Esame 20240725

Esercizio lode

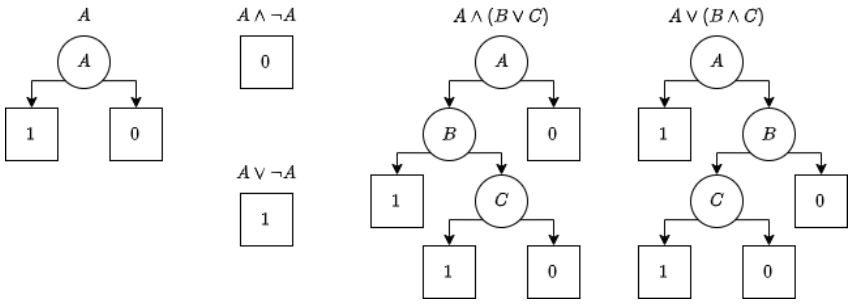
(1) **Esercizio lode**

ESSAY | marked out of 1 | penalty 0 | File picker

In informatica, un albero decisionale ordinato binario è una struttura dati utilizzata per rappresentare una qualunque funzione booleana.

Una qualunque funzione booleana può essere rappresentata come un albero binario costituito da diversi nodi (decisionali) e due nodi terminali. I due nodi terminali sono etichettati 0 (FALSE) e 1 (TRUE). Ciascun nodo (decisionale) u è etichettato da una variabile booleana x_i e ha due nodi figli chiamati figlio di sinistra (o figlio then) e figlio di destra (o figlio else). Seguire il figlio di sinistra (then) rappresenta un'assegnazione del valore TRUE alla variabile x_i, mentre seguire il figlio di destra (else) rappresenta un'assegnazione del valore FALSE alla variabile x_i. Un albero decisionale binario è detto *ordinato* se diverse variabili compaiono nello stesso ordine su tutti i percorsi a partire dalla radice verso le foglie.

Nella figura seguente sono rappresentati diversi alberi decisionali binari ordinati che rappresentano le funzioni booleane A, $A \wedge \neg A$, $A \vee \neg A$, $A \wedge (B \vee C)$, e $A \vee (B \wedge C)$.



Se entrambi i figli di un nodo corrispondente ad una variabile x_i sono nodi isomorfi (ovvero identici), allora la variabile x_i non compare in quella parte dell'albero (si veda l'esempio dell'albero che rappresenta la funzione $A \wedge \neg A$, oppure gli alberi che rappresentano le funzioni $A \wedge (B \vee C)$ e $A \vee (B \wedge C)$).

Dato un albero decisionale ordinato, ed un assegnamento completo alle variabili (ogni variabile ha un valore vero o falso), si può voler valutare se l'assegnamento rende vera la formula Booleana rappresentata dall'albero decisionale. Ad esempio, se $V = \{A, B, C, D\}$, l'assegnamento completo è $A = 1, B = 0, C = 1, D = 1$ e l'albero decisionale è la costante False, allora la valutazione è False, se l'albero è la costante True, allora la valutazione è True, se l'albero è $A \wedge B$, allora la valutazione è False, se l'albero è $A \wedge \neg B$, allora la valutazione è True, mentre se l'albero è $A \vee B$, allora la valutazione è True. L'assegnamento completo $A = 1, B = 0, C = 1, D = 1$ è memorizzato come un array di nodi (**non necessariamente ordinato** rispetto all'ordinamento delle variabili) di dimensione pari alla cardinalità di V (i.e., $|V|$) dove l'elemento i-esimo rappresenta l'albero della variabile diretto se la variabile è considerata True, e negato se la variabile è considerata False.

Nel file `lode.cpp` è presente del codice che definisce una struttura dati `node` che permette di rappresentare un albero decisionale binario ordinato. Il TRUE è rappresentato da un nodo che contiene il carattere '1' e entrambi i figli sono `nullptr`, mentre il FALSE è rappresentato da un nodo che contiene il carattere '0' e entrambi i figli sono `nullptr`. Un nodo generico contiene una lettera dell'alfabeto maiuscola che rappresenta la variabile booleana e due puntatori ai figli. Nel file sono già definite diverse funzioni per manipolare questa struttura dati: `makeNode`,

`makeTrue`, `makeFalse`, `deleteNode`, `printNode`, `isTrue`, `isFalse`, `makeCopy`, `makeVar`, `makeNot`, `getVar`, `getThen`, `getElse`, e `areEquivalent`. La variabile 'A' positiva è rappresentata da un nodo con etichetta 'A' e figlio di sinistra (then) TRUE e figlio destra (else) FALSE, di conseguenza la corrispondente negata ha figlio di sinistra FALSE e figlio destra TRUE.

Si richiede di implementare una nuova funzione ricorsiva `getTruthValue` che prenda come primo argomento un puntatore a `node`,come secondo argomento un array di `node *`, e come terzo argomento in intero n che rappresenta la dimensione dell'array (cardinalità dell'insieme delle variabili con cui costruire gli alberi di decisione ordinati). Tale funzione ritorna `true` se l'assegnamento rende la formula vera, `false` altrimenti.

Il file `lode.cpp` contiene già un `main` con alcuni esempi e alcune invocazioni della funzione `getTruthValue`. Di seguito è riportato l'output di esecuzione.

```
marco > a.out
C=1, B=0, A=1, D=1
A & B := A(B(1,0),0) value = 0
A | B := A(1,B(1,0)) value = 1
A & (B | C) := A(B(1,C(1,0)),0) value = 1
A | (B & C) := A(1,B(C(1,0),0)) value = 1
res[0] := A(1,B(1,0)) value = 1
res[1] := 1 value = 1
res[2] := A(1,B(1,C(1,0))) value = 1
res[3] := A(1,B(1,C(1,D(1,0)))) value = 1
res[4] := A(1,B(1,C(1,0))) value = 1
res[5] := 0 value = 0
res[6] := A(B(1,0),1) value = 0
```

Note:

- Scaricare il file `lode.cpp`, modificarlo per inserire la dichiarazione e la definizione della funzione `makeOperation`, e **caricare il file sorgente risultato delle vostre modifiche a soluzione di questo esercizio** nello spazio apposito.
- All'interno di questo programma **non è ammesso** l'utilizzo di variabili globali o di tipo `static` e di funzioni di libreria al di fuori di quelle definite in `iostream`, `cstdlib`, `cassert`.
- Si ricorda che, gli esempi di esecuzione sono puramente indicativi, e la soluzione proposta NON deve funzionare solo per l'input fornito, ma deve essere robusta a variazioni compatibili con la specifica riportata in questo testo.
- Si ricorda di inserire solo nuovo codice e di **NON MODIFICARE** il resto del programma (pena annullamento dell'esercizio).

lode.cpp

Information for graders:

Total of marks: 1