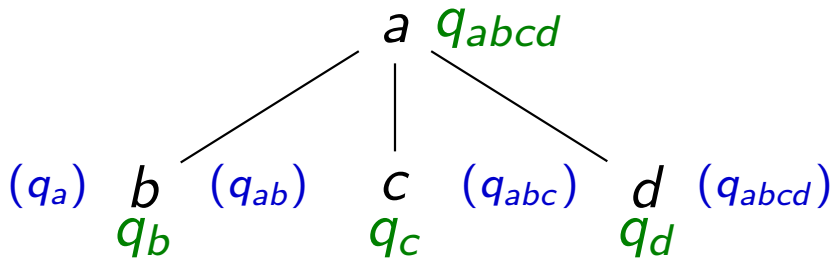


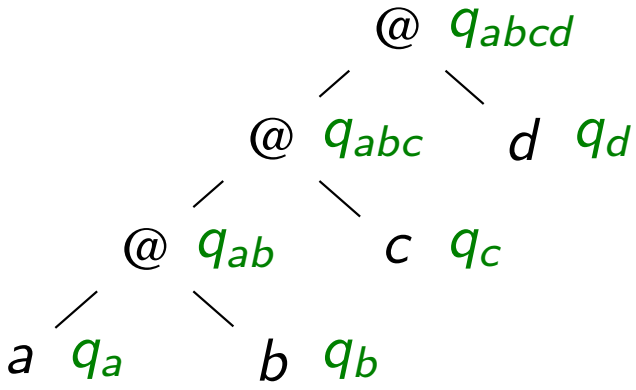




$\text{Final}(A) = \{q_{abcd}\}$

Currying Relation between Automata





Currying: transition relation between automata

- $\xrightarrow{a} q$ in unranked automaton
  $= \varepsilon \xrightarrow{a} q$ in binary automaton
- $q_1 \xrightarrow{p} q_2$ in unranked automaton
  $= (q_1, p) \xrightarrow{@} q_2$ in binary automaton

One-to-one correspondence in transition rules,
which preserves determinism!

Currying Tree Automata Encoding Lemma

So we actually proved:

Lemma (Currying Tree Automata Encoding Lemma)

(1) *For each UTA A, there is a*

$$\textit{BTA } \textit{Curry}(A) \textit{ accepting } \textit{Curry}(\textit{Language}(A))$$

(2) *For each BTA A there is a*

$$\textit{UTA } \textit{Curry}^{-1}(A) \textit{ accepting } \textit{Curry}^{-1}(\textit{Language}(A))$$

Moreover,

- *$\textit{Curry}(A)$ and $\textit{Curry}^{-1}(A)$ are constructible in linear time, and*
- *Curry and $\textit{Curry}^{-1}$ preserve (stepwise) determinism!*

The Result Transfer

- . . . all the results from the previous transfer also follow here
- Minimization is in **PTIME** for stepwise deterministic automata!

(Just use the minimization algorithm for the ranked automaton and merge states / transitions in the unranked one iff they should be merged in the ranked one)

Remark

Minimization is **NP**-hard for the other form of bottom-up determinism, as it's not entirely deterministic

The Result Transfer

- . . . all the results from the previous transfer also follow here
- Minimization is in **PTIME** for stepwise deterministic automata!
- Universality: in **PTIME** for stepwise deterministic automata
- Equivalence: in **PTIME** for stepwise deterministic automata
- Containment / Inclusion: in **PTIME** for stepwise deterministic automata

Remark

Minimization is **NP**-hard for the other form of bottom-up determinism, as it's not entirely deterministic

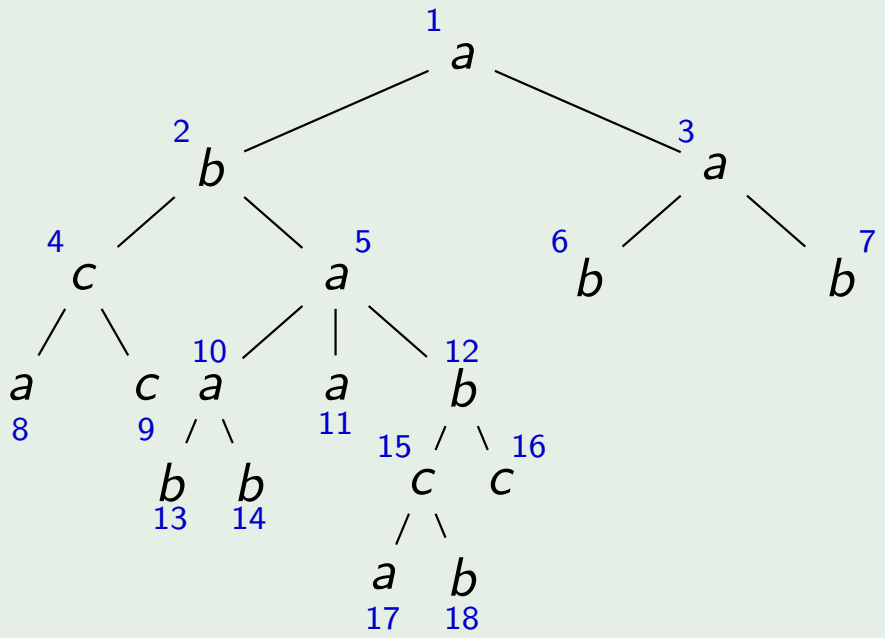
Outline

Unranked Trees as Structures

An unranked tree t over an alphabet $\Sigma = \{\sigma, \ldots, \sigma'\}$ also naturally corresponds to a structure $\underline{t}$ over $V_\Sigma = (E, <, L_\sigma, \ldots, L_{\sigma'})$

Example

Tree



Structure

- $D^{\underline{t}} = \{1, 2, 3, \ldots\}$
- $E^{\underline{t}}(1,2), E^{\underline{t}}(1,3), E^{\underline{t}}(2,4), \ldots$
- $2 <^{\underline{t}} 3, 4 <^{\underline{t}} 5, \ldots$
- $L_a = \{1, 3, 5, \ldots\}$
- $L_b = \{2, 6, 7, \ldots,\}$
- $L_c = \{4, 9, 15, 16\}$

MSO on Unranked Trees

Syntax:

$$\begin{array}{rcl} \phi & ::= & x = y \mid E(x,y) \mid x < y \mid L_a(x) \mid \cdots \mid L_b(y) \\ & \mid & \phi \wedge \phi \mid \neg\phi \mid \exists x\, \phi \mid X \mid X(x) \mid \exists X\, \phi \end{array}$$

with the usual abbreviations $\phi \vee \phi$, $\phi \rightarrow \phi$, $\forall x\, \phi$, $\forall X\, \phi$, ...

Familiar looking example

Every a-labeled node in t has a b-labeled descendant if, and only if

$$\underline{t} \models \forall n\, L_a(n) \rightarrow \exists X\, (\phi \wedge \psi)$$

where

$$\phi := X(n) \wedge \forall m \forall m' \Big(E(m,m') \wedge X(m) \rightarrow X(m') \Big)$$

$$\psi := \exists m\, (X(m) \wedge L_b(m))$$

MSO on Unranked Trees

Syntax:

$$\begin{array}{rcl} \phi & ::= & x = y \mid E(x,y) \mid x < y \mid L_a(x) \mid \cdots \mid L_b(y) \\ & \mid & \phi \wedge \phi \mid \neg\phi \mid \exists x\, \phi \mid X \mid X(x) \mid \exists X\, \phi \end{array}$$

with the usual abbreviations $\phi \vee \phi$, $\phi \rightarrow \phi$, $\forall x\, \phi$, $\forall X\, \phi$, ...

Notation and terminology

- If ϕ is an MSO sentence over V_Σ with Σ an unranked alphabet then

  $$\text{Language}(\phi) := \{t \mid t \text{ an unranked tree over } \Sigma \text{ such that } \underline{t} \models \phi\}$$

- An unranked tree language S is MSO-definable if there is some MSO formula ϕ with $S = \text{Language}(\phi)$

MSO on Unranked Trees

Theorem

An unranked tree language is MSO-definable if, and only if, it is regular.

The proof is exactly the same as in the ranked case.