Azzolini Riccardo 2020-12-03

PDA — Computazioni e linguaggio accettato

1 Descrizione istantanea

Dato un PDA $P = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$, le informazioni che caratterizzano l'automa a un certo punto della computazione sono tre:

- lo stato dell'automa;
- la parte della stringa di input che non è ancora stata letta;
- il contenuto dello stack.

Formalmente, tali informazioni sono caratterizzate dalla **descrizione istantanea** (**ID**, *Instantaneous Description*) dell'automa P, che è una tripla $(q, w, \gamma) \in Q \times \Sigma^* \times \Gamma^*$ in cui:

- q è lo *stato*;
- w è l'*input residuo*;
- γ è il *contenuto dello stack*.

2 Passo di computazione

Si consideri un PDA $P = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$, supponendo che la sua descrizione istantanea sia attualmente $(q, aw, X\beta)$, con $q \in Q$, $a \in \Sigma_\epsilon$, $w \in \Sigma^*$, $X \in \Gamma$ e $\beta \in \Gamma^*$:

- l'automa si trova nello stato q;
- la stringa da leggere è aw, che in particolare inizia con il simbolo a (oppure si può porre $a = \epsilon$, quando l'automa deve eseguire una transizione spontanea, indipendente dai simboli della stringa in input);
- il contenuto dello stack è $X\beta$, dunque in particolare il simbolo in cima è X.

In base a q, a e X, la funzione di transizione stabilisce i passi che l'automa può compiere, dando come risultato un insieme di coppie. Se una di queste coppie è (p, α), allora il passo di computazione di P rispetto alla scelta di tale coppia porta l'automa alla situazione caratterizzata dalla descrizione istantanea $(p, w, \alpha\beta)$:

- lo stato passa da q a p;
- il simbolo in input a viene consumato (se $a \neq \epsilon$, altrimenti $\epsilon w = w$), quindi rimane solo la stringa w;
- lo stack viene modificato sostituendo X con α.

Formalmente, un **passo di computazione** di P è definito come una relazione tra le descrizioni istantanee prima e dopo il passo: se $(p, \alpha) \in \delta(q, a, X)$, allora, per tutte le stringe $w \in \Sigma^*$ e $\beta \in \Gamma^*$,

$$(q, aw, X\beta) \vdash (p, w, \alpha\beta)$$

3 Computazione

Dato $P = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$, una **computazione** di P è una sequenza (possibilmente di lunghezza 0) di passi di computazione di P. In generale, date due descrizioni istantanee I e J, si scrive $I \vdash^* J$ se e solo se esiste una sequenza di ID $K_1, K_2, \ldots, K_n$ (con $n \geq 1$) tale che

- $I = K_1$,
- $J = K_n$,
- per ogni $i = 1, 2, \ldots, n-1$ si ha $K_i \vdash K_{i+1}$,

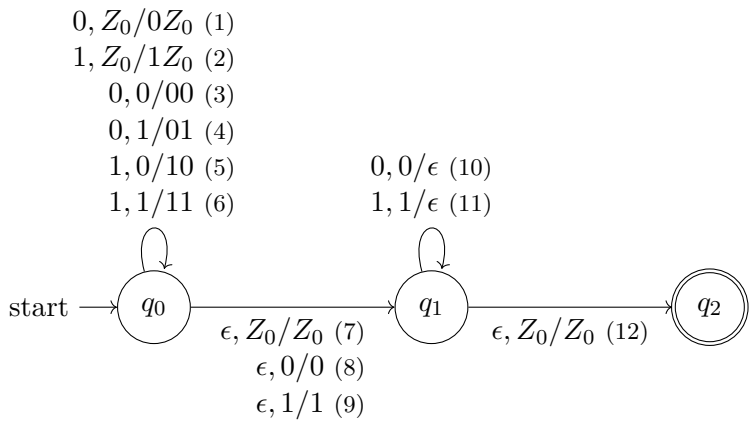
cioè, complessivamente:

$$I = K_1 \vdash K_2 \vdash \cdots \vdash K_n = J$$

Come caso particolare, se $n = 1$, allora $I = K_1$ e $J = K_1$, ovvero si ha una computazione $I \vdash^* J$ in zero passi se e solo se $I = J$, ed è quindi sempre vero che $I \vdash^* I$.

3.1 Esempi

Si consideri l'automa P_{ww^R} visto in precedenza, che riconosce il linguaggio dei palindromi di lunghezza pari su $\{0, 1\}$:



(in questo diagramma sono state numerate le varie transizioni, per farvi riferimento nei seguenti esempi di computazione).

Si supponga di partire dalla descrizione istantanea $(q_0, 0110, Z_0)$. A prima vista, gli elementi rilevanti per la funzione di transizione potrebbero essere q_0, 0 e Z_0, ma siccome sono ammesse le ϵ-mosse, c'è anche la possibilità di considerare invece q_0, ϵ e Z_0 (cioè ϵ invece di 0). Allora, le mosse possibili in questa situazione sono $\delta(q_0, 0, Z_0) = \{(q_0, 0Z_0)\}$ e $\delta(q_0, \epsilon, Z_0) = \{(q_1, Z_0)\}$. Scegliendo la prima di queste due, corrispondente alla transizione (1) nel diagramma, si ottiene il passo di computazione:

$$(q_0, 0110, Z_0) \vdash (q_0, 110, 0Z_0)$$

Adesso, al secondo passo, si ha nuovamente una scelta, tra $\delta(q_0, 1, 0) = \{(q_0, 1Z_0)\}$ e $\delta(q_0, \epsilon, 0) = \{(q_1, 0)\}$. Se questa volta si sceglie l'ϵ-mossa, ovvero la transizione (8), il passo compiuto è

$$(q_0, 110, 0Z_0) \vdash (q_1, 110, 0Z_0)$$

nel quale non viene consumato input e non viene modificato lo stack: cambia solo lo stato (da q_0 a q_1). Adesso, le transizioni possibili sono $\delta(q_1, 1, 0) = \varnothing$ e $\delta(q_1, \epsilon, 0) = \varnothing$, cioè nessuna, quindi la computazione è bloccata. Siccome la stringa in input non è stata consumata per intero, essa non è sicuramente accettata tramite questa computazione.

Una computazione alternativa è quella in cui al secondo passo si sceglie la mossa data da $\delta(q_0, 1, 0) = \{(q_0, 1Z_0)\}$, corrispondente alla transizione (5):

$$\begin{aligned} (q_0, 0110, Z_0) &\vdash (q_0, 110, 0Z_0) && (1) \\ &\vdash (q_0, 10, 10Z_0) && (5) \end{aligned}$$

Da qui, scegliendo opportunamente le transizioni, si può proseguire

$$\begin{aligned} &\vdash (q_1, 10, 10Z_0) && (9) \\ &\vdash (q_1, 0, 0Z_0) && (11) \\ &\vdash (q_1, \epsilon, Z_0) && (10) \\ &\vdash (q_2, \epsilon, Z_0) && (12) \end{aligned}$$

fino a giungere alla situazione caratterizzata dall'ID (q_2, ϵ, Z_0), in cui l'automa è nello stato finale q_2 e ha consumato tutto l'input: come si vedrà in seguito, questa sarà la condizione di accettazione.

4 Configurazione iniziale

Dato un PDA $P = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$, la **configurazione iniziale** di P su input $w \in \Sigma^*$ è la descrizione istantanea (q_0, w, Z_0). Intuitivamente, essa è l'ID da cui partono le computazioni dell'automa, nella quale:

- l'automa si trova nel suo stato iniziale q_0;
- la stringa in input è w;
- lo stack contiene solo il simbolo iniziale di stack Z_0.

5 Accettazione

A differenza degli automi a stati finiti, nel caso di un PDA $P = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$ si hanno due possibili nozioni di accettazione:

- Il linguaggio accettato da P **per stato finale** è

  $$L(P) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (p, \epsilon, \alpha) \text{ con } p \in F\}$$

  cioè l'insieme delle stringhe sull'alfabeto di input per le quali esiste una computazione che, partendo dalla configurazione iniziale, arriva in un'ID (p, ϵ, α) nella quale tutto l'input è stato consumato (l'input "rimanente" è ϵ) e lo stato p è finale.

- Il linguaggio accettato da P **per stack vuoto** è

  $$N(P) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (p, \epsilon, \epsilon)\}$$

  cioè l'insieme delle stringhe sull'alfabeto di input per le quali esiste una computazione che, partendo dalla configurazione iniziale, arriva in un'ID (p, ϵ, ϵ) nella quale tutto l'input è stato consumato e lo stack è vuoto (sia l'input rimanente che lo stack sono ϵ). Non è richiesto che lo stato p sia finale.

Osservazione: Nel caso di accettazione per stack vuoto, l'insieme F degli stati finali è irrilevante, quindi potrebbe essere omesso dalla definizione di PDA, indicando solo i primi sei elementi: $P = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0 \rangle$.

Sebbene la nozione di accettazione per stato finale possa risultare in qualche modo più naturale, in quanto analoga all'accettazione negli automi a stati finiti, quella per stack vuoto renderà più facile la dimostrazione dell'equivalenza tra PDA e CFG.

In generale, dato un qualche PDA P, le due nozioni di accettazione danno luogo a linguaggi diversi, $L(P) \neq N(P)$,[1] ma esse sono equivalenti in un altro senso:

- per ogni PDA P, esiste un PDA P' tale che $N(P') = L(P)$;
- per ogni PDA P, esiste un PDA P' tale che $L(P') = N(P)$.

6 Proprietà delle computazioni

Dati $q, p \in Q$, $x, y \in \Sigma^*$ e $\alpha, \beta \in \Gamma^*$, valgono le seguenti proprietà (che non verranno dimostrate, ma lo si farebbe per induzione sulla lunghezza delle computazioni):

- (P1) Se $(q, x, \alpha) \vdash^* (p, y, \beta)$ allora, per ogni $w \in \Sigma^*$, $(q, xw, \alpha) \vdash^* (p, yw, \beta)$.
- (P2) Se $(q, x, \alpha) \vdash^* (p, y, \beta)$ allora, per ogni $\gamma \in \Gamma^*$, $(q, x, \alpha\gamma) \vdash^* (p, y, \beta\gamma)$.
- (P3) Per ogni $w \in \Sigma^*$, se $(q, xw, \alpha) \vdash^* (p, yw, \beta)$ allora $(q, x, \alpha) \vdash^* (p, y, \beta)$.

Le proprietà (P1) e (P3) affermano in sostanza che le computazioni di un PDA dipendono soltanto dalla parte di input che è stata letta, mentre la (P2) afferma, analogamente, che la computazione dipende solo dalla parte di stack che è stata letta.

[1] Esistono particolari PDA per cui $L(P) = N(P)$, ma questo non vale in generale.