Laboratorio di Linguaggi di Sistema – a.a. 2005/2006
Progetto finale II (avanzato)

Il problema

Si vuole realizzare un *simulatore di Macchina di Turing* in grado di eseguire un programma dato, operando su un nastro simulato. Il comando che implementa il simulatore deve avere la seguente sintassi:

`turing` *programma.txt input* [*timeout in passi*]

in cui il primo parametro è il nome di un file di testo che contiene il programma da eseguire, nel formato che sarà specificato appresso, mentre il secondo parametro è il contenuto iniziale del nastro. Il terzo parametro, opzionale, indica il numero massimo di passi che il simulatore deve compiere – se durante la computazione si eccede questo limite, il simulatore termina.
Il programma è costituito da quintuple (*stato iniziale*, *simbolo letto*, *stato finale*, *simbolo scritto*, *spostamento*). Ad ogni passo della computazione, il simulatore cerca fra le quintuple del programma una che abbia lo *stato iniziale* uguale allo stato corrente della macchina, e il *simbolo letto* uguale al simbolo presente sul nastro nella posizione corrispondente alla testina di lettura/scrittura. Se trova tale quintupla, il simulatore cambia il proprio stato in *stato finale*, scrive sul nastro il *simbolo scritto* nella posizione corrente della testina, e infine sposta la testina su una nuova cella come indicato dallo *spostamento*. Se invece non esiste una tupla corrispondente, il programma termina.
Inizialmente si assume che il simulatore sia nello stato `0`, che il nastro contenga l'input indicato sulla riga di comando, con infinite celle contenenti blank (spazio) a sinistra e a destra, e che la testina sia posizionata sul primo carattere non-blank dell'input.
In ogni caso, al momento della terminazione il simulatore deve stampare su *stdout* l'intero contenuto corrente del nastro. Nella stampa dell'output si dovranno ignorare tutti i blank a sinistra del primo non-blank e tutti quelli a destra dell'ultimo non-blank sul nastro (ovvero, va stampata solo la parte di nastro effettivamente scritta). In caso di terminazione per timeout, alla stampa del nastro deve essere premessa la stringa "`(TIMEOUT)`".

Formato del programma.txt

Il programma è memorizzato in un file di testo; ciascuna riga rappresenta una quintupla, nel formato

`(`*stato iniziale*`,` *simbolo letto*`,` *stato finale*`,` *simbolo scritto*`,` *spostamento*`)`

Lo *stato iniziale* e lo *stato finale* possono essere stringhe alfanumeriche qualunque (per esempio, `0`, `1`, `fine`, `Avanti`, `riporto2`); il *simbolo letto* e il *simbolo scritto* devono essere singoli caratteri alfanumerici, con la convenzione che "`-`" (trattino) rappresenta un blank, ovvero una cella vuota. Non è possibile usare come simboli i caratteri speciali della sintassi: "`(`", "`)`", "`,`", "`-`". *Avanzato*: implementare un meccanismo di escape basato sull'uso del backslash per esprimere tali caratteri*. Infine, lo *spostamento* è indicato da un carattere "`>`" (sposta la testina di una cella a destra), "`<`" (sposta la testina di una cella a sinistra) o "`-`" (mantieni la testina nella stessa posizione).
Il file di testo può anche contenere righe inizianti per `#`, che rappresentano commenti e vanno ignorate.
*Avanzato**: si estenda la macchina in modo da gestire stringhe di simboli (di uguale lunghezza) come simbolo letto e simbolo scritto. Una regola con stringhe di simboli di lunghezza n è equivalente a n regole, ciascuna delle quali ha come simbolo letto e simbolo scritto i simboli presenti, nella stessa posizione, nelle due stringhe. Per esempio,* `(0,abc,0,xyz,>)` = `(0,a,0,0,>)`, `(0,b,0,y,>)`, `(0,c,0,z,>)`.

Tempi e modalità di consegna

L'elaborato deve essere consegnato **improrogabilmente** entro il 23 giugno 2006 e deve essere costituito da una stampa del codice sorgente, adeguatamente formattato e commentato, da una stampa con esempi di esecuzione, e da una breve relazione scritta (2-3 pagine) che descrive il progetto stesso. La versione a stampa può essere consegnata direttamente al docente, oppure depositata presso il centralino del Dipartimento di Informatica. Tutto il materiale deve essere anche inviato via email al docente (gervasi@di.unipi.it) contestualmente alla consegna della parte cartacea.

Altre informazioni

Il programma deve gestire correttamente le situazioni eccezionali, quali ad esempio mancanza di memoria, impossibilità di accedere a certi file in quanto non esistenti o non dotati dei diritti necessari all'operazione, ecc. Gli eventuali errori devono essere segnalati all'utente attraverso messaggi su standard error. In nessun caso il programma deve inviare su standard output materiale diverso dal contenuto del nastro in uscita.

Esempi

Vediamo in questa sezione alcuni esempi di esecuzione.

Un programma per macchina di Turing che scambia fra di loro le `a` e le `b` presenti sul nastro in ingresso:

File scambio.txt

```
# Scambio a con b
(0,a,0,b,>)
(0,b,0,a,>)
```

Invocazione del simulatore

```
turing scambio.txt babba
```

Output

```
abaab
```

Un programma per macchina di Turing che incrementa un numero binario:

File incrementatore.txt

```
# Incrementatore binario
# a. scorro tutto l'input fino ad arrivare in fondo
(0,0,0,0,>)
(0,1,0,1,>)
# b. quando sono in fondo, cambio stato e torno sull'ultima cifra
(0,-,fondo,-,<)
# c. se l'ultima cifra e' uno 0, lo cambio in 1 e ho finito
(fondo,0,fine,1,-)
# d. se invece l'ultima cifra e' un 1, scrivo 0 e porto 1
(fondo,1,porto1,0,<)
# e. applico il riporto. Si potrebbe riusare il codice precedente,
#    ma allunghiamo pure per arricchire l'esempio...
(porto1,0,fine,1,-)
(porto1,1,porto1,0,<)
# f. se arrivo in cima, il numero era della forma 111...11. Aggiungo
#    una cifra 1 in testa; il resto e' diventato tutti di 0,
#    quindi ottengo 1000...00.
(porto1,-,fine,1,-)
# NOTA: non c'e' nessuna tupla con lo stato fine; il programma termina
#       quando si entra in tale stato
```

Invocazione del simulatore

```
turing incrementatore.txt 10011
```

Output

```
10100
```

Un programma che va in loop su “a”:

File loop.txt

```
(0,a,0,a,-)
```

Invocazione del simulatore

```
turing loop.txt a 3000
```

Output

```
(TIMEOUT)a
```