Tree Automata

Kamal Lodaya

Department of Computer Science and Automation
Indian Institute of Science, Bangalore

7, 9 October 2019

Outline

Tree automata: Overview

- Generalize language recognition from strings to trees
- Analogous results lift from string signature to tree signature
- Automata can go bottom-up and top-down
- Satisfiability of monadic second-order logic reduces to nonemptiness of automata
- Help in going beyond regular string languages

Recognition via monoid morphisms

Let $\mathcal{A} = (Q, s, \delta)$ be a deterministic transition system over A

- For $w \in A^*$, let $h_w = \widehat{\delta}(\_, w)$
- h is a monoid morphism from $(A^*, ., \epsilon)$ to the transition monoid $M(\mathcal{A}) = (Q \rightarrow Q, \circ, id)$ of unary functions over the state set Q
- $L \subseteq A^*$ is recognized by morphism $h : A^* \rightarrow M$ if for some F: $L = h^{-1}(F)$
- $\mathcal{A} = (Q, s, \delta, F)$ is a deterministic automaton (DA) over A
- w accepted in $L(\mathcal{A})$ iff $h_w(s) \in F$
- Syntactic congruence $u \cong_L v$ iff $\forall x, y \in A^* : xuy \in L \iff xvy \in L$ matches the equality $h_u = h_v$ derived from the canonical DA

Recognition via monoid morphisms

Let $\mathcal{A} = (Q, s, \delta)$ be a deterministic transition system over A

- For $w \in A^*$, let $h_w = \widehat{\delta}(\_, w)$
- h is a monoid morphism from $(A^*, ., \epsilon)$ to the transition monoid $M(\mathcal{A}) = (Q \to Q, \circ, id)$ of unary functions over the state set Q
- $L \subseteq A^*$ is recognized by morphism $h : A^* \to M$ if for some F: $L = h^{-1}(F)$
- $\mathcal{A} = (Q, s, \delta, F)$ is a deterministic automaton (DA) over A
- w accepted in $L(\mathcal{A})$ iff $h_w(s) \in F$
- Syntactic congruence $u \cong_L v$ iff $\forall x, y \in A^* : xuy \in L \iff xvy \in L$ matches the equality $h_u = h_v$ derived from the canonical DA

Question (Doner 1970, Thatcher-Wright 1968)

Generalize functions to all arities?

Terms over a signature

Example: $\Sigma = (\Sigma_0, \Sigma_1)$ a unary signature of function symbols

- constant $\Sigma_0 = \{\epsilon\}$, unary symbols $\Sigma_1 = A$
- string $a_1 a_2 \ldots a_n \mapsto$ term $a_n(\ldots(a_2(a_1(\epsilon)))\ldots)$
- $\widehat{\delta} : T_\Sigma \to Q$ given by: $\widehat{\delta}(\epsilon) = s$ and $\widehat{\delta}(a(w)) = \delta(a)(\widehat{\delta}(w))$

Terms over a signature

Example: $\Sigma = (\Sigma_0, \Sigma_1)$ a unary signature of function symbols

- constant $\Sigma_0 = \{\epsilon\}$, unary symbols $\Sigma_1 = A$
- string $a_1 a_2 \dots a_n \mapsto$ term $a_n(\dots(a_2(a_1(\epsilon)))\dots)$
- $\widehat{\delta} : T_\Sigma \to Q$ given by: $\widehat{\delta}(\epsilon) = s$ and $\widehat{\delta}(a(w)) = \delta(a)(\widehat{\delta}(w))$

More generally: let $\Sigma = (\Sigma_0, \Sigma_1, \Sigma_2, \dots, \Sigma_m)$ be a finite signature

- Constants in Σ_0 are terms (Σ_0 required to be nonempty), if $t_1, \dots, t_n$ are terms and $f \in \Sigma_n$, $f(t_1, \dots, t_n)$ is a term
- $\delta(c) \in Q$ for $c \in \Sigma_0$, $\delta(f) : Q^n \to Q$ for $f \in \Sigma_n$
- $\widehat{\delta}(c) = \delta(c)$ and $\widehat{\delta}(f(t_1, \dots, f_n)) = \delta(f)(\widehat{\delta}(t_1), \dots, \widehat{\delta}(t_n))$

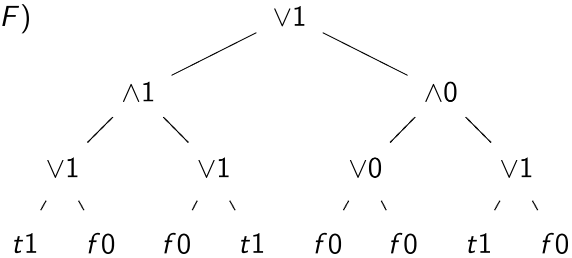
Algebra and automata

$\mathcal{Q} = (Q, \delta)$ is an algebra, accepting states make it a DTA

- $\delta : \Sigma_0 \to Q$, $\delta : \Sigma^n \to (Q^n \to Q)$ for $n > 0$
- $\widehat{\delta}(c) = \delta(c)$ and $\widehat{\delta}(f(t_1, \ldots, f_n)) = \delta(f)(\widehat{\delta}(t_1), \ldots, \widehat{\delta}(t_n))$
- $\mathcal{A} = (Q, \delta, F)$ a deterministic (bottom-up) tree automaton
- t accepted in $L(\mathcal{A})$ iff $\widehat{\delta}(t) \in F$
- Homomorphism $h(f(t_1, \ldots, t_n)) = f(h(t_1), \ldots, h(t_n))$

Tree language $L \subseteq T_\Sigma$ is recognized by $h : T_\Sigma \to Q$ (algebra $\mathcal{Q}$) if for some F: $L = h^{-1}(F)$

$\Sigma_0 = \{t, f\}$,
$\Sigma_1 = \{\neg\}$,
$\Sigma_2 = \{\wedge, \vee\}$,
$Q = \{0, 1\}$,
$F = \{1\}$



Exercises

Problem

Design automata for these tree languages:

1. *There are at least two a-labelled leaves,* $a \in \Sigma_0$
2. *The frontier has no b before an a,* $a, b \in \Sigma_0$
3. *There are an odd number of f-labelled nodes,* $f \in \Sigma_2$
4. *All the three conditions above are satisfied*

Nondeterministic automata, bottom-up and top-down

$\mathcal{A} = (Q, \Delta, F)$ a nondeterministic automaton (NTA), where $\Delta \subseteq \bigcup_{0 \leq i \leq m} (Q^i \times \Sigma_i \times Q)$,
$\mathcal{A} = (Q, \Delta, I)$ a nondeterministic top-down automaton (↓NTA), initial states $I \subseteq Q$ and $\Delta \subseteq \bigcup_{1 \leq i \leq m} (Q \times \Sigma_i \times Q^i) \cup (Q \times \Sigma_0)$.
The last are called final combinations. If there isn't one, a run gets stuck at a leaf and is not accepting.

Theorem

Every ↓NTA has an equivalent NTA, every NTA has an equivalent DTA, accepting the same language. Emptiness of the accepted language can be checked in polynomial time.

Nondeterministic automata, bottom-up and top-down

$\mathcal{A} = (Q, \Delta, F)$ a nondeterministic automaton (NTA), where $\Delta \subseteq \bigcup_{0 \leq i \leq m}(Q^i \times \Sigma_i \times Q)$,
$\mathcal{A} = (Q, \Delta, I)$ a nondeterministic top-down automaton (↓NTA), initial states $I \subseteq Q$ and $\Delta \subseteq \bigcup_{1 \leq i \leq m}(Q \times \Sigma_i \times Q^i) \cup (Q \times \Sigma_0)$. The last are called final combinations. If there isn't one, a run gets stuck at a leaf and is not accepting.

Theorem

Every ↓NTA has an equivalent NTA, every NTA has an equivalent DTA, accepting the same language. Emptiness of the accepted language can be checked in polynomial time.

Proof: Compute the states R that are reachable (respectively, from which runs can reach the leaves) at the roots of input trees. Check whether a final state (respectively, an initial state) is in R.

Deterministic automata, top-down

A deterministic top-down automaton ($\downarrow$DTA) $\mathcal{A} = (Q, \delta, s)$ has a single start state and transition function $\delta : \bigcup_{1 \leq i \leq m} \Sigma_i \to (Q \to Q^i)$ with final combinations $\delta \subseteq (Q \times \Sigma_0)$.

Theorem

The finite tree language $\{f(a, b), f(b, a)\}$ with $a, b \in \Sigma_0$ and $f \in \Sigma_2$ is not accepted by a $\downarrow$DTA

Proof: Construct a DTA accepting these terms. It will also accept $f(a, a)$ and $f(b, b)$.

Nonregular languages

Lemma (Pumping)

For a tree automaton with n states, if it accepts a tree of height $\geq n$, then there are two nodes on some path of the tree such that the tree can be decomposed $r[s[t]]$ at these nodes, where $r[\ ]$ and $s[\ ]$ are singular contexts and t is a tree, such that the trees $r[t]$, $r[s[s[t]]]$, $r[s[\ldots[s[t]]]]$ $\ldots$, where the context s is iterated $i \geq 0$ times, are all accepted by the automaton.

Corollary

The language of all full binary trees is not regular.

Proof: Suppose a tree automaton accepts a full binary tree of height n. Then it can be decomposed $r[s[t]]$ as in the lemma. By the lemma, $r[s[s[t]]]$ is accepted, but this is not a full binary tree.

Myhill-Nerode congruence

Tree language $L \subseteq T_\Sigma$ is recognized by $h : T_\Sigma \to Q$ (algebra $\mathcal{Q}$) if for some F: $L = h^{-1}(F)$

Kernel congruence $s \cong_h t$ if $h(s) = h(t)$

Syntactic congruence $s \cong_L t$ if for all singular contexts $r[\ ]$, $r[s] \in L \iff r[t] \in L$

Theorem (Büchi, Thatcher-Wright 1968)

Let $L \subseteq T_\Sigma$. Then the following are equivalent:

1. *L is a regular tree language*
2. *The syntactic congruence of L has finite index*
3. *L is recognized by a finite algebra*

Monadic second-order logic (MSO) on trees

Signature $\Sigma = (\Sigma_0, \ldots, \Sigma_m)$

Syntax $x = y,\ S_1(x, y), \ldots, S_m(x, y),\ x \leq y,\ P_a(x),\ Z(x)$, closed under boolean operations and first- and second-order quantifiers

Structure $\mathbf{t} = \begin{array}{l} (dom(t) \subseteq (1 \cup \cdots \cup m)^*, \_ \circ 1, \ldots, \_ \circ m, \\ (\bigcup_{i=1,m} \_ \circ i)^*, \{lab(\_) = a \mid a \in \Sigma\}) \end{array}$

Interpretation $\mathbf{t}, \bar{v}, \bar{V} \models \phi(\bar{x}, \bar{Z})$, defined inductively

Extended signature $\Sigma = (\Sigma_0 \cup \cdots \cup \Sigma_m) \times Var_1 \times Var_2$ to encode every first-order and every second-order variable at a position

Theorem (Doner 1970, Thatcher-Wright 1968)

$L \subseteq T_\Sigma$ is regular iff L is definable by an MSO sentence

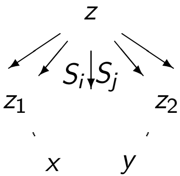
Corollary

Checking satisfiability of MSO formulae reduces to checking nonemptiness of tree automata

Example formulae

Signature $\Sigma = (\Sigma_0, \ldots, \Sigma_m)$

$leaf(x) = \neg\exists x(S_1(x,y) \vee \cdots \vee S_m(x,y))$



$$beforeS(z,z_1,z_2) = \bigvee_{i=1,m-1}(S_i(z,z_1) \wedge \bigvee_{j=i+1,m} S_j(z,z_2))$$

$$\begin{aligned} before(x,y) = \quad & x < y \vee \exists z \exists z_1 \exists z_2(\neg leaf(z) \wedge beforeS(z,z_1,z_2) \wedge \\ & z_1 \leq x \wedge \neg(z_1 \leq y) \wedge z_2 \leq y \wedge \neg(z_2 \leq x)))) \end{aligned}$$

$$first_f(x) = P_f(x) \wedge \neg\exists y(P_f(y) \wedge before(y,x))$$

$$last_f(x) = P_f(x) \wedge \neg\exists y(P_f(y) \wedge before(x,y))$$

$next_f(x,y) =$
$P_f(x) \wedge P_f(y) \wedge before(x,y) \wedge \neg\exists z(P_f(z) \wedge before(x,z) \wedge before(z,y))$

Example sentences

1. There are at least two a-labelled leaves, $a \in \Sigma_0$:
   $\exists x \exists y (x \neq y \wedge leaf(x) \wedge P_a(x) \wedge leaf(y) \wedge P_a(y))$
2. The frontier has no b before an a, $a, b \in \Sigma_0$:
   $\forall x (P_a(x) \wedge leaf(x) \Rightarrow \neg \exists y (P_b(y) \wedge leaf(y) \wedge before(x, y))$
3. There are an odd number of f-labelled nodes, $f \in \Sigma_2$:

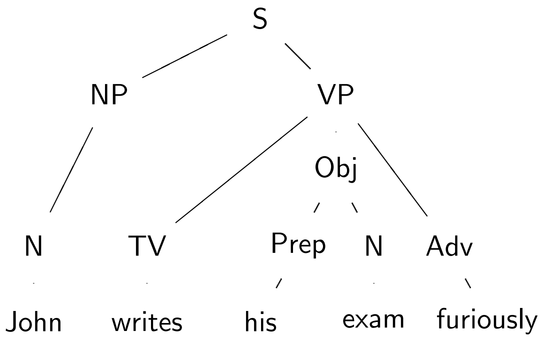
$$\begin{aligned}\exists Z_1 \exists Z_2 \forall x \forall y (\quad & (first_f(x) \Rightarrow Z_1(x)) \wedge \\ & (last_f(y) \Rightarrow Z_1(y)) \wedge \\ & (next_f(x, y) \Rightarrow (Z_1(x) \oplus Z_2(x))))\end{aligned}$$

Definition

$\phi_1 \oplus \phi_2 = (\phi_1 \vee \phi_2) \wedge \neg(\phi_1 \wedge \phi_2)$

Context-free languages

1. $S \to NP\ VP$
2. $NP \to [Adj]\ NP$
3. $VP \to IV\ [Adv] \mid TV\ Obj\ [Adv]$
4. $Obj \to [Prep]\ N$



$\delta(John) = N$
$\delta(writes) = TV$
$\delta(furiously) = Adv$
$\delta(P4.1)(Prep, N) = Obj$
$\delta(P3.2)(IV, Adv) = VP$
$\delta(P3.4)(TV, Obj, Adv) = VP$
$\Sigma_2 = \{S, NP, VP, Obj\},\ \Sigma_3 = \{VP\},\ \Sigma_0 = \{John, writes, \dots\}$

Theorem

All context-free languages are yields of tree automata

Intervals of a string on a virtual tree

- For simplicity, we think of every word as a letter of the alphabet: $((John)((writes)((his)(exam))(furiously))$
- Parentheses depict labelled intervals but they are not present on the string, interpreted as second-order variables
- The MSO-interpretation gives tree-MSO formulas on the imagined structure of intervals

$$Tran(P_a(x)) = leaf(x) \wedge P_a(x)$$
$$Tran(x < y) = leaf(x) \wedge leaf(y) \wedge before(x, y)$$
$$Int(Z) = \forall x \in Z : leaf(x) \wedge (\forall y, z \in Z : x < z \wedge z < x \Rightarrow z \in Z)$$
$$\begin{aligned} Tran(P_f(Z)) = Int(Z) \wedge \exists y(\quad & \neg leaf(y) \wedge P_f(y) \wedge \\ & \forall x(x \in Z \iff leaf(x) \wedge y < x)) \end{aligned}$$
$$Tran(bf(X, Y)) = Int(X) \wedge Int(Y) \wedge \forall x \in X : \forall y \in Y : before(x, y)$$
$$\begin{aligned} Tran(succ(Z, C)) = \quad & Int(Z) \wedge Int(C) \wedge (C \subseteq Z) \wedge \\ & \forall Y(Int(Y) \wedge (C \subseteq Y) \wedge (Y \subseteq Z) \Rightarrow (Y \subseteq C)) \end{aligned}$$
$$Tran(S_1(Z, FC)) = succ(Z, FC) \wedge \forall C(succ(Z, C) \Rightarrow bf(FC, C))$$

Tree transducers realize tree homomorphisms

Tree signatures Σ and Γ, variables $X_n = \{x_1, \ldots, x_n\}$

- $\delta : \Sigma_0 \to Q$, $Val : \Sigma_0 \to T_\Gamma$
- $\delta : \Sigma^n \to (Q^n \to Q)$, $Val : \Sigma^n \to T_\Gamma(X_n)$
- $\widehat{\delta}(f(t_1, \ldots, t_n)) = \delta(f)(\widehat{\delta}(t_1), \ldots, \widehat{\delta}(t_n))$ and $\widehat{Val}(f(t_1, \ldots, t_n)) = Val(f)(x_1, \ldots, x_n)$ where $x_i = \widehat{Val}(t_i)$
- $\mathcal{A} = (Q, \delta, Val, F)$ a deterministic bottom-up tree transducer from Σ to Γ
- Tree homomorphism $h(f(t_1, \ldots, t_n)) = h(f)(x_1, \ldots, x_n)$, where $x_i = h(t_i)$

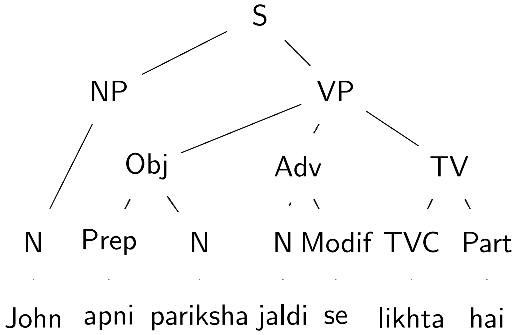
Tree transduction $T(\mathcal{A}) : T_\Sigma \to T_\Gamma = \{(t \mapsto \widehat{Val}(t)) \mid \widehat{\delta}(t) \in F\}$

Problem

String and tree transducers are a subject of current research

Translation as tree transduction

1. $S \to NP\ VP$
2. $NP \to [Adj]\ NP$
3. $VP \to IV\ [Adv] \mid TV\ Obj\ [Adv]$
4. $Obj \to [Prep]\ N$
5. $Adv \to N\ Modif$
6. $TV \to TVC\ Part$



$Val(exam) = pariksha$

$Val(P4.1) = Obj(x_1, x_2)$

$Val(P3.1)(x_1, x_2) = VP(x_2, x_1)$

$Val(P3.2)(TV, Obj, Adv) = VP(x_2, x_3, x_1)$

$Val(Adv) = Adv(N, Modif),\ \ Val(TV) = TV(TVC, Part)$

$\Gamma_2 = \{S, NP, VP, Obj, TV, Adv\},$

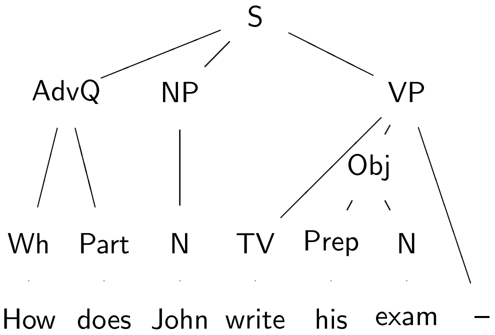
$\Gamma_3 = \{VP\},\ \ \Gamma_0 = \{John, apni, pariksha, likhta, \ldots\}$

Transformation of deep structure

$$S(NP, VP(TV, Obj, Adv)) \Longrightarrow S(AdvQ, NP, VP(TV, Obj, --))$$

1. $S \rightarrow AdvQ\ NP\ VP$
2. $NP \rightarrow [Adj]\ NP$
3. $VP \rightarrow IV \mid TV\ Obj$
4. $Obj \rightarrow [Prep]\ N$
5. $AdvQ \rightarrow Wh\ Part$

$\delta(How) = Wh$
$\delta(does) = Part$
$\delta(P4.1)(Prep, N) = Obj$



$\delta(P3.4)(IV, AdvTr) = VP$, $\delta(P3.2)(TV, Obj, AdvTr) = VP$

Problem

Transformational grammars using tree automata? MSO logic?