Advanced Logic

http://www-sop.inria.fr/members/Martin.Avanzini/teaching/2021/AL/

Martin Avanzini

Summer Semester 2021

Last Lecture

a reachability game is played by two players, players ◆ and ■

- ⋆ the game is played on a graph which determines the current player and her possible moves

$$\mathcal{G} = (V, V_{◆}, V_{■}, E, v_I, Z)$$

- ⋆ the objective of ◆ is to reach a goal Z or make ■ get stuck
- ⋆ the objective of ■ is to prevent this

Theorem

For every arena $\mathcal{G}$, either ◆ or ■ has a (positional) winning strategy.

Theorem

$w \in \mathrm{L}(\mathcal{A})$ *if and only if player ◆ has a winning strategy in* $\mathcal{G}_{\mathcal{A},w}$.

Today's Lecture

- ⋆ bottom-up tree automata
- ⋆ closure properties
- ⋆ top-down tree automata

Bottom-up Tree-Automata

Finite Ordered Trees

Definition

A finite ordered tree (or simply tree here) is a set of sequences of natural numbers $T \subseteq \mathbb{N}^*$ such that, for $w \in \mathbb{N}^*$ and $i \in \mathbb{N}$

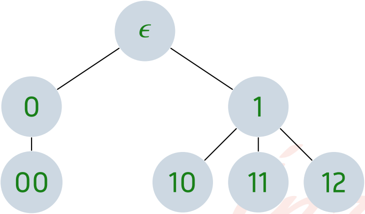
1. if $w \cdot (i+1) \in T$ then $w \cdot i \in T$
2. if $w \cdot 0 \in T$ then $w \in T$

Interpretation

- a tree T is given by the set of nodes
- a node is identified with its position, i.e., its path from the (unique) root ϵ
- the i-th child $(0 \leq i)$ of a node w is $w \cdot i$

Example

$$T = \{\epsilon, 0, 00, 1, 10, 11, 12\}$$



Σ-trees

- let Σ be an alphabet, equipped with an arity $\mathrm{ar} : \Sigma \to \mathbb{N}$
- a Σ-tree is a tuple (T, ℓ) such that
  - T is a tree
  - $\ell : T \to \Sigma$ is a labeling of nodes by letters
  - the labeling respects the arity in the following sense: for all $w \in T$,

$$\mathrm{ar}(\ell(w)) = n \quad \Longleftrightarrow \quad w \text{ has exactly } n \text{ children } w \cdot 0, \ldots, w \cdot (n-1) \in T$$

Σ-trees

- let Σ be an alphabet, equipped with an arity $\mathrm{ar} : \Sigma \to \mathbb{N}$
- a Σ-tree is a tuple (T, ℓ) such that
  - T is a tree
  - $\ell : T \to \Sigma$ is a labeling of nodes by letters
  - the labeling respects the arity in the following sense: for all $w \in T$,

$$\mathrm{ar}(\ell(w)) = n \quad \iff \quad w \text{ has exactly } n \text{ children } w \cdot 0, \ldots, w \cdot (n-1) \in T$$

- Note: Σ-trees can be seen as (ground) terms $t \in \mathcal{T}(\Sigma)$ over function symbols $\mathtt{f} \in \Sigma$

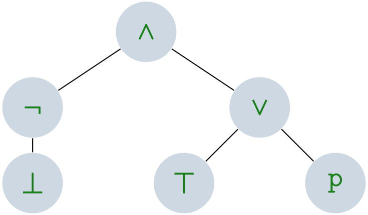
$$t ::= \mathtt{f}(t_1, \ldots, t_{\mathrm{ar}(\mathtt{f})})$$

Example

- consider the alphabet $\Sigma_{\mathbb{B}}$ consisting of usual Boolean connectives and propositional atoms $(\mathrm{p}, \mathrm{q}, \ldots)$

- the following labeled tree (T, t) denotes the propositional formula $\neg\bot \wedge (\top \vee \mathrm{p})$

$w \in T$	$\ell(w)$
ϵ	$\wedge$
0	$\neg$
1	$\vee$
00	$\bot$
10	$\top$
11	p



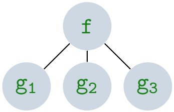
Bottom-Up Tree Automatas (BUTAs)

- A bottom-up tree automata (BUTA) $\mathcal{A}$ is a tuple (Q, Σ, δ, F) consisting of
  - a finite set of states Q
  - an alphabet Σ with associated arities
  - a transition function $\delta \triangleq \{\delta_{\mathtt{f}} \mid \mathtt{f} \in \Sigma\}$ where $\delta_{\mathtt{f}} : Q^{\mathrm{ar}(\mathtt{f})} \to 2^{Q}$
  - a set of final states $F \subseteq Q$

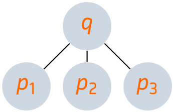
Bottom-Up Tree Automatas (BUTAs)

- A bottom-up tree automata (BUTA) $\mathcal{A}$ is a tuple (Q, Σ, δ, F) consisting of
  - a finite set of states Q
  - an alphabet Σ with associated arities
  - a transition function $\delta \triangleq \{\delta_{\mathtt{f}} \mid \mathtt{f} \in \Sigma\}$ where $\delta_{\mathtt{f}} : Q^{\mathrm{ar}(\mathtt{f})} \to 2^Q$
  - a set of final states $F \subseteq Q$
- An execution on a Σ-tree (T, ℓ) is a Q-tree (T, ℓ_Q) such that for all $w \in T$ with n children,

$$\ell_Q(w) \in \delta_{\ell(w)}(\ell_Q(w \cdot 0), \ldots, \ell_Q(w \cdot (n-1)))$$



fragment of Σ-tree (T, ℓ)

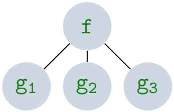


$\delta_{\mathtt{f}}(q) = (p_1, p_2, p_3)$

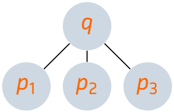
Bottom-Up Tree Automatas (BUTAs)

- A bottom-up tree automata (BUTA) $\mathcal{A}$ is a tuple (Q, Σ, δ, F) consisting of
  - a finite set of states Q
  - an alphabet Σ with associated arities
  - a transition function $\delta \triangleq \{\delta_{\mathtt{f}} \mid \mathtt{f} \in \Sigma\}$ where $\delta_{\mathtt{f}} : Q^{\mathrm{ar}(\mathtt{f})} \to 2^Q$
  - a set of final states $F \subseteq Q$
- An execution on a Σ-tree (T, ℓ) is a Q-tree (T, ℓ_Q) such that for all $w \in T$ with n children,

$$\ell_Q(w) \in \delta_{\ell(w)}(\ell_Q(w \cdot 0), \ldots, \ell_Q(w \cdot (n-1)))$$



fragment of Σ-tree (T, ℓ)



$\delta_{\mathtt{f}}(q) = (p_1, p_2, p_3)$

- the BUTA $\mathcal{A}$ recognises the tree-language

$$\mathrm{L}(\mathcal{A}) \triangleq \{(T, \ell) \mid (T, \ell) \text{ has an execution whose root is in } F\}$$

Example

- consider the BUTA $\mathcal{B} = (\{0,1\}, \Sigma_{\mathbb{B}}, \delta, \{1\})$ where

$$\begin{array}{lll} \delta_\top = \{1\} & \delta_{\mathrm{p}} = \{0,1\} & \delta_\wedge(b_1, b_2) = b_1 \cdot b_2 \\ \delta_\bot = \{0\} & \delta_\neg(b) = \{1 - b\} & \delta_\vee(b_1, b_2) = \{b_1 + b_2 - b_1 \cdot b_2\} \end{array}$$

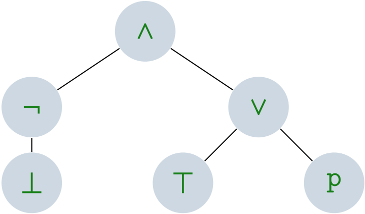
- it recognises the set of satisfiable formulas, for instance

Example

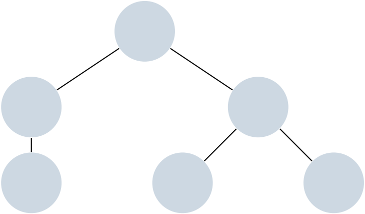
- consider the BUTA $\mathcal{B} = (\{0,1\}, \Sigma_{\mathbb{B}}, \delta, \{1\})$ where

$$\begin{aligned} \delta_{\top} &= \{1\} & \delta_{\mathtt{p}} &= \{0,1\} & \delta_{\wedge}(b_1,b_2) &= b_1 \cdot b_2 \\ \delta_{\perp} &= \{0\} & \delta_{\neg}(b) &= \{1-b\} & \delta_{\vee}(b_1,b_2) &= \{b_1 + b_2 - b_1 \cdot b_2\} \end{aligned}$$

- it recognises the set of satisfiable formulas, for instance



Input Tree (T, ℓ)



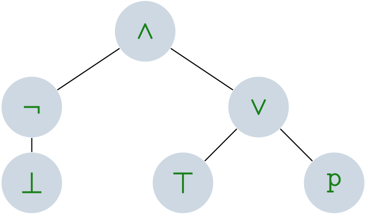
Execution on (T, ℓ)

Example

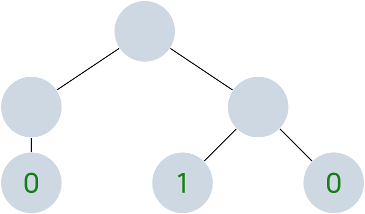
- consider the BUTA $\mathcal{B} = (\{0,1\}, \Sigma_{\mathbb{B}}, \delta, \{1\})$ where

$$\delta_{\top} = \{1\} \qquad \delta_{\mathtt{p}} = \{0,1\} \qquad \delta_{\wedge}(b_1, b_2) = b_1 \cdot b_2$$
$$\delta_{\perp} = \{0\} \qquad \delta_{\neg}(b) = \{1 - b\} \qquad \delta_{\vee}(b_1, b_2) = \{b_1 + b_2 - b_1 \cdot b_2\}$$

- it recognises the set of satisfiable formulas, for instance



Input Tree (T, ℓ)



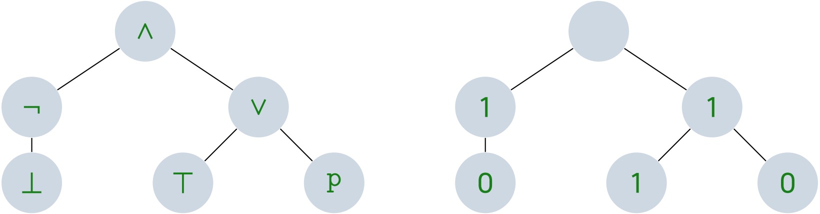
Execution on (T, ℓ)

Example

- consider the BUTA $\mathcal{B} = (\{0,1\}, \Sigma_{\mathbb{B}}, \delta, \{1\})$ where

$$\delta_{\top} = \{1\} \qquad \delta_{\mathtt{p}} = \{0,1\} \qquad \delta_{\wedge}(b_1, b_2) = b_1 \cdot b_2$$
$$\delta_{\perp} = \{0\} \qquad \delta_{\neg}(b) = \{1 - b\} \qquad \delta_{\vee}(b_1, b_2) = \{b_1 + b_2 - b_1 \cdot b_2\}$$

- it recognises the set of satisfiable formulas, for instance



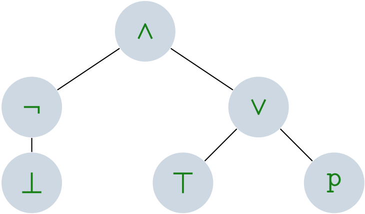
Input Tree (T, ℓ) Execution on (T, ℓ)

Example

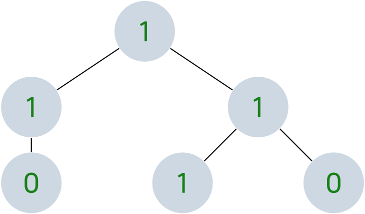
- consider the BUTA $\mathcal{B} = (\{0,1\}, \Sigma_{\mathbb{B}}, \delta, \{1\})$ where

$$
\begin{array}{lll}
\delta_{\top} = \{1\} & \delta_{\mathtt{p}} = \{0,1\} & \delta_{\wedge}(b_1, b_2) = b_1 \cdot b_2 \\
\delta_{\bot} = \{0\} & \delta_{\neg}(b) = \{1 - b\} & \delta_{\vee}(b_1, b_2) = \{b_1 + b_2 - b_1 \cdot b_2\}
\end{array}
$$

- it recognises the set of satisfiable formulas, for instance



Input Tree (T, ℓ)



Execution on (T, ℓ)

Tree Automata Seen as Rewrite Systems

- a transition $\delta_{\mathtt{f}}(q) = (q_1, \ldots, q_n)$ in BUTA $\mathcal{A}$ is seen as rule

$$\mathtt{f}(q_1, \ldots, q_n) \to_{\mathcal{A}} q$$

- an execution on a labeled tree, seen as term t, is a maximal reduction sequence

$$t \to_{\mathcal{A}} \cdots \to_{\mathcal{A}} q$$

- $t \in \mathrm{L}(\mathcal{A}) \iff t \to_{\mathcal{A}}^{*} q$ and $q \in F$

Tree Automata Seen as Rewrite Systems

- a transition $\delta_{\mathtt{f}}(q) = (q_1, \ldots, q_n)$ in BUTA $\mathcal{A}$ is seen as rule

$$\mathtt{f}(q_1, \ldots, q_n) \to_{\mathcal{A}} q$$

- an execution on a labeled tree, seen as term t, is a maximal reduction sequence

$$t \to_{\mathcal{A}} \cdots \to_{\mathcal{A}} q$$

- $t \in \mathrm{L}(\mathcal{A}) \iff t \to_{\mathcal{A}}^{*} q$ and $q \in F$

Example

- The BUTA $\mathcal{B}$ on $\Sigma_{\mathbb{B}}$ as defined before induces the rewrite sytem $\to_{\mathcal{B}}$

$$\begin{array}{ccc} \top \to_{\mathcal{B}} 1 & \mathtt{p} \to_{\mathcal{B}} b & b_1 \wedge b_2 \to_{\mathcal{B}} b_1 \cdot b_2 \\ \bot \to_{\mathcal{B}} 0 & \neg(b) \to_{\mathcal{B}} 1 - b & b_1 \vee b_2 \to_{\mathcal{B}} b_1 + b_2 - b_1 \cdot b_2 \end{array}$$

where b, b_1, b_2 ranges over $\{0, 1\}$

Tree Automata Seen as Rewrite Systems

- a transition $\delta_{\mathtt{f}}(q) = (q_1, \ldots, q_n)$ in BUTA $\mathcal{A}$ is seen as rule

$$\mathtt{f}(q_1, \ldots, q_n) \to_{\mathcal{A}} q$$

- an execution on a labeled tree, seen as term t, is a maximal reduction sequence

$$t \to_{\mathcal{A}} \cdots \to_{\mathcal{A}} q$$

- $t \in \mathrm{L}(\mathcal{A}) \iff t \to^*_{\mathcal{A}} q$ and $q \in F$

Example

- The BUTA $\mathcal{B}$ on $\Sigma_{\mathbb{B}}$ as defined before induces the rewrite sytem $\to_{\mathcal{B}}$

$$\begin{aligned}
\top &\to_{\mathcal{B}} 1 & \mathtt{p} &\to_{\mathcal{B}} b & b_1 \wedge b_2 &\to_{\mathcal{B}} b_1 \cdot b_2 \\
\bot &\to_{\mathcal{B}} 0 & \neg(b) &\to_{\mathcal{B}} 1 - b & b_1 \vee b_2 &\to_{\mathcal{B}} b_1 + b_2 - b_1 \cdot b_2
\end{aligned}$$

  where b, b_1, b_2 ranges over $\{0, 1\}$

- we have $t = \neg\bot \wedge (\top \vee \mathtt{p}) \in \mathrm{L}(\mathcal{A})$ as

$$t = \neg\underline{\bot} \wedge (\underline{\top} \vee \underline{\mathtt{p}}) \to^3_{\mathcal{B}} \underline{\neg 0} \wedge (\underline{1 \vee 0}) \to^2_{\mathcal{B}} 1 \wedge 1 \to_{\mathcal{B}} 1 \in F$$

Determinisation

A deterministic BUTA (DBUTA) is a BUTA $\mathcal{A} = (Q, \Sigma, \delta, F)$ where $\delta_{\mathtt{f}} : Q^{\mathsf{ar}(\mathtt{f})} \to Q$ for all $\mathtt{f} \in \Sigma$.

Determinisation

A deterministic BUTA (DBUTA) is a BUTA $\mathcal{A} = (Q, \Sigma, \delta, F)$ where $\delta_{\mathtt{f}} : Q^{\mathrm{ar}(\mathtt{f})} \to Q$ for all $\mathtt{f} \in \Sigma$.

Theorem

For every BUTA $\mathcal{A}$ with n states there exists a DBUTA $\mathcal{B}$ with at most 2^n states such that $L(\mathcal{A}) = L(\mathcal{B})$.

Determinisation

A deterministic BUTA (DBUTA) is a BUTA $\mathcal{A} = (Q, \Sigma, \delta, F)$ where $\delta_{\mathtt{f}} : Q^{\mathrm{ar}(\mathtt{f})} \to Q$ for all $\mathtt{f} \in \Sigma$.

Theorem

For every BUTA $\mathcal{A}$ with n states there exists a DBUTA $\mathcal{B}$ with at most 2^n states such that $\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{B})$.

Proof Outline.

Let $\mathcal{A} = (Q, \Sigma, \delta, F)$.

The construction corresponds to the subset construction for determinisation of NFAs:

- the states of $\mathcal{B}$ are sets 2^Q of states Q
- the transition relation $\Delta_{\mathtt{f}}$ for $\mathtt{f} \in \Sigma$ of $\mathcal{B}$ is

$$\Delta_{\mathtt{f}}(M_1, \ldots, M_{\mathrm{ar}(\mathtt{f})}) \triangleq \bigcup \{\delta_{\mathtt{f}}(q_1, \ldots, q_{\mathrm{ar}(\mathtt{f})}) \mid q_1 \in M_1, \ldots, q_{\mathrm{ar}(\mathtt{f})} \in M_{\mathrm{ar}(\mathtt{f})}\}$$

- the final states of $\mathcal{B}$ are $\{M \mid M \cap F \neq \varnothing\}$

Closure Properties

Closure Properties of BUTAs

Theorem

The class of languages recognised by BUTAs is closed under the following operations:

1. *union, intersection, and complement*
2. *arity-preserving homomorphism*

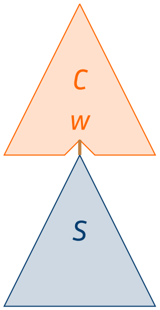
- ⋆ a function $h : \Sigma \to \Gamma$ is arity-preserving if $\mathrm{ar}_\Sigma(\mathtt{f}) = \mathrm{ar}_\Gamma(h(\mathtt{f}))$
- ⋆ the homomorphic application of such a function to a labeled tree $t = (T, \ell)$ is given by $h(t) \triangleq (T, \ell_h)$ where

$$\ell_h(w) = h(\ell(w)) \quad \text{for all } w \in T$$

i.e., $h(t)$ is obtained by re-relabeling letters $\mathtt{f}$ in t with $h(\mathtt{f})$

Pumping Lemma for Tree Automata

- A context is a tuple $C = (T, \ell, w)$ where (T, ℓ) is a tree and $w \in T$ a leaf
- with $C[s]$ we denotes the labeled tree obtained by replacing leaf w in C by the whole tree s
- formally, for $s = (S, \ell_S)$, $C[s] \triangleq (T_{C[s]}, \ell_{C[s]})$ where
  - $T_{C[s]} = T \cup w \cdot S$
  - $\ell_{C[s]}(u) = t(u)$ for all $u \in T \setminus \{w\}$
  - $\ell_{C[s]}(w \cdot u) = \ell_S(u)$ for all $u \in T$



Pumping Lemma for Tree Automata (II)

Theorem

Let L be a language recognised by a BUTA. Then there exists $n \geq 0$ such that for all trees t of height larger than n:

- $t = C[D[s]]$ *for some contexts C, D and trees s; and*
- *for all $k \geq 0$,* $C[\underbrace{D[\dots[D[}_{k \text{ times}}s]]\dots]] \in L$

Pumping Lemma for Tree Automata (II)

Theorem

Let L be a language recognised by a BUTA. Then there exists $n \geq 0$ such that for all trees t of height larger than n:

- $t = C[D[s]]$ *for some contexts C, D and trees s; and*
- *for all* $k \geq 0$, $C[\underbrace{D[\ldots[D[}_{k \text{ times}} s]]\ldots]] \in L$

Proof Outline.

- n is given by the number of states of the corresponding automaton
- since the height of t is larger then n, it has a path from the root longer than n
- on such a path, the automaton has to loop
- the path to the loop defines C, the loop itself D, and the remainder to the leaf defines s

Top-Down Tree Automata

Top-Down Tree Automata

- A top-down tree automata (TDTA) $\mathcal{A}$ is a tuple (Q, Σ, q_I, δ) consisting of
  - a finite set of states Q
  - an alphabet Σ with associated arities
  - a transition function $\delta \triangleq \{\delta_a \mid a \in \Sigma\}$ where $\delta_{\mathtt{f}} : Q \to 2^{Q^{\mathrm{ar}(\mathtt{f})}}$
  - $q_I \in Q$ is an initial state

Top-Down Tree Automata

- A top-down tree automata (TDTA) $\mathcal{A}$ is a tuple (Q, Σ, q_I, δ) consisting of
  - a finite set of states Q
  - an alphabet Σ with associated arities
  - a transition function $\delta \triangleq \{\delta_a \mid a \in \Sigma\}$ where $\delta_{\mathtt{f}} : Q \to 2^{Q^{\mathrm{ar}(\mathtt{f})}}$
  - $q_I \in Q$ is an initial state
- An execution on a Σ-tree (T, ℓ) is a Q-tree (T, ℓ_Q) such that
  - $\ell_Q(\epsilon) = q_I$
  - for all $w \in T$ with n children,

$$(\ell_Q(w \cdot 0), \ldots, \ell_Q(w \cdot (n-1))) \in \delta_{\ell(w)}(\ell_Q(w))$$

Top-Down Tree Automata

- A top-down tree automata (TDTA) $\mathcal{A}$ is a tuple (Q, Σ, q_I, δ) consisting of
  - a finite set of states Q
  - an alphabet Σ with associated arities
  - a transition function $\delta \triangleq \{\delta_a \mid a \in \Sigma\}$ where $\delta_{\mathtt{f}} : Q \to 2^{Q^{\mathrm{ar}(\mathtt{f})}}$
  - $q_I \in Q$ is an initial state
- An execution on a Σ-tree (T, ℓ) is a Q-tree (T, ℓ_Q) such that
  - $\ell_Q(\epsilon) = q_I$
  - for all $w \in T$ with n children,

$$(\ell_Q(w \cdot 0), \ldots, \ell_Q(w \cdot (n-1))) \in \delta_{\ell(w)}(\ell_Q(w))$$

- the TDTA $\mathcal{A}$ recognises the tree-language

$$\mathrm{L}(\mathcal{A}) \triangleq \{(T, \ell) \mid (T, \ell) \text{ has an execution whose root is in } F\}$$

Example

- consider the TDTA $\mathcal{B} = (\{0,1\}, \Sigma_{\mathbb{B}}, \delta, 1)$ where

q	$\delta_\top$	$\delta_\bot$	$\delta_\vee$	$\delta_\wedge$	$\delta_\neg$	$\delta_{\mathtt{p}}$
0	$\varnothing$	$()$	$(0,0)$	$(0,0),(0,1),(1,0)$	1	$()$
1	$()$	$\varnothing$	$(0,1),(1,0),(1,1)$	$(1,1)$	0	$()$

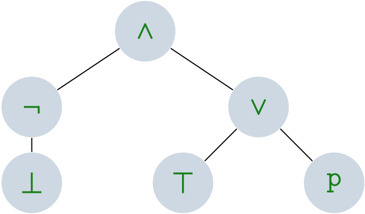
- it again recognises the set of satisfiable formulas, for instance

Example

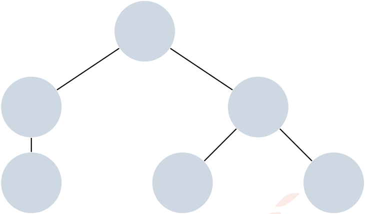
- consider the TDTA $\mathcal{B} = (\{0,1\}, \Sigma_{\mathbb{B}}, \delta, 1)$ where

q	$\delta_\top$	$\delta_\bot$	$\delta_\vee$	$\delta_\wedge$	$\delta_\neg$	$\delta_{\mathtt{p}}$
0	$\varnothing$	$()$	$(0,0)$	$(0,0),(0,1),(1,0)$	1	$()$
1	$()$	$\varnothing$	$(0,1),(1,0),(1,1)$	$(1,1)$	0	$()$

- it again recognises the set of satisfiable formulas, for instance



Input Tree (T, ℓ)



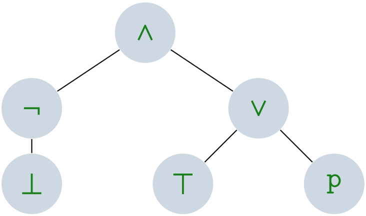
Execution on (T, ℓ)

Example

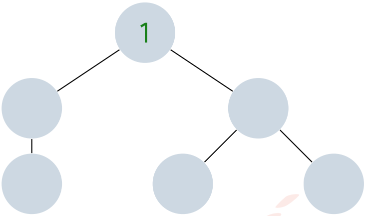
- consider the TDTA $\mathcal{B} = (\{0,1\}, \Sigma_{\mathbb{B}}, \delta, 1)$ where

q	$\delta_\top$	$\delta_\bot$	$\delta_\vee$	$\delta_\wedge$	$\delta_\neg$	$\delta_{\mathtt{p}}$
0	$\varnothing$	$()$	$(0,0)$	$(0,0),(0,1),(1,0)$	1	$()$
1	$()$	$\varnothing$	$(0,1),(1,0),(1,1)$	$(1,1)$	0	$()$

- it again recognises the set of satisfiable formulas, for instance



Input Tree (T, ℓ)



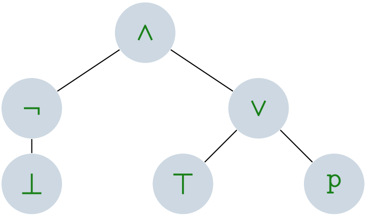
Execution on (T, ℓ)

Example

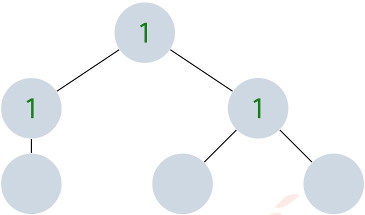
- consider the TDTA $\mathcal{B} = (\{0,1\}, \Sigma_{\mathbb{B}}, \delta, 1)$ where

q	$\delta_\top$	$\delta_\bot$	$\delta_\vee$	$\delta_\wedge$	$\delta_\neg$	$\delta_{\mathtt{p}}$
0	$\emptyset$	$()$	$(0,0)$	$(0,0),(0,1),(1,0)$	1	$()$
1	$()$	$\emptyset$	$(0,1),(1,0),(1,1)$	$(1,1)$	0	$()$

- it again recognises the set of satisfiable formulas, for instance



Input Tree (T, ℓ)



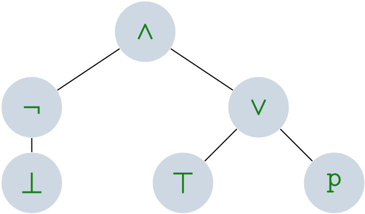
Execution on (T, ℓ)

Example

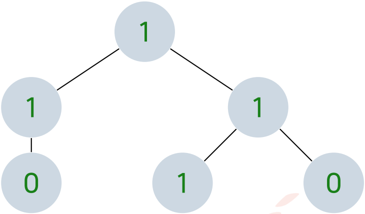
- consider the TDTA $\mathcal{B} = (\{0,1\}, \Sigma_{\mathbb{B}}, \delta, 1)$ where

q	$\delta_\top$	$\delta_\bot$	$\delta_\vee$	$\delta_\wedge$	$\delta_\neg$	$\delta_{\mathtt{p}}$
0	$\varnothing$	$()$	$(0,0)$	$(0,0),(0,1),(1,0)$	1	$()$
1	$()$	$\varnothing$	$(0,1),(1,0),(1,1)$	$(1,1)$	0	$()$

- it again recognises the set of satisfiable formulas, for instance



Input Tree (T, ℓ)



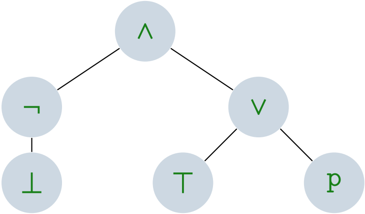
Execution on (T, ℓ)

Example

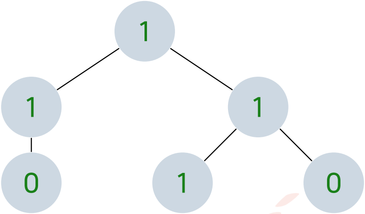
- consider the TDTA $\mathcal{B} = (\{0,1\}, \Sigma_{\mathbb{B}}, \delta, 1)$ where

q	$\delta_\top$	$\delta_\bot$	$\delta_\vee$	$\delta_\wedge$	$\delta_\neg$	$\delta_\mathtt{p}$
0	$\varnothing$	$()$	$(0,0)$	$(0,0),(0,1),(1,0)$	1	$()$
1	$()$	$\varnothing$	$(0,1),(1,0),(1,1)$	$(1,1)$	0	$()$

- it again recognises the set of satisfiable formulas, for instance



Input Tree (T, ℓ)



Execution on (T, ℓ)

Equivalence of BUTAs and TDTAs

Theorem

The following are equivalent:

1. *the set of languages recognized by BUTAs*
2. *the set of languages recognized by TDTAs*

Proof Outline.

- (1) $\Rightarrow$ (2) : Let L be recognised by TDTA $\mathcal{A} = (Q, \Sigma, q_I, \delta)$.
  Then L is recognised by the BUTA $\mathcal{B} = (Q, \Sigma, \delta', \{q_I\})$ where

$$\delta'_{\mathtt{f}}(q_1, \ldots, q_n) \triangleq \{q \mid (q_1, \ldots, q_n) \in \delta_{\mathtt{f}}(q)\}$$

Equivalence of BUTAs and TDTAs

Theorem

The following are equivalent:

1. *the set of languages recognized by BUTAs*
2. *the set of languages recognized by TDTAs*

Proof Outline.

- $(1) \Rightarrow (2)$: Let L be recognised by TDTA $\mathcal{A} = (Q, \Sigma, q_I, \delta)$.
  Then L is recognised by the BUTA $\mathcal{B} = (Q, \Sigma, \delta', \{q_I\})$ where

$$\delta'_{\mathtt{f}}(q_1, \ldots, q_n) \triangleq \{q \mid (q_1, \ldots, q_n) \in \delta_{\mathtt{f}}(q)\}$$

- $(2) \Rightarrow (1)$: Let L be recognised by BUTA $\mathcal{A} = (Q, \Sigma, \delta, F)$.
  Then L is recognised by the TDTA $\mathcal{B} = (Q \uplus \{q_I\}, \Sigma, q_I, \delta')$ where

$$(q_1, \ldots, q_n) \in \delta'_{\mathtt{f}}(q) :\Longleftrightarrow q \in \delta_{\mathtt{f}}(q_1, \ldots, q_n)$$

$$\delta'_{\mathtt{f}}(q_I) \triangleq \{(q_1, \ldots, q_n) \mid \delta_{\mathtt{f}}(q_1, \ldots, q_n) \in F\}$$

Determinisation

A deterministic TDTA (DTDTA) is a TDTA $\mathcal{A} = (Q, \Sigma, \delta, F)$ where $\delta_{\mathtt{f}} : Q \to Q^{\mathsf{ar}(\mathtt{f})} \cup \{\bot\}$ is a partial function for all $\mathtt{f} \in \Sigma$.

Determinisation

A deterministic TDTA (DTDTA) is a TDTA $\mathcal{A} = (Q, \Sigma, \delta, F)$ where $\delta_{\mathtt{f}} : Q \to Q^{\mathrm{ar}(\mathtt{f})} \cup \{\perp\}$ is a partial function for all $\mathtt{f} \in \Sigma$.

Theorem

There are languages recognised by TDTAs which are not recognised by DTDTAs.

Determinisation

A deterministic TDTA (DTDTA) is a TDTA $\mathcal{A} = (Q, \Sigma, \delta, F)$ where $\delta_{\mathtt{f}} : Q \to Q^{\mathsf{ar}(\mathtt{f})} \cup \{\perp\}$ is a partial function for all $\mathtt{f} \in \Sigma$.

Theorem

There are languages recognised by TDTAs which are not recognised by DTDTAs.

Proof Outline.

- ⋆ consider $L \triangleq \{\mathtt{f}(\mathtt{g}, \mathtt{h}), \mathtt{f}(\mathtt{h}, \mathtt{g})\}$ which is clearly recognised by a TDTA

Determinisation

A deterministic TDTA (DTDTA) is a TDTA $\mathcal{A} = (Q, \Sigma, \delta, F)$ where $\delta_{\mathtt{f}} : Q \to Q^{\mathrm{ar}(\mathtt{f})} \cup \{\bot\}$ is a partial function for all $\mathtt{f} \in \Sigma$.

Theorem

There are languages recognised by TDTAs which are not recognised by DTDTAs.

Proof Outline.

- ⋆ consider $L \triangleq \{\mathtt{f}(\mathtt{g}, \mathtt{h}), \mathtt{f}(\mathtt{h}, \mathtt{g})\}$ which is clearly recognised by a TDTA
- ⋆ now suppose $\mathcal{A} = (Q, \Sigma, q_I, \delta)$ recognises L

Determinisation

A deterministic TDTA (DTDTA) is a TDTA $\mathcal{A} = (Q, \Sigma, \delta, F)$ where $\delta_{\mathtt{f}} : Q \to Q^{\mathrm{ar}(\mathtt{f})} \cup \{\bot\}$ is a partial function for all $\mathtt{f} \in \Sigma$.

Theorem

There are languages recognised by TDTAs which are not recognised by DTDTAs.

Proof Outline.

- consider $L \triangleq \{\mathtt{f}(\mathtt{g}, \mathtt{h}), \mathtt{f}(\mathtt{h}, \mathtt{g})\}$ which is clearly recognised by a TDTA
- now suppose $\mathcal{A} = (Q, \Sigma, q_I, \delta)$ recognises L
- then $\mathcal{A}$ has the following shape:
    1. $\delta_{\mathtt{f}}(q_I) = (p, q)$ for some states p, q, otherwise, it would not accept a tree rooted in $\mathtt{f}$

Determinisation

A deterministic TDTA (DTDTA) is a TDTA $\mathcal{A} = (Q, \Sigma, \delta, F)$ where $\delta_{\mathtt{f}} : Q \to Q^{\mathsf{ar}(\mathtt{f})} \cup \{\bot\}$ is a partial function for all $\mathtt{f} \in \Sigma$.

Theorem

There are languages recognised by TDTAs which are not recognised by DTDTAs.

Proof Outline.

- ⋆ consider $L \triangleq \{\mathtt{f}(\mathtt{g},\mathtt{h}), \mathtt{f}(\mathtt{h},\mathtt{g})\}$ which is clearly recognised by a TDTA
- ⋆ now suppose $\mathcal{A} = (Q, \Sigma, q_I, \delta)$ recognises L
- ⋆ then $\mathcal{A}$ has the following shape:
  1. $\delta_{\mathtt{f}}(q_I) = (p, q)$ for some states p, q, otherwise, it would not accept a tree rooted in $\mathtt{f}$
  2. $\delta_{\mathtt{g}}(p) = () = \delta_{\mathtt{h}}(q)$, since $\mathtt{f}(\mathtt{g},\mathtt{h}) \in L$

Determinisation

A deterministic TDTA (DTDTA) is a TDTA $\mathcal{A} = (Q, \Sigma, \delta, F)$ where $\delta_{\mathtt{f}} : Q \to Q^{\mathrm{ar}(\mathtt{f})} \cup \{\bot\}$ is a partial function for all $\mathtt{f} \in \Sigma$.

Theorem

There are languages recognised by TDTAs which are not recognised by DTDTAs.

Proof Outline.

- consider $L \triangleq \{\mathtt{f}(\mathtt{g}, \mathtt{h}), \mathtt{f}(\mathtt{h}, \mathtt{g})\}$ which is clearly recognised by a TDTA
- now suppose $\mathcal{A} = (Q, \Sigma, q_I, \delta)$ recognises L
- then $\mathcal{A}$ has the following shape:
  1. $\delta_{\mathtt{f}}(q_I) = (p, q)$ for some states p, q, otherwise, it would not accept a tree rooted in $\mathtt{f}$
  2. $\delta_{\mathtt{g}}(p) = () = \delta_{\mathtt{h}}(q)$, since $\mathtt{f}(\mathtt{g}, \mathtt{h}) \in L$
  3. dual, $\delta_{\mathtt{h}}(p) = () = \delta_{\mathtt{g}}(q)$, since $\mathtt{f}(\mathtt{h}, \mathtt{g}) \in L$

Determinisation

A deterministic TDTA (DTDTA) is a TDTA $\mathcal{A} = (Q, \Sigma, \delta, F)$ where $\delta_{\mathtt{f}} : Q \to Q^{\mathrm{ar}(\mathtt{f})} \cup \{\perp\}$ is a partial function for all $\mathtt{f} \in \Sigma$.

Theorem

There are languages recognised by TDTAs which are not recognised by DTDTAs.

Proof Outline.

- consider $L \triangleq \{\mathtt{f}(\mathtt{g}, \mathtt{h}), \mathtt{f}(\mathtt{h}, \mathtt{g})\}$ which is clearly recognised by a TDTA
- now suppose $\mathcal{A} = (Q, \Sigma, q_I, \delta)$ recognises L
- then $\mathcal{A}$ has the following shape:
    1. $\delta_{\mathtt{f}}(q_I) = (p, q)$ for some states p, q, otherwise, it would not accept a tree rooted in $\mathtt{f}$
    2. $\delta_{\mathtt{g}}(p) = () = \delta_{\mathtt{h}}(q)$, since $\mathtt{f}(\mathtt{g}, \mathtt{h}) \in L$
    3. dual, $\delta_{\mathtt{h}}(p) = () = \delta_{\mathtt{g}}(q)$, since $\mathtt{f}(\mathtt{h}, \mathtt{g}) \in L$
- by (2) and (3) it follows that $\mathtt{f}(\mathtt{g}, \mathtt{g}), \mathtt{f}(\mathtt{h}, \mathtt{h}) \in \mathrm{L}(\mathcal{A})$, contradicting $\mathrm{L}(\mathcal{A}) = L$

Decision Problems

Theorem

The emptyness problem for BUTAs/TDTAs $\mathcal{A}$ is decidable in time $O(|\mathcal{A}|)$.

Proof.

Exercise. $\square$

Theorem

The universality and equivalence problems for BUTAs/TDTAs $\mathcal{A}$ are decidable in time $O(2^{|\mathcal{A}|})$.

Decision Problems

Theorem

The emptyness problem for BUTAs/TDTAs $\mathcal{A}$ is decidable in time $O(|\mathcal{A}|)$.

Proof.

Exercise. $\square$

Theorem

The universality and equivalence problems for BUTAs/TDTAs $\mathcal{A}$ are decidable in time $O(2^{|\mathcal{A}|})$.

Remark they are in fact EXPTIME-complete, and thus “slightly more difficult” than corresponding problems for NFAs