Tree Automata

Betim Musa

Seminar: Automata Theory

Albert-Ludwigs-University of Freiburg

Introduction

- Automata for tree structures
- Generalization of finite automata
- Two types of tree automata
  - (1) top-down, (2) bottom-up

- Applications:
  - Compiler construction: generate machine code
  - Natural Language Processing: machine translation
  - XML: processing XML documents

Outline

- Tree automata
  - Basics of tree automata
  - Bottom-up tree automata
  - Top-down tree automata
  - Decision problems & complexity
- Connection to logic
  - Monadic Second Order Logic (MSOL)
  - Equivalence between tree automata and MSOL

Basics

- Definition: Σ is a ranked alphabet if
  - It is a non-empty finite set
  - each symbol $a \in \Sigma$ is assigned a finite set $rank(a) \subseteq \mathbb{N}$
  - $\Sigma_i := \{ a \in \Sigma \mid i \in rank(a) \}$
  - $\Sigma = \Sigma_0 \cup \ldots \cup \Sigma_m$

Basics

- Definition: Σ is a ranked alphabet if
  - It is a non-empty finite set
  - each symbol $a \in \Sigma$ is assigned a finite set $rank(a) \subseteq \mathbb{N}$
  - $\Sigma_i := \{a \in \Sigma \mid i \in rank(a)\}$
  - $\Sigma = \Sigma_0 \cup \ldots \cup \Sigma_m$
- Definition: A tree over Σ is inductively defined
  - each symbol $a \in \Sigma_0$ is a tree
  - For $f \in \Sigma_k$ and trees $t_1 \ldots t_k$, $f(t_1 \ldots t_k)$ is also a tree.

Basics

- Definition: Σ is a ranked alphabet if
  - It is a non-empty finite set
  - each symbol $a \in \Sigma$ is assigned a finite set $rank(a) \subseteq \mathbb{N}$
  - $\Sigma_i := \{a \in \Sigma \mid i \in rank(a)\}$
  - $\Sigma = \Sigma_0 \cup \ldots \cup \Sigma_m$
- Definition: A tree over Σ is inductively defined
  - each symbol $a \in \Sigma_0$ is a tree
  - For $f \in \Sigma_k$ and trees $t_1 \ldots t_k$, $f(t_1 \ldots t_k)$ is also a tree.
- The set of all trees over Σ is denoted by $T(\Sigma)$

Bottom-up tree automata

- Definition: A bottom-up tree automaton is a quadruple $B = (\Sigma, Q, F, \Delta)$
  - Σ a ranked alphabet
  - Q finite set of states
  - $F \subseteq Q$ set of final states
  - Δ finite set of transition rules of the form $f(q_1(t_1), \ldots, q_n(t_n)) \rightarrow q$

    where $f \in \Sigma_n$, $q, q_1, \ldots, q_n \in Q$, $t_{1,} \ldots, t_n$ trees

Bottom-up tree automata

- Definition: A bottom-up tree automaton is a quadruple $B=(\Sigma, Q, F, \Delta)$
  - Σ a ranked alphabet
  - Q finite set of states
  - $F \subseteq Q$ set of final states
  - Δ finite set of transition rules of the form $f(q_1(t_1), \dots, q_n(t_n)) \to q$

    where $f \in \Sigma_n, \; q, q_{1,} \dots, q_n \in Q, \; t_{1,} \dots, t_n$ trees
- Rules for constants are „initial rules“ $a \to q_a$

Bottom-up tree automata

- Definition: A bottom-up tree automaton is a quadruple $B=(\Sigma, Q, F, \Delta)$
  - Σ a ranked alphabet
  - Q finite set of states
  - $F \subseteq Q$ set of final states
  - Δ finite set of transition rules of the form
    $f(q_1(t_1), \ldots, q_n(t_n)) \rightarrow q$

    where $f \in \Sigma_n$, $q, q_{1,} \ldots, q_n \in Q$, $t_{1,} \ldots, t_n$ trees
- Rules for constants are „initial rules“ $a \rightarrow q_a$
- Definition: Acceptance of a tree
  - A tree $t \in T(\Sigma)$ is accepted iff $t \rightarrow^* q(t), where\, q \in F$

Bottom-up tree automata

Example 1

- Example: A tree automaton, which accepts all true Boolean expressions over $\Sigma = \{ \wedge_2 , \vee_2 , \neg_1 , 0_0 , 1_0 \}$

Bottom-up tree automata

Example 1

- Example: A tree automaton, which accepts all true Boolean expressions over $\Sigma = \{\wedge_2, \vee_2, \neg_1, 0_0, 1_0\}$
- $B = (\Sigma, Q, F, \Delta)$ with $Q = \{q_{0,} q_1\}$, $F = \{q_1\}$

$$\Delta = \{0 \rightarrow q_0, 1 \rightarrow q_1, \neg(q_0(t)) \rightarrow q_1, \neg(q_1(t)) \rightarrow q_{0,}\}$$

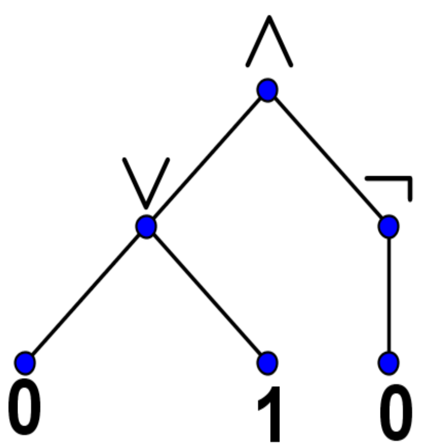
$$\cup \{\wedge(q_i(t_1), q_j(t_2)) \rightarrow q_{min(i,j)}\}$$

$$\cup \{\vee(q_i(t_1), q_j(t_2)) \rightarrow q_{max(i,j)}\}$$

Bottom-up tree automata

Example 1

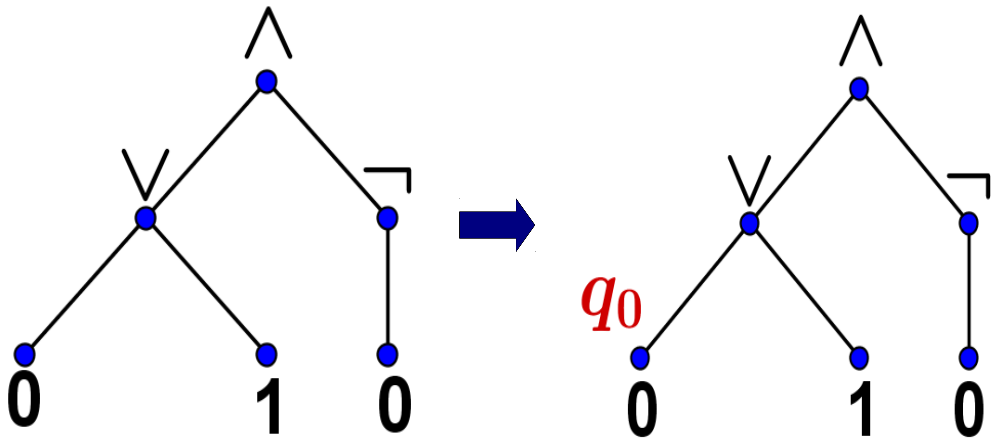
- $B=(\Sigma,Q,F,\Delta)$ with $Q=\{q_0, q_1\}$, $F=\{q_1\}$
  $\Delta=\{0\rightarrow q_0, 1\rightarrow q_1, \neg(q_0(t))\rightarrow q_1, \neg(q_1(t))\rightarrow q_0\}$
  $\cup\{\wedge(q_i(t_1), q_j(t_2))\rightarrow q_{min(i,j)}\}$
  $\cup\{\vee(q_i(t_1), q_j(t_2))\rightarrow q_{max(i,j)}\}$
- Assume we have the following input:
  $t_1=\wedge(\vee(0,1),\neg(0))$



Bottom-up tree automata

Example 1

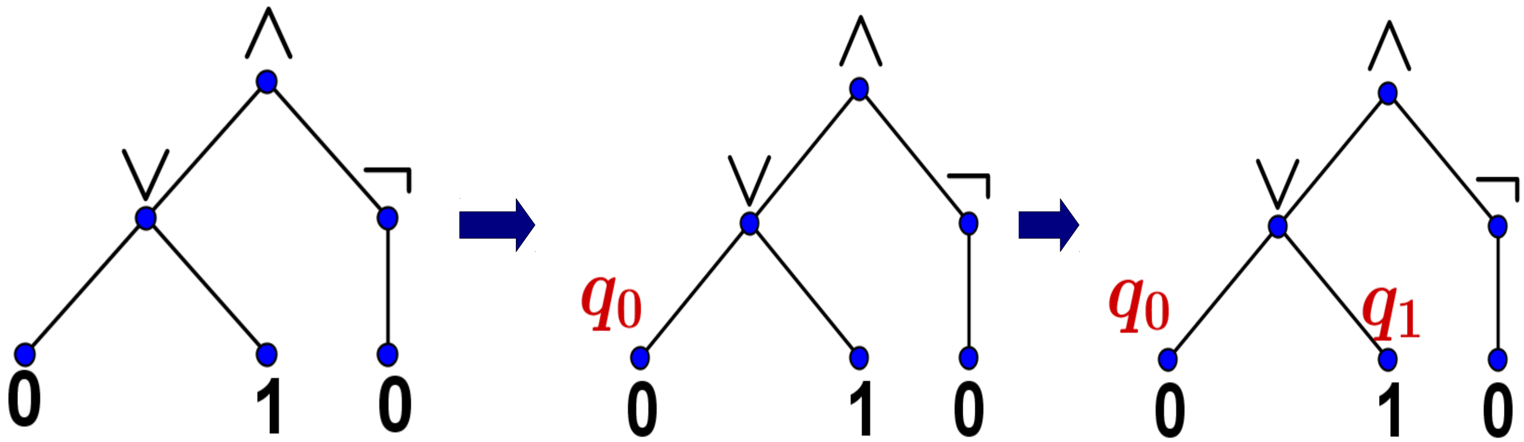
- $B=(\Sigma, Q, F, \Delta)$ with $Q=\{q_{0,} q_1\}$, $F=\{q_1\}$
  $\Delta=\{0\rightarrow q_0, 1\rightarrow q_1, \neg(q_0(t))\rightarrow q_1, \neg(q_1(t))\rightarrow q_{0,}\}$
  $\cup\{\wedge(q_i(t_1), q_j(t_2))\rightarrow q_{min(i,j)}\}$
  $\cup\{\vee(q_i(t_1), q_j(t_2))\rightarrow q_{max(i,j)}\}$
- Assume we have the following input:
  $t_1=\wedge(\vee(0,1), \neg(0))$



Bottom-up tree automata

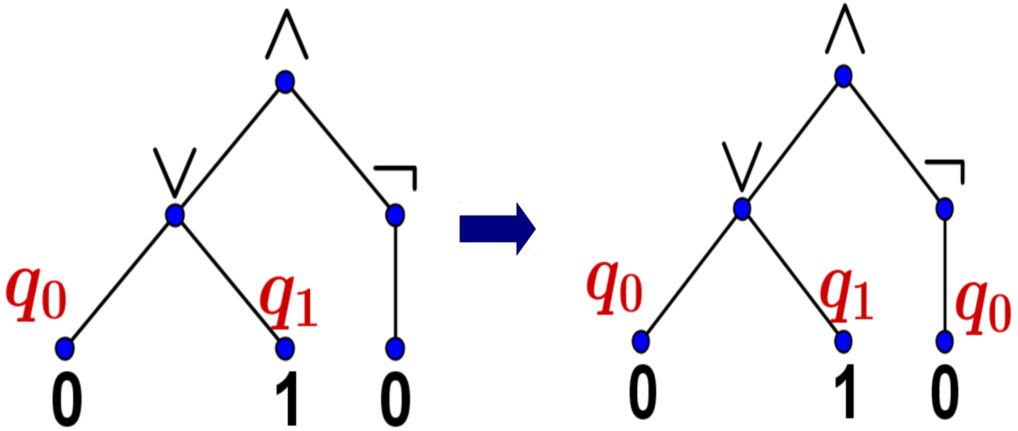
Example 1

- $B=(\Sigma, Q, F, \Delta)$ with $Q=\{q_0, q_1\}$, $F=\{q_1\}$
  $\Delta=\{0 \rightarrow q_0, 1 \rightarrow q_1, \neg(q_0(t)) \rightarrow q_1, \neg(q_1(t)) \rightarrow q_0\}$
  $\cup\{\wedge(q_i(t_1), q_j(t_2)) \rightarrow q_{min(i,j)}\}$
  $\cup\{\vee(q_i(t_1), q_j(t_2)) \rightarrow q_{max(i,j)}\}$
- Assume we have the following input:
  $t_1 = \wedge(\vee(0,1), \neg(0))$



Bottom-up tree automata
Example 1

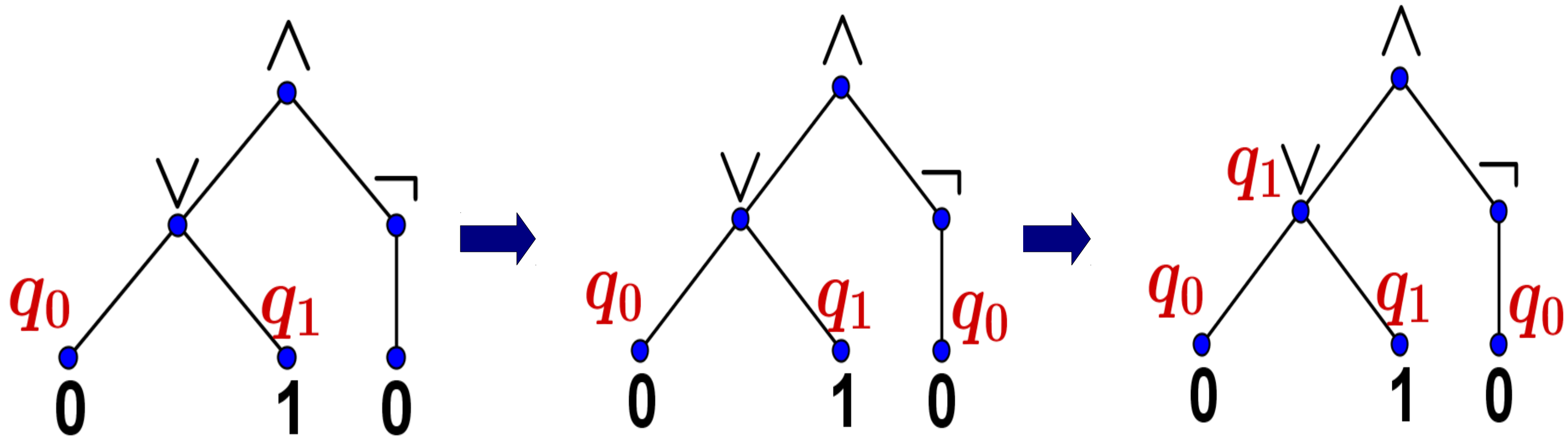
- $B=(\Sigma, Q, F, \Delta)$ with $Q=\{q_{0,} q_1\}$, $F=\{q_1\}$
  $\Delta=\{0 \rightarrow q_0, 1 \rightarrow q_1, \neg(q_0(t)) \rightarrow q_1, \neg(q_1(t)) \rightarrow q_{0,}\}$
  $\cup\{\wedge(q_i(t_1), q_j(t_2)) \rightarrow q_{min(i,j)}\}$
  $\cup\{\vee(q_i(t_1), q_j(t_2)) \rightarrow q_{max(i,j)}\}$
- Assume we have the following input:
  $t_1 = \wedge(\vee(0,1), \neg(0))$



Bottom-up tree automata

Example 1

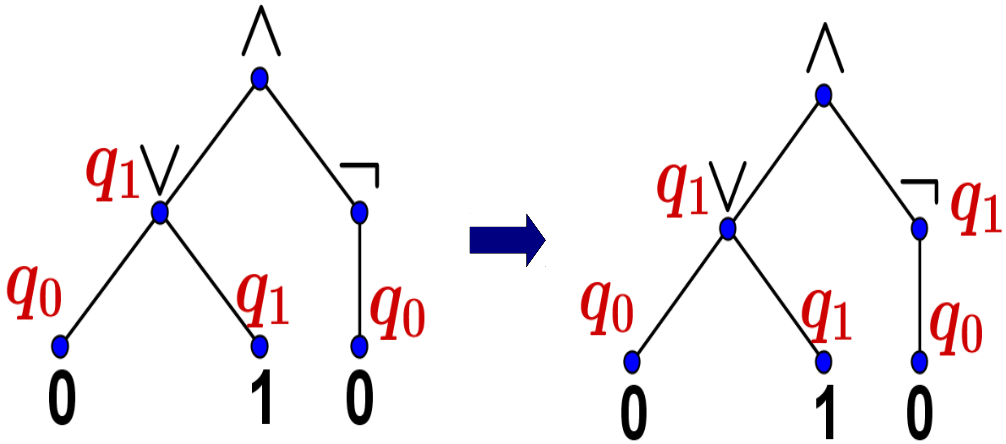
- $B=(\Sigma, Q, F, \Delta)$ with $Q=\{q_{0,} q_1\}$, $F=\{q_1\}$
  $\Delta=\{0\rightarrow q_0, 1\rightarrow q_1, \neg(q_0(t))\rightarrow q_1, \neg(q_1(t))\rightarrow q_{0,}\}$
  $\cup\{\wedge(q_i(t_1), q_j(t_2))\rightarrow q_{min(i,j)}\}$
  $\cup\{\vee(q_i(t_1), q_j(t_2))\rightarrow q_{max(i,j)}\}$
- Assume we have the following input:
  $t_1 = \wedge(\vee(0,1), \neg(0))$



Bottom-up tree automata

Example 1

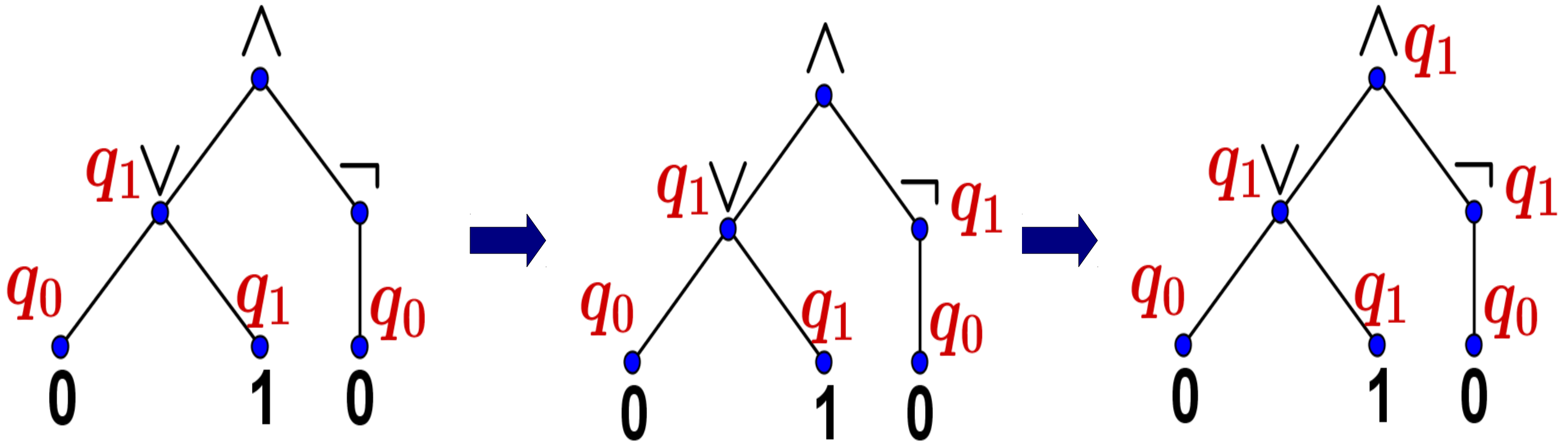
- $B=(\Sigma, Q, F, \Delta)$ with $Q=\{q_0, q_1\}$, $F=\{q_1\}$
  $\Delta=\{0\rightarrow q_0, 1\rightarrow q_1, \neg(q_0(t))\rightarrow q_1, \neg(q_1(t))\rightarrow q_0\}$
  $\cup\{\wedge(q_i(t_1), q_j(t_2))\rightarrow q_{min(i,j)}\}$
  $\cup\{\vee(q_i(t_1), q_j(t_2))\rightarrow q_{max(i,j)}\}$
- Assume we have the following input:
  $t_1=\wedge(\vee(0,1), \neg(0))$



Bottom-up tree automata

Example 1

- $B=(\Sigma, Q, F, \Delta)$ with $Q=\{q_0, q_1\}$, $F=\{q_1\}$
  $\Delta=\{0\rightarrow q_0, 1\rightarrow q_1, \neg(q_0(t))\rightarrow q_1, \neg(q_1(t))\rightarrow q_0,\}$
  $\cup\{\wedge(q_i(t_1), q_j(t_2))\rightarrow q_{min(i,j)}\}$
  $\cup\{\vee(q_i(t_1), q_j(t_2))\rightarrow q_{max(i,j)}\}$
- Assume we have the following input:
  $t_1=\wedge(\vee(0,1), \neg(0))$



Bottom-up tree automata

Further information

- Non-deterministic if there are at least two rules with the same left-hand side

  $$a(q1,\ldots,q_k) \rightarrow q$$
  $$a(q1,\ldots,q_k) \rightarrow q' \qquad where \quad q \neq q'$$

- But expressive power is equal
  - Powerset construction
- Regular expressions definable
  - Equal power to tree automata

Top-down tree automata

- Definition: A top-down automaton is a structure $T = (\Sigma, Q, Q_I, \Delta)$
  - where Σ is a ranked alphabet
  - Q is a finite set of states
  - Q_I is a finite set of initial states
  - Δ finite set of transition rules of the form $q(f(t_{1,} \ldots, t_n)) \rightarrow f(q_1(t_1), \ldots, q_n(t_n))$
  - where $f \in \Sigma_n$, $q, q_1, \ldots, q_n \in Q$, $t_{1,} \ldots, t_n$ different trees

- A tree $t \in T(\Sigma)$ is accepted iff $q(t) \rightarrow^* t \; for\, some\, q \in Q_I$

Top-down tree automata

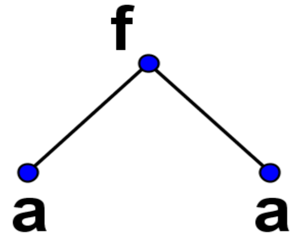
Example 1

- A top-down automaton, which accepts all trees with depth 1 over $\Sigma = \{ f_2, g_1, a_0 \}$

Top-down tree automata

Example 1

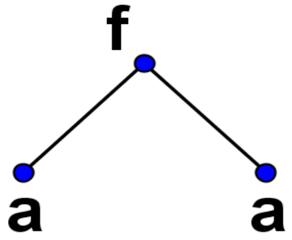
- A top-down automaton, which accepts all trees with depth 1 over $\Sigma=\{f_2, g_1, a_0\}$
- Define $T=(\Sigma, Q, Q_I, \Delta)$ where $Q=\{q_0, q_1\}$, $Q_I=\{q_0\}$
  $\Delta=\{q_0(f(t_{1,} t_2)) \rightarrow f(q_1(t_1), q_1(t_2)), q_0(g(t)) \rightarrow g(q_1(t))\}$
  $\cup\{q_1(a) \rightarrow a\}$
- Instance input is:

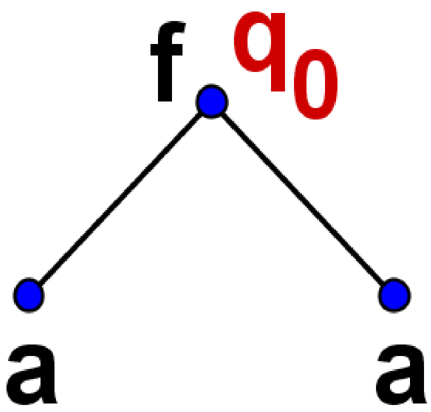


Top-down tree automata

Example 1

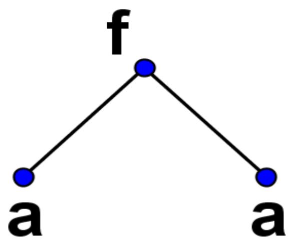
- A top-down automaton, which accepts all trees with depth 1 over $\Sigma=\{f_2,g_1,a_0\}$
- Define $T=(\Sigma,Q,Q_I,\Delta)$ where $Q=\{q_0,q_1\}$, $Q_I=\{q_0\}$
  $\Delta=\{q_0(f(t_1,t_2))\rightarrow f(q_1(t_1),q_1(t_2)),q_0(g(t))\rightarrow g(q_1(t))\}$
  $\cup\{q_1(a)\rightarrow a\}$
- Instance input is:

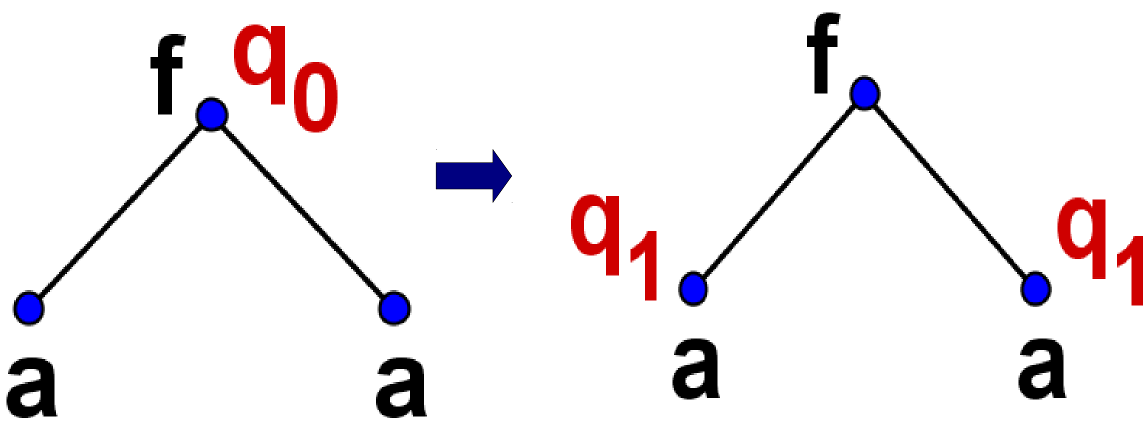




Top-down tree automata
Example 1

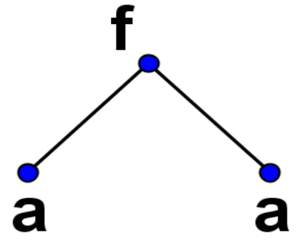
- A top-down automaton, which accepts all trees with depth 1 over $\Sigma=\{f_2, g_1, a_0\}$
- Define $T=(\Sigma, Q, Q_I, \Delta)$ where $Q=\{q_0, q_1\}$, $Q_I=\{q_0\}$
  $\Delta=\{q_0(f(t_{1,} t_2)) \rightarrow f(q_1(t_1), q_1(t_2)), q_0(g(t)) \rightarrow g(q_1(t))\}$
  $\cup \{q_1(a) \rightarrow a\}$
- Instance input is:

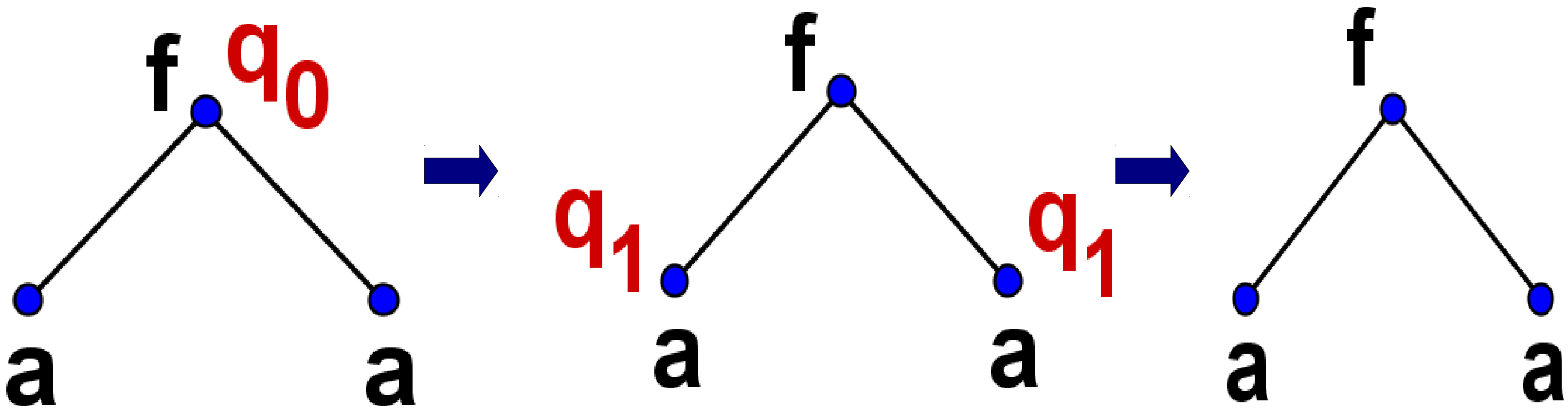




Top-down tree automata
Example 1

- A top-down automaton, which accepts all trees with depth 1 over $\Sigma = \{f_2, g_1, a_0\}$
- Define $T = (\Sigma, Q, Q_I, \Delta)$ where $Q = \{q_0, q_1\}$, $Q_I = \{q_0\}$
  $\Delta = \{q_0(f(t_{1,} t_2)) \rightarrow f(q_1(t_1), q_1(t_2)), q_0(g(t)) \rightarrow g(q_1(t))\}$
  $\cup \{q_1(a) \rightarrow a\}$
- Instance input is:

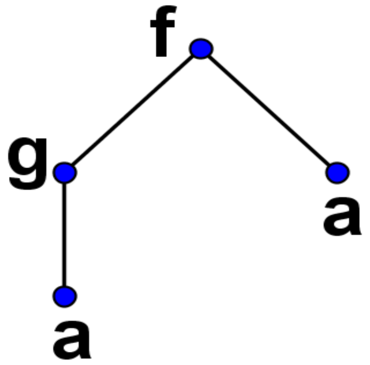




Top-down tree automata

Example 2

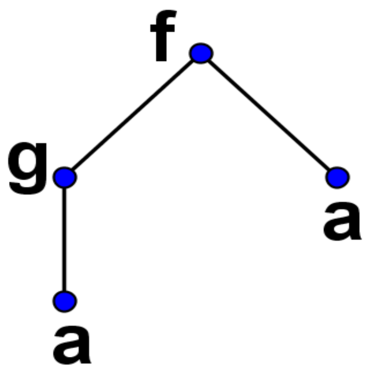
- A top-down automaton, which accepts all trees with depth 1 over $\Sigma=\{f_2, g_1, a_0\}$
- Define $T=(\Sigma, Q, Q_I, \Delta)$ where $Q=\{q_0, q_1\}$, $Q_I=\{q_0\}$
  $\Delta=\{q_0(f(t_{1,}t_2))\rightarrow f(q_1(t_1), q_1(t_2)), q_0(g(t))\rightarrow g(q_1(t))\}$
  $\cup\{q_1(a)\rightarrow a\}$
- Input, which is <u>not</u> accepted:



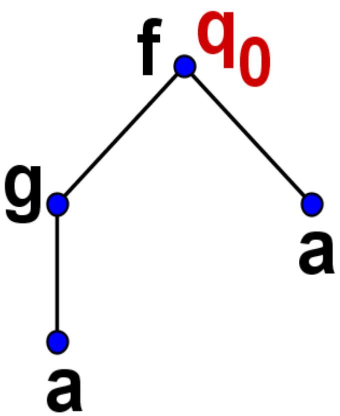
Top-down tree automata

Example 2

- A top-down automaton, which accepts all trees with depth 1 over $\Sigma=\{f_2, g_1, a_0\}$
- Define $T=(\Sigma, Q, Q_I, \Delta)$ where $Q=\{q_0, q_1\}$, $Q_I=\{q_0\}$
  $\Delta=\{q_0(f(t_{1,}t_2))\rightarrow f(q_1(t_1), q_1(t_2)), q_0(g(t))\rightarrow g(q_1(t))\}$
  $\cup\{q_1(a)\rightarrow a\}$



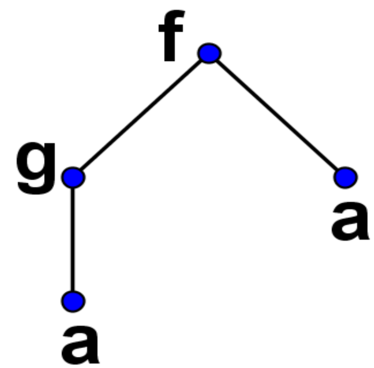
- Input, which is not accepted:

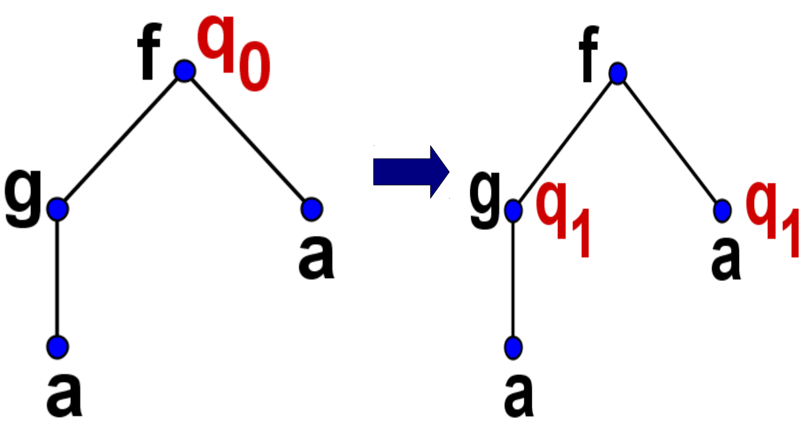


Top-down tree automata

Example 2

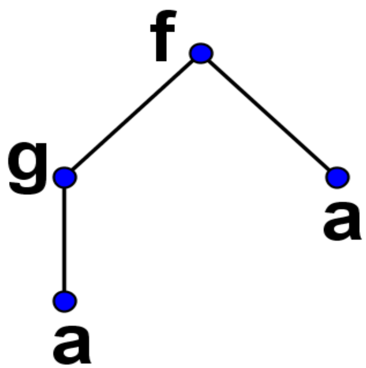
- A top-down automaton, which accepts all trees with depth 1 over $\Sigma = \{f_2, g_1, a_0\}$
- Define $T = (\Sigma, Q, Q_I, \Delta)$ where $Q = \{q_0, q_1\}$, $Q_I = \{q_0\}$
  $\Delta = \{q_0(f(t_{1,} t_2)) \rightarrow f(q_1(t_1), q_1(t_2)), q_0(g(t)) \rightarrow g(q_1(t))\}$
  $\cup \{q_1(a) \rightarrow a\}$
- Input, which is not accepted:



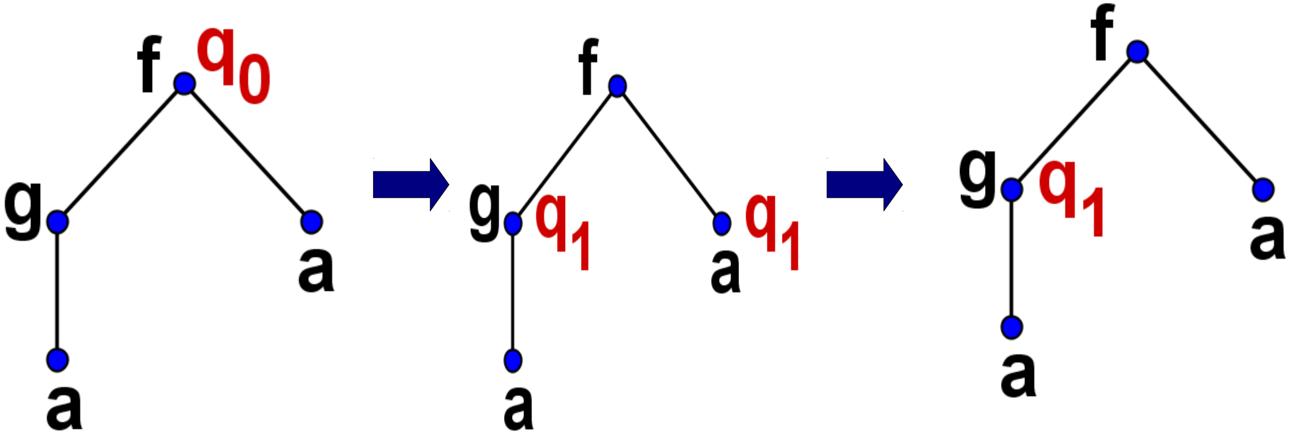


Top-down tree automata
Example 2

- A top-down automaton, which accepts all trees with depth 1 over $\Sigma = \{f_2, g_1, a_0\}$
- Define $T = (\Sigma, Q, Q_I, \Delta)$ where $Q = \{q_0, q_1\}$, $Q_I = \{q_0\}$
  $\Delta = \{q_0(f(t_{1,} t_2)) \rightarrow f(q_1(t_1), q_1(t_2)), q_0(g(t)) \rightarrow g(q_1(t))\}$
  $\cup \{q_1(a) \rightarrow a\}$



- Input, which is not accepted:



Top-down tree automata

Non-deterministic vs. deterministic

- Claim: Deterministic top-down tree automata are strictly less powerful than the non-deterministic ones

Top-down tree automata

Non-deterministic vs. deterministic

- Claim: Deterministic top-down tree automata are strictly less powerful than the non-deterministic ones
- Remark: The doubleton set $DT = \{f(a,b) f(b,a)\}$ is acceptable by non-determinstic top-down tree automata

Top-down tree automata

Non-deterministic vs. deterministic

- Claim: Deterministic top-down tree automata are strictly less powerful than the non-deterministic ones
- Remark: The doubleton set $DT=\{f(a,b) f(b,a)\}$ is acceptable by non-determinstic top-down tree automata
- $T=(\Sigma,Q,Q_I,\Delta)$ where $\Sigma=\{f_2,a_0,b_0\},Q=\{q_0,q_a,q_b\},Q_I=\{q_0\}$

$$\Delta=\{q_0(f(t_1,t_2))\rightarrow f(q_a(t_1),q_b(t_2)),q_0(f(t_1,t_2))\rightarrow f(q_b(t_1),q_a(t_2))\}$$
$$\cup\{q_a(a)\rightarrow a,q_b(b)\rightarrow b\}$$

Top-down tree automata

Non-deterministic vs. deterministic

- Claim: Deterministic top-down tree automata are strictly less powerful than the non-deterministic ones
- $T=(\Sigma, Q, Q_I, \Delta)$ where $\Sigma=\{f_2, a_0, b_0\}, Q=\{q_0, q_a, q_b\}, Q_I=\{q_0\}$
  $\Delta=\{q_0(f(t_1, t_2))\rightarrow f(q_a(t_1), q_b(t_2)), q_0(f(t_1, t_2))\rightarrow f(q_b(t_1), q_a(t_2))\}$
  $\cup\{q_a(a)\rightarrow a, q_b(b)\rightarrow b\}$
- Assume that there is a determinstic top-down automaton which recognizes the doubleton set

Top-down tree automata

Non-deterministic vs. deterministic

- Claim: Deterministic top-down tree automata are strictly less powerful than the non-deterministic ones
- $T=(\Sigma, Q, Q_I, \Delta)$ where $\Sigma=\{f_2, a_0, b_0\}, Q=\{q_0, q_a, q_b\}, Q_I=\{q_0\}$
  $\Delta=\{q_0(f(t_1,t_2))\rightarrow f(q_a(t_1),q_b(t_2)), q_0(f(t_1,t_2))\rightarrow f(q_b(t_1),q_a(t_2))\}$
  $\cup\{q_a(a)\rightarrow a, q_b(b)\rightarrow b\}$
- Assume that there is a determinstic top-down automaton which recognizes the doubleton set
- It must have the following transition rules

$$\Delta=\{q_0(f(t_1,t_2))\rightarrow f(q_1(t_1),q_2(t_2))\}$$

Top-down tree automata

Non-deterministic vs. deterministic

- Claim: Deterministic top-down tree automata are strictly less powerful than the non-deterministic ones
- $T=(\Sigma, Q, Q_I, \Delta)$ where $\Sigma=\{f_2, a_0, b_0\}, Q=\{q_0, q_a, q_b\}, Q_I=\{q_0\}$
  $\Delta=\{q_0(f(t_1,t_2)) \rightarrow f(q_a(t_1), q_b(t_2)), q_0(f(t_1,t_2)) \rightarrow f(q_b(t_1), q_a(t_2))\}$
  $\cup\{q_a(a) \rightarrow a, q_b(b) \rightarrow b\}$
- Assume that there is a determinstic top-down automaton which recognizes the doubleton set
- It must have the following transition rules

$$\Delta=\{q_0(f(t_1,t_2)) \rightarrow f(q_1(t_1), q_2(t_2)), q_1(a) \rightarrow a, q_2(b) \rightarrow b\}$$

Top-down tree automata

Non-deterministic vs. deterministic

- Claim: Deterministic top-down tree automata are strictly less powerful than the non-deterministic ones
- $T=(\Sigma, Q, Q_I, \Delta)$ where $\Sigma=\{f_2, a_0, b_0\}, Q=\{q_0, q_a, q_b\}, Q_I=\{q_0\}$
  $\Delta=\{q_0(f(t_1,t_2))\rightarrow f(q_a(t_1), q_b(t_2)), q_0(f(t_1,t_2))\rightarrow f(q_b(t_1), q_a(t_2))\}$
  $\cup\{q_a(a)\rightarrow a, q_b(b)\rightarrow b\}$
- Assume that there is a determinstic top-down automaton which recognizes the doubleton set
- It must have the following transition rules
  $\Delta=\{q_0(f(t_1,t_2))\rightarrow f(q_1(t_1), q_2(t_2)), q_1(a)\rightarrow a, q_2(b)\rightarrow b\}$
  $\cup\{q_2(a)\rightarrow a, q_1(b)\rightarrow b\}$

Top-down tree automata

Non-deterministic vs. deterministic

- Claim: Deterministic top-down tree automata are strictly less powerful than the non-deterministic ones
- $T=(\Sigma, Q, Q_I, \Delta)$ where $\Sigma=\{f_2, a_0, b_0\}, Q=\{q_0, q_a, q_b\}, Q_I=\{q_0\}$
$\Delta=\{q_0(f(t_1,t_2))\to f(q_a(t_1), q_b(t_2)), q_0(f(t_1,t_2))\to f(q_b(t_1), q_a(t_2))\}$
$\cup\{q_a(a)\to a, q_b(b)\to b\}$
- Assume that there is a determinstic top-down automaton which recognizes the doubleton set
- It must have the following transition rules
$\Delta=\{q_0(f(t_1,t_2))\to f(q_1(t_1), q_2(t_2)), q_1(a)\to a, q_2(b)\to b\}$
$\cup\{q_2(a)\to a, q_1(b)\to b\}$
- It accepts also f(a, a) → Contradiction.

Decision problems & complexity

	NDTA	NWA	PDA
$\cup, \cdot, *$	✔	✔	✔
complement	✔	✔	✘
intersection	✔	✔	✘
emptiness			
equivalence			
inclusion			

Decision problems & complexity

	NDTA	NWA	PDA
$\cup, \cdot, *$	✔	✔	✔
complement	✔	✔	✘
intersection	✔	✔	✘
emptiness	linear time	PTIME	PTIME
equivalence	EXPTIME	PTIME	undecidable
inclusion	EXPTIME	PTIME	undecidable

Outline

- Tree automata
  - Basics of tree automata
  - Bottom-up tree automata
  - Top-down tree automata
  - Decision problems & complexity
- Connection to logic
  - Monadic Second Order Logic (MSOL)
  - Equivalence between tree automata and MSOL

Monadic Second Order Logic

Why?

- Why consider logic on trees?
  - To specify languages in a more comfortable way

- L = „There is a path which consists of only *a*“
- Regular expression of L would become too large
- A formula for L:
  - $\phi := \exists x \exists y (x < y \wedge \forall z ((x < z \wedge z < y) \rightarrow P_a(z)))$

Monadic Second Order Logic

- Extension of first-order logic
- Second-order because quantification over sets is allowed
  - $\exists X \left(X(min) \rightarrow P_a(min) \right)$
- Monadic because quantification is restricted to sets (unary relations)

Monadic Second Order Logic

over trees

- Formulae are built up from
  - Variables x, y, z denoting positions of branches
  - Constant min , Position of the root node
  - Set variables X, Y, Z denoting sets of positions

Monadic Second Order Logic

over trees

- Formulae are built up from
  - Variables x, y, z denoting positions of branches
  - Constant min , Position of the root node
  - Set variables X, Y, Z denoting sets of positions
  - Atomic formulae (with explicit semantics)
  - x = y (equality)
  - $\leqslant$ prefix relation
  - $S_i(x, y)$ i-th successor relation
  - $P_a(x)$ „at position x there is an *a*“
  - X(y) „y is element of X“
  - And the usual connectors, quantifiers $\wedge, \vee, \neg, \ldots, \exists, \forall$

Monadic Second Order Logic

over trees

- How can we describe the properties of trees in terms of MSOL-formulae?

Monadic Second Order Logic
over trees

- How can we describe the properties of trees in terms of MSOL-formulae?
- Let $\Sigma = \Sigma_0 \cup .. \cup \Sigma_m$ be a ranked alphabet. Following structure encodes a tree t:
- $\underline{t} = (\mathrm{dom}_t, S_1^{\ t}, .., S_m^{\ t}, \leqslant^t, (P_a^t)_{a \in \Sigma})$

UNI FREIBURG

Monadic Second Order Logic

over trees

- How can we describe the properties of trees in terms of MSOL-formulae?
- Let $\Sigma = \Sigma_0 \cup .. \cup \Sigma_m$ be a ranked alphabet. Following structure encodes a tree t:
- $\underline{t} = (\mathrm{dom}_t, S_1{}^t, .., S_m{}^t, \leqslant^t, (P_a^t)_{a \in \Sigma})$
  - dom_t domain of t (i.e. set of all positions in t)
  - S_i^t the i-th successor relation on the domain
  - $\leqslant^t$ prefix relation (between two positions in t that are on the same path)
  - P_a^t set of all positions of t labeled with an *a*

Monadic Second Order Logic

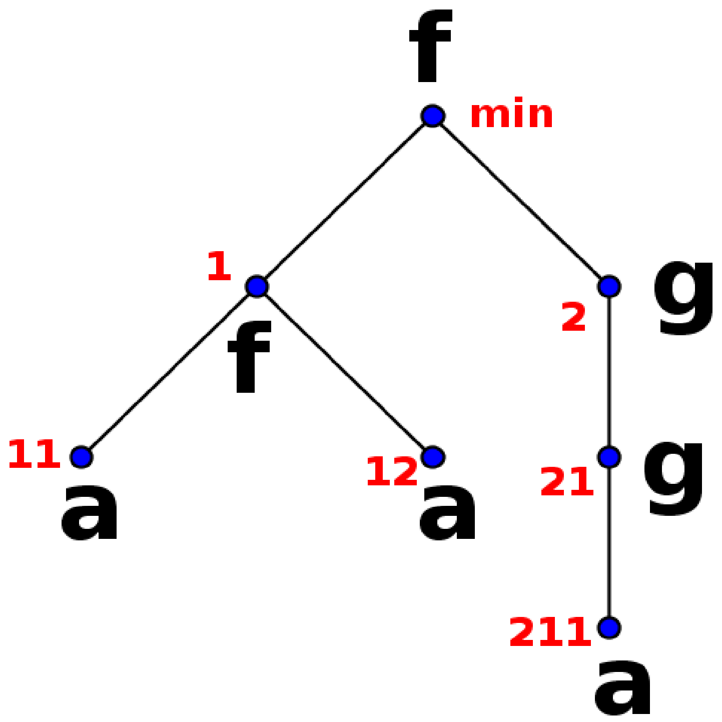
Example

- Let $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$ where $\Sigma_0 = \{a\}, \Sigma_1 = \{g\}, \Sigma_2 = \{f\}$
- $\underline{t} = (dom_t, S_1{}^t, .., S_m{}^t, \leqslant^t, (P_a^t)_{a \in \Sigma})$

Monadic Second Order Logic

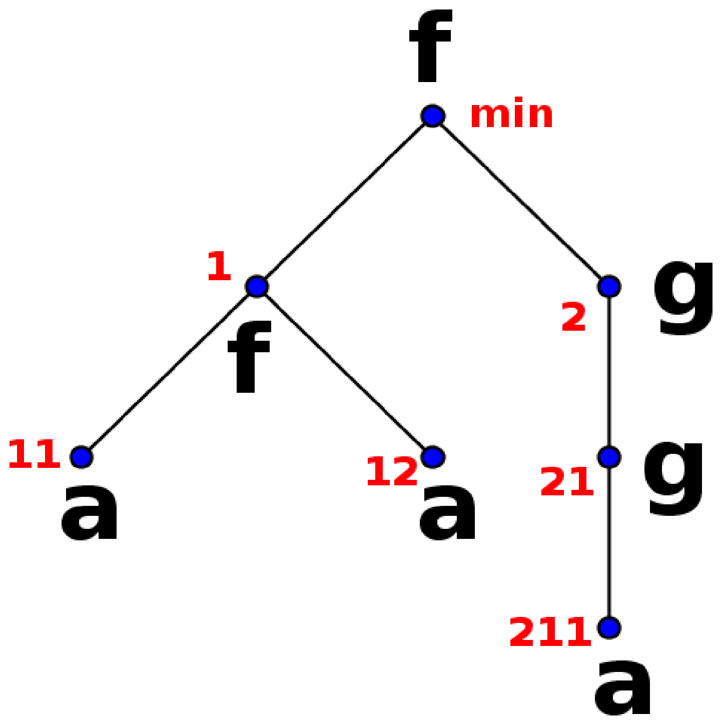
Example

- Let $\Sigma=\Sigma_0\cup\Sigma_1\cup\Sigma_2$ where $\Sigma_0=\{a\},\Sigma_1=\{g\},\Sigma_2=\{f\}$
- $\underline{t}=(\mathrm{dom}_t, S_1{}^t,..,S_m{}^t, \leqslant^t,(P_a^t)_{a\in\Sigma})$



Monadic Second Order Logic
Example

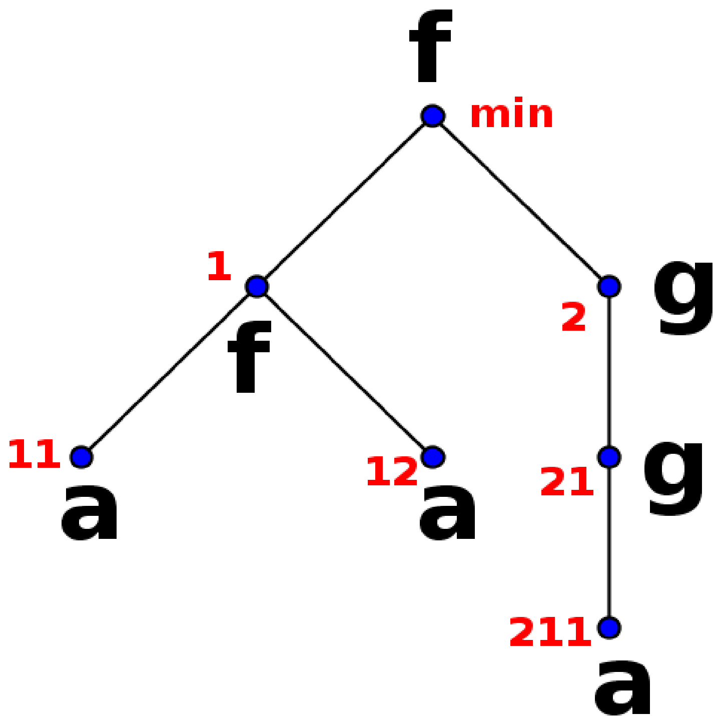
- Let $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$ where $\Sigma_0 = \{a\}, \Sigma_1 = \{g\}, \Sigma_2 = \{f\}$
- $\underline{t} = (dom_t, S_1^{\,t}, .., S_m^{\,t}, \leqslant^t, (P_a^t)_{a \in \Sigma})$
- $dom_t = \{min, 1, 11, 12, 2, 21, 211\}$



Monadic Second Order Logic
Example

- Let $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$ where $\Sigma_0 = \{a\}, \Sigma_1 = \{g\}, \Sigma_2 = \{f\}$
- $\underline{t} = (dom_t, S_1{}^t, .., S_m{}^t, \leqslant^t, (P^t_a)_{a \in \Sigma})$
- $dom_t = \{min, 1, 11, 12, 2, 21, 211\}$

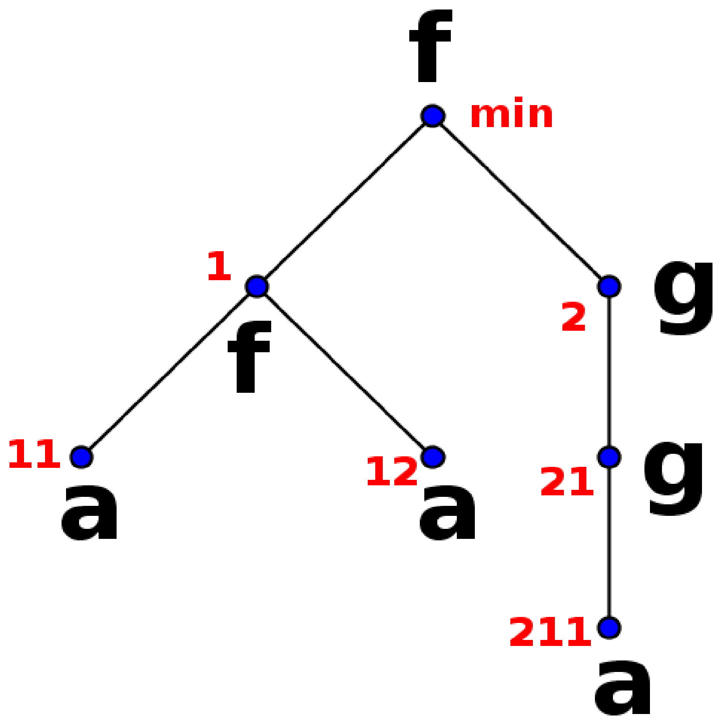
- S_1^t, S_2^t
  - $S_2^t(min, 2), S_1^t(2, 21)$



Monadic Second Order Logic

Example

- Let $\Sigma=\Sigma_0 \cup \Sigma_1 \cup \Sigma_2$ where $\Sigma_0=\{a\}, \Sigma_1=\{g\}, \Sigma_2=\{f\}$
- $\underline{t}=(dom_t, S_1{}^t, .., S_m{}^t, \leqslant^t, (P_a^t)_{a\in\Sigma})$
- $dom_t=\{min, 1, 11, 12, 2, 21, 211\}$



- S_1^t, S_2^t
  - $S_2^t(min, 2), S_1^t(2, 21)$
- $P_a=\{11, 12, 211\}, P_f=\{min, 1\}$
  $P_g=\{2, 21\}$

Monadic Second Order Logic

- Given a sentence ϕ in MSO-L, the expression
- $(\mathrm{dom}_t, S_1{}^t, .., S_m{}^t, \leqslant^t, (P_a^t)_{a \in \Sigma}) \models \phi$
- states that $\underline{t}$ satisfies ϕ if there is an automaton A, which accepts t.

- Tree languages
  - ϕ defines $T(\phi) := \{ t \in T_\Sigma \mid \underline{t} \models \phi \}$
  - $T(\phi)$ is called MSO-definable

Equivalence

between tree automata and MSOL

- Theorem(Doner, Thatcher-Wright, 1968):

  A tree language is recognizable by a finite tree automaton iff it is MSO-definable.

Equivalence

between tree automata and MSOL

- Theorem(Doner, Thatcher-Wright, 1968):

  A tree language is recognizable by a finite tree automaton iff it is MSO-definable.

- Proof: Direction from tree automata to MSOL
- Given automaton A, specify a formula such that:
- $t \in L(A) \Leftrightarrow \underline{t} \vDash \phi$

Equivalence

between tree automata and MSOL

- Observation 1: If states of A are $\{q_{1,}\dots,q_n\}$ then every run of of A on a tree t can be represented by sets of nodes $Q_1,\dots,Q_n$

Equivalence

between tree automata and MSOL

- Observation 1: If states of A are $\{q_{1,}, \ldots, q_n\}$ then every run of of A on a tree t can be represented by sets of nodes $Q_1, \ldots, Q_n$
- Observation 2: We can define that $Q_1, \ldots, Q_n$ represent an accepting run

Equivalence

between tree automata and MSOL

- Observation 1: If states of A are $\{q_{1,},\ldots,q_n\}$ then every run of of A on a tree t can be represented by sets of nodes $Q_1,\ldots,Q_n$
- Observation 2: We can define that $Q_1,\ldots,Q_n$ represent an accepting run
  - every node is labeled with at most one state

$$\phi_1 := \bigwedge_{i \neq j} \forall x \left(Q_i(x) \rightarrow \neg Q_j(x) \right)$$

Equivalence

between tree automata and MSOL

- Observation 1: If states of A are $\{q_{1,} \dots , q_n\}$ then every run of of A on a tree t can be represented by sets of nodes $Q_1, \dots , Q_n$
- Observation 2: We can define that $Q_1, \dots , Q_n$ represent an accepting run
  - every node is labeled with at most one state

    $$\phi_1 := \bigwedge_{i \neq j} \forall x \left(Q_i(x) \rightarrow \neg Q_j(x) \right)$$

  - root node is labeled with an accepting state

    $$\phi_2 := \bigvee_{q_i \in F} Q_i(\min)$$

UNI FREIBURG

Equivalence

between tree automata and MSOL

- Observation 2: We can define that $Q_1, \ldots, Q_n$ represents an accepting run
  - Leaf nodes are labeled with a state according to the rules

$$\phi_3 := \bigwedge_{a \in \Sigma_0} \forall x \left(P_a(x) \rightarrow \bigvee_{a \rightarrow q_i \in \Delta} Q_i(x) \right)$$

Equivalence
between tree automata and MSOL

- Observation 2: We can define that $Q_1,\ldots,Q_n$ represents an accepting run
  - Leaf nodes are labeled with a state according to the rules

$$\phi_3 := \bigwedge_{a \in \Sigma_0} \forall x \left(P_a(x) \rightarrow \bigvee_{a \rightarrow q_i \in \Delta} Q_i(x) \right)$$

  - Inner nodes are labeled as follows:

$$\phi_4 := \bigwedge_{a \in \Sigma_r} \forall x \forall y_1 \ldots \forall y_n$$

$$\left(P_a(x) \wedge S_r(x, y_1) \wedge \ldots \wedge S_r(x, y_n) \wedge y_1 < y_2 < .. y_{n-1} < y_n \right)$$

$$\rightarrow \bigvee_{a(q_{i_1}, .., q_{i_n}) \rightarrow q_i \in \Delta} \left(Q_{i_1}(y_1) \wedge .. \wedge Q_{i_n}(y_n) \wedge Q_i(x) \right)$$

Equivalence

between tree automata and MSOL

- Observation 3: In MSO we can guess $Q_1, \ldots, Q_n$
- $\phi := \exists Q_1 .. \exists Q_n \phi_1 \wedge \phi_2 \wedge \phi_3 \wedge \phi_4$

Equivalence

between tree automata and MSOL

- Observation 3: In MSO we can guess $Q_1, \ldots, Q_n$
- $\phi := \exists Q_1 .. \exists Q_n \phi_1 \wedge \phi_2 \wedge \phi_3 \wedge \phi_4$
- Then A accepts t iff $\underline{t} \models \phi$
- It is clear, that $L(A) = L(\phi)$
- Hence, every finite tree language is MSO-definable.

Equivalence

between tree automata and MSOL

- Proof: Direction from formulae to tree automata

Equivalence

between tree automata and MSOL

- Proof: Direction from formulae to tree automata
  - Induction over construction of MSO-L formulae
  - Use closure properties of tree automata
- If a tree language is MSO-definable, then it is recognizable by a tree automaton A.

Summary

- Tree automata
  - Basics of tree automata
  - Bottom-up tree automata
  - Top-down tree automata
  - Decision problems & complexity
- Connection to logic
  - Monadic Second Order Logic (MSOL)
  - Equivalence between tree automata and MSOL

References

- M.Dauchet, H.Comon,..
  Tree Automata Techniques and Applications
  (TATA), chapter 1, 2008
- Prof. Dr. W.Thomas, RWTH Aachen
  Applied Automata Theory, chapter 3, 2005
- Wim Martens, Stijn Vansummeren
  Automata and Logic on Trees
  University of Dortmund