

Linguaggi

17: Logica del prim'ordine

Claudio Sacerdoti Coen

`<sacerdot@cs.unibo.it>`

Università di Bologna

04/12/2019

Wikipedia: “Nella logica i *quantificatori* sono espressioni come “qualcosa” (quantificatore esistenziale) e “ogni cosa”. Il nome “quantificatori” è legato al fatto che danno una informazione su quanto è grande l'estensione in cui è valido un predicato.”

Premessa e motivazioni

La **logica proposizionale** è quella logica le cui **connotazioni denotano tutte valori di verità** (almeno classicamente).

La sintassi della logica proposizionale permette di formalizzare una sentenza del linguaggio naturale a un certo livello di dettaglio.

A volte il livello di dettaglio è sufficiente a catturare il ragionamento informale.

Esempio:

linguaggio naturale

2 è un numero pari

se 2 è un numero pari allora 3 è dispari

quindi 3 è un numero dispari

formalizzazione

P

$P \Rightarrow D$

$\Vdash D$

Premessa e motivazioni

Tuttavia, quando il dominio del discorso non è finito, la logica proposizionale diventa insufficiente e/o innaturale in quanto non riesce con un numero finito di sentenze a catturare quanto espresso **con una sola sentenza** in linguaggio naturale.

Esempio (non riesco con una formula sola!):

linguaggio naturale **formalizzazione**

tutti gli $x + 1$ sono positivi $P1, P2, P3, \dots$

Inoltre non catturo la vera **struttura** della frase.

Esempio

(forzando ad avere una formula sola perdo conseguenza logica):

linguaggio naturale

2 è un numero pari

quando x è un numero pari allora $x + 1$ è dispari

quindi 3 è un numero dispari

formalizzazione

P

$Q \Rightarrow R$

$\not\vdash D$

In particolare la logica proposizionale non cattura i **quantificatori**.

Esempi:

- tutti gli x sono biondi (**quantificatore universale**, \forall)
- qualche x è biondo (**quantificatore esistenziale**, \exists)
- più di un x è biondo
- molti x sono biondi

Per tale motivo si introduce la **logica del prim'ordine** per poter parlare esplicitamente (sintatticamente) dei quantificatori universale ed esistenziale.

Vogliamo catturare frasi tipo $\forall x.\exists y.x + 1 \leq y$.

Ci limiteremo ai quantificatori **universale** ed **esistenziale**.

Alcuni altri tipi di quantificatori sono catturabili a partire da questi e dalla nozione di uguaglianza (p.e. “almeno 2”); altri non saranno catturati (p.e. “molti”).

Per farlo, oltre ai quantificatori, ci servono:

- variabili x, y, z, \dots
denotano un oggetto ignoto e variabile sul dominio
- costanti $0, 1, \dots, \pi, e, \dots, \text{marco}, \text{luca}, \dots c, d, \dots$
denotano un oggetto noto e fissato del dominio
- simboli di funzione $+, *, \dots$, il fratello di, il figlio di \cdot e \cdot, \dots
applicati a connotazioni di oggetti del dominio, denotano un oggetto del dominio
- simboli di predicato
 $<, \leq, =, \dots$, essere sposato con, essere vedovo, \dots
applicati a connotazioni di oggetti del dominio, denotano un valore di verità

Recapitolando, siamo interessati a due tipi di connotazioni:

- **Connotazioni che denotano elementi del dominio del discorso** e **funzioni** che applicati a connotazioni per il dominio del discorso formano altre connotazioni per il dominio del discorso.
- **Connotazioni che denotano valori di verità**, **predicati** che applicati a connotazioni per il dominio del discorso formano connotazioni per valori di verità, **quantificatori** che cambiano la denotazione di una connotazione per un valore di verità e **connettivi** che applicati a connotazioni per valori di verità formano altre connotazioni per valori di verità.

Introduciamo quindi due categorie sintattiche distinte per i due tipi di connotazioni.

Termini:

$t ::= x \mid c \mid f^n(t_1, \dots, t_n)$

variabili, costanti, funzioni di arietà n totalmente applicate

Proposizioni:

$P ::= P^n(t_1, \dots, t_n) \mid \perp \mid \top \mid P \wedge P \mid P \vee P \mid P \Rightarrow P \mid \forall x.P \mid \exists x.P$

predicati di arietà n totalmente applicati, connettivi,
quantificatori

dove le costanti c , le funzioni f^n e i predicati P^n sono presi da tre insiemi distinti fissati a priori. Presi assieme, essi formano la **specificata** sintattica di un **linguaggio al prim'ordine**.

Esempio (sotto-linguaggio dell'aritmetica):

costanti: 0, 1

funzioni: +, *

predicati: =, <, >, ≤, ≥

Termine sintatticamente valido: $x + y * (1 + 1)$

Termine sintatticamente invalido: $x + y * 2$

Proposizione sintatticamente valida: $\forall x. \neg(x = x)$

Proposizione sintatticamente invalida: $x \leq x^x$

Precedenza degli operatori

I quantificatori hanno precedenza massima.

Esempio: $\forall x.P \Rightarrow \exists x.Q$ si legge come $(\forall x.P) \Rightarrow (\exists x.Q)$

In Matita invece i quantificatori hanno precedenza minima!

Esempio: $\forall x.P \Rightarrow \exists x.Q$ si legge come $\forall x.(P \Rightarrow (\exists x.Q))$

Notazione alternativa per i quantificatori: $(\forall x)P$ e $(\exists x)P$

La semantica di una costante dipende dal mondo.

La semantica di una funzione dipende dal mondo.

La semantica di un predicato dipende dal mondo.

La semantica di una variabile è determinata dal quantificatore che la lega.

La semantica dei connettivi è fissata.

La semantica dei quantificatori è fissata: $\forall x.P$ significa “per tutti gli x vale P ” e $\exists x.P$ significa “per un qualche x vale P ” **senza alcun'altra restrizione.**

In particolare, variabili distinte legate da quantificatori distinti **possono assumere lo stesso valore**.

Esempio: in $\exists x.\exists y.P(x, y)$ può essere che x e y siano uguali!

Esempio: $\forall x.\exists y.x = y$ è vera (quando interpretiamo $=$ come uguaglianza nel mondo)

Funzioni e predicati costanti

Nota: ammettiamo anche funzioni e predicati 0-ari.

Una funzione 0-aria è la stessa cosa di una costante: è una “funzione” che non prende argomenti e restituisce una connotazione (costante!) per un oggetto del dominio.

Un predicato 0-ario è la stessa cosa delle proposizioni atomiche A, B, \dots della logica proposizionale: è un predicato che non prende alcun argomento e restituisce una connotazione (costante!) per un valore di verità.

Nota: le denotazioni delle costanti e delle proposizioni atomiche non variano una volta fissato il mondo (sono costanti). Variano al variare dei mondi (o interpretazioni) così come varia la semantica delle funzioni e dei predicati n -ari.

Pertanto la logica del prim'ordine estende quella proposizionale con predicati n -ari per $n > 0$ i quali vanno applicati a termini che vanno anch'essi aggiunti alla logica proposizionale.

Formalizzare le seguenti frasi (facile):

- Ogni quadrilatero con quattro angoli retti è un quadrato.
- Nessuno è padre di se stesso.
- Non tutti i numeri primi sono dispari.

Formalizzare le seguenti frasi (difficile). Suggerimento: basarsi sul predicato di uguaglianza.

- C'è un solo numero x tale che per ogni y si ha $x + y = y$
- C'è un solo colpevole e quel colpevole andrà in galera
- Ci sono almeno due innocenti
- Ci sono esattamente due innocenti

I quantificatori sono casi particolari di **binder**.

Un binder **lega** una variabile in uno **scope**.

I binder occorrono in diversi contesti. Esempi:

- Matematica: $\int_x f(x) dx$, $\frac{df}{dx}$, $\Sigma_x f(x)$, ...
- Linguaggi di programmazione: parametri delle funzioni
- Logica: $\forall x.P$, $\exists x.P$

Vi sono una serie di problematiche sintattiche comuni a tutti i tipi di binder.

Variabili libere e variabili legate

Osserviamo la seguente formula

$$\forall x.(P(x) \wedge \exists x.Q(x))$$

Lo **scope** associato al \forall è l'intera formula $P(x) \wedge \exists x.Q(x)$.

Quindi la x in $P(x)$ si riferisce (è **legata**) al quantificatore universale.

Lo scope associato all' \exists è la formula $Q(x)$.

Poichè l' \exists ri-lega la variabile x , la x che occorre in $Q(x)$ si riferisce (è legata) al quantificatore esistenziale. Non è più possibile riferirsi alla x legata dal \forall nello scope dell' \exists (**shadowing**).

Capiamo meglio questo concetto.

Diagrammi di legame

In maniera informale si possono introdurre i diagrammi di legame: una volta scritta un'espressione

- 1 **collegare** ogni occorrenza di una **variabile legata con il binder** che la lega per mezzo di una freccia
- 2 le variabili non legate sono dette **libere**: indicarle con una freccia che punta all'infinito su cui scrivete **il nome della variabile**

La definizione formale (= per ricorsione strutturale) di occorrenze libere/legate è più complessa.

Variabili libere e variabili legate

Definizione di **variabili libere** (free variables): per induzione strutturale sulle proposizioni e sui termini.

$$FV(x) := \{x\}$$

$$FV(f^n(t_1, \dots, t_n)) := FV(t_1) \cup \dots \cup FV(t_n)$$

$$FV(P^n(t_1, \dots, t_n)) := FV(t_1) \cup \dots \cup FV(t_n)$$

$$FV(\perp) = FV(\top) := \emptyset$$

$$FV(\neg P) := FV(P)$$

$$FV(P \wedge Q) = FV(P \vee Q) = FV(P \Rightarrow Q) := FV(P) \cup FV(Q)$$

$$FV(\forall x.P) = FV(\exists x.P) := FV(P) \setminus \{x\}$$

Un quantificatore, legando una variabile, la rende non più libera.

Variabili libere e variabili legate

Definizione di **variabili legate**: sia $P[F]$ una proposizione per la quale F sia una sottoformula (termine o proposizione) e $P[\]$ il suo contesto con un solo buco. L'insieme delle **variabili legate da P in F** (bound variables) è definito per induzione strutturale su P come segue:

$$BV(\[]) := \emptyset$$

$$BV(P^n(t_1, \dots, t_n)) := \emptyset$$

$$BV(\perp) = BV(\top) := \emptyset$$

$$BV(\neg P) := BV(P)$$

$$BV(P \wedge Q) = BV(P \vee Q) = BV(P \Rightarrow Q) := BV(P) \text{ se } \[] \text{ occorre in } P$$

$$BV(P \wedge Q) = BV(P \vee Q) = BV(P \Rightarrow Q) := BV(Q) \text{ se } \[] \text{ occorre in } Q$$

$$BV(\forall x.P) = BV(\exists x.P) := \{x\} \cup BV(P)$$

Esempio: la variabile x è **al tempo stessa libera in $P(x) \wedge \forall x.Q(x)$** (per via della x in $P(x)$) **e legata nella sottoformula $Q(x)$** .

I nomi scelti per le variabili legate **non hanno alcuna importanza se generano lo stesso diagramma di legame.**

Esempio: quali di queste formule sono equivalenti? Quali no e perchè?

- $\forall x.P(x) \wedge \exists y.P(x, y)$
- $\forall y.P(y) \wedge \exists x.P(y, x)$
- $\forall x.P(x) \wedge \exists y.P(y, x)$
- $\forall z.P(z) \wedge \exists z.P(z, z)$
- $\forall x.P(x) \wedge \exists y.P(z, y)$

Le formule equivalenti si dicono **α -convertibili.**

Intuizione sull' α -convertibilità:

- Le occorrenze **legate** sono solo un modo user-friendly per scrivere puntatori al binder corrispondente: il nome scelto non conta e la semantica è determinata dal binder.
- Per le occorrenze **libere** il nome conta: osservando solo l'espressione non sappiamo quale sarà il binder che legherà l'occorrenza, che verrà determinato solo **inserendo l'espressione in un contesto** che contenga il binder (il che è fuori dal nostro controllo).
- Esempio (linguaggi di programmazione): le occorrenze legate sono le **variabili locali** che possiamo ridenominare ovunque; le libere sono **variabili globali/funzioni di libreria** dichiarate da un'altra parte che non possiamo ridenominare non avendo accesso alla dichiarazione.

Definizione: due formule $P[Qx.F]$ e $P[Qy.G]$ dove Q è un quantificatore (binder) si dicono **α -convertibili** quando:

- 1 $y \notin FV(F)$ ovvero y non viene catturata in G e non in F
- 2 x non è legata in alcun sotto-terminine H di F tale per cui $y \in BV(H)$
- 3 G si ottiene a partire da F sostituendo tutte le **occorrenze libere** di x in F con y

Controesempi: le seguenti formule NON SONO α -convertibili

- $\exists y.\forall x.P(x, y)$ vs $\exists y.\forall y.P(y, y)$: violazione del primo vincolo
- $\forall x.P(x, y)$ vs $\forall y.P(y, y)$: violazione del primo vincolo
- $\forall x.\exists y.P(x, y)$ vs $\forall y.\exists y.P(y, y)$: violazione del secondo vincolo

DA QUESTO MOMENTO IN AVANTI IDENTIFICHEREMO
SEMPRE FORMULE α -CONVERTIBILI

Il senso del quantificatore universale $\forall x.P(x)$ è che da esso dobbiamo essere in grado di dedurre $P(t)$ per un qualunque termine t .

Esempio: se per ogni x si ha $x = x$ allora deve essere $3^2 = 3^2$.

Abbiamo quindi bisogno di una funzione di **sostituzione** di un termine t al posto di una variabile precedentemente legata x .

Dobbiamo però fare molta attenzione a non catturare variabili che erano libere e a rispettare il diagramma di legame.

Esempio di sostituzione sbagliata: dedurre da $\forall x.(Q(x) \Rightarrow \exists y.P(x, y))$ che vale $Q(y^2) \Rightarrow \exists y.P(y^2, y)$

Definizione di sostituzione di un termine per una variabile: per induzione strutturale sul termine in cui avviene la sostituzione.

$$x[t/x] = t$$

$$y[t/x] = y$$

$$f^n(t_1, \dots, t_n)[t/x] = f^n(t_1[t/x], \dots, t_n[t/x])$$

$$P^n(t_1, \dots, t_n)[t/x] = P^n(t_1[t/x], \dots, t_n[t/x])$$

$$\perp[t/x] = \perp$$

$$\top[t/x] = \top$$

$$(\neg F)[t/x] = \neg(F[t/x])$$

$$(F_1 \wedge F_2)[t/x] = F_1[t/x] \wedge F_2[t/x]$$

$$(F_1 \vee F_2)[t/x] = F_1[t/x] \vee F_2[t/x]$$

$$(F_1 \Rightarrow F_2)[t/x] = F_1[t/x] \Rightarrow F_2[t/x]$$

$$(\forall x.P)[t/x] = \forall x.P$$

$$(\forall y.P)[t/x] = \forall z.(P[z/y][t/x]) \text{ per } z \notin FV(t) \cup FV(P)$$

$$(\exists x.P)[t/x] = \exists x.P$$

$$(\exists y.P)[t/x] = \exists z.(P[z/y][t/x]) \text{ per } z \notin FV(t) \cup FV(P)$$

Esercizio (2 punti)

Nel seguente frammento di programma C fare l'inlining della funzione `f` in `main` (ovvero, espandere il codice della `f` nel corpo del `main` per evitare il costo associato alla chiamata di funzione), minimizzando il numero di cambi di nome alle variabili.

...

```
int f(int z, int y) {
    return g(x+z, y);
}

int main() {
    int x, c;
    return f(d+c, x);
}
```

Esercizio (2 punti)

Si calcoli il risultato della seguente sostituzione minimizzando il numero di cambi di nome di variabili.

$$((\exists x.P(x + z)) \wedge (\forall y.P(y + z)))[(w \cdot x + \prod_{y=0}^n y)/z]$$