Computabilità, Complessità e Logica

Lezione 7

Funzioni computabili

Abbiamo visto come possiamo rispondere “sì” o “no” usando una MdT

Possiamo usare una MdT per calcolare funzioni con un codominio più ampio? Per esempio funzioni da $\Sigma^\star$ a $\Sigma^\star$?

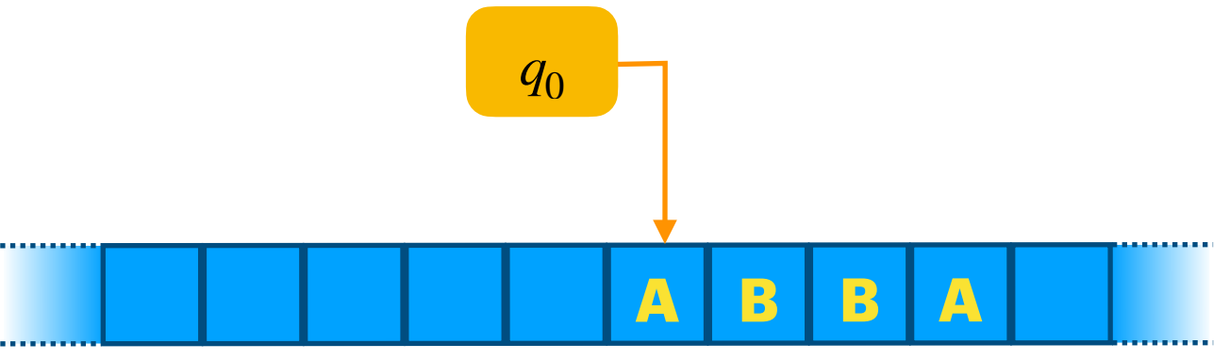
Sì, possiamo usare come “output” il contenuto del nastro della MdT quando la macchina si arresta

Questo ci permette di definire la nozione di **funzione computabile**

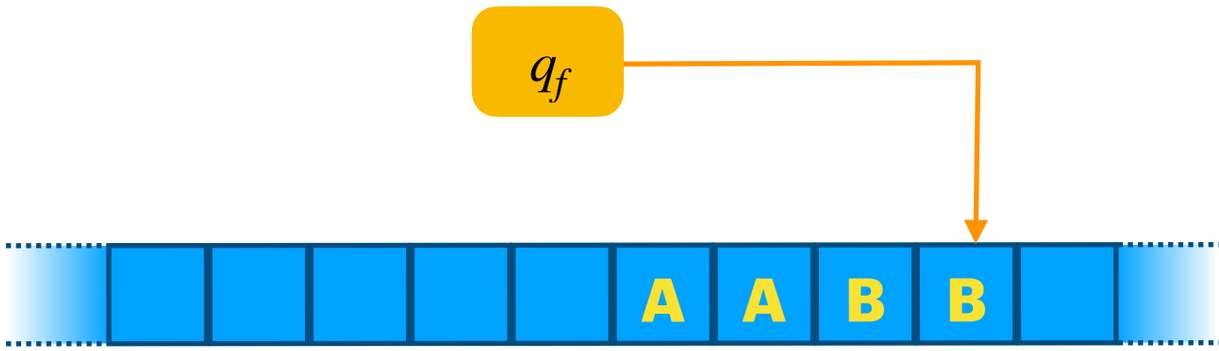
Funzioni computabili

Una funzione $f : \Sigma^\star \to \Sigma^\star$ si dice **computabile** se esiste una macchina di Turing M tale per cui input $w \in \Sigma^\star$ la macchina M si arresta con solo $f(w)$ sul nastro

$$w = ABBA$$



$$f(ABBA) = AABB$$



Funzioni computabili

Le normali operazioni aritmetiche sugli interi sono computabili: per esempio possiamo costruire un MdT che, data una coppia $\langle m, n \rangle$ ritorna il valore $m + n$

Una funzione computabile può anche lavorare su codifiche di macchine di Turing.
Per esempio una macchina che prende in input una rappresentazione di una MdT $\langle M \rangle$ e ritorna in output la codifica di $\langle M \rangle$ con invertiti i ruoli di stato accettante e rifiutante

Esercizio: la composizione di funzioni computabili è computabile

Funzioni computabili

Non tutte le funzioni sono computabili!

Quante funzioni ci sono? Quante macchine di Turing?

Possiamo sfruttare le funzioni computabili per mostrare che alcuni problemi sono indecidibili/decidibili?

Sì, usando la nozione di **riduzione** tra linguaggi

L'idea è che se possiamo trasformare un linguaggio in un altro con una funzione computabile allora possiamo "trasportare" alcuni risultati di decidibilità

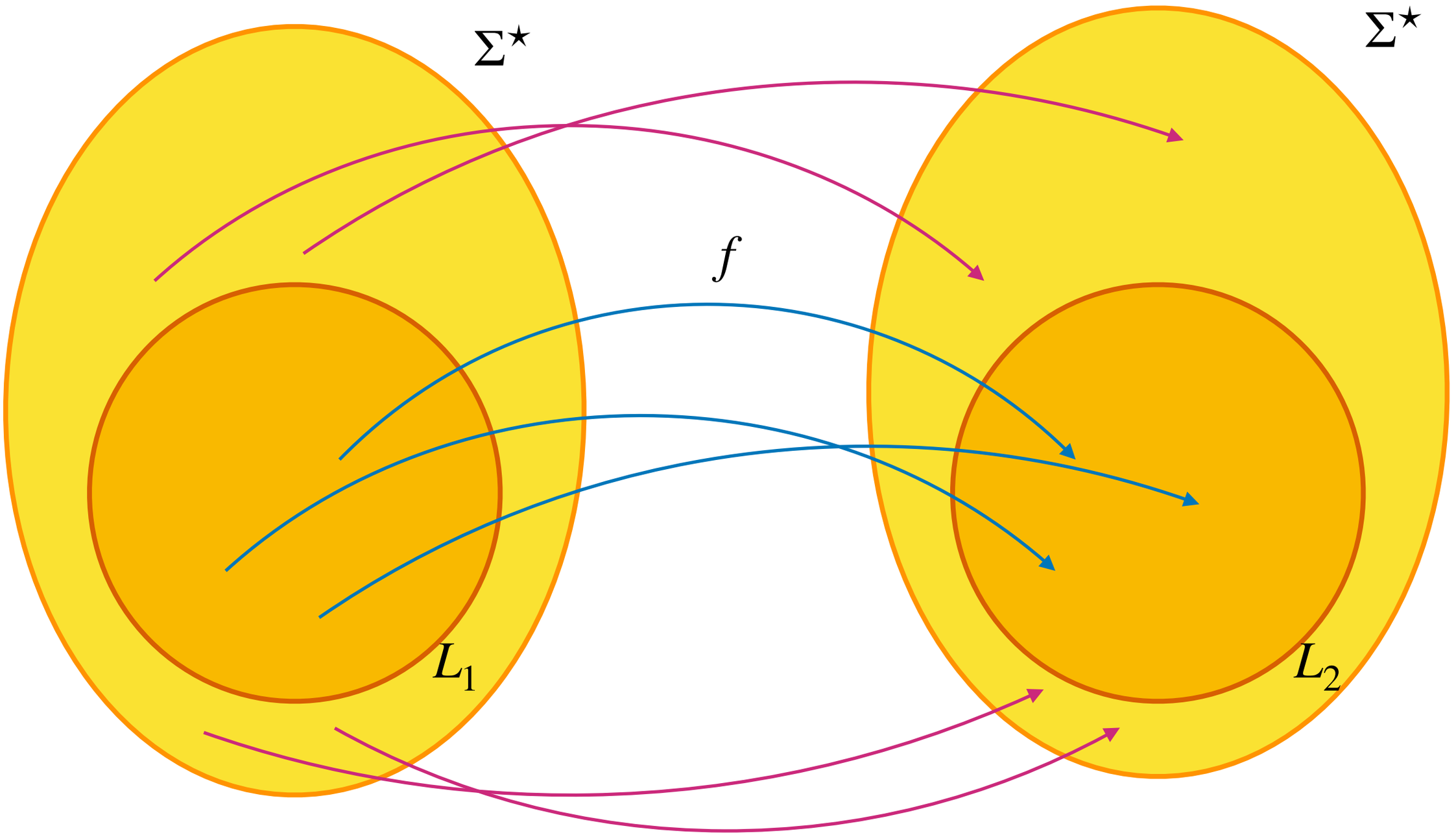
Riducibilità

Un linguaggio L_1 si dice **riducibile** a un linguaggio L_2, denotato con $L_1 \leq_m L_2$, se esiste una funzione computabile $f : \Sigma^\star \to \Sigma^\star$ tale per cui, per ogni $w \in \Sigma^\star$

$$w \in L_1 \iff f(w) \in L_2$$

La funzione f è chiamata una **riduzione** da L_1 a L_2

Riducibilità



f sposta tutti gli elementi di L_1 in elementi di L_2

f sposta tutti gli elementi fuori da L_1 in elementi fuori da L_2

Riducibilità

Teorema

Se $L_1 \leq_m L_2$ e L_2 è *decidibile* allora L_1 è *decidibile*.

Dimostrazione

Costruiamo una MdT che decide L_1.

Dato che L_2 è decidibile esiste una macchina M_2 che lo riconosce.

Dato che L_1 è riducibile a L_2 allora esiste una riduzione f da L_1 a L_2

Riducibilità

Sia M_1 la MdT definita nel seguente modo:

Su input $w \in \Sigma^\star$

1. Computa $f(w)$

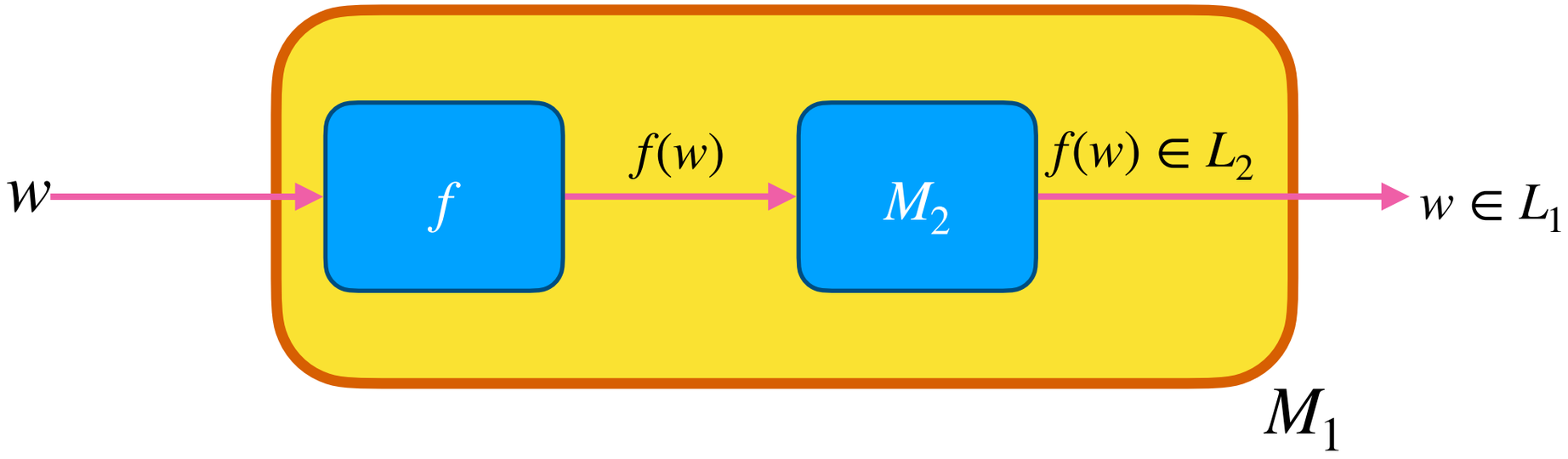
2. Simula M_2 su input $f(w)$

3. Se M_2 accetta (risp. rifiuta) allora M_1 accetta (risp. rifiuta)

Se $w \in L_1$ allora $f(w) \in L_2$ perché f è una riduzione. Quindi M_1 accetta se e solo se $w \in L_1$ ∎

Riducibilità

Rappresentazione grafica



Possiamo usare la macchina M_2 e la riduzione f per costruire una macchina che decide L_1

Riducibilità

Corollario

Se $L_1 \leq_m L_2$ e L_1 è *indecidibile* allora L_2 è *indecidibile*.

Dimostrazione

Mostriamo che se esistesse una macchina che decide L_2 allora ne esisterebbe una che decide L_1.

Se L_2 fosse decidibile potremmo costruire la macchina della dimostrazione precedente per decidere L_2 ∎

Altri problemi indecidibili

Possiamo dimostrare che altri problemi sono indefinibili tramite riduzioni

Per esempio se troviamo il modo di ridurre il problema dell'arresto a un altro problema allora quel problema è indecidibile

Un classico esempio è il
problema di corrispondenza di Post (PCP)

Problema di Corrispondenza di Post

Data due sequenze finite di lunghezza $n \in \mathbb{N}$ di parole

$A = u_1, u_2, \ldots, u_n$ e $B = v_1, v_2, \ldots, v_n$

Ci si chiede se esista una sequenza di indici $i_1, \ldots, i_k$ per qualche $k \in \mathbb{N}$ tale per cui:

$$u_{i_1} u_{i_2} \ldots u_{i_k} = v_{i_1} v_{i_2} \ldots v_{i_k}$$

È più semplice visualizzare il problema come una sequenza di tessere del domino

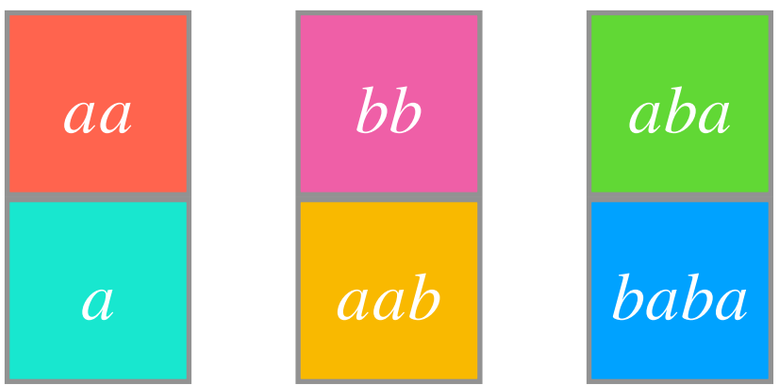
PCP come domino

Formulazione “visuale” del problema

$$A = (\boxed{aa}, \boxed{bb}, \boxed{aba})$$

$$B = (\boxed{a}, \boxed{aab}, \boxed{baba})$$

Possiamo costruire delle “tessere del domino” con le coppia di parole di cui la prima è presa da A e la seconda da B:



PCP come domino

Formulazione “visuale” del problema

Possiamo mettere assieme le tessere del domino (ne abbiamo infinite per tipo) in modo da poter leggere la stessa parola sopra e sotto?

aa	aa	bb	aba
a	a	aab	$baba$

Questa è una soluzione (che forma la parola $aaaabbaba$) con le tessere dell’esempio.

Possiamo trovare una soluzione in ogni caso?

Indecidibilità del PCP

Proviamo a fare una riduzione dal problema dell'arresto

Prendiamo una coppia $\langle M, w \rangle$ di una MdT e di un input

Costruiamo un insieme di domino (due sequenze di parole) per il quale esiste una sequenza di indici che fa coincidere le due parole che si formano se e solo se M su input w si arresta

In quel caso PCP non può essere decidibile perché altrimenti potremmo trasformare una istanza del problema dell'arresto in una istanza del problema di corrispondenza di Post e deciderla!

Indecidibilità del PCP

Mostriamo che una versione modificata del PCP è indecidibile:

MPCP è come PCP ma la prima “tessera” è fissata.

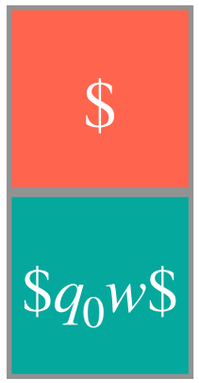
Mostreremo che questa restrizione può essere poi eliminata

Assumiamo di avere un MdT
$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$
e una parola di input $w = w_1 w_2 \cdots w_n \in \Sigma^\star$

Assumiamo inoltre che esista un simbolo $\$ \notin \Gamma$ che useremo come “separatore di configurazioni di una MdT”

Indecidibilità del PCP: parte 1

Come prima tessera inseriamo:



Questa tessera costringerà a ricostruire la configurazione iniziale della MdT M nelle tessere “sopra”

Questa sarà anche la tessera iniziale del MPCP

Indecidibilità del PCP: parte 2

Come seconda tipologia di tessere inseriamo:

Per ogni $q, r \in Q$ e $a, b \in \Gamma$ tali per cui $\delta(q, a) = (r, b, \rightarrow)$

Questo forza a codificare il passaggio di stato, la scrittura di un simbolo e il movimento della testina verso destra

Indecidibilità del PCP: parte 3

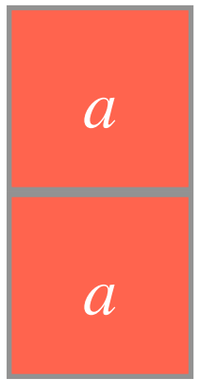
Come terza tipologia di tessere inseriamo:

Per ogni $q, r \in Q$ e $a, b, c \in \Gamma$ tali per cui
$\delta(q, a) = (r, b, \leftarrow)$

Questo forza a codificare il passaggio di stato, la scrittura di un simbolo e il movimento della testina verso sinistra. Per via della codifica usata queste tessere sono diverse dal secondo tipo

Indecidibilità del PCP: parte 4

Come quarta tipologia di tessere inseriamo:

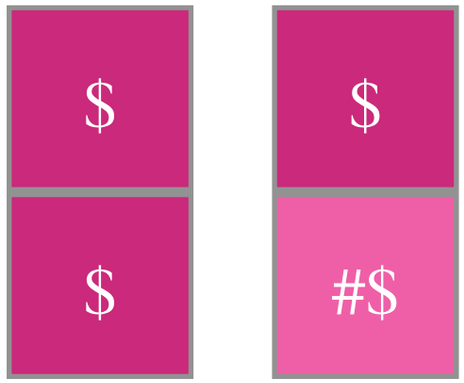


Per ogni $a \in \Gamma$

Questo forza a mantenere invariato lo stato del nastro della macchine quando non è presente la testina a modificarlo

Indecidibilità del PCP: parte 5

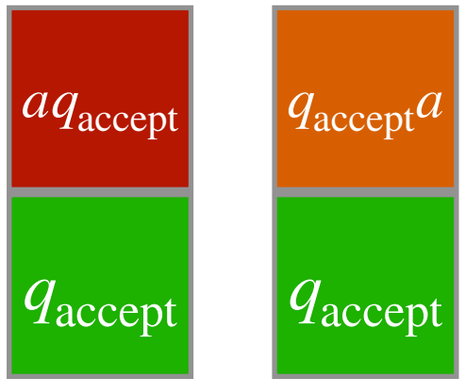
Come quinta tipologia di tessere inseriamo:



Questo permette di terminare una configurazione della MdT. La seconda tessera rende esplicito lo spazio vuoto normalmente non codificato in una configurazione

Indecidibilità del PCP: parte 6

Come sesta tipologia di tessere inseriamo:



Per ogni $a \in \Gamma$. Questo permette allo stato accettante di "mangiare" i simboli adiacenti (ci serve per arrivare in uno stato in cui rimane solo q_{accept})

Indecidibilità del PCP: parte 7

Come settima tipologia di tessere inseriamo:

Questo permette di completare il match tra le due stringhe.

Vediamo con un esempio come funziona questa costruzione.

Indecidibilità del PCP: esempio

Consideriamo una MdT che accetta se legge AA sul nastro (riscrivendolo in BB). L'input è $w = AA$

$	q_0A	A	$	B	q_1A	$	B	Bq_{accept}	$	Bq_{accept}	$	$q_{accept}\$\$$
$\$q_0AA\$$	Bq_1	A	$	B	Bq_{accept}	$	B	q_{accept}	$	q_{accept}	$	$

Divisione in passi di computazione:

$	q_0A	A	$	B	q_1A	$	B	Bq_{accept}	$	Bq_{accept}	$	$q_{accept}\$\$$
$\$q_0AA\$$	Bq_1	A	$	B	Bq_{accept}	$	B	q_{accept}	$	q_{accept}	$	$

Da MPCP a PCP

Dobbiamo ora passare da MPCP a PCP

Definiamo, per una stringa $u = u_1 u_2 \cdots u_n$ le seguenti operazioni:

$$\star u = * u_1 * u_2 * \cdots * u_n$$

$$\star u \star = * u_1 * u_2 * \cdots * u_n *$$

$$u \star = u_1 * u_2 * \cdots * u_n *$$

Da MPCP a PCP

Data una sequenza di tessere

$$\left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \ldots, \left[\frac{t_n}{b_n}\right]$$

La ridefiniamo come

$$\left[\frac{\star\, t_1}{\star\, b_1\, \star}\right], \left[\frac{\star\, t_1}{b_1\, \star}\right]\left[\frac{\star\, t_2}{b_2\, \star}\right], \ldots, \left[\frac{\star\, t_n}{b_n\, \star}\right], \left[\frac{\star\, \blacklozenge}{\blacklozenge}\right]$$

Questo forza la tessera $\left[\frac{\star\, t_1}{\star\, b_1\, \star}\right]$ a essere la prima

Indecidibilità del PCP

Se esiste una computazione accettante di M su input w esiste un modo di fare “match” tra le tessere del domino e quindi la risposta all’istanza di PCP corrispondente è positiva.

Viceversa, se M non accetta (o non termina) su input w la risposta alla corrispondente istanza di PCP è negativa.

Se potessimo decidere PCP allora potremmo risolvere il problema dell’arresto. Quindi PCP è indecidibile.