Bernd Finkbeiner Date: June 28, 2011

Automata, Games and Verification: Lecture 12

19 Complementation of Parity Tree Automata

Reference: W. Thomas: *Languages, Automata and Logic,* Handbook of formal languages, Volume 3.

Theorem 1 *For each parity tree automaton $\mathcal{A}$ over Σ there is a parity tree automaton $\mathcal{A}'$ with $\mathcal{L}(\mathcal{A}') = T_\Sigma - \mathcal{L}(\mathcal{A})$.*

Proof:

- $\mathcal{A}$ does *not* accept some tree t iff Player 1 has a winning memoryless strategy f in $\mathcal{G}_{\mathcal{A},t}$ from (ε, s_0)
- Strategy
$$f : \{0,1\}^* \times M \to \{0,1\}^* \times S$$
can be represented as
$$f' : \{0,1\}^* \times M \to \{0,1\}$$
(where $f(u, (q, \sigma, q'_0, q'_1)) = (u \cdot i, q'_i)$ iff $f'(u, \tau) = i$).
- f' is isomorphic to
$$g : \{0,1\}^* \to (M \to \{0,1\})$$
($M \to \{0,1\}$ is the finite "local strategy")
- Hence, $\mathcal{A}$ does not accept t iff

  (1) there is a $(M \to \{0,1\})$-tree v such that
  (2) for all $i_0, i_1, i_2, \ldots \in \{0,1\}^\omega$
  (3) for all $\tau_0, \tau_1, \ldots \in M^\omega$
  (4) <u>if</u>
  - for all j,
    $\tau_j = (q, a, q'_0, q'_1)$
    $\Rightarrow a = t(i_0, i_1, \ldots, i_j)$ and
  - $i_0 i_1 \ldots = v(\varepsilon)(\tau_0) v(i_0)(\tau_1) \ldots$

  <u>then</u> the generated state sequence $q_0 q_1 \ldots$
  with $q_0 = s_0, (q_j, a, q^0, q^1) = \tau_j$,
  $q_{j+1} = q^{v(i_0,\ldots,i_{j-1})(\tau_j)}$ for all j
  violates c.

- Condition **(4)** is a property of words over

$$\Sigma' = \underbrace{(M \to \{0,1\})}_{v} \times \underbrace{\Sigma}_{t} \times \underbrace{M}_{\tau} \times \underbrace{\{0,1\}}_{i}$$

  and can be checked by a parity word automaton $\mathcal{A}_4 = (S_4, \{s_4\}, T_4, c_4)$:
  - $S_4 = S \cup \{\perp\}$;
  - $s_4 = s_0$;
  - $T_4 = \{(q, (f, a, (q, a, q_0', q_1'), i), q_i') \mid q \in S, f : M \to \{0,1\},$
    $(q, a, q_0', q_1') \in M, i = f(q, a, q_0', q_1')\}$
    $\cup \ \{(q, (f, a, (q, a', q_0', q_1'), i), \perp) \mid a \neq a' \text{ or } i \neq f(q, a', q_0', q_1')\}$
    $\cup \ \{(\perp, a, \perp) \mid a \in \Sigma'\}$;
  - $c_4(q) = c(q) + 1$ for $q \in S$;
  - $c_4(\perp) = 0$.
- Condition **(3)** is a property of words $(M \to \{0,1\}) \times \Sigma \times \{0,1\}$ which results from **(4)** by universal quantification (= complement; project; complement) $\Rightarrow$ there is a deterministic parity word automaton $\mathcal{A}_3$ that checks **(3)**.
- Condition **(2)** defines a property of $(M \to \{0,1\}) \times \Sigma$-trees. It can be checked by a tree automaton $\mathcal{A}_2 = (S_2, s_2, M_2, c_2)$, simulating $\mathcal{A}_3$ along each path:
  - $S_2 = S_3$;
  - $s_2 = s_3$;
  - $M_2 = \{(q, (f, a), q_0', q_1') \mid (q, (f, a, 0), q_0') \in T_3, (q, (f, a, 1), q_1') \in T_3\}$;
  - $c_2 = c_1$.
- Condition **(1)** is a property on Σ-trees: Use nondeterminism to guess $M \to \{0,1\}$ label: $\mathcal{A}_1 = (S_1, s_1, M_1, c_1)$, where
  - $S_1 = S_2$;
  - $s_1 = s_2$;
  - $M_1 = \{(q, a, q_0', q_1') \mid \exists f : M \to \{0,1\}.(q, (f, a), q_0', q_1') \in M_2\}$;
  - $c_1 = c_2$.

■

20 Monadic Second-Order Theory of Two Successors (S2S)

Syntax:

- first-order variable set $V_1 = \{x_0, x_1, \ldots\}$
- second-order variable set $V_2 = \{X_0, X_1, \ldots\}$
- Terms t:

$$t ::= \epsilon \mid x \mid t0 \mid t1$$

- Formulas φ:

$$\varphi ::= t \in X \mid t_1 = t_2 \mid \neg\varphi \mid \varphi_0 \vee \varphi_1 \mid \exists x.\varphi \mid \exists X.\varphi$$

Semantics:

- first-order valuation $\sigma_1 : V_1 \to \mathbb{B}^*$
- second-order valuation $\sigma_2 : V_2 \to 2^{\mathbb{B}^*}$

Semantics of terms:

- $[\![\epsilon]\!] = \epsilon$
- $[\![x]\!]_{\sigma_1} = \sigma_1(x)$
- $[\![t0]\!]_{\sigma_1} = [\![t]\!]_{\sigma_1}0$
- $[\![t1]\!]_{\sigma_1} = [\![t]\!]_{\sigma_1}1$

Semantics of formulas:

- $\sigma_1, \sigma_2 \models t \in X$ iff $[\![t]\!]_{\sigma_1} \in \sigma_2(X)$
- $\sigma_1, \sigma_2 \models t_1 = t_2$ iff $[\![t_1]\!]_{\sigma_1} = [\![t_2]\!]_{\sigma_1}$
- $\sigma_1, \sigma_2 \models \neg\varphi$ iff $\sigma_1, \sigma_2 \not\models \varphi$
- $\sigma_1, \sigma_2 \models \varphi_0 \vee \varphi_1$ iff $\sigma_1, \sigma_2 \models \varphi_0$ or $\sigma_1, \sigma_2 \models \varphi_1$
- $\sigma_1, \sigma_2 \models \exists x_i.\varphi$ iff there is a $a \in \mathbb{B}^*$ s.t.

  $$\sigma_1'(y) = \begin{cases} \sigma_1(y) & \text{if} x \neq y, \\ a & \text{otherwise;} \end{cases}$$

  and $\sigma_1', \sigma_2 \models \varphi$
- $\sigma_1, \sigma_2 \models \exists X_i.\varphi$ iff there is a $A \subseteq \mathbb{B}^*$ s.t.

  $$\sigma_2'(Y) = \begin{cases} \sigma_2(Y) & \text{if} X \neq Y \\ A & \text{otherwise;} \end{cases}$$

  and $\sigma_1, \sigma_2' \models \varphi$

Examples:

- "node x is a prefix of node y"

  $$x \leq y \quad \Leftrightarrow \quad \forall X.((y \in X \wedge \forall z(z0 \in X \Rightarrow z \in X) \wedge \forall z.(z1 \in X \Rightarrow z \in X)) \Rightarrow x \in X)$$

- "X is linearly ordered by $\leq$"

  $$\text{Chain}(X) \quad \Leftrightarrow \quad \forall x.\forall y.((x \in X \wedge y \in X) \Rightarrow (x \leq y \vee y \leq x))$$

- "X is a path"

$$\begin{array}{rcl}
\mathrm{Path}(X) & \Leftrightarrow & \mathrm{Chain}(X) \land \neg\exists Y.\ (X \subseteq Y \land X \neq Y \land \mathrm{Chain}(Y)) \\
X \subseteq Y & \Leftrightarrow & \forall z.(z \in X \Rightarrow z \in Y) \\
X = Y & \Leftrightarrow & X \subseteq Y \land Y \subseteq X
\end{array}$$

- "X is infinite"

$$\mathrm{Inf}(X) \quad \Leftrightarrow \quad \exists Y.(Y \neq \emptyset \land \forall y \in Y.\exists y' \in Y.\exists x' \in X.\ (y < y' \land y < x'))$$

Theorem 2 *For each Muller tree automaton $\mathcal{A} = (S, s_0, M, \mathcal{F})$ over $\Sigma = 2^{V_2}$ there is a S2S formula φ over V_2 s.t. $t \in \mathcal{L}(\mathcal{A})$ iff $\sigma_2 \models \varphi$ where $\sigma_2(P) = \{q \in \{0,1\}^* \mid P \in t(q)\}$.*

Proof:

Use $\overline{R} = (R_q)_{q \in S}$ to encode the run tree.

$$\begin{array}{rcl}
\varphi & \Leftrightarrow & \exists \overline{R}.(\mathrm{Part} \land \mathrm{Init} \land \mathrm{Trans} \land \mathrm{Accept}) \\
\mathrm{Part} & \Leftrightarrow & \forall x. \bigvee_{q \in S} \mathrm{State}_q(x) \\
\mathrm{State}_q(x) & \Leftrightarrow & R_q(x) \land \bigwedge_{q' \in S \setminus \{q\}} \neg R_{q'}(x) \\
\mathrm{Init} & \Leftrightarrow & \mathrm{State}_{s_0}(\epsilon) \\
\mathrm{Trans} & \Leftrightarrow & \forall x. \bigvee_{(q,A,q'_0,q'_1) \in M} (\mathrm{State}_q(x) \land (\bigwedge_{V \in A} V(x) \land \bigwedge_{V \notin A} \neg V(x)) \land \\
& & \land \mathrm{State}_{q'_0}(x0) \land \mathrm{State}_{q'_1}(x1)) \\
\mathrm{InfOcc}_q(P) & \Leftrightarrow & \exists Q.(Q \subseteq P \land Q \subseteq R_q \land \mathrm{Inf}(Q)) \\
\mathrm{Inf}(P) & \Leftrightarrow & \exists P'.(P' \neq \emptyset \land \forall x' \in P'.\exists y \in P.\exists y' \in P'.(x' < y \land y < y')) \\
\mathrm{Muller}(P) & \Leftrightarrow & \bigvee_{F \in \mathcal{F}} (\bigwedge_{q \in F} \mathrm{InfOcc}_q(P) \land \bigwedge_{q \notin F} \neg \mathrm{InfOcc}_q(P)) \\
\mathrm{Accept} & \Leftrightarrow & \forall P.(\mathrm{Path}(P) \Rightarrow \mathrm{Muller}(P))
\end{array}$$

■

Theorem 3 *For every S2S formula φ over V_1, V_2 there is a Muller tree automaton $\mathcal{A}$ over $\Sigma = 2^{V_1 \cup V_2}$ such that $t \in \mathcal{L}(\mathcal{A})$ iff $\sigma_1, \sigma_2 \models \varphi$ where*

$$\begin{array}{rcl}
\sigma_1(x) & = & q \text{ iff } x \in t(q); \\
\sigma_2(X) & = & \{q \in \{0,1\}^* \mid X \in t(q)\}.
\end{array}$$

Proof:

First, we rewrite S2S formulas to a normal form, for which we only have the following types of equalities:

$$x = \epsilon,\ x = y0,\ x = y1,\ x \in Y,\ x = y$$

Next we inductively translate S2S formulas to tree automata. (Analogous to the proof for S1S in Lecture 7.)

- $x \in Y$:
  - $S = \{q_0, q_1\}$
  - $s_0 = q_0$
  - $M = \{(q_0, A, q_0, q_1) \mid x \notin A\}$
    $\cup \{(q_0, A, q_1, q_0) \mid x \notin A\}$
    $\cup \{(q_0, A, q_1, q_1) \mid x \in A, Y \in A\}$
    $\cup \{(q_1, A, q_1, q_1) \mid x \notin A\}$

  - $\mathcal{F} = \{q_1\}$
- $x = y0$:
  - $S = \{q_0, q_1, q_2\}$
  - $s_0 = q_0$
  - $M = \{(q_0, A, q_0, q_2) \mid \{x, y\} \cap A = \emptyset\}$
    $\cup \{(q_0, A, q_2, q_0) \mid \{x, y\} \cap A = \emptyset\}$
    $\cup \{(q_0, A, q_1, q_2) \mid x \notin A, y \in A\}$
    $\cup \{(q_1, A, q_2, q_2) \mid x \in A, y \notin A\}$
    $\cup \{(q_2, A, q_1, q_2) \mid x \notin A, y \in A\}$
    $\cup \{(q_2, A, q_2, q_2) \mid \{x, y\} \cap A = \emptyset\}$

  - $\mathcal{F} = \{q_2\}$
- etc.

■

Corollary 1 *S2S is decidable.*

SnS is the monadic second order theory of n successors.

Corollary 2 *SnS is decidable.*

Proof:

Repeat exercise for automata on n-ary trees. ■

SωS is the monadic second order theory of ω successors.

Theorem 4 *SωS is decidable.*

Proof:

We give an effective translation from SωS to S2S.

- Bijection β from ω^* to $0\mathbb{B}^*$:
  - $\beta(\epsilon) := \epsilon$
  - $\beta(vn) := \beta(v)01^n$

- One-to-many relation R between SωS and S2S structures: label a position $\beta(x)$ in the binary tree with σ iff x is labeled with σ in the ω-ary tree.
- Bring given SωS formula in normal form and translate as follows:
  - $x = \epsilon \mapsto x = \epsilon$
  - $x = yn \mapsto x = y01^n$ for $n \in \omega$
  - $x \in Y \mapsto x \in Y$
  - $x = y \mapsto x \in Y$
  - $\exists X \ldots \mapsto \exists X\ .(\forall y \in X\ .\ \neg 1 \leq y) \wedge \ldots$

■

WS2S is the weak monadic second order theory of two successors. It has the same syntax as S1S and the following difference in the semantics:
$\sigma_1, \sigma_2 \models \exists X.\varphi$ iff there is a **finite** $A \subseteq \mathbb{B}^*$ s.t.

$$\sigma_2'(Y) = \begin{cases} \sigma_2(Y) & \text{if} X \neq Y \\ A & \text{otherwise} \end{cases}$$

and $\sigma_1, \sigma_2' \models \varphi$.

Corollary 3 *WS2S is decidable.*

Theorem 5 *For a language $L \subseteq T_\Sigma$, the following are equivalent:*

1. *Both L and its complement are recognizable by a Büchi tree automaton.*
2. *L is WS2S-definable.*

Corollary 4 *WS2S is strictly weaker than S2S.*