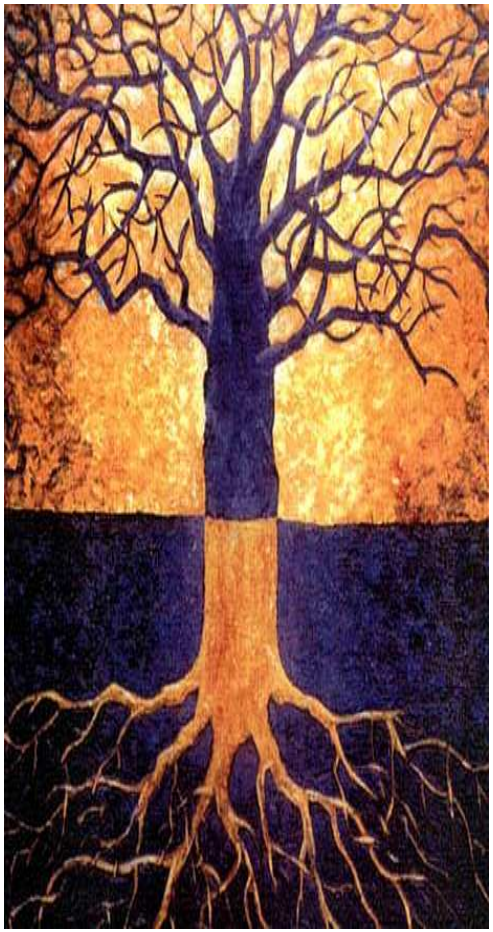


Tree Automata with Global Constraints

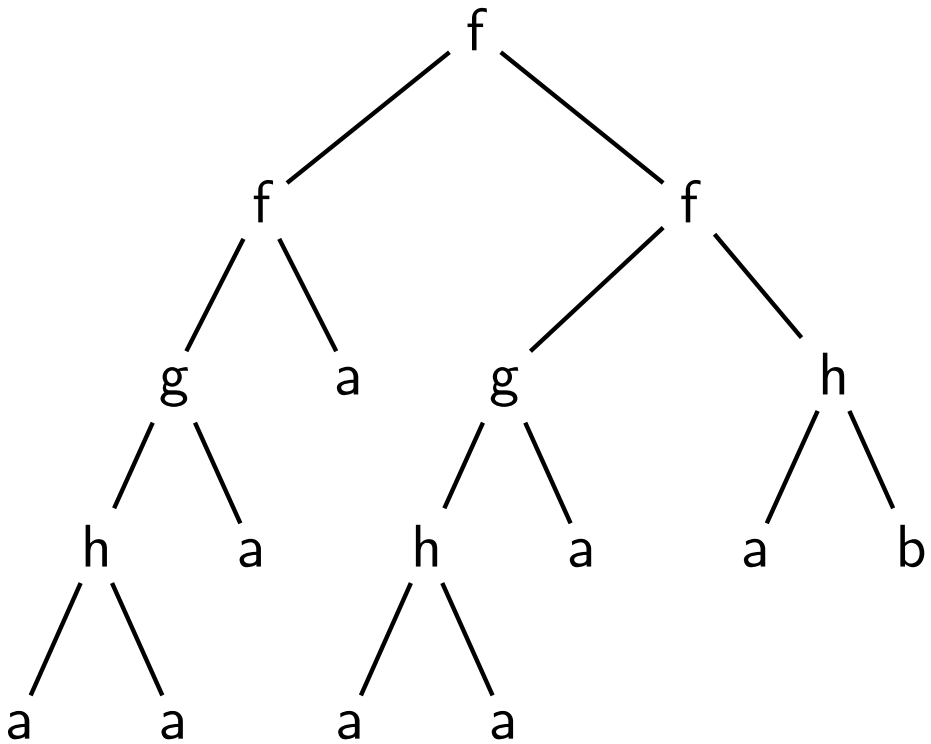
Emmanuel Filiot INRIA Futurs, Lille, Mostrare Project
Jean-Marc Talbot University of Provence, LIF, Marseille
Sophie Tison University of Lille1, LIFL, Mostrare Project

DLT'08

Motivations

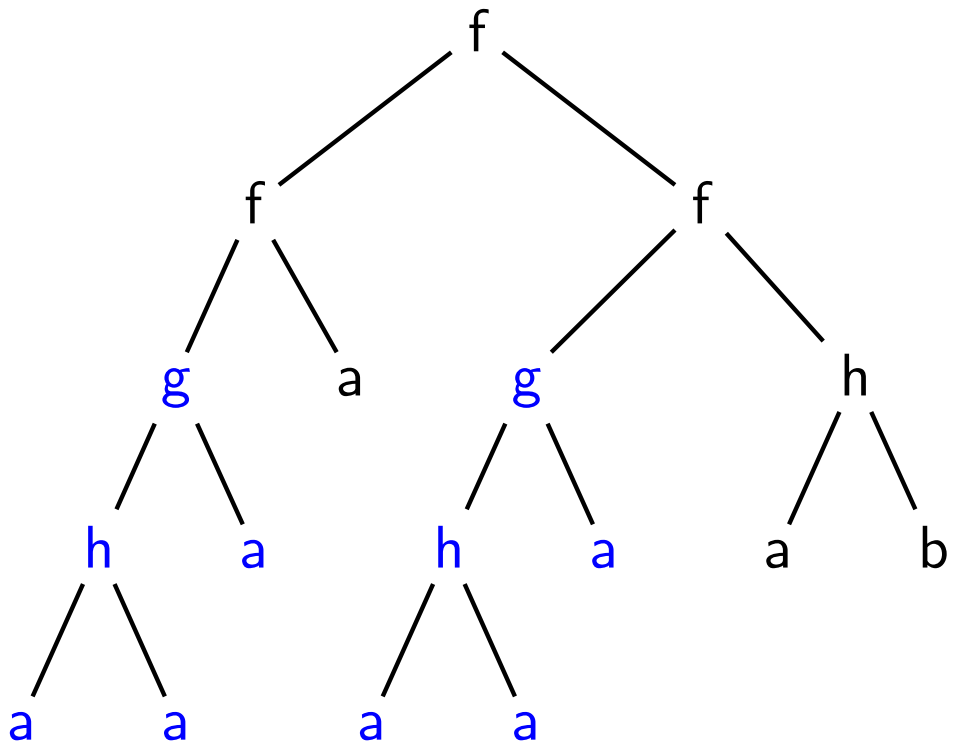
- query languages for XML
- TQL spatial logic (Cardelli, Ghelli, ICALP'02) (F., Talbot, Tison, CSL'07)
- express tree patterns with tree (dis)equality tests
- equivalent automaton model
- equality tests are **not** local
- a new model is needed: TAGED
- study TAGED more deeply
- give other applications (other logics, unification)

Examples of TQL-definable languages



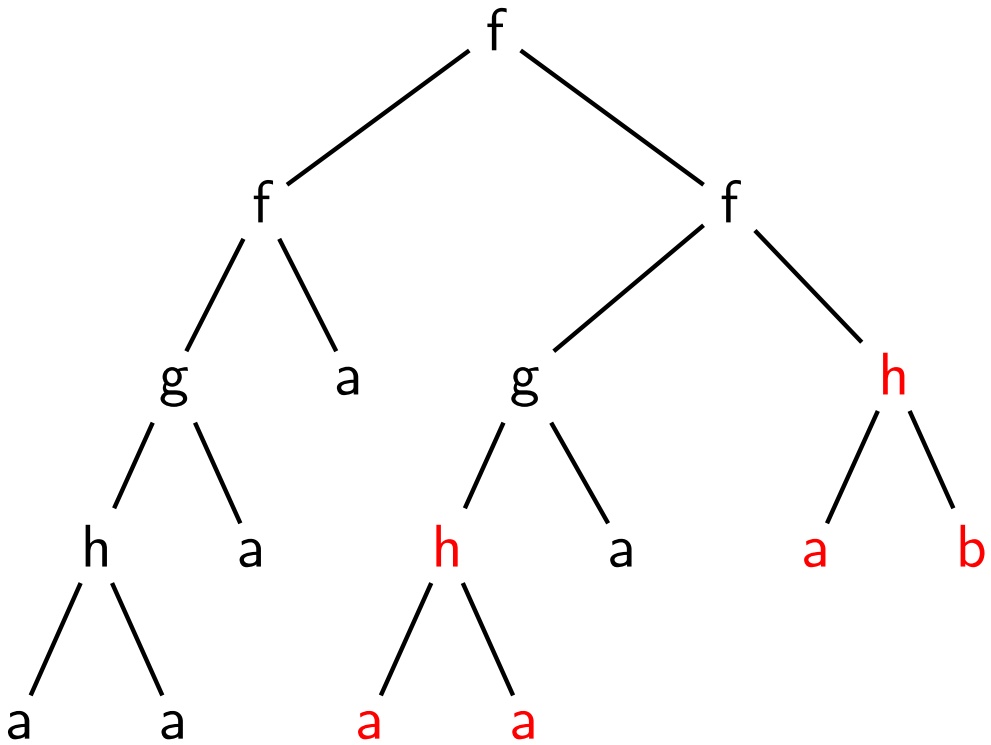
1. all regular languages

Examples of TQL-definable languages



1. all regular languages
2. all subtrees rooted at g are equal

Examples of TQL-definable languages



1. all regular languages
2. all subtrees rooted at g are equal
3. there are at least two different subtrees rooted at h

Related Work: Automata with Constraints

- rules have the following form:

$$f(q_1, q_2) \rightarrow_{1.2=2,\ 1.2\neq 1.3} q$$

- emptiness is undecidable, but several decidable classes exist
  - between children (Bogaert, Tison, STACS'92)
  - deterministic + boundedness condition (Dauchet, Caron, Coquidé, JCS'95)
  - extension to unranked trees (Karianto, Löding, ICALP'07)
- tests are **local**, one needs **global** tests.

Outline

1. Definition, Examples and Properties of TAGEDs
2. Emptiness Problem
3. Applications: MSO and Unification

Bottom-up Tree Automata

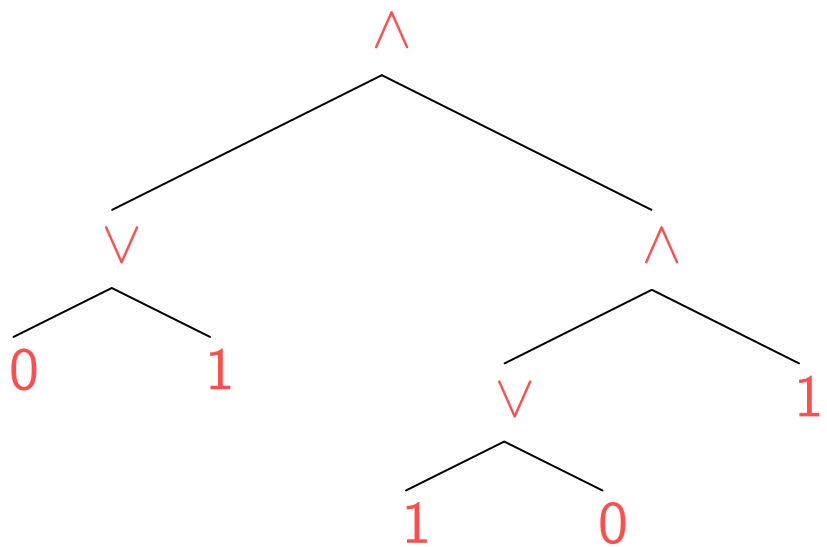
- trees are finite and binary over a finite alphabet $\Sigma = f, g, h, a, b \ldots$
- Q: set of states
- $F \subseteq Q$: set of final states
- Δ: rules of the form $f(q_1, q_2) \longrightarrow q$ or $a \longrightarrow q$

Bottom-up Tree Automata

- trees are finite and binary over a finite alphabet $\Sigma = f, g, h, a, b \ldots$
- Q: set of states
- $F \subseteq Q$: set of final states
- Δ: rules of the form $f(q_1, q_2) \longrightarrow q$ or $a \longrightarrow q$

Example (variable-free satisfiable Boolean formulas)

a tree and a successful run



transitions

$$0 \longrightarrow q_0 \quad 1 \longrightarrow q_1$$
$$\wedge(q_{b_1}, q_{b_2}) \longrightarrow q_{b_1 \wedge b_2}$$
$$\vee(q_{b_1}, q_{b_2}) \longrightarrow q_{b_1 \vee b_2}$$

final states

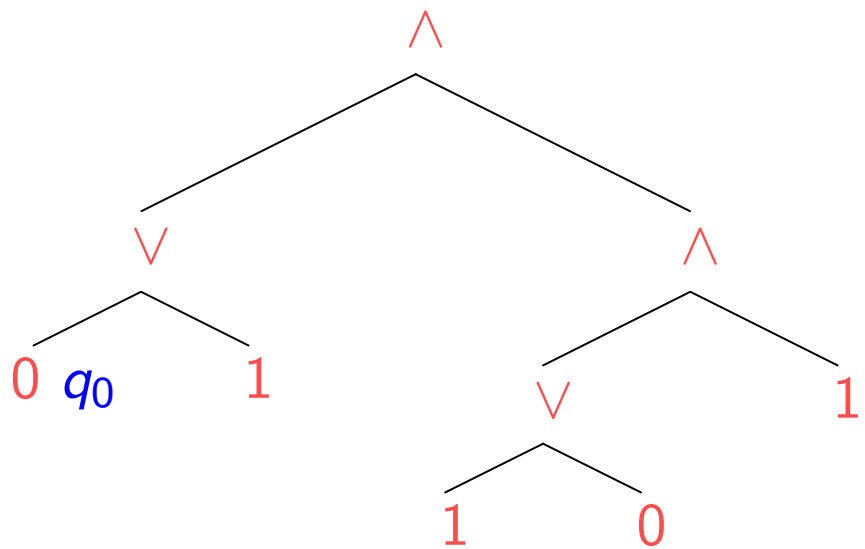
$$F = \{q_1\}$$

Bottom-up Tree Automata

- trees are finite and binary over a finite alphabet $\Sigma = f, g, h, a, b \ldots$
- Q: set of states
- $F \subseteq Q$: set of final states
- Δ: rules of the form $f(q_1, q_2) \longrightarrow q$ or $a \longrightarrow q$

Example (variable-free satisfiable Boolean formulas)

a tree and a successful run



transitions

$0 \longrightarrow q_0 \quad 1 \longrightarrow q_1$

$\wedge(q_{b_1}, q_{b_2}) \longrightarrow q_{b_1 \wedge b_2}$

$\vee(q_{b_1}, q_{b_2}) \longrightarrow q_{b_1 \vee b_2}$

final states

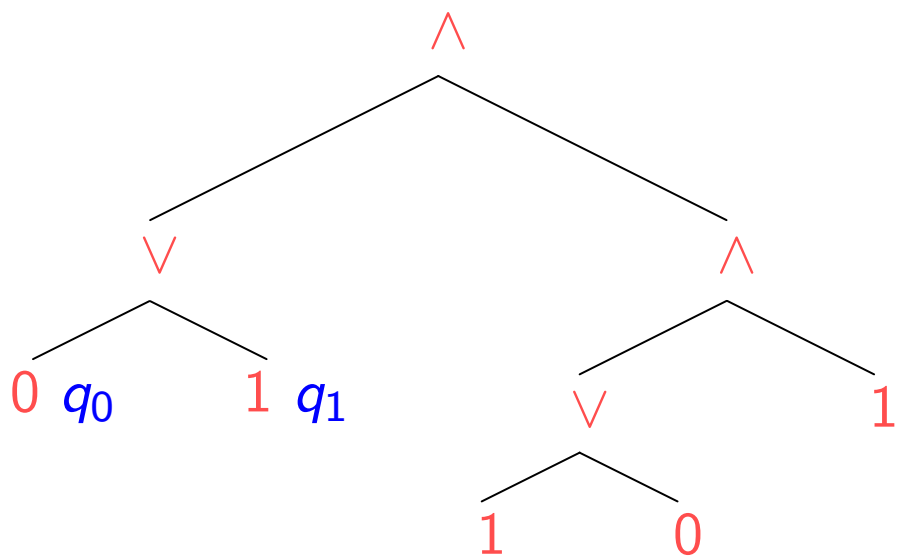
$F = \{q_1\}$

Bottom-up Tree Automata

- trees are finite and binary over a finite alphabet $\Sigma = f, g, h, a, b \ldots$
- Q: set of states
- $F \subseteq Q$: set of final states
- Δ: rules of the form $f(q_1, q_2) \longrightarrow q$ or $a \longrightarrow q$

Example (variable-free satisfiable Boolean formulas)

a tree and a successful run



transitions

$0 \longrightarrow q_0 \quad 1 \longrightarrow q_1$

$\wedge(q_{b_1}, q_{b_2}) \longrightarrow q_{b_1 \wedge b_2}$

$\vee(q_{b_1}, q_{b_2}) \longrightarrow q_{b_1 \vee b_2}$

final states

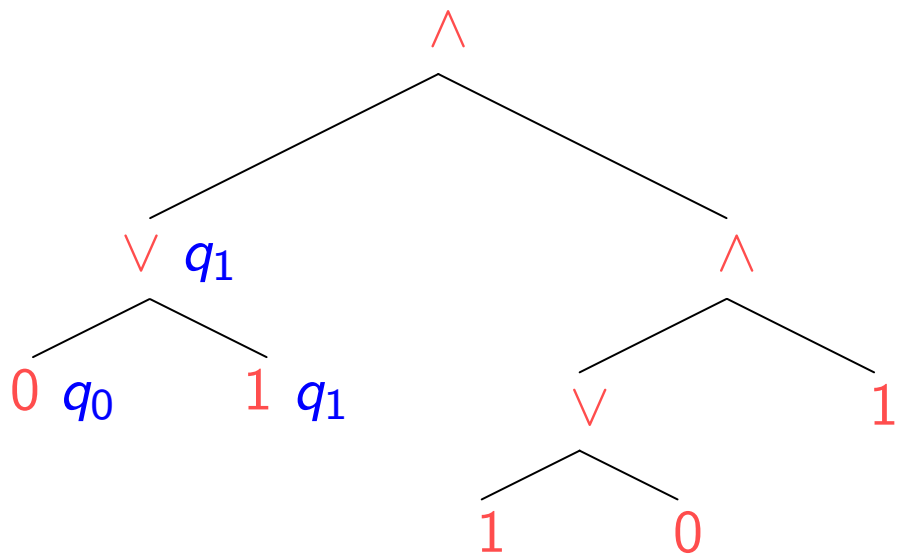
$F = \{q_1\}$

Bottom-up Tree Automata

- trees are finite and binary over a finite alphabet $\Sigma = f, g, h, a, b \ldots$
- Q: set of states
- $F \subseteq Q$: set of final states
- Δ: rules of the form $f(q_1, q_2) \longrightarrow q$ or $a \longrightarrow q$

Example (variable-free satisfiable Boolean formulas)

a tree and a successful run



transitions

$0 \longrightarrow q_0 \quad 1 \longrightarrow q_1$

$\wedge(q_{b_1}, q_{b_2}) \longrightarrow q_{b_1 \wedge b_2}$

$\vee(q_{b_1}, q_{b_2}) \longrightarrow q_{b_1 \vee b_2}$

final states

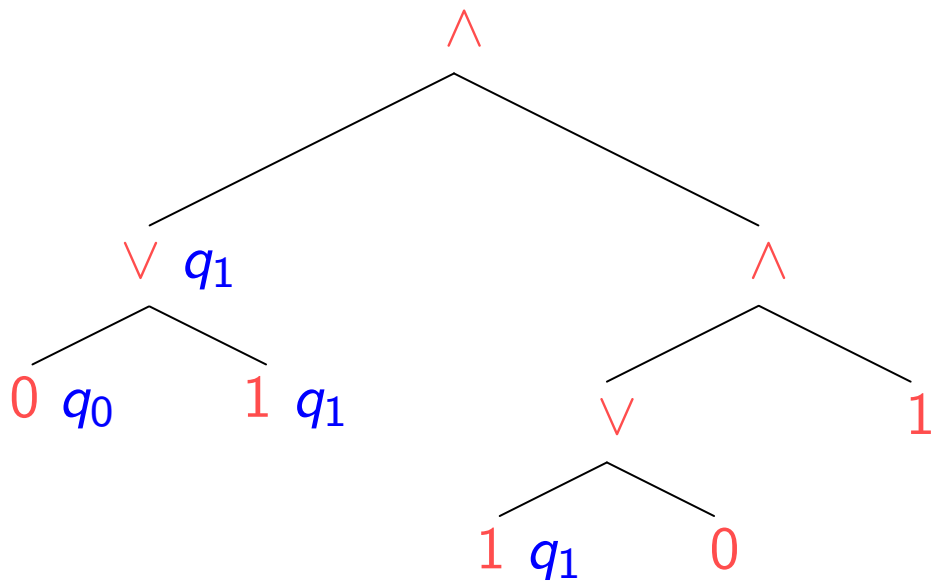
$F = \{q_1\}$

Bottom-up Tree Automata

- trees are finite and binary over a finite alphabet $\Sigma = f, g, h, a, b \ldots$
- Q: set of states
- $F \subseteq Q$: set of final states
- Δ: rules of the form $f(q_1, q_2) \longrightarrow q$ or $a \longrightarrow q$

Example (variable-free satisfiable Boolean formulas)

a tree and a successful run



transitions

$$0 \longrightarrow q_0 \quad 1 \longrightarrow q_1$$
$$\wedge(q_{b_1}, q_{b_2}) \longrightarrow q_{b_1 \wedge b_2}$$
$$\vee(q_{b_1}, q_{b_2}) \longrightarrow q_{b_1 \vee b_2}$$

final states

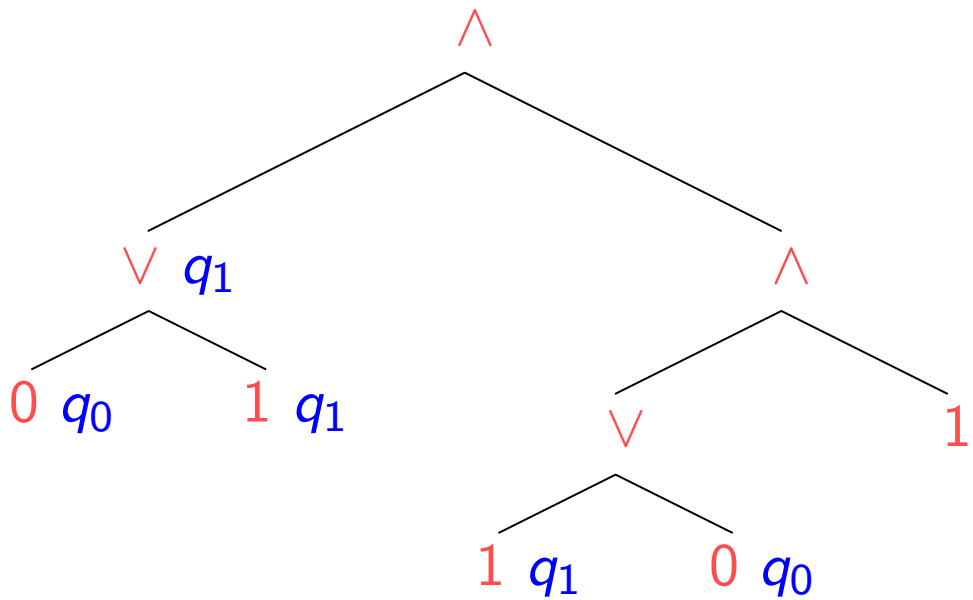
$$F = \{q_1\}$$

Bottom-up Tree Automata

- trees are finite and binary over a finite alphabet $\Sigma = f, g, h, a, b \ldots$
- Q: set of states
- $F \subseteq Q$: set of final states
- Δ: rules of the form $f(q_1, q_2) \longrightarrow q$ or $a \longrightarrow q$

Example (variable-free satisfiable Boolean formulas)

a tree and a successful run



transitions

$0 \longrightarrow q_0 \quad 1 \longrightarrow q_1$

$\wedge(q_{b_1}, q_{b_2}) \longrightarrow q_{b_1 \wedge b_2}$

$\vee(q_{b_1}, q_{b_2}) \longrightarrow q_{b_1 \vee b_2}$

final states

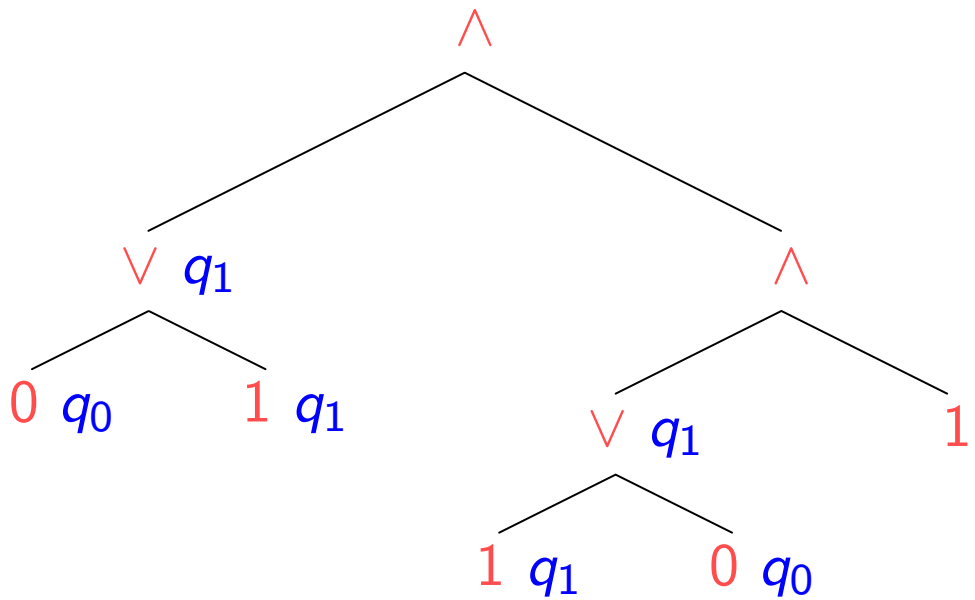
$F = \{q_1\}$

Bottom-up Tree Automata

- trees are finite and binary over a finite alphabet $\Sigma = f, g, h, a, b \ldots$
- Q: set of states
- $F \subseteq Q$: set of final states
- Δ: rules of the form $f(q_1, q_2) \longrightarrow q$ or $a \longrightarrow q$

Example (variable-free satisfiable Boolean formulas)

a tree and a successful run



transitions

$0 \longrightarrow q_0 \quad 1 \longrightarrow q_1$

$\wedge(q_{b_1}, q_{b_2}) \longrightarrow q_{b_1 \wedge b_2}$

$\vee(q_{b_1}, q_{b_2}) \longrightarrow q_{b_1 \vee b_2}$

final states

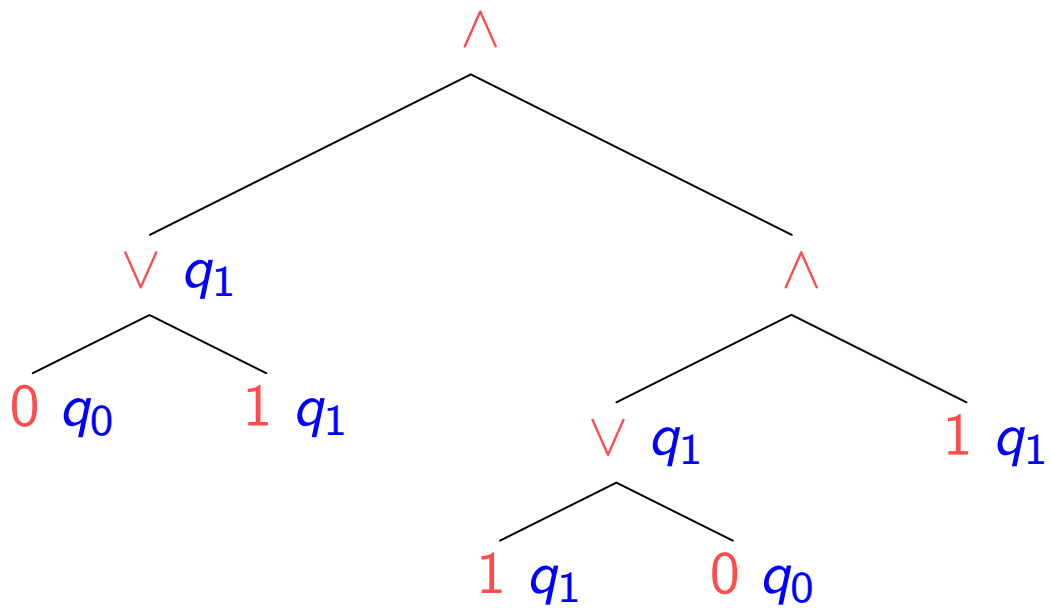
$F = \{q_1\}$

Bottom-up Tree Automata

- trees are finite and binary over a finite alphabet $\Sigma = f, g, h, a, b \ldots$
- Q: set of states
- $F \subseteq Q$: set of final states
- Δ: rules of the form $f(q_1, q_2) \longrightarrow q$ or $a \longrightarrow q$

Example (variable-free satisfiable Boolean formulas)

a tree and a successful run



transitions

$0 \longrightarrow q_0 \quad 1 \longrightarrow q_1$

$\wedge(q_{b_1}, q_{b_2}) \longrightarrow q_{b_1 \wedge b_2}$

$\vee(q_{b_1}, q_{b_2}) \longrightarrow q_{b_1 \vee b_2}$

final states

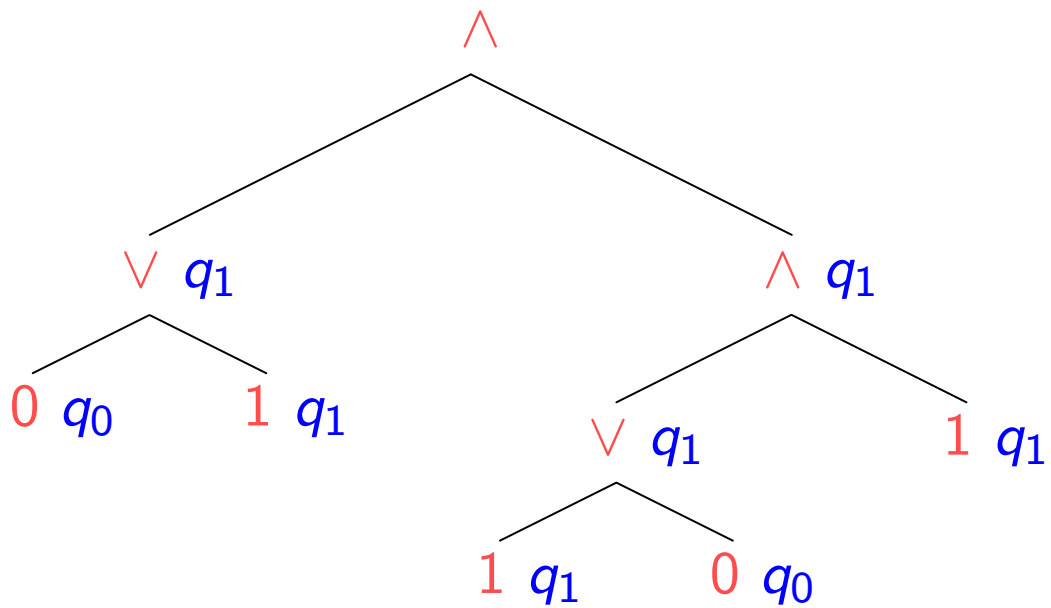
$F = \{q_1\}$

Bottom-up Tree Automata

- trees are finite and binary over a finite alphabet $\Sigma = f, g, h, a, b \ldots$
- Q: set of states
- $F \subseteq Q$: set of final states
- Δ: rules of the form $f(q_1, q_2) \longrightarrow q$ or $a \longrightarrow q$

Example (variable-free satisfiable Boolean formulas)

a tree and a successful run



transitions

$0 \longrightarrow q_0 \quad 1 \longrightarrow q_1$

$\wedge(q_{b_1}, q_{b_2}) \longrightarrow q_{b_1 \wedge b_2}$

$\vee(q_{b_1}, q_{b_2}) \longrightarrow q_{b_1 \vee b_2}$

final states

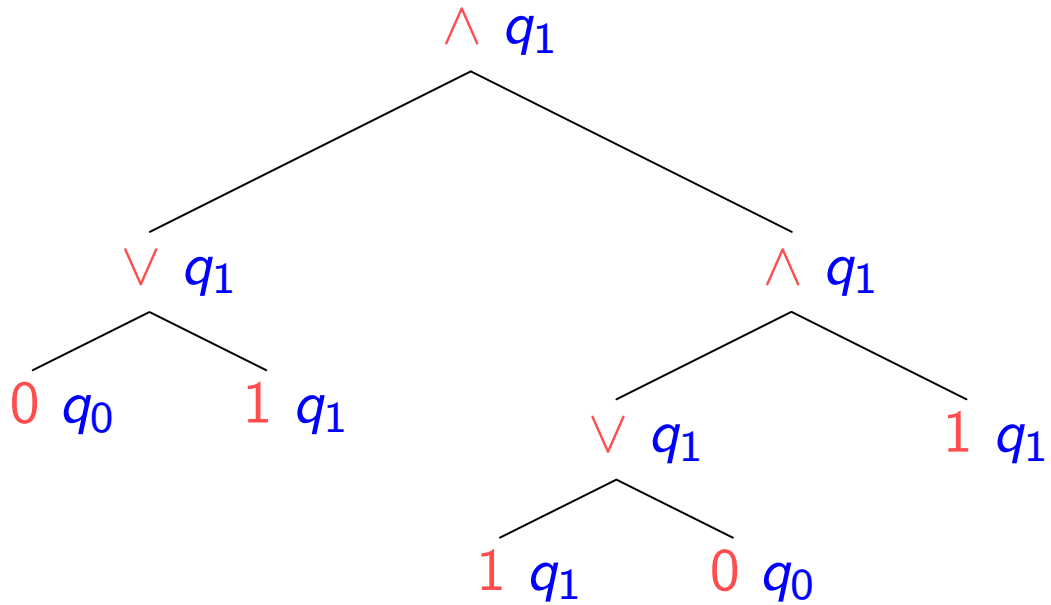
$F = \{q_1\}$

Bottom-up Tree Automata

- trees are finite and binary over a finite alphabet $\Sigma = f, g, h, a, b \ldots$
- Q: set of states
- $F \subseteq Q$: set of final states
- Δ: rules of the form $f(q_1, q_2) \longrightarrow q$ or $a \longrightarrow q$

Example (variable-free satisfiable Boolean formulas)

a tree and a successful run



transitions

$0 \longrightarrow q_0 \quad 1 \longrightarrow q_1$

$\wedge(q_{b_1}, q_{b_2}) \longrightarrow q_{b_1 \wedge b_2}$

$\vee(q_{b_1}, q_{b_2}) \longrightarrow q_{b_1 \vee b_2}$

final states

$F = \{q_1\}$

Tree Automata with Global Equalities and Disequalities

A **tree automata A with global equalities and disequalities** (TAGED) is given by:

Σ alphabet
Q set of states
F set of final states
Δ set of rules

} tree automaton

Tree Automata with Global Equalities and Disequalities

A **tree automata A with global equalities and disequalities** (TAGED) is given by:

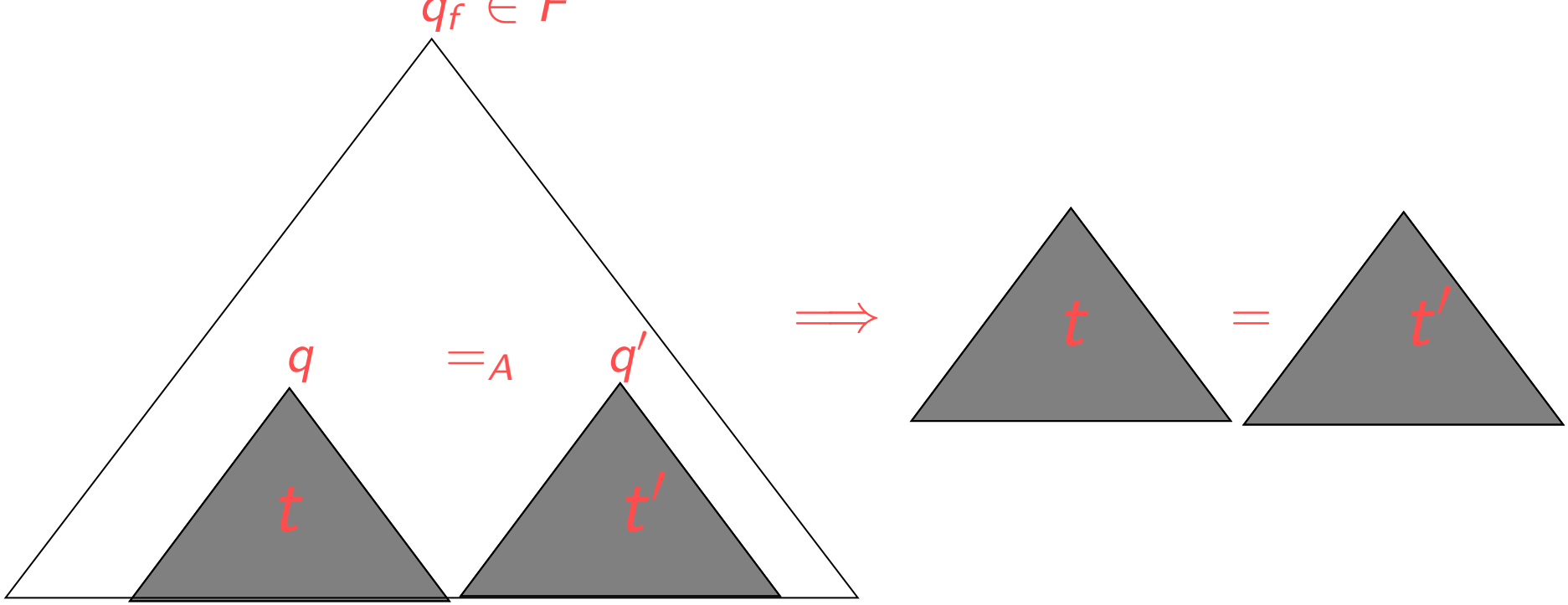
Σ alphabet
Q set of states
F set of final states
Δ set of rules

} tree automaton

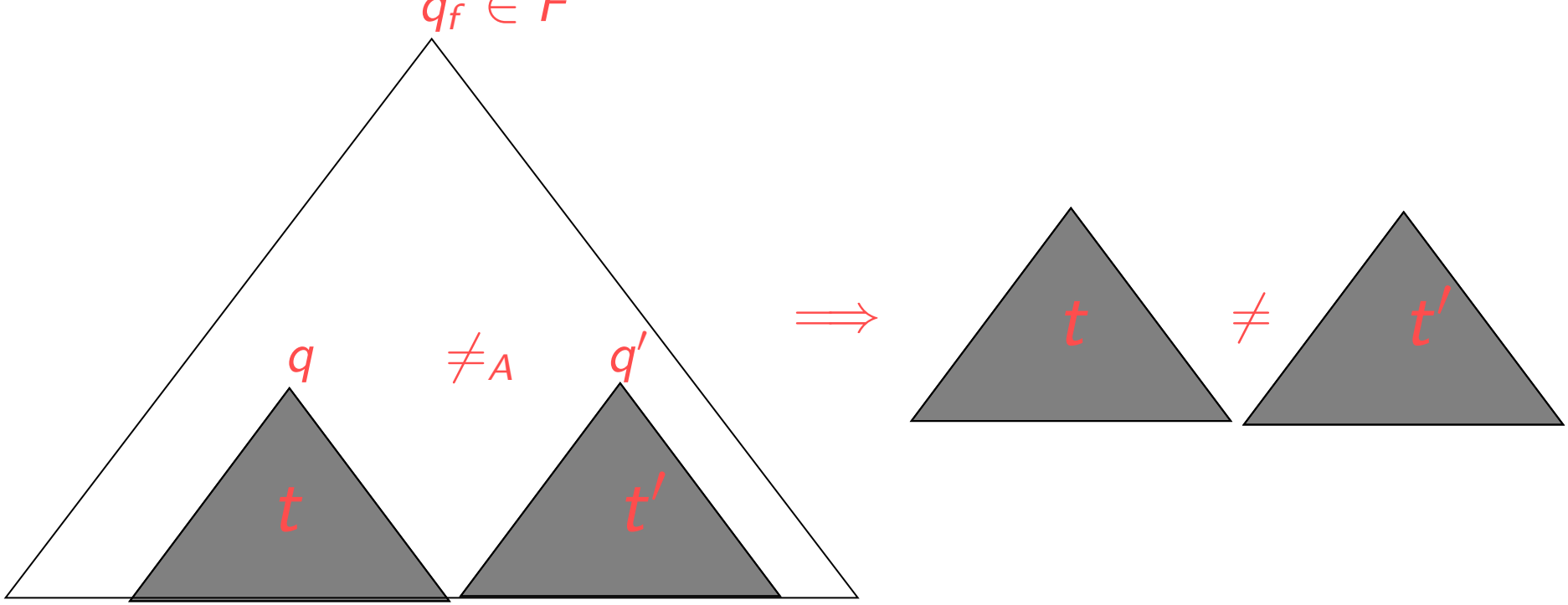
$=_A \subseteq Q^2$ reflexive and symmetric relation on a **subset** of Q

$\neq_A \subseteq Q^2$ irreflexive and symmetric relation

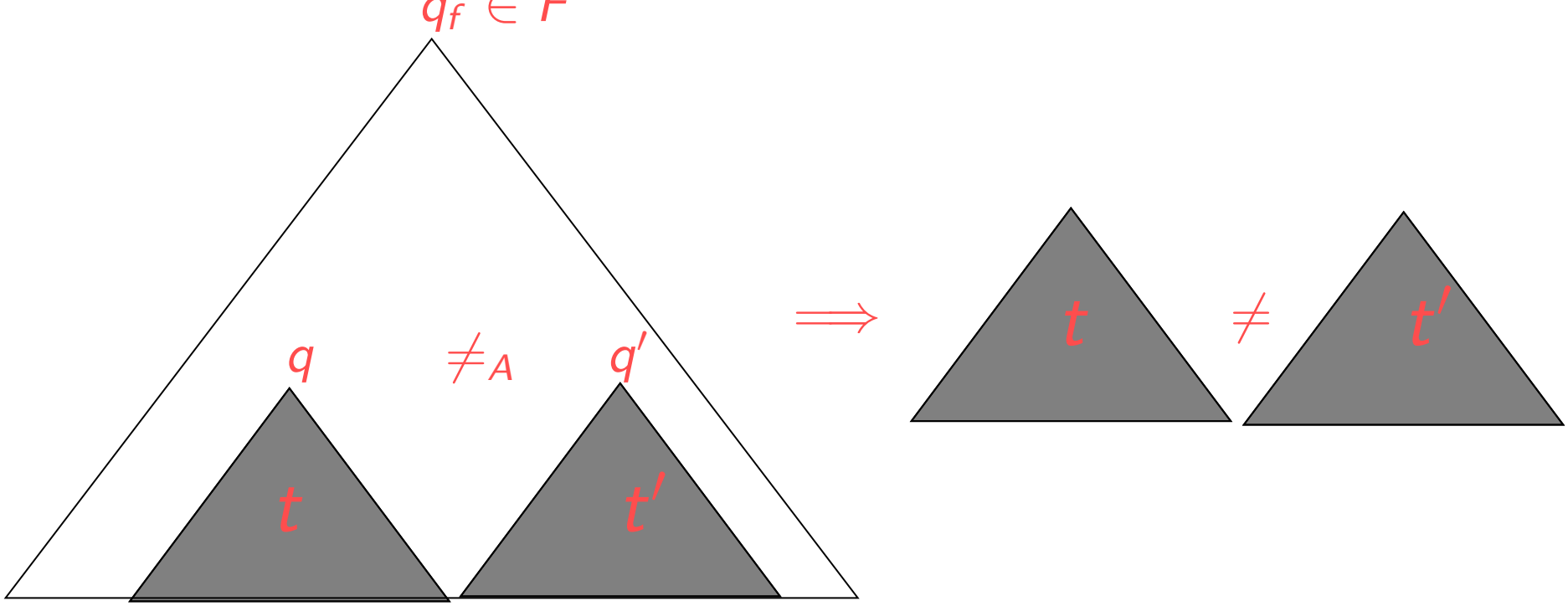
Successful Runs



Successful Runs



Successful Runs

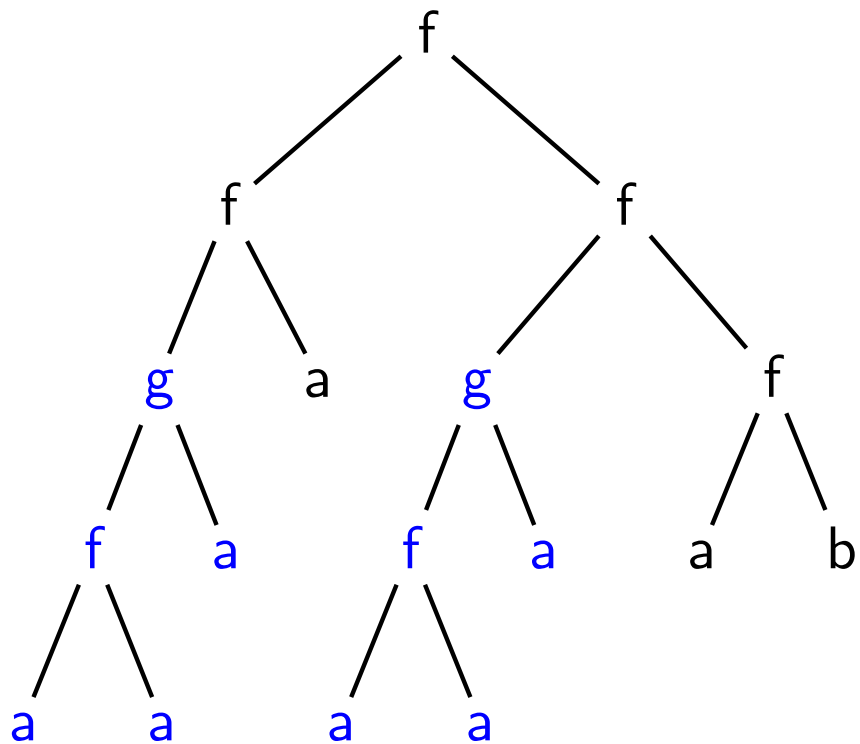


TAGEDs and automata with local constraints are orthogonal

Given a successful run, two nodes are equivalent if they have been "successfully tested" to be equal in the run

	number of eq. classes	cardinality of eq. classes
local constraints	unbounded	bounded
TAGEDs	bounded	unbounded

Example: all subtrees rooted at g are equal

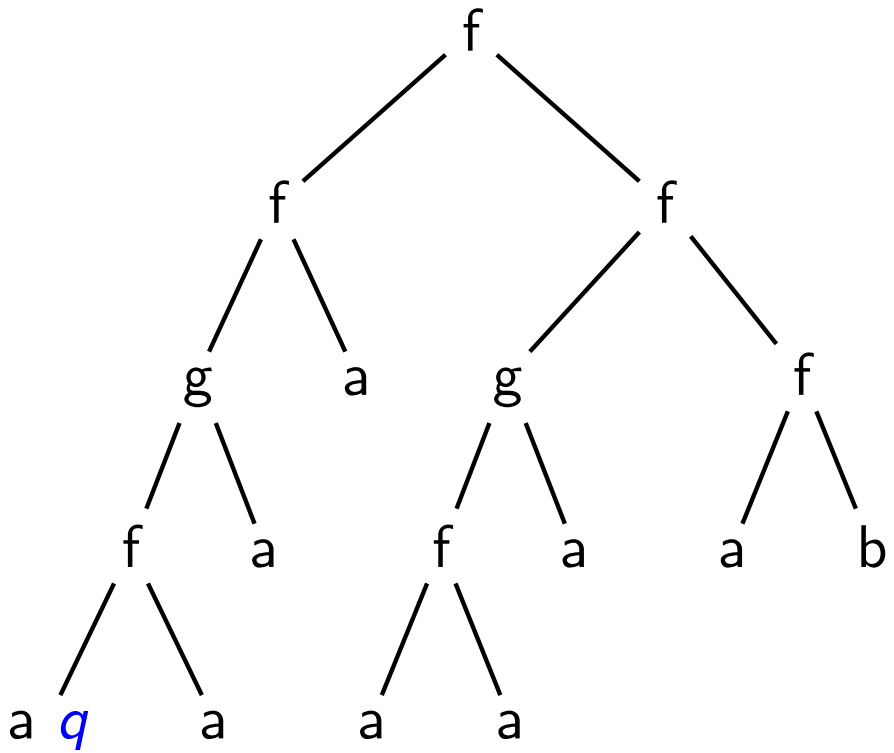


- $\Sigma = \{f, a, g\}$
- $Q = F = \{q, q_g, q_g^{\downarrow}\}$
- $\Delta =$

$$
\begin{aligned}
& a \longrightarrow q \\
& g(q, q) \longrightarrow q_g \\
& f(q, q) \longrightarrow q \\
& f(q_g, q) \longrightarrow q_g^{\downarrow} \\
& f(q, q_g) \longrightarrow q_g^{\downarrow} \\
& f(q_g, q_g) \longrightarrow q_g^{\downarrow} \\
& f(q_g^{\downarrow}, q_g) \longrightarrow q_g^{\downarrow} \\
& \ldots
\end{aligned}
$$

- $q_g \ =_A \ q_g$

Example: all subtrees rooted at g are equal

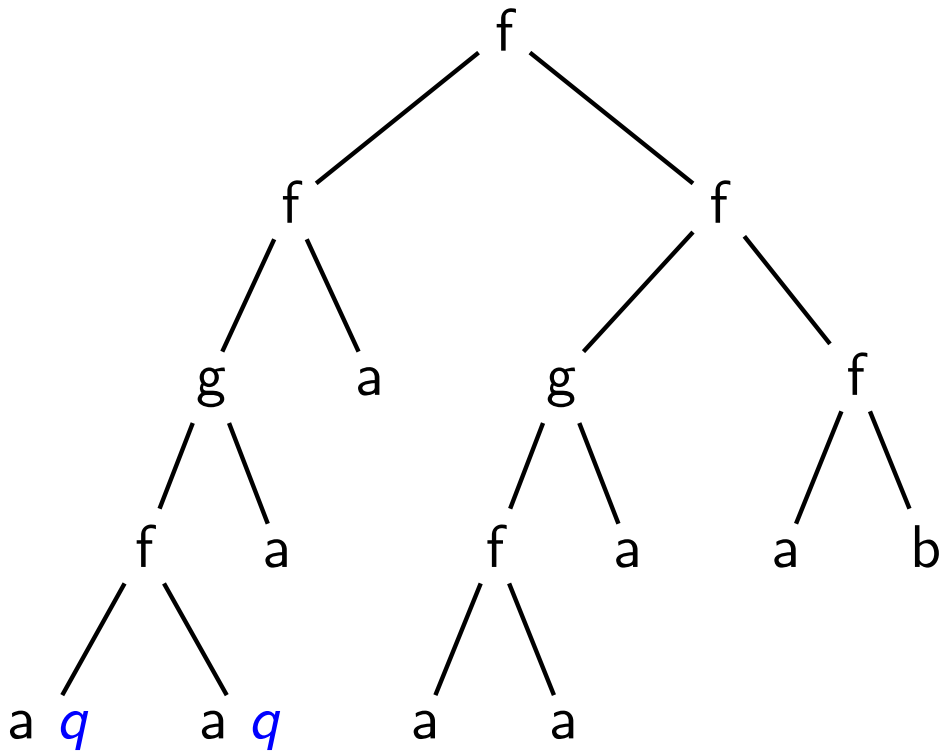


- $\Sigma = \{f, a, g\}$
- $Q = F = \{q, q_g, q_g^{\downarrow}\}$
- $\Delta =$

$$
\begin{aligned}
a &\to q \\
g(q,q) &\to q_g \\
f(q,q) &\to q \\
f(q_g,q) &\to q_g^{\downarrow} \\
f(q,q_g) &\to q_g^{\downarrow} \\
f(q_g,q_g) &\to q_g^{\downarrow} \\
f(q_g^{\downarrow},q_g) &\to q_g^{\downarrow} \\
&\dots
\end{aligned}
$$

- $q_g =_A q_g$

Example: all subtrees rooted at g are equal

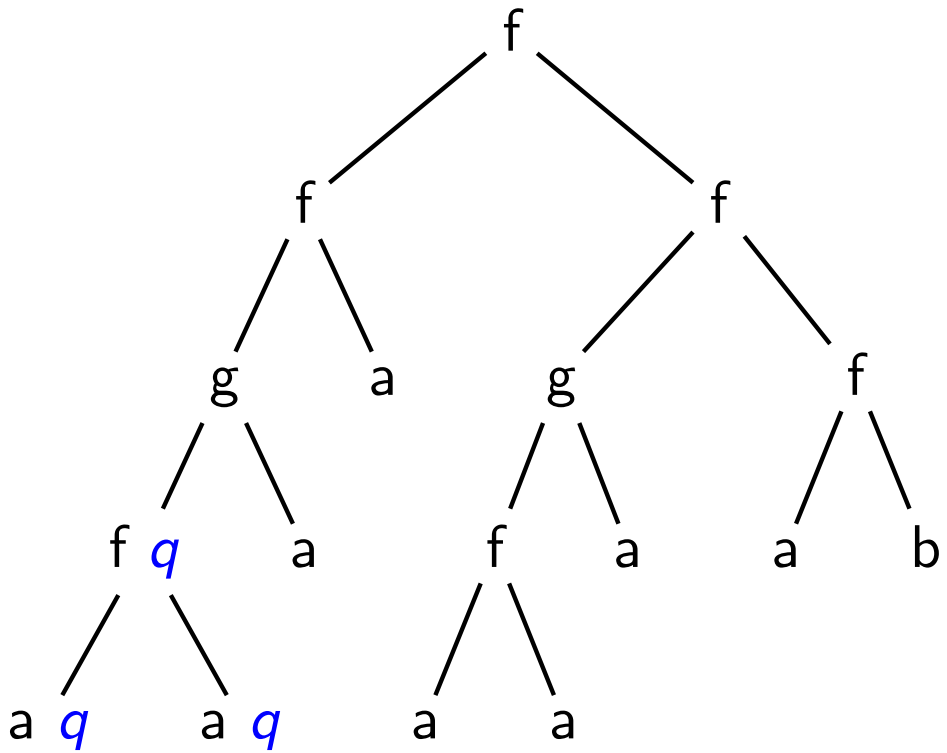


- $\Sigma = \{f, a, g\}$
- $Q = F = \{q, q_g, q_g^{\downarrow}\}$
- $\Delta =$

$$
\begin{aligned}
& a \longrightarrow q \\
& g(q, q) \longrightarrow q_g \\
& f(q, q) \longrightarrow q \\
& f(q_g, q) \longrightarrow q_g^{\downarrow} \\
& f(q, q_g) \longrightarrow q_g^{\downarrow} \\
& f(q_g, q_g) \longrightarrow q_g^{\downarrow} \\
& f(q_g^{\downarrow}, q_g) \longrightarrow q_g^{\downarrow} \\
& \ldots
\end{aligned}
$$

- $q_g \ =_A \ q_g$

Example: all subtrees rooted at g are equal

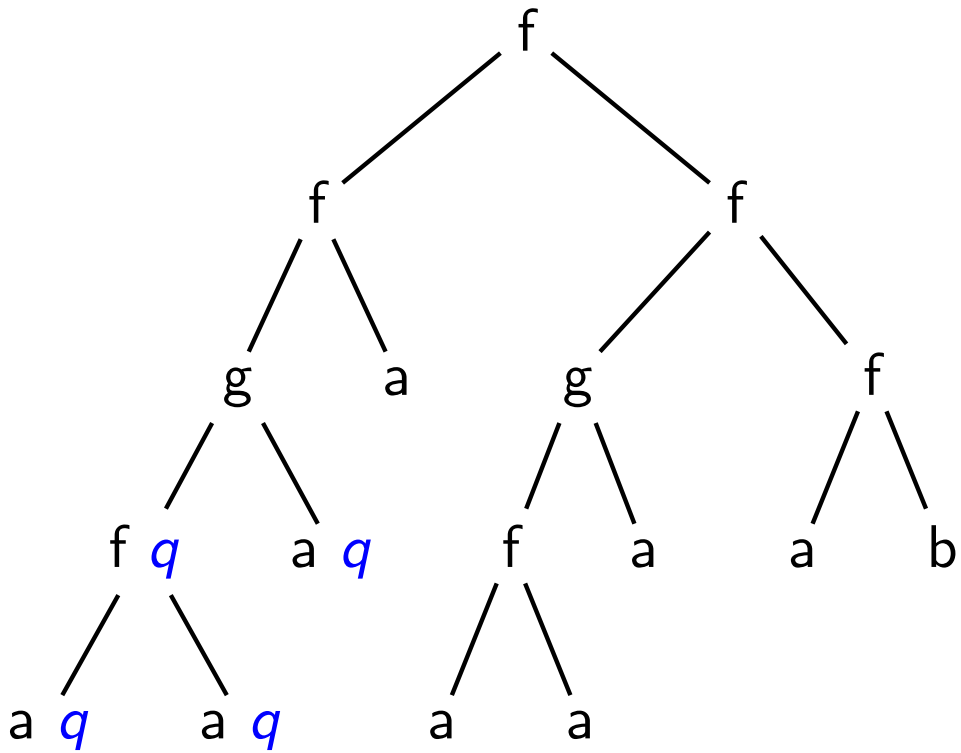


- $\Sigma = \{f, a, g\}$
- $Q = F = \{q, q_g, q_g^{\downarrow}\}$
- $\Delta =$

$$
\begin{aligned}
& a \longrightarrow q \\
& g(q, q) \longrightarrow q_g \\
& f(q, q) \longrightarrow q \\
& f(q_g, q) \longrightarrow q_g^{\downarrow} \\
& f(q, q_g) \longrightarrow q_g^{\downarrow} \\
& f(q_g, q_g) \longrightarrow q_g^{\downarrow} \\
& f(q_g^{\downarrow}, q_g) \longrightarrow q_g^{\downarrow} \\
& \ldots
\end{aligned}
$$

- $q_g \ =_A \ q_g$

Example: all subtrees rooted at g are equal

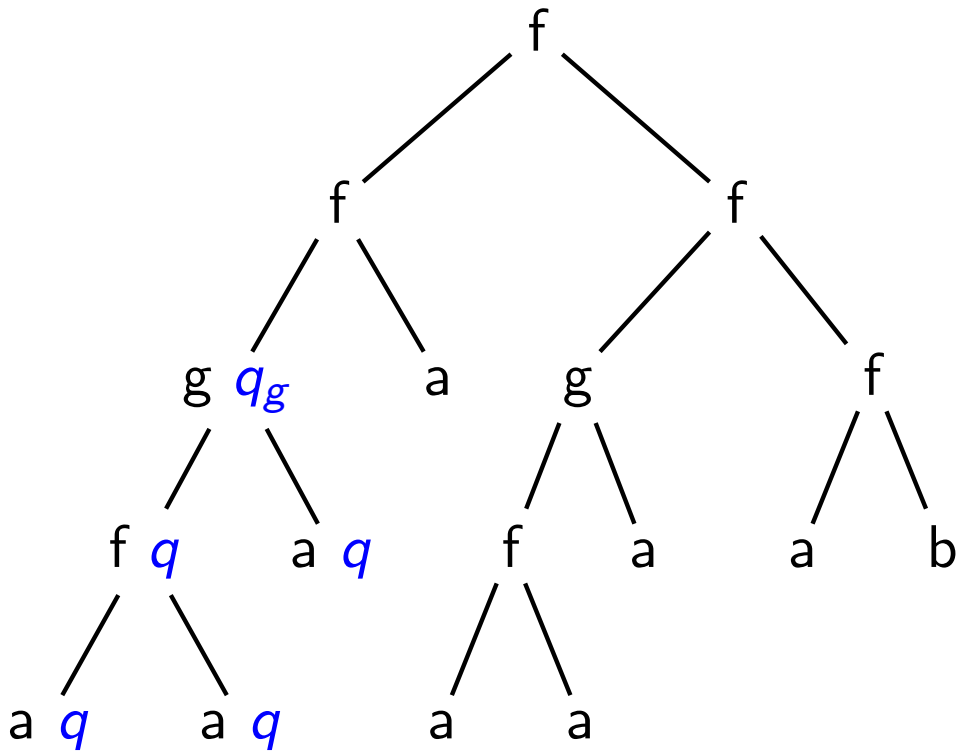


- $\Sigma = \{f, a, g\}$
- $Q = F = \{q, q_g, q_g^{\downarrow}\}$
- $\Delta =$

$$
\begin{aligned}
& a \longrightarrow q \\
& g(q, q) \longrightarrow q_g \\
& f(q, q) \longrightarrow q \\
& f(q_g, q) \longrightarrow q_g^{\downarrow} \\
& f(q, q_g) \longrightarrow q_g^{\downarrow} \\
& f(q_g, q_g) \longrightarrow q_g^{\downarrow} \\
& f(q_g^{\downarrow}, q_g) \longrightarrow q_g^{\downarrow} \\
& \ldots
\end{aligned}
$$

- $q_g \ =_A \ q_g$

Example: all subtrees rooted at g are equal

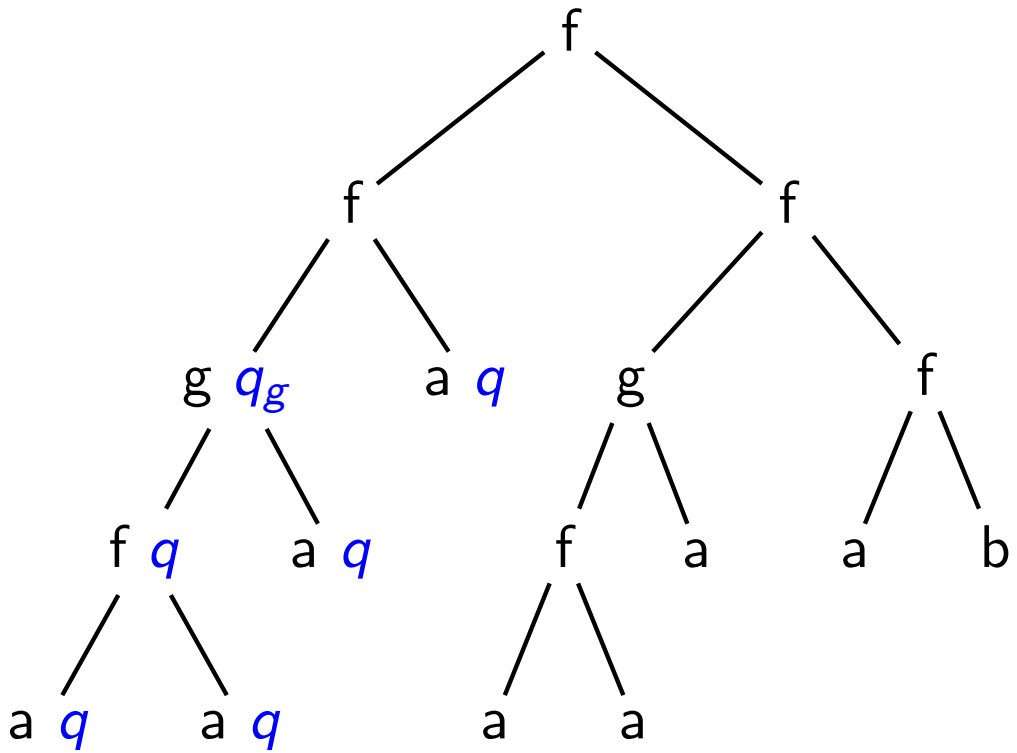


- $\Sigma = \{f, a, g\}$
- $Q = F = \{q, q_g, q_g^{\downarrow}\}$
- $\Delta =$

$$
\begin{aligned}
& a \longrightarrow q \\
& g(q,q) \longrightarrow q_g \\
& f(q,q) \longrightarrow q \\
& f(q_g,q) \longrightarrow q_g^{\downarrow} \\
& f(q,q_g) \longrightarrow q_g^{\downarrow} \\
& f(q_g,q_g) \longrightarrow q_g^{\downarrow} \\
& f(q_g^{\downarrow},q_g) \longrightarrow q_g^{\downarrow} \\
& \ldots
\end{aligned}
$$

- $q_g \ =_A \ q_g$

Example: all subtrees rooted at g are equal

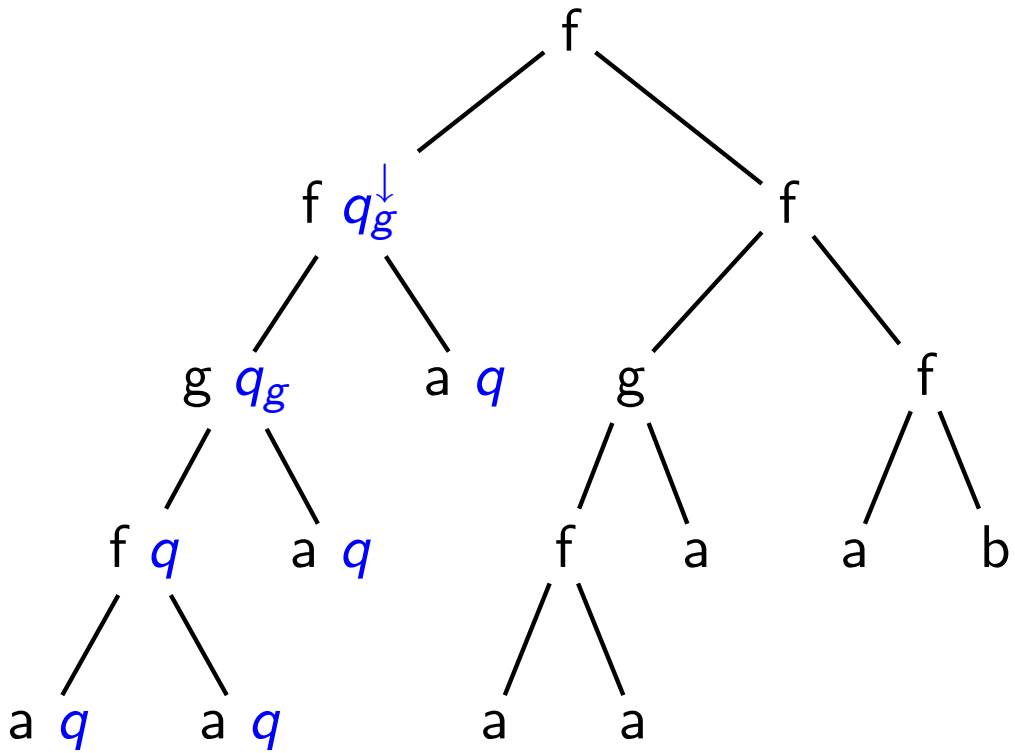


- $\Sigma = \{f, a, g\}$
- $Q = F = \{q, q_g, q_g^{\downarrow}\}$
- $\Delta =$

$$
\begin{aligned}
& a \longrightarrow q \\
& g(q,q) \longrightarrow q_g \\
& f(q,q) \longrightarrow q \\
& f(q_g,q) \longrightarrow q_g^{\downarrow} \\
& f(q,q_g) \longrightarrow q_g^{\downarrow} \\
& f(q_g,q_g) \longrightarrow q_g^{\downarrow} \\
& f(q_g^{\downarrow},q_g) \longrightarrow q_g^{\downarrow} \\
& \ldots
\end{aligned}
$$

- $q_g \ =_A \ q_g$

Example: all subtrees rooted at g are equal

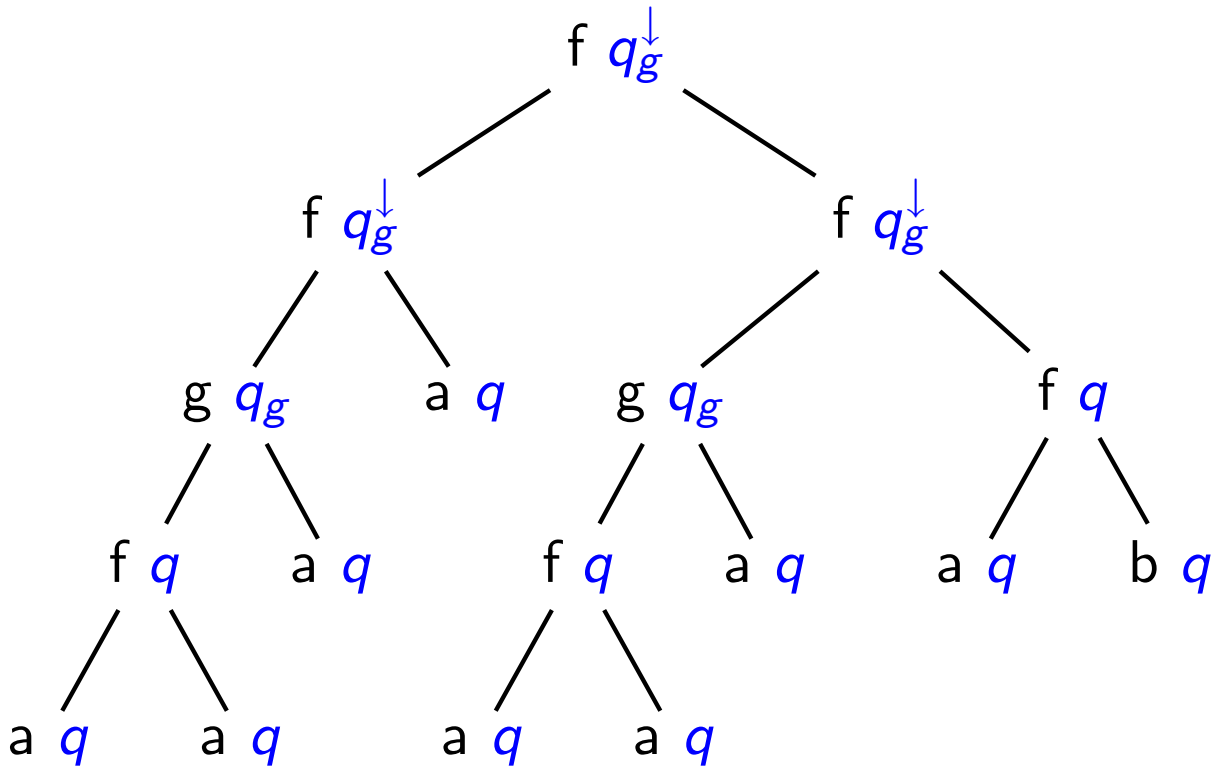


- $\Sigma = \{f, a, g\}$
- $Q = F = \{q, q_g, q_g^{\downarrow}\}$
- $\Delta =$

$$
\begin{aligned}
& a \longrightarrow q \\
& g(q,q) \longrightarrow q_g \\
& f(q,q) \longrightarrow q \\
& f(q_g,q) \longrightarrow q_g^{\downarrow} \\
& f(q,q_g) \longrightarrow q_g^{\downarrow} \\
& f(q_g,q_g) \longrightarrow q_g^{\downarrow} \\
& f(q_g^{\downarrow},q_g) \longrightarrow q_g^{\downarrow} \\
& \ldots
\end{aligned}
$$

- $q_g \ =_A \ q_g$

Example: all subtrees rooted at g are equal



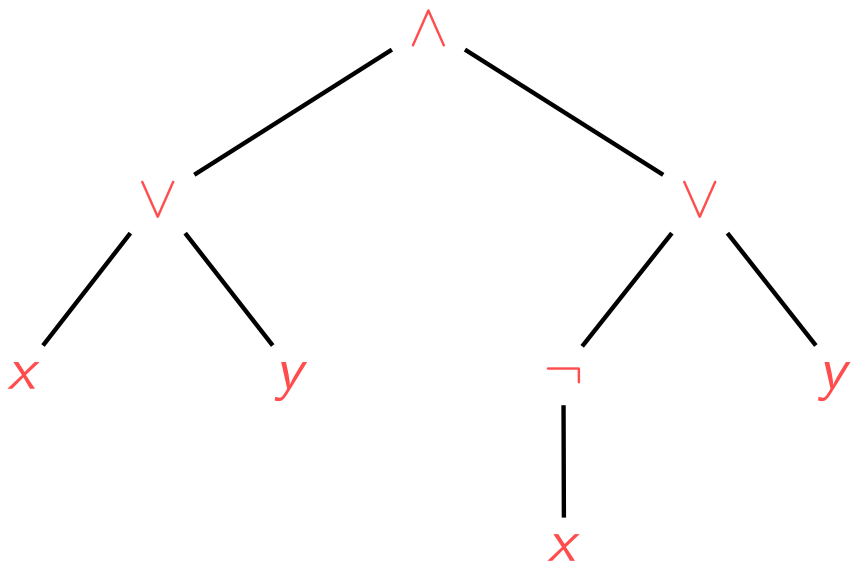
- $\Sigma = \{f, a, g\}$
- $Q = F = \{q, q_g, q_g^{\downarrow}\}$
- $\Delta =$

$$
\begin{aligned}
& a \longrightarrow q \\
& g(q, q) \longrightarrow q_g \\
& f(q, q) \longrightarrow q \\
& f(q_g, q) \longrightarrow q_g^{\downarrow} \\
& f(q, q_g) \longrightarrow q_g^{\downarrow} \\
& f(q_g, q_g) \longrightarrow q_g^{\downarrow} \\
& f(q_g^{\downarrow}, q_g) \longrightarrow q_g^{\downarrow} \\
& \ldots
\end{aligned}
$$

- $q_g \ =_A \ q_g$

Example: satisfiable CNF formulas

$\Phi = (x \vee y) \wedge (\neg x \vee y)$



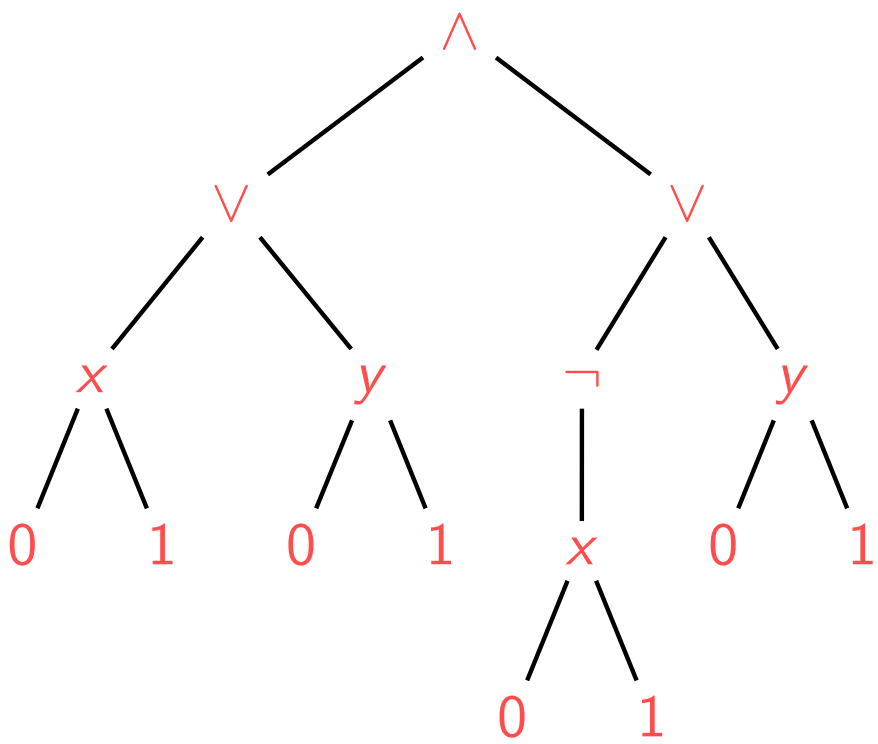
- $\Sigma = \{\vee, \wedge, x, y\}$
- $Q = \{q_0, q_1\}$
  $F = \{q_1\}$
- $\Delta =$

  $x \longrightarrow q_0 \quad x \longrightarrow q_1$

  ...

Example: satisfiable CNF formulas

$\Phi = (x \vee y) \wedge (\neg x \vee y)$



- $\Sigma = \{\vee, \wedge, x, y\}$
- $Q = \{q_0, q_1, q, q_x, q_y\}$
  $F = \{q_1\}$
- $\Delta =$

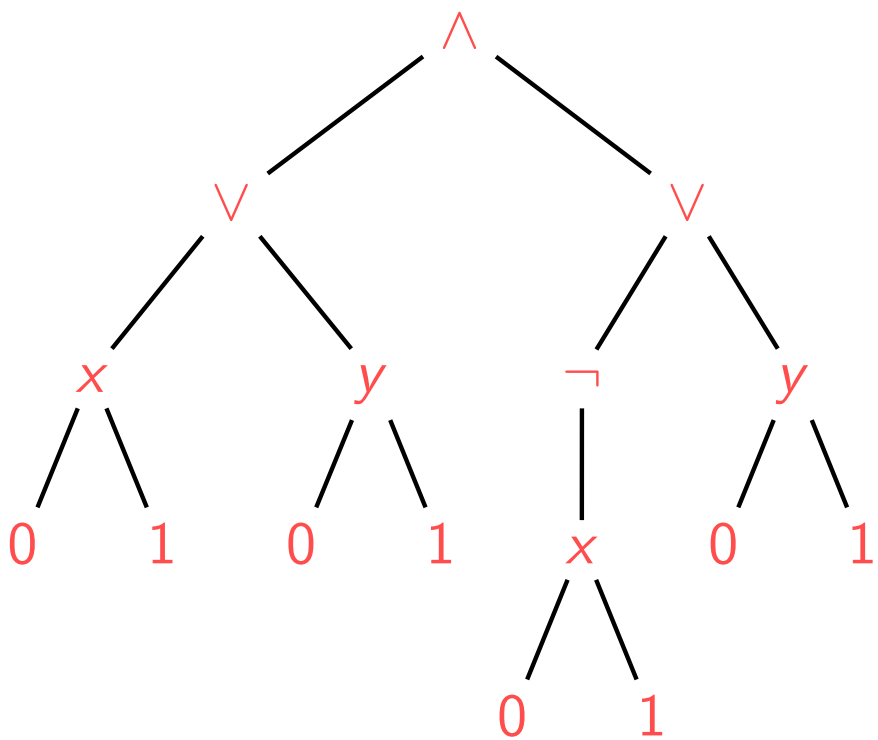
  $0 \to q_x \quad 0 \to q$
  $1 \to q_x \quad 1 \to q$
  $x(q, q_x) \to q_1$
  $x(q_x, q) \to q_0$

  $0 \to q_y \quad 0 \to q$
  ...

- $q_x =_A q_x \; q_y =_A q_y$

Example: satisfiable CNF formulas

$\Phi = (x \vee y) \wedge (\neg x \vee y)$



Prop. The uniform membership problem $\{(A, t) \mid t \in L(A)\}$ is NP-complete.

- $\Sigma = \{\vee, \wedge, x, y\}$
- $Q = \{q_0, q_1, q, q_x, q_y\}$
  $F = \{q_1\}$
- $\Delta =$

  $0 \to q_x \quad 0 \to q$
  $1 \to q_x \quad 1 \to q$
  $x(q, q_x) \to q_1$
  $x(q_x, q) \to q_0$

  $0 \to q_y \quad 0 \to q$
  ...

- $q_x =_A q_x \; q_y =_A q_y$

Determinization and Complement

Proposition

TAGEDs are not determinizable.

Proposition

TAGED-definable languages are not closed by complement.

Idea. The set of trees such that there is a subtree of the form $g(t, t')$ with $t \neq t'$ is TAGED-definable, but not its complement.

Closure by Union and Intersection

Proposition

TAGED-languages are closed by union and intersection.

Closure by Union and Intersection

Proposition

TAGED-languages are closed by union and intersection.

- union: take the disjoint union of the two TAGEDs

Closure by Union and Intersection

Proposition

TAGED-languages are closed by union and intersection.

- union: take the disjoint union of the two TAGEDs
- intersection: product automaton $A_1 \times A_2$ with

$$(q_1, q_2) =_{A_1 \times A_2} (p_1, p_2) \quad \text{iff} \quad q_1 =_{A_1} p_1 \text{ or } q_2 =_{A_1} p_2$$

$$(q_1, q_2) \neq_{A_1 \times A_2} (p_1, p_2) \quad \text{iff} \quad q_1 \neq_{A_1} p_1 \text{ or } q_2 \neq_{A_1} p_2$$

Outline

1. Definition, Examples and Properties of TAGEDs
2. **Emptiness Problem**
3. Applications: MSO and Unification

Emptiness Problem

- **Input**: a TAGED A
- **Output**: is there a tree accepted by A?

Theorem

Emptiness is decidable for positive, negative and bounded TAGEDs.

Testing equalities can be done with the same state

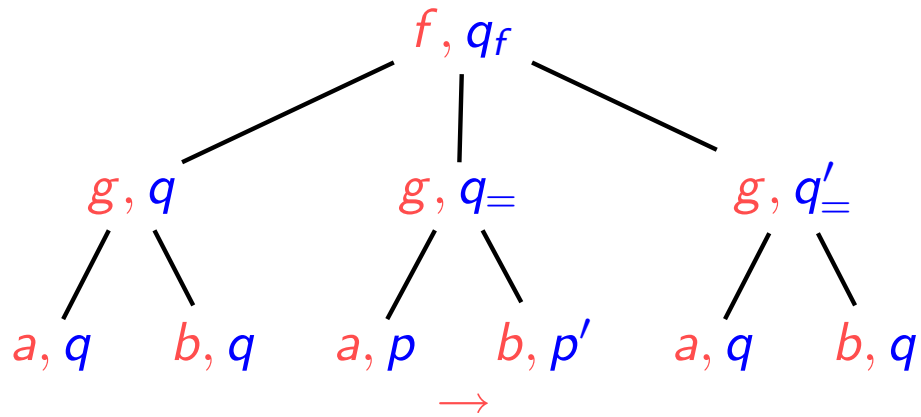
Lemma

In a successful run, the same state can be used to test equalities. In other words, every TAGED A *is equivalent to a TAGED* $A' = (Q', F', \Delta', =_{A'}, \neq_{A'})$ *such that* $=_{A'} \subseteq id_{Q'}$

Testing equalities can be done with the same state

Lemma

In a successful run, the same state can be used to test equalities. In other words, every TAGED A is equivalent to a TAGED $A' = (Q', F', \Delta', =_{A'}, \neq_{A'})$ such that $=_{A'} \subseteq id_{Q'}$



Testing equalities can be done with the same state

Lemma

In a successful run, the same state can be used to test equalities. In other words, every TAGED A *is equivalent to a TAGED* $A' = (Q', F', \Delta', =_{A'}, \neq_{A'})$ *such that* $=_{A'} \subseteq id_{Q'}$

Testing equalities can be done with the same state

Lemma

In a successful run, the same state can be used to test equalities. In other words, every TAGED A *is equivalent to a TAGED* $A' = (Q', F', \Delta', =_{A'}, \neq_{A'})$ *such that* $=_{A'} \subseteq id_{Q'}$

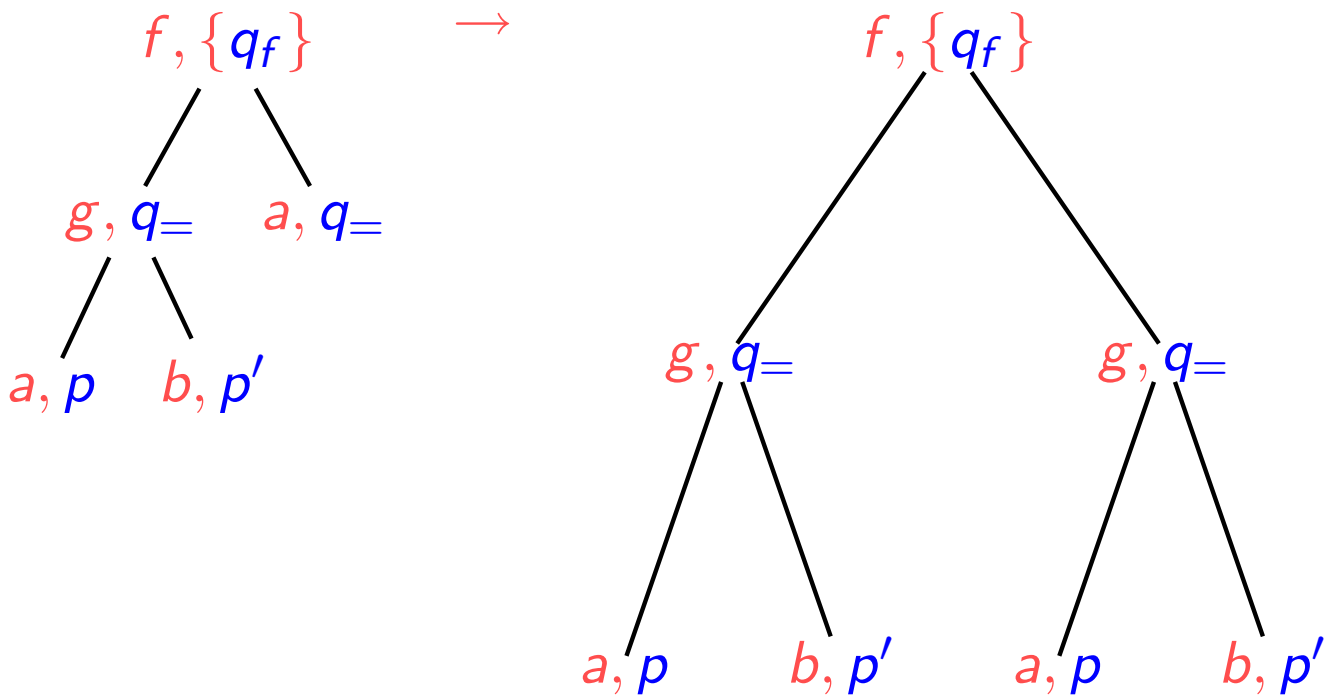
1. guess which states are used to test equalities
2. subset construction
3. $\rightarrow |A'| = 2^{O(|A|)}$

Emptiness of Positive TAGED (Upper Bound)

Theorem

Emptiness of positive TAGED is decidable in EXPTIME, and in linear time if $=_A \subseteq id_Q$.

1. transform A into an equivalent TAGED A' such that $=_{A'} \subseteq id_{Q'}$;
2. test emptiness of A' as for classical tree automata:



Emptiness of Positive TAGEDs (Lower Bound)

Theorem

Emptiness of positive TAGEDs is EXPTIME-hard.

Sketch. Reduction from the emptiness test of the intersection of n tree automata $A_1, \ldots, A_n$.

1. let A define the set of trees $\#(t_1, \ldots, t_n)$ such that $\forall i, t_i \in L(A_i)$
2. let $A_=$ define the set of trees $\#(t, \ldots, t)$
3. let $A_\cap$ define the set of trees $L(A) \cap L(A_=)$
4. test emptiness of $A_\cap$

Emptiness of Negative TAGEDs

Theorem

Emptiness of negative TAGEDs is decidabe in NEXPTIME.

- Encoding into a system of set constraints

$$\begin{cases} X_q \subseteq \bigcup_{f(q_1,q_2)\to q\in\Delta} f(X_{q_1}, X_{q_2}) \cup \bigcup_{a\to q\in\Delta} a \\ \bigcup_{q\in F} X_q \not\subseteq \varnothing \\ X_q \cap X_p = \varnothing \text{ for all } q, p \in Q \text{ such that } q \neq_A p \end{cases}$$

- can be solved in NEXPTIME (Aiken,Kozen,Wimmers,95) (Charatonik,Pacholski,94)

Main Result: Bounded TAGEDs

Definition

A bounded TAGED is a pair (A, k) where A is a TAGED and $k \in \mathbb{N}$ is a natural number.

Definition (Successful Runs)

Additional condition: every state in the domain of $\neq_A$ must occur at most k times along any root-to-leaves path.

Main Result: Bounded TAGEDs

Definition

A bounded TAGED is a pair (A, k) where A is a TAGED and $k \in \mathbb{N}$ is a natural number.

Definition (Successful Runs)

Additional condition: every state in the domain of $\neq_A$ must occur at most k times along any root-to-leaves path.

Theorem

Emptiness of bounded TAGEDs is deciable in 2NEXPTIME.

- assume that $=_A \subseteq id_Q$
- put the same run below equality states
- pump in parallel below equality states
- repair the unsatisfied disequality constraints

Outline

1. Definition, Examples and Properties of TAGEDs
2. Emptiness Problem
3. **Applications: MSO and Unification**

MSO with Tree Isomorphism Tests: MSO($\sim$)

- first-order variables x, y denote **nodes**
- second-order variables X, Y denote **set of nodes**
- an new predicate $x \sim y$ to test tree isomorphisms between subtrees rooted at x and y respectively

Theorem

Satisfiability of MSO($\sim$) is undecidable.

Existential Fragment: $\mathrm{MSO}^{\exists}_{=}$

- remove $\sim$ and add new predicates:

$$\begin{aligned} eq(X) &= \forall x, y \in X,\ x \sim y \\ diff_k(X, Y) &= \forall x \in X \forall y \in Y, x \not\sim y \wedge \\ & \quad DescChain(X) \leq k \wedge DescChain(Y) \leq k \end{aligned}$$

- formulas of the form:

$$\exists X_1 \ldots \exists X_n \psi(X_1, \ldots, X_n)$$

- tests $eq(X_i)$ or $diff_k(X_i, X_j)$ allowed **only** on $X_1, \ldots, X_n$ in ψ

Theorem

- **expressivity**: *$\mathrm{MSO}^{\exists}_{=}$ sentences and bounded TAGEDs effectively define the same tree languages;*
- **satisfiability**: *decidable for $\mathrm{MSO}^{\exists}_{=}$.*

Unification with membership constraints

- atoms of the form $s = s'$ or $x \in L$ (true if $\exists \sigma, \sigma(s) = \sigma(s')$
- s, s' are terms with variables
- FO over these atoms is decidable (Comon, Delor, ICALP'90)

Unification with membership constraints

- atoms of the form $s = s'$ or $x \in L$ (true if $\exists \sigma, \sigma(s) = \sigma(s')$
- s, s' are terms with variables
- FO over these atoms is decidable (Comon, Delor, ICALP'90)

- add context variables C and atoms $C \in L$
- **restriction:** cannot use the same context variable in two different terms
- full FO is undecidable (even with the restriction)
- decidable for the existential fragment (by using bounded TAGEDs)

Summary and Future Work

Summary

- automata with global constraints orthogonal to classical automata with local constraints
- emptiness for three subclasses
- correspondence with a super MSO logic
- unification with membership constraints

Future Work

- emptiness of full TAGEDs
- extension to unranked trees (via a binary encoding)
- regularity: given a TAGED, does it define a regular language?
- deterministic class