Bernd Finkbeiner
Sven Schewe

Date: July 17, 2008

Automata, Games and Verification: Lecture 12

18 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 $q_0 = s_0, (q_j, a, q'_0, q'_1) = \tau_j$,
        $q_{j+1} = q_{v(i_1,\ldots,i_j)(\tau_j)}$
        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 $\mathcal{A}_4 = (S_4, \{s_4\}, T_4, c_4)$:
  - $S_4 = S \cup \{\bot\}$;
  - $s_4 = s_0$;
  - $T_4 = \{(q, (f, a, (q, a, q'_0, q'_1), i), q'_i) \mid q \in S, f : M \to \{0,1\},$
    $(q, a, q'_0, q'_1) \in M, i = f(q, a, q'_0, q'_1)\}$
    $\cup\ \{(q, (f, a, (q, a', q'_0, q'_1), i), \bot) \mid a \neq a' \text{ or } i \neq f(q, a', q'_0, q'_1)\}$
    $\cup\ \{(\bot, a, \bot) \mid a \in \Sigma'\}$;
  - $c_4(q) = c(q) + 1$ for $q \in S$;
  - $c_4(\bot) = 0$.
- 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
  - $S_1 = S_2$;
  - $s_1 = s_2$;
  - $M_1 = \{(q, a, q'_0, q'_1) \mid \exists f : M \to \{0,1\}.(q, (f, a), q'_0, q'_1) \in M_2\}$;
  - $c_1 = c_2$.

■

19 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"

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

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

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

- "X is a path"

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

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 $\sigma_2(P) = \{q \in \{0,1\}^* \mid P \in t(q)\}$.*

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*

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

Theorem 4 *S2S is decidable.*

SnS is the monadic second order theory of n successors.

Theorem 5 *SnS is decidable.*

20 Synthesis

The Synthesis Problem: Let i be a Boolean input variable, and O be a set of Boolean output variables. Given an LTL specification φ over $O \cup \{i\}$, decide if *there exists* an implementation that satisfies φ for all possible inputs.

Construction:

LTL specification φ

↓

Alternating Büchi word automaton $\mathcal{A}_\varphi$

↓

Nondeterministic Büchi word automaton $\mathcal{A}'_\varphi$

↓

Deterministic parity word automaton $\mathcal{A}''_\varphi$

↓

Deterministic parity *tree* automaton $\mathcal{A}'''_\varphi$

↓

Empty?

↙ Yes: φ *not realizable*

↘ No: φ *realizable*
winning strategy
in emptiness game
defines implementation.