Roberto Barile

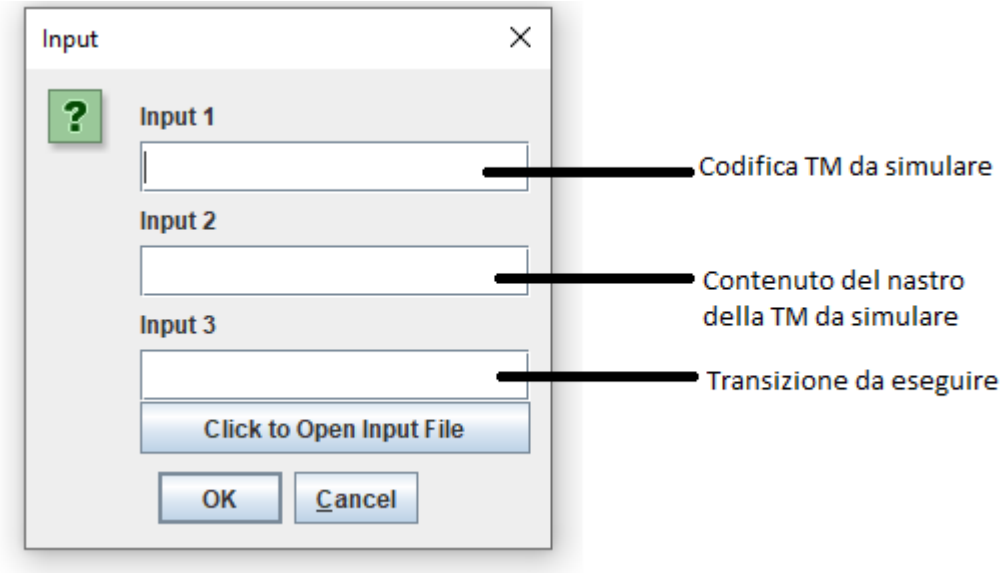
Macchina di Turing Universale

Introduzione

Il file jflap allegato è un'implementazione della universal turing machine (UTM), la quale permette di simulare qualsiasi Turing Machine risolvendo la limitazione di poter eseguire solo un programma per volta, cioè di poter calcolare solo la funzione per cui la Turing Machine è stata programmata.

Implementazione UTM

La UTM è stata implementata tramite una Turing Machine 3 tape nel seguente modo:



Tape 1 - Codifica

La Turing Machine da simulare viene descritta come una stringa di simboli.
Si definisce la funzione
$\pi: \Sigma \cup Q \cup \{R, L\} \rightarrow \Sigma^*_{encode}$ con $\Sigma^*_{encode} = \{ 1 \}$
che permette di codificare i simboli dell'alfabeto, gli stati e i movimenti della testina, ad esempio:

Simboli alfabeto:	Stati:	Movimenti testina:
$\pi(0) = 1$	$\pi(q_0) = 1$	$\pi(R) = 1$
$\pi(1) = 11$	$\pi(q_1) = 11$	$\pi(L) = 11$
$\pi(2) = 111$	$\pi(q_2) = 111$	
$\pi(i) = 111…1$		

Ricordando che una transizione è identificabile con una quintupla **(q_{in}, σ_r, q_{out}, σ_w, op_code)** dove:
qin = stato corrente
σ_r = simbolo letto
qout = nuovo stato
σ_w = carattere scritto
op_code = movimento della testina

la codifica della transizione diventa: **$\pi(q_{in})$ 0 $\pi(\sigma_r)$ 0 $\pi(q_{out})$ 0 $\pi(\sigma_w)$ 0 π(op_code)**
Per descrivere la transizione si è introdotto un separatore quindi si estende l'alfabeto di codifica:
Σ*encode = Σ*encode U { 0 } = {0, 1}

La codifica Δ dell'intera macchina si ottiene concatenando le codifiche delle transizioni (usando 00 come separatore tra le transizioni) :
Δ = $\pi(q_{in})$ 0 $\pi(\sigma_r)$ 0 $\pi(q_{out})$ 0 $\pi(\sigma_w)$ 0 π(op_code) 00
$\pi(q_{in})$ 0 $\pi(\sigma_r)$ 0 $\pi(q_{out})$ 0 $\pi(\sigma_w)$ 0 π(op_code) 00
$\pi(q_{in})$ 0 $\pi(\sigma_r)$ 0 $\pi(q_{out})$ 0 $\pi(\sigma_w)$ 0 π(op_code) 00

Per ottimizzare l'analisi delle transizioni si modifica la loro struttura, cioè si elimina q_{in} e si modifica l'ordine di op_code, σ_w e q_{out} ottenendo quindi la quadrupla **(σ_r, σ_w, op_code, q_{out})**.
Come conseguenza della modifica della struttura delle transizioni, in particolare dell'eliminazione dello stato iniziale q_{in} si ha che Δ deve essere totalmente ordinata rispetto a stato e carattere letto, quindi va innanzitutto resa totale introducendo delle transizioni dummy dagli stati per i quali non è definita alcuna transizione per alcuni simboli dell'alfabeto, cioè:

$$\forall q_i \in Q - F - \{q_e\}, \forall \sigma_i \in \Sigma: (q_i, \sigma_i) = \emptyset \Rightarrow (q_i, \sigma_i) = (q_e, E, S)$$

Inoltre vanno introdotti dei caratteri appositi per gli stati speciali, altrimenti questi ultimi sarebbero trattati come stati normali perdendo il loro ruolo, quindi si codificano stato finale e stato dummy rispettivamente come $ e $$. In seguito a questa modifica è nuovamente esteso l'alfabeto di codifica: Σ*encode = Σ*encode U { $ } = {0, 1, $}

Tape 2 – Input
Sul tape 2 si inserisce l'input della macchina che si sta simulando.
Si ricorda che la macchina implementata è universale solo per l'alfabeto **Γ** = {1, 0, □} perciò l'input deve essere una stringa di alfabeto **Γ**.

Dettagli implementativi

Transition search – Stato iniziale

La macchina universale inizia la simulazione della macchina codificata scegliendo la transizione da eseguire tra le transizioni che partono dallo stato iniziale della macchina codificata, in questo caso tra le prime tre transizioni dato che questa macchina universale lavora su un alfabeto di tre caratteri.
La scelta è effettuata in base al carattere indicato dalla testina sul nastro 2, cioè:
0: prima transizione di Δ
1: seconda transizione di Δ
□: terza transizione di Δ

Copy Unit

Successivamente la transizione viene copiata sul nastro 3 (compreso il separatore 00), quest'ultimo viene riavvolto fino all'inizio della transizione e si analizza la transizione scelta e copiata, cioè si esegue il Sigma Out Analyzer e il Move Analyzer.

Sigma out analyzer

Si effettua la scelta del carattere da scrivere sul nastro 2 nella posizione indicata dalla sua testina; la scelta è basata su σ_w nella codifica della transizione che si sta analizzando sul nastro 3.
Anche in questo il carattere appartiene all'alfabeto $\Gamma = \{1, 0, \square\}$.

Move analyzer

Si effettua la scelta del movimento della testina del nastro 2 analizzando il campo op_code della transizione che si sta analizzando sul nastro 3.
Si ricorda che op_code può essere R (destra) o L (sinistra).
Nell'implementazione della macchina universale è stato usato anche S (nessun movimento) per comodità, mentre nella macchina codificata S deve essere simulato con la concatenazione di R e L.

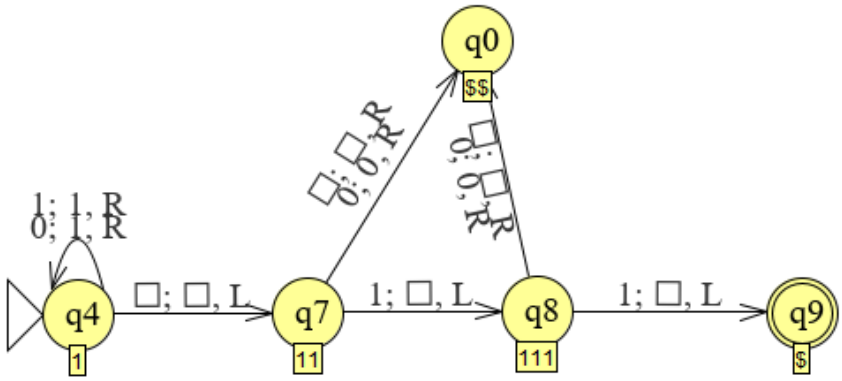
Transition search

Successivamente basandosi sull'ultima sezione della transizione (q_{out}) ci si posiziona nel nastro 1 sullo stato q_{out}, ossia sulla prima delle 3 transizioni che partono da q_{out}.
Ciò viene fatto saltando p transizioni dove
$p = (k - 1) * |\Gamma|$
con k = numero di 1 usato per la codifica di q_{out} nella transizione che si sta analizzando sul nastro 3.
Dopo essersi posizionati sulla prima transizione tra le tre transizioni che partono da q_{out} si pulisce il nastro 3 e in seguito si effettua la scelta della transizione da analizzare in base al carattere indicato dalla testina sul nastro 2 analogamente al procedimento descritto precedentemente per lo stato iniziale.

Error / Final states

E' prevista anche una sezione che gestisce le transizioni che portano ad una computazione errata o ad uno stato finale.
In particolare la macchina di Turing universale termina in uno stato finale scrivendo una E sul secondo nastro in caso di computazioni errate, mentre si ferma senza agire sui nastri quando la macchina codificata termina correttamente.

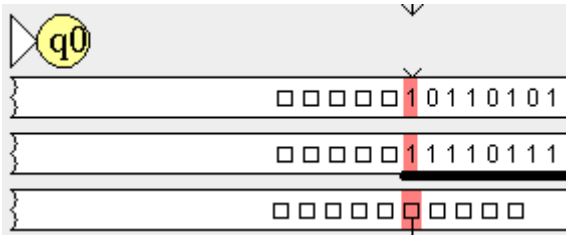
Esempi
Somma unaria
L'input è inserito separando i due addendi col carattere 0
Macchina:

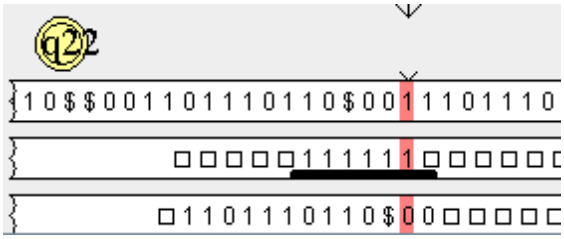


Codifica:

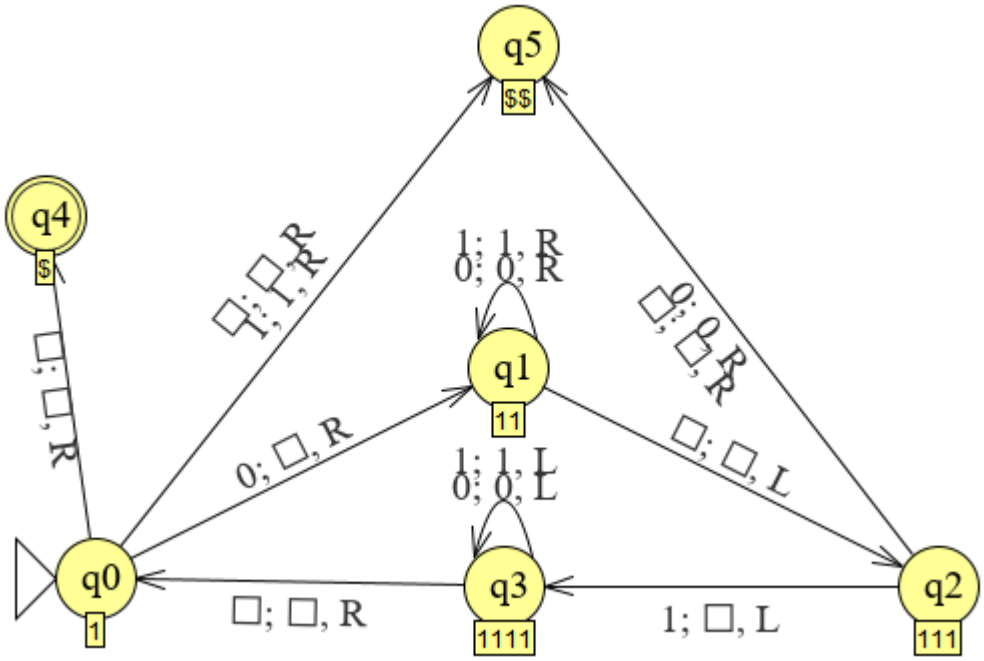
1011010100
11011010100
111011101101100
101010$$00
110111011011100
1110111010$$00
101010$$00
1101110110$00
1110111010$$00

Simulazione su utm:





Linguaggio 0^n1^n
Macchina:



Codifica:
101110101100
11011010$$00
1110111010$00
1010101100
110110101100
1110111011011100
101010$$00
1101110110111100
1110111010$$00
1010110111100
110110110111100
1110111010100

Simulazione su utm: