Tree Automata (UMC 205)

Himesh(23775), Kuldeep(23684), Shobhnik(23697)

5th April 2025

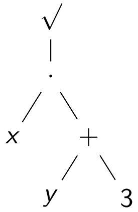
Outline

Outline

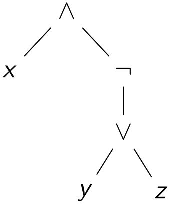
From Strings to Trees

- Most automata models work on strings over finite alphabets
- But many problems have inputs more structured than strings
- Trees arise in a variety of contexts:
  - Arithmetic expressions
  - Logical formulas
  - Parse trees of grammars
  - HTML documents
- Tree automata provide a natural way to process such structured inputs

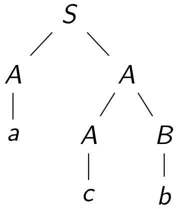
Examples of Tree Structures



Expression $\sqrt{x \cdot (y+3)}$



Formula $x \wedge \neg(y \vee z)$



Parse tree of grammar

$S \to AA$

$A \to a|c|AB$

$B \to b$

derivation tree for "acb"

Outline

Tree Domains and Labeled Trees

Definition

An n-ary tree domain, dom_t, is a prefix closed subset of $\{0, 1, 2, \ldots, n-1\}^*$ such that if $ui \in \mathrm{dom}_t$ then $uj \in \mathrm{dom}_t$ for every $j < i$.

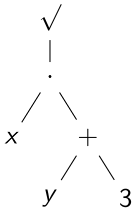
Definition

A Γ-labeled n-ary tree is a pair $t = (\mathrm{dom}_t, \mathrm{val}_t)$, where dom_t is an n-ary tree domain and $\mathrm{val}_t : \mathrm{dom}_t \to \Gamma$ is a labeling function.

Example

For the tree representing $\sqrt{x \cdot (y+3)}$:

- Root is named ε (labeled $\sqrt{}$)
- Its child is 0 (labeled $\cdot$)
- Other vertices are 00 (labeled x), 01 (labeled $+$)
- And 010 (labeled y), 011 (labeled 3)



Expression $\sqrt{x \cdot (y+3)}$

Tree Operations

Building Trees

Given Γ-labeled trees $t_0 = (\text{dom}_{t_0}, \text{val}_{t_0})$ and $t_1 = (\text{dom}_{t_1}, \text{val}_{t_1})$, and $A \in \Gamma$, the tree $A(t_0, t_1)$ is given by $t = (\text{dom}_t, \text{val}_t)$ where:

$$\text{dom}_t = \{\varepsilon\} \cup \{0u | u \in \text{dom}_{t_0}\} \cup \{1u | u \in \text{dom}_{t_1}\}$$

$$\text{val}_t(u) = \begin{cases} A & \text{if } u = \varepsilon \\ \text{val}_{t_0}(v) & \text{if } u = 0v \\ \text{val}_{t_1}(v) & \text{if } u = 1v \end{cases}$$

Tree Operations

Building Trees

Given Γ-labeled trees $t_0 = (\mathrm{dom}_{t_0}, \mathrm{val}_{t_0})$ and $t_1 = (\mathrm{dom}_{t_1}, \mathrm{val}_{t_1})$, and $A \in \Gamma$, the tree $A(t_0, t_1)$ is given by $t = (\mathrm{dom}_t, \mathrm{val}_t)$ where:

$$\mathrm{dom}_t = \{\varepsilon\} \cup \{0u | u \in \mathrm{dom}_{t_0}\} \cup \{1u | u \in \mathrm{dom}_{t_1}\}$$

$$\mathrm{val}_t(u) = \begin{cases} A & \text{if } u = \varepsilon \\ \mathrm{val}_{t_0}(v) & \text{if } u = 0v \\ \mathrm{val}_{t_1}(v) & \text{if } u = 1v \end{cases}$$

Subtrees

Given a Γ-labeled tree $t = (\mathrm{dom}_t, \mathrm{val}_t)$ and vertex/position $p \in \mathrm{dom}_t$, subtree rooted at position p is the tree $t|_p = (\mathrm{dom}_{t|_p}, \mathrm{val}_{t|_p})$ given by:

$$\mathrm{dom}_{t|_p} = \{u | pu \in \mathrm{dom}_t\}$$

$$\mathrm{val}_{t|_p}(u) = \mathrm{val}_t(pu)$$

Outline

Deterministic Tree Automata (DTA)

Definition

A deterministic tree automaton (DTA) on Σ-labeled n-ary trees is $M = (Q, \Sigma, \delta, F)$ where:

- Q is a finite set of states
- $F \subseteq Q$ is a set of final/accepting states
- $\delta = \cup_{i=0}^{n} \delta_i$ is the transition function, where $\delta_i : Q^i \times \Sigma \to Q$

Deterministic Tree Automata (DTA)

Definition

A deterministic tree automaton (DTA) on Σ-labeled n-ary trees is $M = (Q, \Sigma, \delta, F)$ where:

- Q is a finite set of states
- $F \subseteq Q$ is a set of final/accepting states
- $\delta = \cup_{i=0}^{n} \delta_i$ is the transition function, where $\delta_i : Q^i \times \Sigma \to Q$

Run of a DTA

The run of a DTA $M = (Q, \Sigma, \delta, F)$ on a tree $t = (\mathrm{dom}_t, \mathrm{val}_t)$ is a Q-labeled tree $\rho = (\mathrm{dom}_\rho, \mathrm{val}_\rho)$ where $\mathrm{dom}_\rho = \mathrm{dom}_t$ and for any vertex $u \in \mathrm{dom}_t$ with i children:

$$\mathrm{val}_\rho(u) = \delta_i(\mathrm{val}_\rho(u0), \ldots \mathrm{val}_\rho(u(i-1)), \mathrm{val}_t(u))$$

Acceptance and Language

Acceptance

A run $\rho = (\mathrm{dom}_\rho, \mathrm{val}_\rho)$ of M on t is accepting if $\mathrm{val}_\rho(\varepsilon) \in F$.
A tree t is accepted by M if M has an accepting run on t.

Acceptance and Language

Acceptance

A run $\rho = (\mathrm{dom}_\rho, \mathrm{val}_\rho)$ of M on t is accepting if $\mathrm{val}_\rho(\varepsilon) \in F$.
A tree t is accepted by M if M has an accepting run on t.

Language

The language recognized by M is the set of all Σ-labeled n-ary trees it accepts:

$$L(M) = \{t | M \text{ accepts } t\}$$

Acceptance and Language

Acceptance

A run $\rho = (\text{dom}_\rho, \text{val}_\rho)$ of M on t is accepting if $\text{val}_\rho(\varepsilon) \in F$.
A tree t is accepted by M if M has an accepting run on t.

Language

The language recognized by M is the set of all Σ-labeled n-ary trees it accepts:

$$L(M) = \{t | M \text{ accepts } t\}$$

Definition

A set A of Σ-labeled n-ary trees is regular if there is a DTA M such that $A = L(M)$.

Example: Boolean Expression Evaluator

DTA for Boolean Expressions

Let $\Sigma = \{0, 1, \neg, \wedge, \vee\}$. Consider the DTA $M_p = (\{q_0, q_1, q_r\}, \Sigma, \delta, \{q_1\})$ where:

$$\delta_0(0) = q_0 \qquad \delta_0(1) = q_1$$

$$\delta_1(q_0, \neg) = q_1 \qquad \delta_1(q_1, \neg) = q_0$$

For $\wedge$:

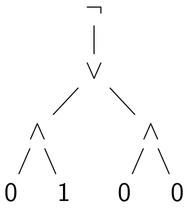
$$\delta_2(q_i, q_j, \wedge) = \begin{cases} q_1 & \text{if } q_i = q_j = q_1 \\ q_0 & \text{if } \{q_i, q_j\} \cap \{q_r\} = \emptyset \text{ and } \{q_i, q_j\} \cap \{q_0\} \neq \emptyset \end{cases}$$

For $\vee$:

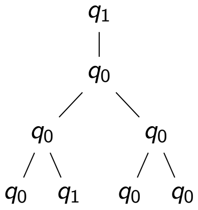
$$\delta_2(q_i, q_j, \vee) = \begin{cases} q_0 & \text{if } q_i = q_j = q_0 \\ q_1 & \text{if } \{q_i, q_j\} \cap \{q_r\} = \emptyset \text{ and } \{q_i, q_j\} \cap \{q_1\} \neq \emptyset \end{cases}$$

For all other cases, M_p transitions to state q_r.

Example: Boolean Expression Evaluator



Input tree $\neg((0 \wedge 1) \vee (0 \wedge 0))$



Run of DTA M_p on the input

- Since the label of the root in the run is $q_1 \in F$, this input is accepted
- $L(M_p)$ = set of syntactically correct boolean expressions that evaluate to true

Example: Arithmetic Modulo 3

DTA for Arithmetic Expressions

For $\Sigma = \{0, 1, 2, +, \cdot\}$. Consider the DTA $M_a = (\{q_0, q_1, q_2, q_r\}, \Sigma, \delta, \{q_0\})$ where:

$$\delta_0(0) = q_0 \qquad \delta_0(1) = q_1 \qquad \delta(2) = q_2$$

$$\delta_2(q_i, q_j, +) = q_{(i+j) \bmod 3} \qquad \text{if } \{q_i, q_j\} \cap \{q_r\} = \emptyset$$

$$\delta_2(q_i, q_j, \cdot) = q_{(i \cdot j) \bmod 3} \qquad \text{if } \{q_i, q_j\} \cap \{q_r\} = \emptyset$$

In all other cases not considered above, δ returns q_r.

- Trees over Σ represent arithmetic expressions
- If they are syntactically correct, M_a's run will have root labeled q_i, where i is the remainder when the value of the expression is divided by 3
- Since the final state is q_0, the automaton accepts expressions that evaluate to multiples of 3

Outline

Nondeterministic Tree Automata

Definition

A nondeterministic tree automaton (NTA) on Σ-labeled n-ary trees is $M = (Q, \Sigma, \delta, F)$ where Q is a finite set of states, $F \subseteq Q$ is a set of final/accepting states, and $\delta = \cup_{i=0}^{n} \delta_i$ is the transition function, where $\delta_i : Q^i \times \Sigma \to 2^Q$.

Nondeterministic Tree Automata

Definition

A nondeterministic tree automaton (NTA) on Σ-labeled n-ary trees is $M = (Q, \Sigma, \delta, F)$ where Q is a finite set of states, $F \subseteq Q$ is a set of final/accepting states, and $\delta = \cup_{i=0}^{n} \delta_i$ is the transition function, where $\delta_i : Q^i \times \Sigma \rightarrow 2^Q$.

Theorem

Let A be a tree language recognized by an NTA. Then A is regular.

Nondeterministic Tree Automata

Definition

A nondeterministic tree automaton (NTA) on Σ-labeled n-ary trees is $M = (Q, \Sigma, \delta, F)$ where Q is a finite set of states, $F \subseteq Q$ is a set of final/accepting states, and $\delta = \cup_{i=0}^{n} \delta_i$ is the transition function, where $\delta_i : Q^i \times \Sigma \to 2^Q$.

Theorem

Let A be a tree language recognized by an NTA. Then A is regular.

Proof Sketch.

The standard subset construction extends to tree automata:

- Convert NTA $N = (Q, \Sigma, \delta, F)$ to DTA $D = (2^Q, \Sigma, \delta', F')$
- $F' = \{P \subseteq Q \mid P \cap F \neq \emptyset\}$
- $\delta'_0 = \delta_0$ and $\delta'_k(Q_1, Q_2, \ldots, Q_k, f) = \{q \mid \exists q_1, q_2, \ldots, q_k . q_i \in Q_i \text{ and } q \in \delta_k(q_1, q_2, \ldots, q_k, f)\}$

$\square$

Outline

Claim

Every finite set of trees is regular

Claim

Every finite set of trees is regular

Proof.

Let $T = \{t_1, t_2 ..., t_n\}$, be the set of finite number of trees

- **States**:
  - Let $S = \{\text{all subtrees of } t_i \in T\}$ (finite set)
  - $Q = S \cup \{q_{\text{sink}}\}$
- **Accepting States**:
  - $F = \{t_1, t_2, \dots, t_n\} \subseteq Q$
- **Transition Function** δ:
  - For node 'a' with children in states $s_1, \dots, s_k$:

$$\delta(s_1, \dots, s_k, a) = \begin{cases} s & \text{if } \exists s \in S \text{ matches subtree rooted at '}a\text{'} \\ q_{\text{sink}} & \text{otherwise} \end{cases}$$

$\square$

DTA for Parse Trees

Claim

The set of derivation trees/parse trees of a context-free grammar $G = (N, T, P, S)$ is regular.

DTA for Parse Trees

Claim

The set of derivation trees/parse trees of a context-free grammar $G = (N, T, P, S)$ is regular.

Proof.

Take $\Sigma = N \cup T$. The automaton recognizing the parse trees is:

- **States:** $Q = T \cup N \cup \{*\}$
- **Final States:** $F = \{S\}$
- **Transitions:**
  - For $i = 0$: $\delta_0(a) = a$ where $a \in T$
  - For $i > 0$: $\delta_i(\alpha_1, \ldots, \alpha_i, X) = \begin{cases} X & \text{if } X \to \alpha_1 \cdots \alpha_i \in P \\ * & \text{otherwise} \end{cases}$

$\square$

Non Regularity

Can you think of a tree language that is not regular ?

Non Regularity

Can you think of a tree language that is not regular ?

Example Non-Regular Tree Languages

$$L = \{A(t, t) \mid t \in T(\Sigma)\}$$

where $T(\Sigma)$ is the collection of all full binary trees with leaves labeled by b and internal vertices labeled by A.

Proof by contradiction: If L is recognized by a DTA with finitely many states, then we can find two distinct trees $t_1 \neq t_2$ such that the DTA reaches the same state after reading either tree, which implies $A(t_1, t_2) \in L$, a contradiction.

Closed under Union

Union

Let L_1 and L_2 be two recognizable tree languages and $T_1 = (Q_1, \delta_1, F_1)$ and $T_2 = (Q_2, \delta_2, F_2)$ be the corresponding tree automata. Then $L_1 \cup L_2$ is also a recognizable tree language and is accepted by the automaton $T_1 \times T_2$

Closed under Union

Union

Let L_1 and L_2 be two recognizable tree languages and $T_1 = (Q_1, \delta_1, F_1)$ and $T_2 = (Q_2, \delta_2, F_2)$ be the corresponding tree automata. Then $L_1 \cup L_2$ is also a recognizable tree language and is accepted by the automaton $T_1 \times T_2$

Construction

- $Q = Q_1 \times Q_2$
- $\delta = \delta_1 \times \delta_2$ i.e $\delta(t) : (Q_1 \times Q_2)^k \to Q_1 \times Q_2$ for $t \in \Sigma_k$
- $F = (F_1 \times Q_2) \cup (F_2 \times Q_1)$

Closed under Complement

Complement

Let L be a recognizable tree language and $T = (Q, \delta, F)$ be the corresponding automaton over the alphabet Σ. Then the complement of L is also a recognizable tree language and is accepted by the automaton $T' = (Q, \delta, Q \backslash F)$

Closed under Intersection

Intersection

Let L_1 and L_2 be two recognizable tree languages and $T_1 = (Q_1, \delta_1, F_1)$ and $T_2 = (Q_2, \delta_2, F_2)$ be the corresponding automata. Then $L_1 \cap L_2$ is also a recognizable tree language and is accepted by the automaton $T_1 \cap T_2$

Closed under Intersection

Intersection

Let L_1 and L_2 be two recognizable tree languages and $T_1 = (Q_1, \delta_1, F_1)$ and $T_2 = (Q_2, \delta_2, F_2)$ be the corresponding automata. Then $L_1 \cap L_2$ is also a recognizable tree language and is accepted by the automaton $T_1 \cap T_2$

Construction

- $Q = Q_1 \times Q_2$
- $\delta = \delta_1 \times \delta_2$
- $F = F_1 \times F_2$

Outline

Decision Problems

Theorem

Given an NTA $M = (Q, \Sigma, \delta, F)$ there is a algorithm to determine if $L(M) = \emptyset$.

Decision Problems

Theorem

Given an NTA $M = (Q, \Sigma, \delta, F)$ there is a algorithm to determine if $L(M) = \emptyset$.

Proof Sketch.

Inductively compute the set of states that can be reached on some input:

$$E_0 = \{q \mid \exists a \in \Sigma \ \text{ and } \ q \in \delta_0(a)\}$$

$$E_i = E_{i-1} \cup \bigcup_k \{q \mid \exists a \in \Sigma, \exists\ q_0, \ldots q_{k-1} \in E_{i-1}, q \in \delta_k(q_0, q_1, \ldots q_{k-1}, a)\}$$

Since M has finitely many states, for some ℓ (at most $|Q|$), $E_\ell = E_{\ell+1} = \{q \mid \exists t.q \text{ is reached on } t\}$. Therefore, $L(M) = \emptyset$ iff $E_\ell \cap F = \emptyset$. $\square$

Decision Problems

Theorem

Given an NTA $M = (Q, \Sigma, \delta, F)$ there is a algorithm to determine if $L(M) = \emptyset$.

Proof Sketch.

Inductively compute the set of states that can be reached on some input:

$$E_0 = \{q \mid \exists a \in \Sigma \text{ and } q \in \delta_0(a)\}$$

$$E_i = E_{i-1} \cup \bigcup_k \{q \mid \exists a \in \Sigma, \exists\ q_0, \ldots q_{k-1} \in E_{i-1}, q \in \delta_k(q_0, q_1, \ldots q_{k-1}, a)\}$$

Since M has finitely many states, for some ℓ (at most $|Q|$), $E_\ell = E_{\ell+1} = \{q \mid \exists t.q \text{ is reached on } t\}$. Therefore, $L(M) = \emptyset$ iff $E_\ell \cap F = \emptyset$. □

Time Complexity : $O(|Q||\delta|)$

where $|\delta|$ is the number of transitions and $|Q|$ is the number of states.

Universality

Corollary

Given a DTA M, there is a polynomial time algorithm to check if $L(M)$ contains all Σ-labeled trees.

Outline

Applications: Two-Player Games

Game Setup

A game graph or arena is a $G = (Q_A, Q_B, E)$, where Q_A and Q_B are finite disjoint sets of vertices, and $E \subseteq (G_A \cup G_B) \times (G_A \cup G_B)$ is the edge relation with the property that every vertex has at least one outgoing edge.

Applications: Two-Player Games

Game Setup

A game graph or arena is a $G = (Q_A, Q_B, E)$, where Q_A and Q_B are finite disjoint sets of vertices, and $E \subseteq (G_A \cup G_B) \times (G_A \cup G_B)$ is the edge relation with the property that every vertex has at least one outgoing edge.

Game Rules

- Initially the "token" is placed in some vertex
- In each move, the token is moved along an adjacent edge
- Who makes a move is decided by whose vertex it is

Applications: Two-Player Games

Game Setup

A game graph or arena is a $G = (Q_A, Q_B, E)$, where Q_A and Q_B are finite disjoint sets of vertices, and $E \subseteq (G_A \cup G_B) \times (G_A \cup G_B)$ is the edge relation with the property that every vertex has at least one outgoing edge.

Game Rules

- Initially the "token" is placed in some vertex
- In each move, the token is moved along an adjacent edge
- Who makes a move is decided by whose vertex it is

Winnig Criteria

Reachability to any final state $q \in F$ where $F \subseteq Q_A \cup Q_B$

Applications: Two-Player Games

Game Setup

A game graph or arena is a $G = (Q_A, Q_B, E)$, where Q_A and Q_B are finite disjoint sets of vertices, and $E \subseteq (G_A \cup G_B) \times (G_A \cup G_B)$ is the edge relation with the property that every vertex has at least one outgoing edge.

Game Rules

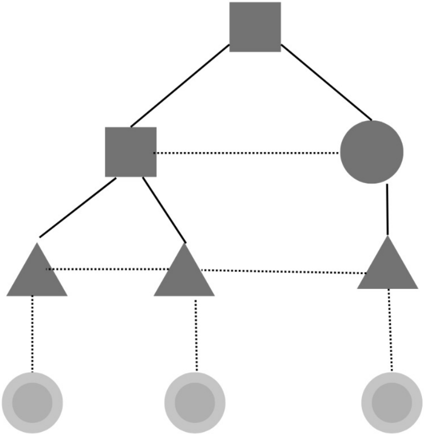
- Initially the "token" is placed in some vertex
- In each move, the token is moved along an adjacent edge
- Who makes a move is decided by whose vertex it is

Winnig Criteria

Reachability to any final state $q \in F$ where $F \subseteq Q_A \cup Q_B$

Can you figure out the initial states for which Bob has winning strategy?

Solution



Solution

Theorem. The collection of winning strategies for Bob from position q is a regular tree language. The tree automaton recognizing this language can be effectively constructed.

Solution

Theorem. The collection of winning strategies for Bob from position q is a regular tree language. The tree automaton recognizing this language can be effectively constructed.

Proof Sketch:

- `States:` Either game positions $\overline{Q}$ (copy of $Q_A \cup Q_B$) or error state $*$.
- `Transition Rules:`
  - $\delta_0(q) = \bar{q}$ if $q \in F$
  - $\delta_1(\bar{q_1}, q) = \bar{q}$ if $(q, q_1) \in E$ and $q \in Q_B$
  - $\delta_i(\overline{q_1}, \overline{q_2}, \ldots, \overline{q_i}, q) = \overline{q}$ if $q \in Q_A$ and $(q, q_1), \ldots, (q, q_i)$ are the only outgoing edges of q

`Formal Automaton Definition:`

$$M = \left(\overline{Q} \cup \{*\},\ Q_A \cup Q_B,\ \delta,\ \{q\}\right)$$

Solution

Theorem : It can be decided whether Bob has a winning strategy from position q. Moreover if there is a winning strategy, it can be effectively constructed.

Solution

Theorem : It can be decided whether Bob has a winning strategy from position q. Moreover if there is a winning strategy, it can be effectively constructed.

Proof : One can check the emptiness of the language associated with the DTA recognizing winning strategies from previous theorem.

Outline

Refrences

- Automata on Trees by Mahesh Viswanathan

Thank You!