

11-711: Algorithms for NLP

Recitation #5

October 9, 2009

Equivalence of Acceptance by Final State and Empty Stack

Direction 1: From Final State to Empty Stack

Given a PDA P_F that accepts a language L by final state, we can construct a PDA P_N that accepts the same language L by empty stack.

Idea We build P_N to simulate P_F and then empty its stack at the end of the computation. We add a bottom-of-stack marker to make sure that P_N does not ‘accidentally’ accept words not in the language.

Construction

- We use a new symbol, \perp , which must not be a symbol of Γ to mark the bottom of the stack.
- We add a new initial state q'_0 to push Z_0 , the start symbol of P_N , onto the stack.
- We add a new final state q_f that pops the stack without consuming any input.

$$P_N = (Q \cup \{q'_0, q_f\}, \Sigma, \Gamma \cup \{\perp\}, \delta_N, q'_0, \perp)$$

where δ_N is defined by:

1. $\delta_N(q'_0, \epsilon, \perp) = \{(q_0, Z_0 \perp)\}$
2. $\delta_N(q, a, Y) = \delta_F(q, a, Y) \forall q \in Q, a \in \Sigma \cup \{\epsilon\}, Y \in \Gamma$
3. $\delta_N(q, \epsilon, Y) \ni (q_f, \epsilon) \forall q \in F, Y \in \Gamma \cup \{\perp\}$
4. $\delta_N(q_f, \epsilon, Y) = \{(q_f, \epsilon)\} \forall Y \in \Gamma \cup \{\perp\}$

Proof

$$w \in L(P_F) \implies w \in L(P_N)$$

$$\begin{array}{ll} (q_0, w, Z_0) \vdash_{P_F}^* (q, \epsilon, \alpha) & \text{by definition of } P_F \\ (q_0, w, Z_0) \vdash_{P_N}^* (q, \epsilon, \alpha) & \text{by (2)} \\ (q_0, w, Z_0 \perp) \vdash_{P_N}^* (q, \epsilon, \alpha \perp) & \text{by Theorem 6.5} \\ (q'_0, w, \perp) \vdash_{P_N} (q_0, w, Z_0 \perp) \vdash_{P_N}^* (q, \epsilon, \alpha \perp) & \text{by (1)} \\ (q'_0, w, \perp) \vdash_{P_N} (q_0, w, Z_0 \perp) \vdash_{P_N}^* (q, \epsilon, \alpha, \perp) \vdash_{P_N}^* (q_f, \epsilon, \epsilon) & \text{by (3) and (4)} \end{array}$$

$$w \in L(P_N) \implies w \in L(P_F)$$

The only way P_N can empty its stack is by entering state q_f since \perp is sitting at the bottom of the stack and \perp is not a symbol on which P_F has any moves. The only way P_N can enter state q_f is if the simulated P_F enters an accepting state. Between IDs $(q_0, w, Z_0 \perp)$ and $(q, \epsilon, \alpha \perp)$ all the moves are moves of P_F . In particular \perp was never the top stack symbol prior to reaching ID $(q, \epsilon, \alpha \perp)$. Thus, we conclude that the same computation can occur in P_F .

Direction 2: From Empty Stack to Final State

Given a PDA P_N that accepts a language L by empty stack, we can construct a PDA P_F that accepts the same language L by final state.

Idea We build P_F to simulate P_N and accept with some state q_f if the stack is empty. To detect when the stack is empty we again add \perp to Γ .

subsubsection*Construction

- We use a new symbol, \perp , which must not be a symbol of Γ to mark the bottom of the stack.
- We add a new initial state q'_0 to push Z_0 , the start symbol of P_N , onto the stack.
- We add a final state q_f .

$$P_F = (Q \cup \{q'_0, q_f\}, \Sigma, \Gamma \cup \{\perp\}, \delta_F, q'_0, \perp, \{q_f\})$$

where δ_F is defined by:

1. $\delta_F(q'_0, \epsilon, \perp) = \{(q_0, Z_0 \perp)\}$
2. $\delta_F(q, a, Y) = \delta_N(q, a, Y) \forall q \in Q, a \in \Sigma \cup \{\epsilon\}, Y \in \Gamma$
3. $\delta_F(q, \epsilon, \perp) \ni (q_f, \epsilon) \forall q \in Q$

Proof

$$w \in L(P_N) \implies w \in L(P_F)$$

$$\begin{array}{ll} (q_0, w, Z_0) \vdash_{P_N}^* (q, \epsilon, \epsilon) & \text{by definition of } P_N \\ (q_0, w, Z_0) \vdash_{P_F}^* (q, \epsilon, \epsilon) & \text{by (2)} \\ (q_0, w, Z_0 \perp) \vdash_{P_F}^* (q, \epsilon, \perp) & \text{by Theorem 6.5} \\ (q'_0, w, \perp) \vdash_{P_F} (q_0, w, Z_0 \perp) \vdash_{P_F}^* (q, \epsilon, \perp) & \text{by (1)} \\ (q'_0, w, \perp) \vdash_{P_F} (q_0, w, Z_0 \perp) \vdash_{P_F}^* (q, \epsilon, \perp) \vdash_{P_F}^* (q_f, \epsilon, \epsilon) & \text{by (3)} \end{array}$$

$$w \in L(P_F) \implies w \in L(P_N)$$

The additional rules (1) and (3) of δ_F give us very limited ways to accept w by final state. We must use rule (3) at the last step, and we can only use that rule if the stack contains \perp . Furthermore, we must use rule (1) as the first step to place \perp on the stack. Between IDs $(q_0, w, Z_0 \perp)$ and (q, ϵ, \perp) must be a computation of P_N with \perp below the stack because P_F cannot use any transition that is not a transition of P_N . In addition, \perp cannot be exposed or the computation would end. Thus, we conclude that the same computation can occur in P_N .