LEZIONI 30/09/09 - 2/10/09

1. Introduzione

La *Logica* viene usata in Informatica in tutte quelle situazioni in cui un sistema informatico viene modellato da un oggetto matematico e si vuole ragionare in modo formale sulle proprietà del sistema. Ragionare può significare, ad esempio:

(1) verificare determinate proprietà del sistema;
(2) descrivere comportamenti comuni caratterizzanti una classe di sistemi;
(3) stabilire se certi comportamenti di un sistema ne implichino altri etc.

Il termine sistema informatico è lasciato volutamente vago e comprende sia hardware che software: ad esempio lo stesso tipo di strumenti logici puó essere utilizzato sia nell'analisi di un circuito digitale che nella verifica della correttezza di un programma.

La logica è un linguaggio che permette di formalizzare le proprietà delle strutture matematiche: se noi modelliamo un sistema informatico con una struttura matematica, le proprietà del sistema possono essere espresse nel linguaggio logico:

Realtà	Modello Formale
sistema informatico	struttura matematica M
proprietà	formula F
verifica di proprietà del sistema	F è vera in M?
insieme di proprietà del sistema	insieme di formule Γ
classe di sistemi	classe di strutture $\mathcal{C}$
comportamenti comuni caratterizzanti una classe di sistemi	$M \in \mathcal{C} \Leftrightarrow M$ verifica Γ
comportamenti di un sistema implicano altri	Γ ha conseguenza logica F

Esempio 1.1. *Consideriamo il programma* P_1*:*

(1) $v := 0, w := 1$*;*
(2) while $w \neq 0$ *do* $v := v + w \ (mod\ 4)$, $w := w + 2 \ (mod\ 3)$*;*
(3) while $w < 3$ *do* $w := w + v$, $v := v + v \ (mod\ 5)$*;*
(4) goto 2;

Possiamo rappresentare le possibili computazioni del programma con il grafo etichettato

$$\{v = 0, w = 1\} \to \{v = 1, w = 0\} \to \{v = 2, w = 1\} \to \{v = 4, w = 3\} \to \{v = 3, w = 2\} \to \dots$$

rappresentabile anche più schematicamente così:

$$(0, 1) \to (1, 0) \to (2, 1) \to (4, 3) \to (3, 2) \to \dots$$

o anche

$$s_1 \to s_2 \to s_3 \to s_4 \to s_5 \to \dots$$

(continua...)

Più in generale, una struttura adatta alla modellizzazione delle possibili computazioni di un programma su un insieme di variabili $v_1, v_2, \ldots$ è dato da un grafo etichettato $G = (S, R)$ in cui l'insieme S nodi rappresenta i possibili stati che le variabili assumono durante una computazione, mentre gli archi rappresentano le transizioni del programma.

A seconda delle proprietà del programma che si è interessati a verificare, si considerano un certo numero di predicati unari che rappresentino proprietà atomiche degli stati.

Esempio 1.2. *(Esempio* **??** *continua) Nel programma precedente, possiamo considerare la proprietà: la somma delle variabili è pari ad* 1 *indicandola con l'etichetta* $sum_{=1}$*. Nel grafo che rappresenta il programma,* $sum_{=1}$ *vale ad esempio per gli stati* s_1, s_2 *mentre non vale per* s_3, s_4, s_5*.*

In altre parole, le computazioni del nostro programma sono rappresentate dal grafo

$$s_1 \to s_2 \to s_3 \to s_4 \to s_5 \to \ldots$$

in cui alcuni nodi (tra cui i nodi s_1, s_2) vengono etichettati con $sum_{=1}$.

Questa struttura matematica si chiama *grafo etichettato* e può essere vista nel formalismo logico come un'interpretazione del linguaggio predicativo $\mathcal{L} = \{a, r, sum_{=1}\}$, dove a è un simbolo di costante, r è un predicato binario e $sum_{=1}$ è un predicato unario.

Abbiamo:

$$D^I = \{(i,j) : i,j \in \mathbb{N}\}, \quad a^I = (0,1), \qquad sum^I_{=1} = \{(i,j) \in D^I : i + j = 1\},$$
$$r^I = \{(s,t) \in D^I : s \text{ si trasforma in } t \text{ con un passo del programma}\}.$$

In generale, dato un programma ci chiederemo: quale linguaggio formale è adatto allo studio delle proprietà del programma? Quale logica ci permette di esprimere le proprietà che ci interessa verificare?

Esempio 1.3. *(Esempio* **??** *continua)*

Esempi di proprietà del programma P_1 *che vogliamo verificare o confutare:*

(1) la somma del valore iniziale delle variabili è 1*;*
(2) dopo due transizioni dallo stato iniziale la somma delle variabili raggiunge un valore diverso da 1*;*
(3) in una computazione del programma, a parte i primi due passi, non si raggiunge mai più uno stato con somma 1*;*
(4) il numero di stati raggiungibili dal programma è infinito.

Queste proprietà sono vere o false nel modello considerato?

Notiamo che, proseguendo con lo sviluppo della computazione di P_1 si ottiene:

$$(0,1) \to (1,0) \to (2,1) \to (4,3) \to (3,2) \to (1,1) \to (2,0) \to (4,2) \to$$
$$\to (3,6) \to (5,2) \to (3,1) \to (0,0) \to (0,0) \to \ldots$$

Quindi, le prime tre proprietà sono vere, l'ultima è falsa.

Piĭn generale come possiamo verificare in modo automatico la validità o meno di una proprietà?

Le prima proprietà da verificare è tradotta dalla formula proposizionale (cioè: priva di quantificatori)

$$sum_{=1}(a).$$

La veritá della prima proprietà per P_1 corrisponde infatti a

$$I \models sum_{=1}(a).$$

Se invece consideriamo il programma P_2 che differisce da P_1 solo nello stato iniziale $(1,1)$ otteniamo che per la rappresentazione J varrà

$$I \models \neg sum_{=1}(a).$$

La seconda proprietà è traducibile con la formula predicativa (al prim'ordine, cioè con quantificazioni sulle variabili individuali)

$$\forall s \forall t(r(a,s) \wedge r(s,t) \rightarrow \neg sum_{=1}(t)).$$

Come facciamo ad esprimere la terza proprietà in un linguaggio logico?

Questo esempio esempio ci mostra come, quando vogliamo risolvere problemi di questo tipo utilizzando la logica, ci si imbatte facilmente nelle seguenti domande:

(1) Quali strutture matematiche utilizzare per modellare il sistema?
(2) In quale logica formale riesco ad esprimere le proprietà del sistema ?
(3) Possiamo verificare la verità della formula nell'interpretazione data in un tempo finito (ragionevole?)
(4) Da un insieme di (formule che rappresentano) proprietà del sistema, abbaimo un procedimento che ci permette di derivarne altre?

Nel caso dei programmi (e di molti altri sistemi informatici) i grafi etichettati (detti anche Sistemi di Transizione Etichettati, o Labeled Transition Systems, o modelli di Kripke) sono largamente utilizzati per la modellizzazione dei sistemi. Le logiche utilizzate per esprimere le proprietà di tali sistemi invece variano notevolmente. Più in generale, l'uso della logica come strumento per l'Informatica Teorica ha favorito la nascita e l'affermazione o riaffermazione di Logiche *non classiche* che spesso sono risultate insieme più *trattabili* e/o più espressive delle Logiche che le hanno precedute.

Un esempio notevole è dato dalla Logica Modale, che, nata in ambito filosofico e di teoria del linguaggio, ha avuto un enorme sviluppo grazie alle sue applicazioni informatiche negli ultimi 20 anni.

In questo corso tratteremo principalmente le logiche classiche, incominciando con un breve ripasso di Logica Proposizionale e Predicativa (vedi dispense del Prof. Marcone di Elementi di Logica Matematica [ElLog], capitoli 1,2,3,5,6).

Considereremo poi il problema della conseguenza logica fra formule (quindi soddisfacibilità e validità), e la possibilità di trovare algoritmi efficienti per risolvere tali problemi. Ci interesseremo inoltre a problemi di espressività di queste logiche, e ai fondamenti teorici di un linguaggio di programmazione basato sulla logica predicativa.

Bibliografia per i Prerequisiti al corso

[ElLog] Alberto Marcone, Dispense di Elementi di Logica Matematica, reperibile alla pagina web: http://users.dimi.uniud.it/ alberto.marcone/ElLogica.html

2. Linguaggi Formali ed Esprimibilità di Proprietà

Per la teoria, vedi [ElLog].

Esercizio 2.1. *Dato il programma P_1 dell'esempio* **??***, ed il grafo che lo rappresenta, considera il linguaggio proposizionale $\mathcal{P} = \{p_1, p_2, \ldots\}$. Possiamo utilizzare il linguaggio proposizionale per esprimere alcune proprietà dello stato iniziale del sistema. Ad esempio, possiamo utilizzare la*

variabile p_1 per esprimere la proprietà:
la somma delle variabili nello stato iniziale è maggiore di 8,
mentre possiamo utilizzare la variabile p_2 per esprimere la proprietà
la somma delle variabili nello stato iniziale è minore di 7
Interpretando così le variabile p_1, p_2 nel nostro grafo

$$(0,1) \to (1,0) \to (2,1) \to (4,3) \to (3,2) \to \dots$$

otteniamo la seguente valutazione v sulle variabili p_1, p_2:

$$v(p_1) = \mathbf{F}, v(p_2) = \mathbf{T}.$$

Con il linguaggio proposizionale possiamo allora esprimere le proprietà seguenti (vere o false che siano):

(1) la somma delle variabili nello stato iniziale è minore o uguale ad 8*;* $\neg p_1$*;*
(2) se la somma ... è minore di 7 allora non può essere maggiore di 8*;* $p_2 \to \neg p_1$*;*
(3) non vale che, se la somma è maggiore o uguale a 7 o minore uguale ad 8 allora è minore di 7 e maggiore di 8*;* $\neg(\neg p_2 \vee \neg p_1 \to \neg p_2 \wedge p_1)$.

Esercizio 2.2. *Sia $G = (V, E)$ un grafo non orientato con n nodi. Esprimere con una formula F di un linguaggio proposizionale la proprietà G* è 4-colorabile*, dove G si dice 4-colorabile se è possibile colorare i nodi del grafo con 4 colori diversi in modo che nodi adiacenti abbiano colore diverso. Suggerimento: usare l'insieme di variabili proposizionali*

$$\mathcal{P} = \{R_1, \dots, R_n, V_1, \dots, V_n, G_1, \dots, G_n, B_1, \dots, B_n\}$$

dove, ad esempio, la variabile R_i esprime la proprietà il vertice i è rosso.

Soluzione:

$$\bigwedge_{i=1}^{n}(R_i \vee V_i \vee G_i \vee B_i) \wedge (R_i \to \neg(V_i \vee G_i \vee B_i) \wedge (V_i \to \neg \dots) \wedge \bigwedge_{(v_i,v_j)\in E}(R_i \leftrightarrow \neg R_j) \wedge (V_i \leftrightarrow \neg V_j) \dots .$$

Se il grafo fosse infinito, non potremmo più usare la logica proposizionale per esprimere la 4 colorabilità del grafo.

2.1. **Esempi di LinguaggiProposizionali e al Prim'ordine.**

- Consideriamo una scacchiera ed i pezzi del gioco degli scacchi. Per ogni pezzo (ad esempio: la regina o il primo pedone) e per ogni prosizione (ad esempio: B5), usiamo una variabile proposizionale per esprimere la proprietà *il pezzo dato è nella posizione data* (ad esempio: Reg_{B5} oppure Ped^1_{B5}). Ogni posizionamento dei pezzi nella scacchiera corrisponde ad una valutazione delle variabili proposizionali date, seguendo la regola:
  il pezzo P è nella posizione Xy se e solo se $v(P_{Xy}) = V$
  Ad esempio, se la regina è in $B5$ avrò $v(Reg_{B5}) = V, v(Reg_{A2}) = F, \dots$.
  Non a tutte le valutazioni però corrisponde un posiizonamento di pezzi. Ad esempio alla valutazione $v(Reg_{B5)} = V$ e $v(Reg_{A2}) = V$ non corrisponde una reale disposizione di pezzi, visto che abbiamo una sola regina da sistemare.
  Possiamo scrivere una formula proposizionale A tale che

$$v \models A \Leftrightarrow v \text{ corrisponde ad un posizionamento dei pezzi sulla scacchiera.}$$

Possiamo anche scrivere una formula che esprima la proprietà: la regina non fa scacco al re in una mossa (in questo caso non ci interessa la posizione degli altri pezzi, cioè la valutazione sulle variabili che non parlano di re o regina).

- Dato il linguaggio predicativo $\mathcal{L} = \{a, r, p, f\}$, con a costante, r relazione binaria, p relazione unaria, f funzione unaria, esprimere le seguenti proprietà:
  (1) (con una variabile libera x): x è un nodo r adiacente ad a;
  $r(a, x)$;
  (2) (con una variabile libera x): da x parte un r ciclo di lunghezza 3;
  $\exists y \exists z(r(x, y) \wedge r(y, z) \wedge r(y, x))$;
  (3) r non ha cicli di lunghezza minore di 3;
  $\forall x \neg A(x)$, dove $A(x)$ è la formula del punto precedente.
  (4) ogni elemento è in relazione con la sua immagine tramite f;
  $\forall x r(x, fx)$;
  (5) la controimmagine tramite f di p è contenuta in p;
  $\forall x(p(fx) \rightarrow px)$;
  (6) tutti i punti sono raggiungibili da a con un r cammino di lunghezza minore di 3;
  $\forall x(x = a \vee r(a, x) \vee \exists y(r(a, y) \wedge r(y, x)))$;
  (7) a partire da a possiamo raggiungere un punto in cui vale p in meno di 3 passi;
  $p(a) \vee \exists x(p(x) \wedge r(a, x)) \vee \exists x \exists y(p(y) \wedge (r(a, x) \wedge r(x, y)))$.

  Vedremo più avanti che non è possibile trovare formule al prim'ordine per esprimere le proprietà:
  (1) r non ha cicli;
  (2) tutti gli r cammini sono finiti;
  (3) ci sono infiniti punti;
  (4) a partire da a non è possibile raggiungere in un numero finito di passi un punto in cui vale p.

2.2. **ESERCIZI.**

(1) Scrivere una formula nel linguaggio predicativo contenente un simbolo di relazione unario p e un simbolo funzionale unario f che esprima la proprietà: *p contiene l'immagine di f.*

(2) Scrivere una formula della logica predicativa (utilizzando solo il simbolo relazionale binario $=$) che esprima (in tutte le interpretazioni dove $=$ è interpretato correttamente) la proprietà: *il dominio contiene esattamente tre elementi.*

(3) Dato il linguaggio $\{a, S, g\}$ dove a è una costante S un simbolo funzionale unario e g un simbolo funzionale binario, considerare l' interpretazione I tale che: D^I è l'insieme dei numeri naturali;
$a^I = 0$,
$S^I(n) = n + 1$,
$g(n, m) = nm$, il prodotto di n ed m.
Scrivere una formula $A(x)$ nel linguaggio $\{a, S, g\}$ (utilizzando anche $=$) con una variabile libera x tale che:

$$I, \sigma \models A(x) \qquad \Leftrightarrow \qquad \sigma(x) \text{ è un numero primo.}$$

(4) Una formula predicativa si dice soddisfacibile se esiste un' interpretazione che la rende vera.
Dimostrare che le formule seguenti non sono soddisfacibili (senza usare le tavole semantiche, ma ragionando informalmente sulla nozione di verità)

i) $SIMM \wedge TRANS \wedge \forall x \exists y r(x,y) \wedge \exists x \neg r(x,x)$, dove $SIMM := \forall x \forall y (r(x,y) \rightarrow r(y,x))$ esprime la simmetria della relazione r e $TRANS := \forall x \forall z (r(x,y) \wedge r(y,z) \rightarrow r(x,z))$ esprime la transitività della relazione r.

ii)
$$\forall x \forall y (r(x,y) \rightarrow p(y)) \wedge \exists x r(x, fx) \wedge \neg \exists y p y.$$

iii) $\exists x \forall y (r(x,y) \leftrightarrow \neg r(y,y))$.

iv) Dimostrare che la formula seguente è soddisfacibile, fornendo un esempio di interpretazione che la rende vera.
$$\exists x \forall y (r(x,y) \leftrightarrow r(y,y)).$$

v) Dimostrare che la formula seguente non è soddisfatta da alcun modello finito. Trovare inoltre una interpretazione che la soddisfi.
$$TRANS \wedge \forall x \exists y r(x,y) \wedge \neg \exists x r(x,x).$$

3. Ripasso di Logica Proposizionale

4. Semantica

Definitione 4.1. *Sia v una* valutazione *proposizionale, cioè una funzione dall'insieme delle variabili proposizionali $P_1, \ldots, P_n, \ldots$ all 'insieme dei valori di verità $\{V, F\}$. La funzione v si estende alle formule proposizionali utilizzando le definizioni viste nel corso di Elementi di Logica.*

(1) Una valutazione v è modello *di una formula α se $v(\alpha) = V$; utilizzeremo anche la notazione:*
$$v \models \alpha$$
come sinonimo di $v(\alpha) = V$

(2) una formula si dice soddisfacibile *se ha un modello,* insoddisfacibile *se non ne ha;*

(3) α è valida *se, per ogni valutazione v vale $v(\alpha) = V$ (in ambito proposizionale, le formule valide si chiamano anche* tautologie*);*

(4) un insieme di formule Σ si dice soddisfacibile *se esiste un'interpretazione v tale che $v(\alpha) = V$, per ogni $\alpha \in \Sigma$;*

(5) un insieme di formule Σ si dice insoddisfacibile *se non è soddisfacibile.*

Esempio
Sia v una valutazione tale che $v(A) = V, v(B) = v(C) = v(D) = V$. Se α è la formula $\alpha = (\neg A \vee B) \rightarrow C \wedge D$ allora $v(\alpha) = V$ e v è modello di α.
La formula α è quindi soddisfacibile, mentre la formula $A \wedge \neg A$ non lo è, visto che tutte le possibili valutazioni la rendono falsa.
L'insieme $\Sigma = \{P, \neg P \rightarrow P, Q \vee \neg P\}$ è soddisfacibile, ed un suo modello è dato da ogni valutazione che rende vere sia P che Q.
La formula $\alpha = ((A \rightarrow B) \rightarrow A) \rightarrow A$ è valida: data una qualsiasi valutazione v, se $v(A) = V$ allora v rende vero il conseguente dell'implicazione α e quindi rende vera α; se invece $v(A) = F$, allora $(A \rightarrow B)$ è vero e quindi è falso l'antecedente $(A \rightarrow B) \rightarrow A)$ di α (perché è vero $A \rightarrow B$,

ma falso A). Ne segue che $v(\alpha) = V$.
L'insieme

$$\Sigma = \{P, P \to Q, \neg Q\}$$

è insoddisfacibile.

Dato un insieme insoddisfacibile Σ, ogni insieme Σ' che lo contiene è ancora insoddisfacibile.

Per costruire un insieme infinito insoddisfacibile è allora sufficiente partire da un insieme finito insoddisfacibile e completarlo con un arbitrario insieme infinito di formule. Il Teorema di compattezza, di cui vedremo la dimostrazione nelle prossime lezioni, ci assicura che questa condizione è anche necessaria: ogni insieme infinito di formule, se è insoddisfacibile, deve contenere un sottoinsieme finito di formule che è a sua volta insoddisfacibile.

Definitione 4.2. Equivalenza Logica

(1) *Due proposizioni α, β sono* logicamente equivalenti *(notazione: $\alpha \equiv \beta$) se ogni modello di α è modello di β e viceversa, ogni modello di β è modello di α;*
(2) *una formula α ha come* conseguenza logica *una formula β (notazione $\alpha \models \beta$) se ogni modello di α è anche modello di β;*
(3) *un insieme di formule Σ ha come conseguenza logica una formula β (notazione $\Sigma \models \beta$) se ogni modello di Σ è anche modello di β, cioè, se per ogni valutazione v con $v(\alpha) = V$ per ogni $\alpha \in \Sigma$ si ha $v(\beta) = V$.*

Se $\Sigma = \{\alpha_1, \ldots, \alpha_n\}$ è un insieme finito di formule, allora $\alpha_1, \ldots, \alpha_n \models \beta$ sta per $\Sigma \models \beta$.

NOTA BENE Per ogni coppia di proposizioni α, β vale:

$$\alpha \equiv \beta \quad \Leftrightarrow \quad \alpha \leftrightarrow \beta \text{ è una tautologia;}$$
$$\alpha \models \beta \quad \Leftrightarrow \quad \alpha \to \beta \text{ è una tautologia;}$$

Tutte le valutazioni soddisfano (sono modelli) dell'insieme vuoto di formule. Ne segue che

$$\emptyset \models \alpha \quad \Leftrightarrow \quad \alpha \text{ è una tautologia.}$$

Scriveremo anche $\models \alpha$ al posto di $\emptyset \models \alpha$.

Esempio

$$\begin{array}{rcl} (A \vee B) \wedge C & \equiv & (A \wedge C) \vee (B \wedge C) \\ A \to B & \equiv & \neg A \vee B \\ \neg(A \to B) & \equiv & A \wedge \neg B \end{array}$$

4.1. **Esercizi.** Dimostrare che:

(1) $\Sigma \models \alpha \quad \Leftrightarrow \quad \Sigma \cup \{\neg\alpha\}$ è insoddisfacibile;
(2) se $\Sigma = \{\alpha_1, \ldots, \alpha_n\}$ è un insieme finito di formule, allora

$$\Sigma \models \alpha \quad \Leftrightarrow \quad \models \alpha_1 \wedge \ldots \alpha_n \to \alpha \quad \Leftrightarrow \quad \models \alpha_1 \to (\alpha_2 \to (\ldots (\alpha_n \to \alpha) \ldots).$$

(3) Verificare la validità o meno delle formule

$$A \to (B \to A \wedge B), \quad (A \to B) \to (\neg A \to \neg B).$$