Immagine che contiene testo

Descrizione generata automaticamente

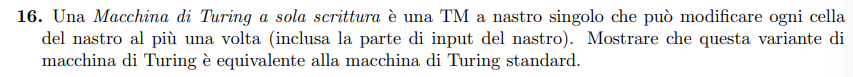
L’equivalenza tra un automa a coda ed una TM deve essere mostrata, quindi che il linguaggio può essere riconosciuto dall’automa e dunque dalla TM. Si dimostra creando una simulazione che fa capite che l’automa Q si comporti esattamente come la TM M.

*La simulazione di Q come M funziona nel modo seguente:*

si consideri l’intero nastro come una coda. Ciascun simbolo viene modificato e il movimento del nastro avviene a destra. Quando più di un numero viene pushato nella coda, si shiftano tutti i valori a destra. Se si raggiunge la fine del nastro, allora il simbolo più a sinistra viene considerato e si capisce di essere arrivati alla fine.

*La simulazione di M come Q funziona nel modo seguente:*

l’alfabeto della macchina M viene espanso aggiungendo un simbolo extra e un simbolo marcatore a sinistra viene inserito nella coda. I simboli sono pushati a sinistra e letti (pop) a destra.



*(Soluzione estesa del libro, tradotta e riscritta)*

Per prima cosa simuliamo una normale macchina di Turing con una macchina di Turing che scrive due volte.

La macchina a scrittura doppia simula un singolo passaggio della macchina originale copiando

l'intero nastro su una parte nuova del nastro sul lato destro della porzione attualmente utilizzata. La procedura di copiatura opera carattere per carattere, marcando un carattere mentre viene copiato. Questa procedura altera ogni quadrato del nastro due volte: una volta per scrivere il personaggio per la prima volta, e ancora per sottolineare che lo è stato copiato. La posizione della testina del nastro della macchina di Turing originale è contrassegnata sul nastro.

Quando si copiano le celle in corrispondenza o vicino alla posizione contrassegnata, il contenuto del nastro è aggiornato secondo le regole della macchina di Turing originale. Per eseguire la simulazione con una macchina scrivibile una volta, si opera come prima, eccetto che ogni cella del nastro precedente è ora rappresentata da due celle. La prima di queste contiene il simbolo del nastro della macchina originale e il secondo è per il simbolo utilizzato nella procedura di copiatura. L'input non viene presentato alla macchina nel formato con due celle per simbolo, quindi la prima volta che il nastro viene copiato, i segni di copiatura sono messi direttamente sopra il simbolo originale

*(Soluzione mia, che segue tipo quelle degli ε-NFA)*

Partendo dalla macchina originale, si copia l’intero nastro tra le transizioni. In questo nuovo nastro viene aggiunto un simbolo delimitatore (caso TM a nastro semi-infinito) ed un alfabeto espanso per registrare la posizione del nastro quando è stato copiato. In questo caso si hanno due scritture, la prima usata quando si copia il simbolo sul nuovo nastro, la seconda per indicato che sia stato copiato.

Avendo le transizioni uguali, ma TM a sola scrittura è equivalente a quelle standard.

Immagine che contiene testo

Descrizione generata automaticamente

1. Si considera un linguaggio CF, che viene riconosciuto da un PDA.

Ad esempio vediamo L={anbncn | n >= 0}

Usiamo il PL per linguaggi CF dimostrando che non si tratta di CFL.

Ad esempio consideriamo, per s=uvxyz, una stringa s=1p0p1p0p

Quindi consideriamo che s sia nella forma detta e:

* se v e v contengono al più un tipo di simbolo dell’alfabeto, la stringa sarà nella forma uv2xy2z e gli 0 ed 1 saranno di lunghezza diversa
* se v ed y contengono più di un tipo di alfabeto per simbolo e comunque questi non saranno nell’ordine corretto, portando ad una stringa sbilanciata

Dunque l’1-PDA (PDA regolare) non risconosce L.

Se siamo come visto nella forma uv2xy2z allora viene riconosciuto dal 2-PDA:

* si fa push di tutte le “a” che appaiono di fronte al nastro di input sul primo stack
* si fa push di tutte le “b” che seguono le “a” nella seconda pila. Se si vede qualche “a”, si rifiuta
* quando vede la prima “c”, si fa pop da entrambi gli stack allo stesso momento. Dopo questo, si rifiuta l’input se si vede qualche “a” o qualche “b”
* pop da entrambe le pile per ogni carattere “c” letto. Se l’input termina quando entrambi gli stack sono vuoti, accetta, altrimenti rifiuta.

Quindi in questo modo, L è riconosciuto dai 2-PDA.

Perciò i 2-PDA sono più potenti degli 1-PDA.

1. I 3-PDA non devono essere più potenti di una TM e quindi se si mostra i 2-PDA possono simulare una Turing Machine, allora i 3-PDA non sono più potenti dei 2-PDA.

Simuliamo una TM su 2-PDA:

* la prima pila memorizza i caratteri a sinistra della testa, con il fondo della pila che memorizza il carattere più a sinistra della TM
* la seconda pila memorizza i caratteri a destra della testa, con il fondo della pila che memorizza il carattere più a destra della TM

Vediamo quindi le transizioni:

* a sinistra, avendo δ(qi, wi)=(qj,wj,L) e poi wi sulla cima della seconda pila

Se ci sta il $ alla cima si rifiuta, altrimenti si procede a sinistra.

In altri casi, si fa pop del simbolo della prima pila facendo push sulla seconda pila. Si cambia poi stato sulla macchina qj. Se qj è stato accettante, allora accetta *w*; viceversa se qj rifiuta, allora rifiuta *w.*

* a destra, avendo δ(qi, wi)=(qj,wj,L) e poi wi sulla cima della seconda pila

Si fa il pop di wi dalla cima della seconda pila, pushando wj nella seconda pila. Se il $ è in cima alla prima pila, si fa il push del blank sulla seconda pila. In entrambi i cas, si camvia lo stato nella macchina qj. Se qj è stato accettante, allora accetta *w*; viceversa se qj rifiuta, allora rifiuta *w.*

In questo modo si dimostra che i 2-PDA riconoscono esattamente la classe dei linguaggi Turing-riconoscibili

1. Per dimostrare che i 3-PDA *non* sono più potenti dei 2-PDA, si considera di simulare una macchina a 4 nastri non deterministici con i 3-PDA.

* il primo nastro è quello di input, mentre gli altri 3 (secondo, terzo, quarto) corrispondono a 3 pile (prima, seconda, terza pila/stack).
* se lo stack si cambia con un push e non con un pop, allora si muove il nastro a destra senza cambia il simbolo corrente. Viene poi scritto il simbolo sul nastro pushato in cima alla pila. In questo caso tutte le cime del nastro sono tenuto eccetto gli stack affetti da push senza pop. Muove a destra per prendere il simbolo blank nel nastro
* se lo stack cambia con un pop e non con un push, N rimpiazza la cima del nastro con un simbolo vuoto e si muove a sinistra
* se lo stack è cambiato da entrambe le operazioni, N cambia il nastro con il simbolo scritto pushato nella cella corrente
* se lo stack non è cambiato, N scrive lo stesso simbolo all’inizio del nastro e poi di nuovo nel nastro stesso

Si usa un simbolo per capire quando N muove il primo nastro a destra. Se non viene usato, allora, N copia gli elementi all’inizio del primo nastro. Questa azioni sono parallele e fatte nello stesso momento in tutti i nastri, dato che possono cambiare una sola vola. N per questi motivi e si nota, è nondeterministica, capendo quale transizione prendere.

Dunque, alla fine di tutto questo, si prova che una TM non è potente quanto i 3-PDA, ma anche che i 3-PDA non sono pi potenti dei 2-PDA. Entrambi sono ugualmente potenti; ciò che cambia è la gestione effettiva dei valori. Ma entrambi sono affetti dal non-determinismo; semplicemente la TM per ciascuna variante numerica di PDA, aggiungerà un nastro, mantenendo di massima lo stesso comportamento dei nastri, risultando sempre ugualmente potente.