Esercizi proposti – 11

Prima Parte

1. Sia $F = p \wedge q \rightarrow \neg p \vee r$ e $\mathcal{M}$ l'interpretazione tale che $\mathcal{M}(p) = T$ e $\mathcal{M}(q) = \mathcal{M}(r) = F$. Determinare se $\mathcal{M} \models F$ oppure $\mathcal{M} \not\models F$, giustificando la risposta mediante la definizione ricorsiva di $\models$.

2. Sia $F = (p \rightarrow (q \wedge r)) \vee (q \rightarrow s)$. Definire interpretazioni $\mathcal{M}_1$ e $\mathcal{M}_2$ tali che $\mathcal{M}_1 \models F$ e $\mathcal{M}_2 \not\models F$, e giustificare la definizione delle interpretazioni utilizzando la definizione ricorsiva di $\models$.

3. Dimostrare mediante ragionamento semantico che:

   (a) $q \rightarrow (p \rightarrow r), \neg r, q \models \neg p$

   (b) $p \vee q, p \vee (q \wedge r) \models p \vee r$

   (c) $\neg(p \rightarrow q), p \rightarrow r \vee q \models r$

4. Dimostrare mediante ragionamento semantico che:

   (a) $p \rightarrow q \not\models \neg p \rightarrow \neg q$

   (b) $p \wedge (q \vee r) \not\models \neg(q \rightarrow r)$

Seconda Parte

Per questo gruppo di esercizi si presuppone di aver definito il seguente tipo di dati per rappresentare formule della logica proposizionale (senza la doppia implicazione $\equiv$):

```
type form =
    True
  | False
  | Prop of string
  | Not of form
  | And of form * form
  | Or of form * form
  | Imp of form * form
```

1. La complessità di una formula è il numero di operatori logici ($\neg, \wedge, \vee \rightarrow$) che contiene. Ad esempio, un atomo ha complessità 0, la formula $(p \wedge q) \rightarrow \neg r$ ha complessità 3. Definire una funzione `complessita: form -> int` che calcoli la complessità di una formula.

2. Definire una funzione `mkand: form list -> form`, che, data una lista di formule `[f1;...;fn]` ne riporti la congiunzione: $f_1 \wedge .... \wedge f_n$. L'ordine in cui si associano i congiunti è indifferente: la formula può essere $f_1 \wedge (f_2 \wedge (....(f_{n-1} \wedge f_n)...))$, oppure $((...(f_1 \wedge f_2).... \wedge f_{n-1}) \wedge f_n)$. Se la lista è vuota, la funzione riporterà $\top$.

3. Definire una funzione `mkor: form list -> form` che, applicata a una lista di formule, riporti la disgiunzione di tutte le formule nella lista. Se la lista è vuota, la funzione riporta $\bot$. Anche qui, l'ordine in cui si associano i disgiunti è indifferente.

4. Un *letterale* è un atomo o la negazione di un atomo. Il letterale *complementare* di un atomo p è $\neg p$; il complementare di $\neg p$ è p; il complementare di $\top$ è $\bot$, e viceversa. Scrivere una funzione `complementare: form -> form` che, applicata a un letterale ne riporti il complementare. Se la formula non è un letterale solleverà un'eccezione.

5. Una formula si dice in *forma normale negativa (NNF)* se è ottenuta a partire da letterali applicando soltanto gli operatori $\wedge$ e $\vee$. In altri termini, la formula contiene soltanto gli operatori $\neg$, $\wedge$ e $\vee$ e la negazione domina soltanto atomi. Ad esempio $p \to q$ e $\neg(p \wedge q)$ non sono in NNF, mentre lo sono $\neg p \vee q$ e $p \vee (\neg q \wedge \neg r)$. Scrivere una funzione `test_nnf: form -> bool` che verifichi se una formula è in NNF.

6. Se F e G sono formule in NNF, si dice che F e G sono *duali* l'una dell'altra se F si può ottenere da G sostituendo:

   - ogni $\wedge$ con $\vee$,
   - ogni $\vee$ con $\wedge$,
   - ogni letterale con il suo complementare.

   Scrivere una funzione `duale: form -> form` che, data una formula in NNF, riporti la sua duale. Se la formula non è in NNF, la funzione solleverà un'eccezione.

7. Sia F una *congiunzione di letterali*, cioè una formula della forma $\ell_1 \wedge .... \wedge \ell_n$ dove ogni ℓ_i è un *letterale*. Scrivere una funzione `and2list: form -> form list` che, applicata a una congiunzione di letterali $\ell_1 \wedge .... \wedge \ell_n$ (dove le parentesi possono essere in qualunque modo), riporti la lista dei letterali che la compongono `[`ℓ_1`;....;`ℓ_n`]`. Se la formula non è una congiunzione di letterali, la funzione solleverà un'eccezione.

8. Il controllo di soddifacibilità per congiunzioni di letterali è meno complesso del controllo di soddifacibilità per formule in generale: se $F = \ell_1 \wedge .... \wedge \ell_n$, F è soddisfacibile se e solo se l'insieme $\{\ell_1, ..., \ell_n\}$ non contiene alcuna coppia di letterali complementari (cioè se nessun ℓ_i è il complementare di qualche ℓ_j), e, ovviamente, nessun ℓ_i è $\bot$ o $\neg\top$. Scrivere una funzione `satxand_of_lits: form -> bool` che, applicando questo metodo, controlli se una congiunzione di letterali è soddisfacibile o no. Se la formula cui viene applicata non è una congiunzione di letterali, la funzione solleverà un'eccezione.

9. A partire da un'interpretazione $\mathcal{M}$ si può costruire una congiunzione di letterali F che è vera in $\mathcal{M}$ e soltanto in $\mathcal{M}$: per ogni atomo p del linguaggio, F contiene p se p è vera in $\mathcal{M}$, e $\neg p$ altrimenti. Diciamo in questo caso che F rappresenta $\mathcal{M}$.

   Rappresentiamo un'interpretazione mediante una lista associativa di tipo `(string * bool) list`: ad ogni atomo del linguaggio (identificato dalla stringa che è il suo "nome") è associato il suo valore di verità.

   Dichiariamo dunque un tipo per rappresentare le interpretazioni:

```
type interpretation = (string * bool) list
```

Scrivere una funzione `int2form: interpretation -> form` che, data un'interpretazione così rappresentata, costruisca la formula che la rappresenta.

Ad esempio, il valore di `int2form [("p",true); ("q", false); ("r",true)]` sarà una `form` che rappresenta la formula $p \wedge \neg q \wedge r$ (dove l'ordine in cui si associano i congiunti è indifferente).

10. Definire una funzione `dnf: form -> form`, che trasformi una formula in una forma normale disgiuntiva (DNF) equivalente.

    È possibile utilizzare uno degli algoritmi seguenti:

    - una algoritmo analogo a quello utilizzato nell'implementazione della funzione `cnf` vista a lezione (si trasforma la formula in FNN e si applicano le leggi distributive).
    - Costruire l'insieme delle interpretazioni in cui la formula è vera e, per ciascuna di esse, costruire la formula che la rappresenta; infine, costruire la disgiunzione dell'insieme di formule che si ottengono.
    - Costruire un tableau completo per la formula data e collezionarne i rami aperti. Per ciascuno di essi, costruire la congiunzione dei letterali del ramo ed infine, costruire la disgiunzione di tali congiunzioni.

11. Una forma normale congiuntiva (CNF) di una formula F si può ottenere utilizzando il metodo dei tableaux: si raccolgono i complementi dei letterali dei rami aperti in un tableau completo per $\neg F$, ottenendo una lista di liste di letterali $[\texttt{flist}_1;\ldots;\texttt{flist}_n]$; da ciascuna delle liste $\texttt{flist}_i$ si ottiene la formula D_i costituita dalla disgiunzione dei letterali in $\texttt{flist}_i$; infine, si costruisce la congiunzione di tali disgiunzioni $D_1 \wedge \ldots \wedge D_n$.

    Scrivere una funzione `tab_cnf: form -> form` che riporti la forma normale disgiuntiva di una formula seguendo tale metodo.

12. Utilizzando la funzione `sat` vista a lezione, scrivere una funzione `logical_consequence: form list -> form -> bool` che, applicata ad una lista di formule `flist` e una formula `f`, determini se `f` è una conseguenza logica di `flist`.