Logica proposizionale vs. logica del prim'ordine

La logica proposizionale ha pregi e limitazioni:

- Permette un'analisi abbastanza semplice delle deduzioni logiche.
- Quest'analisi è implementabile algoritmicamente: i connettivi sono *porte logiche*.
- La validità di un ragionamento è controllabile mediante le tavole di verità.
- L'analisi è applicabile a una varietà di situazioni discretamente ampia.
- **Tuttavia** vi sono numerosi esempi di deduzione che non sono modellabili dalla logica proposizionale.

Esempio.
Chi tace acconsente. I colleghi della madre di Pino tacciono, quindi acconsentono.

Logica proposizionale vs. logica del prim'ordine

Per analizzare ragionamenti di questo tipo si utilizzano strumenti sintattici più complessi:

- *connettivi e parentesi* come in logica proposizionale.
- *virgole*.
- *variabili* $x, y, \ldots$ per denotare elementi generici del dominio in considerazione.
- *quantificatori* $\exists$ e $\forall$ per indicare per quanti oggetti vale un'affermazione (qualcuno, tutti).

Nota. Le parentesi e le virgole non sono necessarie. Si può introdurre una sintassi senza parentesi e virgole, ma la lettura delle formule sarebbe molto più complicata.

Logica proposizionale vs. logica del prim'ordine

- *simboli di relazione* per rappresentare proprietà (ovvero relazioni) di e tra oggetti del dominio
  Esempi.
  - $T(x)$: x tace
  - $A(x)$: x acconsente
  - $C(x, y)$: x è collega di y

  Nota. Tra i simboli di relazione c'è sempre il simbolo binario $=$, che è sempre interpretato come l'uguaglianza tra i termini a cui è applicato.
- *simboli di costante* per designare elementi specifici del dominio.
  Esempio.
  - p: Pino
- *simboli di funzione* per rappresentare operazioni (ovvero funzioni) a uno o più argomenti definite sul dominio.
  Esempio.
  - $m(x)$: la madre di x

Linguaggi del prim'ordine

Avendo a disposizione i simboli degli esempi precedenti, l'asserzione

> *Chi tace acconsente. I colleghi della madre di Pino tacciono, quindi acconsentono.*

si può formalizzare come

$$\forall x(T(x) \to A(x)) \wedge \forall x(C(x, m(p)) \to T(x)) \to \forall x(C(x, m(p)) \to A(x))$$

Dalla correttezza di questo ragionamento (cioè dalla validità della formula) la logica del prim'ordine permette di ottenere la validità di tutte le asserzioni che possono essere formalizzate nello stesso modo.

Esempio

Tutti i gatti sono quadrupedi, gli amici del panettiere di Gino sono dei gatti, quindi gli amici del panettiere di Gino sono quadrupedi.

- $T(x)$: x è un gatto
- $A(x)$: x è un quadrupede
- $C(x, y)$: x è amico di y
- p: Gino
- $m(x)$: il panettiere di x

$$\forall x(T(x) \to A(x)) \land \forall x(C(x, m(p)) \to T(x)) \to \forall x(C(x, m(p)) \to A(x))$$

Esempio

Nell'insieme dei numeri interi $\mathbb{Z}$ la differenza di due numeri esiste sempre.

Per esprimere con una formula questa asserzione si può usare un *linguaggio* che consiste di un simbolo di funzione binaria $+$, la cui interpretazione è:

- $x + y$: la somma di x e y

L'asserzione si può esprimere allora come:

$$\forall x \forall y \exists z \; x = y + z$$

Note

- Essendo $+$ un simbolo di funzione, la sua applicazione a degli argomenti dovrebbe scriversi $+(x, y)$. Tuttavia per i simboli di binari (di funzione o di relazione) si preferisce spesso nella pratica scrivere il simbolo di funzione in mezzo ai due argomenti: $x + y$.
- A differenza dell'esempio precedente, l'enunciato

$$\forall x \forall y \exists z \; x = y + z$$

  non è valido (cioè vero in tutti i contesti possibili). Infatti:
  - È vero in $\mathbb{Z}$ interpretando $+$ con l'operazione di somma
  - È falso in $\mathbb{N}$ interpretando $+$ con l'operazione di somma
  - È falso in $\mathbb{Z}$ interpretando $+$ con l'operazione di moltiplicazione

Una logica del prim'ordine estende quindi la logica proposizionale e consiste:

- nella scelta di un linguaggio L che contenga i simboli di relazione, di costante e di funzione adatti per trattare un particolare problema;
- nello stabilire una **sintassi** per costruire (algoritmicamente) le formule;
- nel definire la nozione **semantica** di *modello*, cioè un contesto in cui interpretare le affermazioni e dotarle di significato.

Sintassi della logica del prim'ordine

Si fissa una collezione di *simboli logici*: sono simboli presenti in tutti i linguaggi del prim'ordine e che hanno lo stesso significato in qualunque linguaggio del prim'ordine:

- parentesi: $(,)$
- virgola: $,$
- connettivi: $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$
- quantificatori: $\exists, \forall$
- simbolo d'uguaglianza: $=$
- un insieme infinito di simboli, detti *variabili*: $Vbl = \{v_0, v_1, v_2, \ldots\}$

Nota: le metavariabili

In genere, non è essenziale quali siano le variabili di questa lista effettivamente usate in una data situazione, a patto di essere consistenti.

Pertanto si useranno simboli come $x, y, z, x_0, x', \ldots$ per designare delle variabili generiche. Questi nuovi simboli **non** fanno parte del linguaggio, servono solo per indicare che si stanno usando delle variabili (distinte) prese dalla lista *Vbl*.
Tecnicamente, questi simboli $x, y, z, x_0, x', \ldots$ si chiamano *metavariabili*, perché tengono il posto di variabili generiche.

Per esempio, nella formula

$$\forall v_5 \forall v_3 \exists v_7 \ v_5 = v_3 + v_7$$

non è importante che si siano usate le variabili v_3, v_5, v_7.
Per parlare di questa formula (come di tutte le altre scritte usando altre variabili invece di queste tre) si scriverà

$$\forall x \forall y \exists z \ x = y + z$$

intendendo che x, y, z siano *variabili distinte*, cioè elementi distinti dell'insieme *Vbl* (anche se possono designare lo stesso elemento del dominio che si sta considerando).

Linguaggi del prim'ordine

Un linguaggio L *del prim'ordine* consiste in

- Un insieme di *simboli di costante*: $Const = \{a, b, c, d, e, \ldots\}$
- Un insieme di *simboli di funzione*: $Funct = \{f, g, h, \ldots\}$
- Un insieme di *simboli di relazione* (*simboli di predicato*): $Rel = \{P, Q, R, \ldots\}$

Quindi

$$L = Const \cup Funct \cup Rel$$

A ogni simbolo di funzione e simbolo di relazione è associato un numero detto *arità*, denotato ar: è il numero di argomenti a cui quel simbolo si applica.

- L'arità del simbolo di funzione f si indica $ar(f)$
- L'arità del simbolo di relazione P si indica $ar(P)$

Esempi

I seguenti sono esempi di linguaggi del prim'ordine:

- $\emptyset$
- $\{R, f, g, a, c\}$, dove R è simbolo di relazione binario, f è simbolo di funzione unario, g è simbolo di funzione binario, a, c sono simboli di costante
- $\{P\}$, dove P è simbolo di relazione ternario
- $\{*, \preceq\}$ dove $*$ è simbolo di funzione binario, $\preceq$ è simbolo di relazione binario
- $\{+, \leq\}$, dove $+$ è simbolo di funzione binario, $\leq$ è simbolo di relazione binario
- $\{+, \cdot, \leq, 0, 1, 2, 3, \ldots\}$, dove $+, \cdot$ sono simboli di funzione binari, $\leq$ è simbolo di relazione binario, $0, 1, 2, 3, \ldots$ sono simboli di costante

Termini

I *termini* sono espressioni che servono per designare *individui*, cioè elementi del dominio a cui si è interessati.

I termini sono definiti induttivamente:

- Ogni variabile è un termine
- Ogni simbolo di costante è un termine
- Se $t_1, t_2, \ldots t_k$ sono termini e se f è un simbolo funzionale k-ario, allora $f(t_1, \ldots, t_k)$ è un termine

Termini

Questo significa che, dato un linguaggio $L = Const \cup Funct \cup Rel$, l'insieme *Term* dei termini di L è definito induttivamente nel modo seguente:

- $Term_0 = Vbl \cup Const$
- $Term_{n+1} = Term_n \cup \{f(t_1, \ldots, t_k) \mid f \in Funct; ar(f) = k; t_1, \ldots, t_k \in Term_n\}$

$$Term = \bigcup_{n \in \mathbb{N}} Term_n$$

Se t è un termine, il minimo n tale che $t \in Term_n$ si dice *altezza* di t, e si denota $ht(t)$.

Esempi

Sia $L = \{R, f, g, c\}$, dove

- R è simbolo relazionale binario
- f è simbolo funzionale unario
- g è simbolo funzionale binario
- c è simbolo di costante

Allora:

- $Term_0 = \{c, v_0, v_1, v_2, v_3, \ldots\}$
- $Term_1 = Term_0 \cup \{f(c), f(v_i), g(c, c), g(c, v_i), g(v_i, c), g(v_i, v_j) \mid i, j \in \mathbb{N}\}$
- $f(g(c, v_0)), g(v_5, f(v_7)) \in Term_2$
- $g(g(v_3, v_3), f(f(v_3))) \in Term_3$

Esempi

Sia $\{\leq, +, \cdot, 0, 1, 2, 3, \dots\}$, dove

- $\leq$ è simbolo relazionale binario
- $+, \cdot$ sono simboli funzionali binari
- $0, 1, 2, 3, \dots$ sono simboli di costante

Allora:

- $Term_0 = \{0, 1, 2, 3, \dots, v_0, v_1, v_2, v_3, \dots\}$
- $Term_1 = Term_0 \cup \{i+j, i+v_j, v_i+j, v_i+v_j, i \cdot j, i \cdot v_j, v_i \cdot j, v_i \cdot v_j \mid i, j \in \mathbb{N}\}$
- $(3 + v_8) \cdot 1, (2 + 3) + (v_0 \cdot 5) \in Term_2$
- $((2 + 3) + (v_0 \cdot 5)) + 6 \in Term_3$

Alberto sintattico di un termine

L'*albero sintattico* di un termine descrive algoritmicamente la costruzione del termine. L'algoritmo può essere applicato a qualunque stringa i cui simboli sono variabili, simboli di costanti, simboli funzionali, parentesi e virgole. Se tale stringa non è un termine, l'algoritmo lo riconosce.

L'albero sintattico di un termine è un albero etichettato finito (ma non necessariamente binario: il numero di successori immediati di un nodo dipende dall'arità del simbolo di funzione applicato).

- La radice è etichettata con il termine (o stringa) dato
- Se un nodo è etichettato con una variabile o una costante, non si aggiungono successori a quel nodo: è una foglia
- Se un nodo è etichettato con un termine della forma $f(t_1, \dots, t_n)$, dove $ar(f) = n$, allora si aggiungono n successori immediati a quel nodo, etichettandoli rispettivamente $t_1, t_2, \dots, t_n$