Equivalenza tra 2-PDA e Macchine di Turing

Teorema Principale

Teorema: I 2-PDA sono computazionalmente equivalenti alle Macchine di Turing.

Formalmente: L(2-PDA) = L(TM) = L(RE) (linguaggi ricorsivamente enumerabili)

Definizioni Preliminari

k-PDA

Un **k-PDA** è una tupla $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ dove:

- Q: insieme finito di stati
- Σ: alfabeto di input
- Γ: alfabeto delle pile (stack alphabet)
- δ: $Q \times \Sigma \cup \{\varepsilon\} \times \Gamma^k \to P(Q \times \Gamma^k)$ (funzione di transizione)
- $q_0 \in Q$: stato iniziale
- $Z_0 \in \Gamma$: simbolo iniziale delle pile
- $F \subseteq Q$: stati finali

Gerarchia dei k-PDA

- **0-PDA** ≡ **NFA** (automi a stati finiti non deterministici)
- **1-PDA** ≡ **PDA** (automi a pila convenzionali)
- **k-PDA** per $k \geq 2$

Dimostrazione dell'Equivalenza

Direzione 1: 2-PDA ⊆ TM

Lemma 1: Ogni linguaggio riconosciuto da un 2-PDA può essere riconosciuto da una TM.

Dimostrazione: Simulazione diretta. Data una TM M_TM, simuliamo il 2-PDA M_2PDA su M_TM mantenendo:

- Stati del 2-PDA negli stati interni della TM
- Le due pile sui due lati del nastro della TM (separate da un marcatore)
- Simulazione delle operazioni push/pop con movimenti del nastro

La simulazione è algoritmica e sempre terminante se il 2-PDA termina. □

Direzione 2: TM ⊆ 2-PDA (Costruzione Principale)

Lemma 2: Ogni linguaggio riconosciuto da una TM può essere riconosciuto da un 2-PDA.

Idea Chiave: Simulare il nastro infinito della TM usando due pile.

Rappresentazione del Nastro

Per una configurazione del nastro TM:

```
... b b a₁ a₂ a₃ [a₄] a₅ a₆ b b ...
                  ↑
               testina
```

Rappresentiamo con due pile:

- **Pila Sinistra (S_1)**: contenuto a sinistra della testina (invertito)
- **Pila Destra (S_2)**: contenuto dalla testina in poi

```
S₁: [a₃, a₂, a₁, b, b, ...]   (top = a₃)
S₂: [a₄, a₅, a₆, b, b, ...]   (top = a₄)
```

Costruzione Formale

Dato: TM $M = (Q_{TM}, \Sigma, \Gamma, \delta_{TM}, q_0, b, F_{TM})$

Costruiamo: 2-PDA $M' = (Q', \Sigma, \Gamma', \delta', q_0', Z_0, F')$

Stati: $Q' = Q_{TM} \cup \{q_0', q_{move\_R}, q_{move\_L}\}$

Alfabeto delle pile: $\Gamma' = \Gamma \cup \{Z_0\}$

Algoritmo di Simulazione

Fase 1: Inizializzazione

$$\delta'(q_0', w_1, Z_0, Z_0) = (q_0, Z_0, w_1 Z_0)$$

Input $w = w_1 w_2 ... w_n$ va in S_2, S_1 rimane vuota.

Fase 2: Simulazione delle Transizioni

Per ogni transizione TM: $\delta_{TM}(q, a) = (q', a', D)$

Caso D = R (movimento destro):

$$\delta'(q, \varepsilon, X, a) = (q\_move\_R, Xa', Y) \text{ se } S_2.top = a$$
$$\delta'(q\_move\_R, \varepsilon, X, Y) = (q', X, Y)$$

Caso D = L (movimento sinistro):

$$\delta'(q, \varepsilon, a, Y) = (q\_move\_L, X, a'Y) \text{ se } S_1.top = a$$
$$\delta'(q\_move\_L, \varepsilon, X, Y) = (q', X, Y)$$

Caso D = S (stazionario):

$$\delta'(q, \varepsilon, X, a) = (q', X, a') \text{ se } S_2.top = a$$

Gestione dei Blank

Quando una pila diventa vuota, assumiamo simboli blank infiniti:

$$\delta'(q, \varepsilon, X, Z_0) = (q', X, bZ_0) \quad // S_2 \text{ vuota} \rightarrow \text{aggiungi blank}$$
$$\delta'(q, \varepsilon, Z_0, Y) = (q', bZ_0, Y) \quad // S_1 \text{ vuota} \rightarrow \text{aggiungi blank}$$

Correttezza della Costruzione

Invariante Fondamentale

Invariante: Ad ogni passo, la configurazione (S_1, S_2) rappresenta esattamente il contenuto del nastro TM nella posizione corrente della testina.

Teorema di Correttezza

Teorema: M' accetta w $\iff$ M accetta w

Dimostrazione:

- $\rightarrow$: Se M accetta w, esiste una computazione accettante. La simulazione preserva ogni transizione, quindi M' raggiungerà lo stato finale corrispondente.
- $\leftarrow$: Se M' accetta w, ogni passo corrisponde a una transizione valida di M, quindi M accetterà w.

Conseguenze Teoriche

Corollario 1: Complessità Computazionale

2-PDA possono simulare qualsiasi algoritmo (come le TM), ma con overhead polinomiale nella simulazione delle operazioni di nastro.

Corollario 2: Problema della Fermata

Il problema della fermata per 2-PDA è indecidibile (ereditato dalle TM).

Corollario 3: Linguaggi Context-Sensitive

2-PDA riconoscono strettamente più dei linguaggi context-free:

$$L(REG) \subsetneq L(CFL) \subsetneq L(CSL) \subsetneq L(2\text{-}PDA) = L(RE)$$

Esempio Concreto

Linguaggio: $L = \{a^n b^n c^n \mid n \geq 1\}$

Questo linguaggio non è context-free ma è riconoscibile da un 2-PDA:

1. Pila 1: conta le 'a'
2. Pila 2: conta le 'b'
3. Confronta entrambe con le 'c'

Limite Teorico

Teorema: Per $k \geq 2$, k-PDA $\equiv$ TM

Dimostrazione: La costruzione si estende naturalmente usando solo 2 delle k pile disponibili.

Questo stabilisce che **2 pile sono sufficienti** per la completezza di Turing - pile aggiuntive non aumentano il potere computazionale.