Algoritmi e Principi dell'Informatica
Appello del 9 Settembre 2019
Chi deve sostenere l'esame integrato (API) deve svolgere tutti gli esercizi in 2 ore.
Chi deve sostenere solo il modulo di *Informatica teorica* deve svolgere gli Esercizi 1 e 2 in 1 ora.
Chi deve sostenere solo il modulo di *Informatica 3* deve svolgere gli Esercizi 3 e 4 in 1 ora.
NB: i punti attribuiti ai singoli esercizi hanno senso solo con riferimento all'esame integrato e hanno valore puramente indicativo.

Esercizio 1 (10 punti)
Si vuole modellare il comportamento di una macchina che vende bottiglie di latte. La macchina accetta 2 tipi di monete: da 50 centesimi e da 1 euro. Nel momento in cui la macchina ha almeno 1 euro di credito, essa produce automaticamente una bottiglia di latte. Dopo che l'utente ha preso il latte, si può ricominciare a inserire monete nella macchina. Inoltre, l'utente può chiedere di avere indietro il resto che è presente in quel momento nella macchina.

Il comportamento della macchina viene modellato descrivendo sequenze di "eventi" rilevanti. Più precisamente, gli eventi di interesse sono i seguenti:

- `in50c`, e `in1`, che corrispondono al fatto che viene inserita una moneta da 50¢ o 1€, rispettivamente;
- `cred0`, e `cred50`, che corrispondono al fatto che il credito presente nella macchina è di 0 o 50, centesimi, rispettivamente (si noti che nel momento in cui si inserisce una moneta che farebbe aumentare il credito fino a essere uguale a o più di 1€, viene erogato il latte, quindi il credito è subito consumato);
- `latte`, che corrisponde al fatto che viene erogato il latte;
- `resto`, che corrisponde al fatto che si richiede il resto presente in quel momento.

Le sequenze di simboli che descrivono il comportamento della macchina sono alternanze di simboli corrispondenti a comandi dati alla macchina (`in50c`, `in1`, `resto`) e simboli che rappresentano il credito residuo (`cred0`, `cred50`). Se però l'inserimento di una nuova moneta porta all'erogazione del latte (perché si raggiungerebbe l'ammontare di 1€), allora il comando di inserimento è subito seguito dal simbolo `latte` che indica l'erogazione del latte, che a sua volta è seguito dall' indicazione del credito residuo. Le sequenze di simboli iniziano e terminano con credito uguale a 0.

Immaginiamo, per esempio, una sequenza in cui l'utente inserisce prima una moneta da 50¢, poi un'altra da 50¢ (a questo punto la macchina avrebbe raggiunto 1€, quindi deve essere erogato il latte e aggiornato il credito), poi una da 50¢, poi una da 1€ (nel qual caso, di nuovo viene subito erogato il latte), poi viene chiesto il resto, poi viene inserita una moneta da 50¢, poi di nuovo viene chiesto il resto, e a questo punto la sequenza termina.
Tale sequenza corrisponde alla seguente sequenza di simboli:

`cred0`, `in50c`, `cred50`, `in50c`, `latte`, `cred0`, `in50c`, `cred50`, `in1`, `latte`, `cred50`, `resto`, `cred0`, `in50c`, `cred50`, `resto`, `cred0`

Si formalizzino in logica i seguenti vincoli e proprietà:

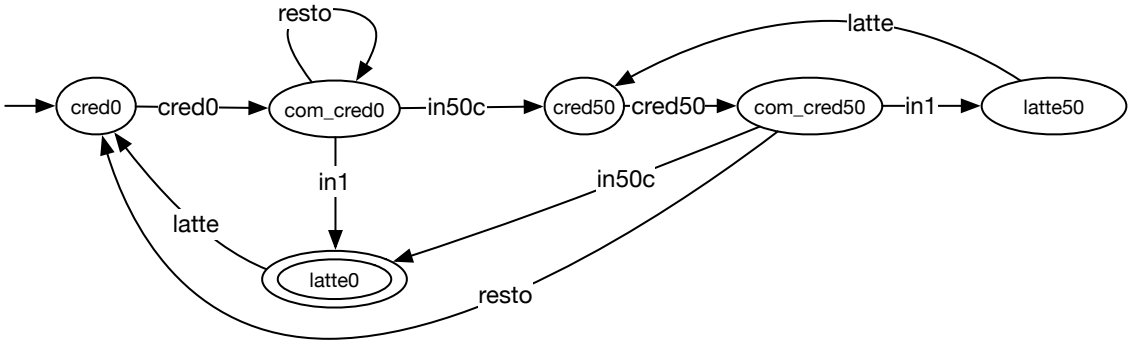
1. All'inizio e alla fine il credito è 0.
2. Un simbolo rappresentante un credito, se non è l'ultimo della sequenza, deve essere subito seguito da un simbolo rappresentante un comando.
3. L'effetto dell'inserimento di una moneta sul credito, cioè come cambia il credito in seguito all'inserimento di una moneta, ed eventualmente l'emissione del latte (si noti che, in caso di emissione del latte, prima viene erogato il latte e poi viene segnalato il nuovo credito).
4. L'effetto di un comando di resto.

Il punteggio sarà tanto più alto, quanto meno potente è il formalismo usato per definire le sequenze.

Esercizio 2 (7 punti)

1. È decidibile il problema di stabilire se, dato un programma C generico, esso è in grado di discriminare tra le sequenze di "eventi" che rispettano i vincoli formalizzati al punto 1, e quelle che non lo sono?

2. È decidibile il problema di stabilire se le sequenze accettate dal seguente automa a stati finiti rispettano il vincolo 2 (cioè il fatto che un simbolo di credito è seguito da un simbolo di comando)?



3. È decidibile il problema di stabilire se le sequenze accettate da un *generico* automa a stati finiti rispettano il vincolo 2 (cioè il fatto che un simbolo di credito è seguito da un simbolo di comando)?

Esercizio 3 (8 punti)

Si consideri la seguente variazione della tradizionale macchina di Turing a k nastri: la macchina non può modificare il contenuto di una cella dopo averci scritto.
Questo tipo di macchina può riconoscere qualsiasi linguaggio riconosciuto da una MT tradizionale? In caso positivo, dette T(n) e S(n) le complessità temporale e spaziale della macchina originaria, con che complessità può effettuare il riconoscimento la macchina "non cancellante"?

Esercizio 4 (8 punti)

Si vuole implementare un vocabolario, ossia un insieme di stringhe *statico*, in cui si effettuano principalmente due operazioni:

1. Exact-Search: data una stringa w, verificare se la stringa è contenuta nel vocabolario
2. Range-Search: date due stringhe w1, w2, restituire tutte le parole w del vocabolario tali che w1 <= w <= w2, dove la relazione d'ordine è rappresentata dal confronto lessicografico tra stringhe

Si progetti una struttura dati per memorizzare le stringhe contenute nel vocabolario e si descrivano gli algoritmi corrispondenti per le operazioni di Exact-Search e Range-Search, considerando questi due casi:

1. Le stringhe w1, w2 per l'operazione di Range-Search devono appartenere al vocabolario
2. Le stringhe w1, w2 per l'operazione di Range-Search possono non appartenere al vocabolario

Si calcoli la complessità degli algoritmi descritti in funzione del numero *n* di stringhe del vocabolario e, nel caso della Range-Search, del numero di stringhe *m* del vocabolario comprese tra w1 e w2. La struttura dati e gli algoritmi scelti devono minimizzare la complessità temporale di entrambe le operazioni di Exact-Search e Range-Search *nel caso medio*; in caso si identifichino più soluzioni che non minimizzano entrambe le complessità, si consideri la soluzione che minimizza il costo dell'operazione più onerosa. Nella scelta della soluzione e nella stima della complessità, non è necessario considerare il costo della costruzione della struttura dati, in quanto si tratta di un'operazione una tantum, né la complessità di eventuali operazioni di modifica del vocabolario, trattandosi di un insieme statico.

Tracce delle soluzioni

Esercizio 1

Le sequenze desiderate si possono definire mediante formule MFO, usando i predicati `in50c(x)`, `credo0(x)`, `latte(x)`, ecc., per esempio nel modo seguente:

1. Inizialmente e alla fine il credito è 0:
`x = 0 ∨ last(x) ⇒ cred0(x)`

2. Un simbolo di credito deve essere seguito da un simbolo di comando:
`y = x + 1 ∧ (cred0(x) ∨ cred50(x)) ⇒ (in50c(y) ∨ in1(y) ∨ resto(y))`

Effetto di un inserimento di moneta:

```
y = x + 1 ⇒
 ( (cred0(x) ∧ in50c(y) ⇒ ∃z ( z = y+1 ∧ cred50(z) ))
   (cred0(x) ∧ in1(y) ⇒ ∃z,w ( z = y+1 ∧ w = z+1 ∧ latte(z) ∧ cred0(w) ))
   (cred50(x) ∧ in1(y) ⇒ ∃z,w ( z = y+1 ∧ w = z+1 ∧ latte(z) ∧ cred50(w) ))
   (cred50(x) ∧ in50c(y) ⇒ ∃z,w ( z = y+1 ∧ w = z+1 ∧ latte(z) ∧ cred0(w) ))
 )
```

Effetto di una richiesta di resto:
`resto(x) ⇒ ∃y ( y = x+1 ∧ cred0(y) )`

Esercizio 2

1. No. Siccome il linguaggio C ha la stessa potenza della MT, questo è un classico caso di applicazione del teorema di Rice, essendo l'insieme delle stringhe che soddisfano i requisiti delle formule logiche un linguaggio decidibile.

2. Questo invece è un caso di problema chiuso, avendo da una parte una singola macchina e dall'altra un singolo linguaggio; quindi indipendentemente dal fatto che l'automa proposto riconosca esattamente l'insieme delle stringhe che soddisfano il requisito 2, cosa in effetti *vera*, anche se l'automa risulta globalmente scorretto rispetto all'insieme completo dei requisiti, il problema è decidibile.

3. In questo caso, anche se l'automa non è fissato ma è un input del problema, il problema è decidibile, perché la logica MFO può essere tradotta in un automa a stati finiti e quindi il problema diventa il problema dell'equivalenza (o del contenimento) tra linguaggi accettati da due automi a stati finiti, notoriamente decidibile.

Esercizio 3

La macchina non cancellante può simulare ogni singola mossa della macchina tradizionale, ricopiando con una passata completa il contenuto dei nastri modificando quello originario in modo da tener conto dell'effetto che avrebbe la mossa della macchina originaria. Ad esempio, si possono utilizzare 2.k nastri in modo che ogni nastro originario abbia una sua copia: ad ogni passata il contenuto originario, marcato da opportuni delimitatori, viene ricopiato in alternanza di seguito al precedente contenuto della "copia gemella". La complessità spaziale diventa quindi il quadrato della complessità originaria: ad ogni passata il contenuto del nastro della macchina simulata, ossia la parte compresa tra due delimitatori del nastro della macchina simulante -di lunghezza m- diventa lungo al massimo m+1 e viene aggiunto al contenuto del "nastro copia" di quello corrente che quindi cresce di m+1. La lunghezza complessiva dl ogni nastro della macchina simulante cresce quindi in modo quadratico rispetto alla crescita della macchina simulata. Lo stesso vale per la complessità temporale, perché ad ogni passata si scandiscono m celle per ogni nastro.

Esercizio 4

Nel primo caso, si può utilizzare una hash table in cui ogni entry contiene un puntatore alla stringa del vocabolario successiva in ordine lessicografico. In particolare, ogni cella della tabella contiene la chiave (che può essere la stringa stessa o un numero associato biunivocamente alla stringa) e un puntatore alla cella che contiene la stringa del vocabolario successiva in ordine lessicografico. Utilizzando questa struttura dati, le ricerche esatte possono essere fatte semplicemente verificando la presenza della stringa nella tabella hash, mentre la Range-Search può essere effettuata trovando w1 nella hash table e seguendo i vari puntatori agli elementi successivi fino a quando non si arriva a w2. Assumendo l'utilizzo di una buona funzione di hash che garantisce un numero limitato di collisioni, si possono effettuare le ricerche di stringhe nella tabella con complessità O(1), per cui le due operazioni hanno costo O(1) e O(m), rispettivamente.

Nel secondo caso, utilizzare la stessa struttura dati renderebbe inefficienti le operazioni di Range-Search, in quanto bisognerebbe scansionare linearmente tutti gli elementi della tabella hash per trovare la prima stringa w > w1. Le operazioni di Range-Search possono essere effettuate efficientemente tramite un albero binario di ricerca bilanciato:

1. Si cerca la stringa w1 nell'albero, includendola nell'insieme di stringhe cercate in caso sia presente
2. Si scansionano tutti i successori di w1 nell'albero fino a quando non si trova un nodo la cui stringa w > w2

La ricerca di w1 costa O(log(n)), mentre la scansione dei successori costa O(m+log(n)). Infatti, sebbene l'algoritmo di calcolo del successore costa O(log(n)) nel caso pessimo, in questo caso applicandolo ripetutamente si visitano al più O(log(n)) nodi che non sono compresi tra w1 e w2, ovvero i nodi antenati di w1 le cui stringhe non sono maggiori di w1. In conclusione, l'operazione di Range-Search può essere effettuata con costo O(m+log(n)); tuttavia il costo della ricerca esatta diventa O(log(n)), dovuto alla ricerca della stringa nell'albero. Si può ritornare a un costo O(1) per la ricerca esatta memorizzando i nodi dell'albero binario in una tabella di hash: in questo caso, ogni cella contiene la chiave e due puntatori, che puntano alle celle della tabella contenenti i figli destro e sinistro nell'albero di ricerca. Mantenendo un puntatore alla cella della tabella che contiene la radice dell'albero, è possibile visitare la tabella come fosse un albero binario di ricerca, preservando l'efficienza delle Range-Search; allo stesso tempo, le ricerche esatte possono essere effettuate con costo O(1) tramite una usuale ricerca della stringa nella tabella hash.