Lezione24

Table of contents

PROLOG e Logica

Robert Anthony Kowalski (1941)

Come mai la programmazione logica ha questo nome?
Per capirlo, dobbiamo vedere la logica su cui andremo a lavorare e poi vedere i metodi che ci permettano di partire da verità assodate e regole di deduzione, per generare nuove affermazioni. Il metodo per dimostrare i problemi matematici, vedremo che alla fine, è PROLOG.

Il linguaggio logico su cui lavoriamo è la **logica dei predicati** (logica del primo ordine), che estende la logica delle proposizioni, sia in senso semantico che sintattico. La strada che andremo a percorrere, fu studiata e realizzata dall'informatico inglese Robert A. Kowalski. Un linguaggio logico dei predicati è caratterizzato da una struttura che lo descrive *signature* a 4 insiemi:

$$\langle P, C, F, V \rangle$$

- l'insieme dei *simboli di predicato* P non vuoto, tipicamente finito (predicati);
- l'insieme dei *simboli di costante* C (costanti);
- l'insieme dei *simboli di funzione* F (funzioni);
- l'insieme dei *simboli di variabile* V (variabili).

La nomenclatura degli ultimi 3 punti, assomiglia molto a quella dei termini, perché quelli sarebbero; i simboli di predicato assomigliano ai simboli per la costruzione dei termini tranne che questi vengono utilizzati in un posto soltanto. Possiamo vedere la rappresentazione di questo schema, pensando a un albero, dove le radici sono elementi di P.

Se ci viene data una signature del linguaggio di primo ordine, escludendo P, capiamo che possiamo costruire *termini* t di primo ordine. Con $\texttt{vars}(t)$ costruiamo l'insieme delle variabili contenute in un termine:

- se t è una variabile, $\texttt{vars}(t) = \{t\}$;
- se t è una costante, $\texttt{vars}(t) = \emptyset$;
- se t è un termine costruito con simbolo di funzione e n argomenti, il simbolo di funzione non contribuisce mai variabili, per cui ci basta calcolare $\texttt{vars}(t_n)$ e unire tutti i risultati per ottenere $\texttt{vars}(t)$.

Se per un termine t, $\texttt{vars}(t) = \emptyset$, allora il termine viene detto *ground* (chiuso), ovvero senza variabili; non c'è nulla da riempire, siccome albero i cui nodi non possono essere sostituiti con qualcos'altro.

> ⓘ **Immutabilità in linguaggi funzionali**

La caratteristica dei termini ground, è una delle diverse che compare per i linguaggi funzionali e non logici, per i così detti valori immutabili.

Dal punto di vista matematico, è molto più difficile ragionare su programmi per cui la mutabilità è prevista.

I design pattern non hanno alcuna base matematica, piuttosto, sono stati ideati tramite ripetuti test ed errori. Per i programmatori Haskell il problema non sussiste, essendo il linguaggio funzionale, dove ciascun "functor", "monad", "co-monad" e "zipper" ha una base matematica.

Why do immutable objects enable functional programming?

Costruire termini ci permette di costruire *formule ben formate* (o ben formulate). Ricordandoci i concetti della logica delle proposizioni, sappiamo che in realtà le frasi che andiamo a scrivere le costruiamo medianti i simboli di proposizione e connettivi. Qui, avere a disposizione i termini, permette la costruzione di formule, usando quello rimasto, ovvero i simboli di predicato P. La struttura della logica del primo ordine, come consueto dirsi, ha 2 livelli: il livello dei termini, arbitrariamente libero e profondo con utilizzo di variabili ovunque; il livello delle formule ben formate.

1. Esiste un simbolo detto $\top$ (top) e uno detto $\bot$ (bottom), che presi da soli sono proposizioni.
2. Se A è una proposizione, allora $\neg(A)$ è una proposizione. A viene detto letterale affermato, la sua negazione viene detta *letterale negato*.
3. Se A e B sono due proposizioni, $(A \land B), (A \lor B), (A \implies B), (A \iff B)$ sono proposizioni.

Queste prime 3 regole, sono quelle già definite nella logica delle proposizioni.

4. Se prendiamo n termini t, con p un simbolo di predicato a cui possiamo associare arità $n \in \mathbb{N}_+$, allora $p(t_1, t_2, \ldots, t_n)$ è proposizione e viene detta *letterale affermato*.
5. Se abbiamo un simbolo x appartenente ai simboli di variabile V, $\exists x.\,(A)$ e $\forall x.\,(A)$ sono proposizioni. Possiamo costruire proposizioni dove i quantificatori vengono utilizzati per definire uno "scope" identificante la validità della variabile x (la variabile è tale all'interno di A).
6. Nient'altro è proposizione.

Se una variabile, come accennato alla regola #5, non è nel *campo d'azione* di un quantificatore, allora viene detta *variabile libera* (od occorrenza libera della variabile): la stessa variabile x è così ovunque. La parte del cambio del nome di una variabile è importante perché concetto delle variabili fresh: siccome PROLOG il ragionamento lo svolge su una di queste variabili, dobbiamo capire come cambiare nome alle variabili, dove in realtà stiamo dicendo "se sono quantificate, allora sappiamo l'inizio e fine di x", caso che useremo.

A sintassi definita, dobbiamo ora attribuire un significato.
Per farlo, nella logica delle proposizioni attribuiamo un *valore di verità* a ogni simbolo proposizionale, costruendo il valore di verità della proposizione combinando i valori di verità dei simboli proposizionali (tabella di verità di `AND`, `OR`, `NOT`). Dal punto di vista d'interpretazione del mondo, tuttavia, questa descrizione non è molto fine:

$$\text{"L'erba è verde"}$$

è un'affermazione vera, ha senso, ma non riusciamo ad attribuire il concetto di "colore" all'erba. Nell logica dei predicati vogliamo descrivere un mondo generale, formato da oggetti con delle proprietà e relazioni, applicabile anche a mondi reali/veri, "fotografando" ogni volta lo stato: se siamo in grado di descrivere il mondo, siamo in grado di attribuire un valore di verità agli elementi atomici della sintassi (predicati + argomenti).

Presa una signature di un linguaggio dei predicati e un dominio insieme d'interpretazioni chiamato *dominio del linguaggio*, si costruisce una *funzione d'interpretazione* I che lavora direttamente sui simboli della signature C, F, P.

- Fissata una costante $c \in C$, qualsiasi sia essa, la sua interpretazione $I(c) \in D$ è elemento del dominio del linguaggio. Per esempio: il numero di matricola costante per ogni studente universitario, può essere interpretato come lo studente.
- Preso un simbolo di funzione $f \in F$, con arità n, interpretando quello che andiamo a costruire è una funzione $I(f) : D^n \to D$ che genera un nuovo elemento del dominio del linguaggio. È una funzione di calcolo, per cui l'interpretazione attribuirà significato.
- Dato un simbolo di predicato $p \in P$ con arità n, l'interpretazione di p è un sottoinsieme $I(p) \subseteq D^n$. Un predicato è relazione non necessariamente funzionale.

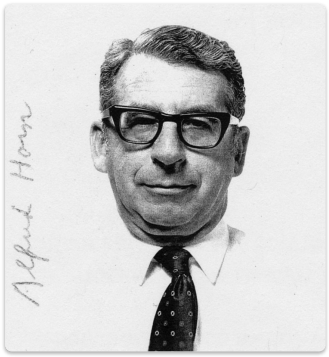
Per lavorare su variabili, è necessaria un'*assegnazione* che parte dall'elemento sintattico e porta all'elemento semantico $s : V \to D$.

- L'interpretazione di costanti rimane la stessa, $I_s(c) = I(c)$.
- L'interpretazione del simbolo di variabile rimane la stessa, ma anziché usare I per interpretare i termini, utilizziamo I_s, $I_s(x) = s(x)$.
- L'interpretazione dei predicati rimane la stessa, $I_s(t) = I(f)(I_s(t_1), I_s(t_2), \dots, I_s(t_n))$.

Fissata un'interpretazione I e fissata un'assegnazione s, possiamo dire quando la copia soddisfa lo proposizione $(I, s) \models A$. Le regole che stiamo per vedere, non sono altro che un'esplicitazione delle tabelle di verità dei connettivi, dando un significato ai quantificatori.

1. $(I, s) \models p(t_1, t_2, \dots, t_n)$ se e soltanto se $(I_s(t_1), I_s(t_2), \dots, I_s(t_n)) \in I(p)$
2. $(I, s) \models \top$ e $(I, s) \not\models \bot$
3. $(I, s) \models \neg A$ se e soltanto se $(I, s) \not\models A$
4. $(I, s) \models A \wedge B$ se e soltanto se $(I, s) \models A$ e $(I, s) \models B$
5. $(I, s) \models A \vee B$ se e soltanto se $(I, s) \models A$ oppure $(I, s) \models B$
6. $(I, s) \models A \implies B$ se e soltanto se $(I, s) \not\models A$ oppure $(I, s) \models B$
7. $(I, s) \models A \iff B$ se e soltanto se $(I, s) \models A$ e $(I, s) \models B$, oppure $(I, s) \not\models A$ e $(I, s) \not\models B$
8. $(I, s) \models \exists x.\, A$ se e soltanto se esiste un $d \in D$ tale che $(I, s_{x \to d}) \models A$
   Siccome le variabili vengono abbinate agli elementi del dominio tramite la sostituzione, se troviamo un elemento del dominio che mappato nella variabile x rende la proposizione A vera, allora è lecito dire che $\exists x.\, A$.
9. $(I, s) \models \forall x.\, A$ se e soltanto se per ogni $d \in D$ vale $(I, s_{x \to d}) \models A$
   La formula ben formata $\forall x.\, A$ è vera nella copia (I, s) se e soltanto se qualsiasi sia l'elemento d, se andiamo ad attribuire l'elemento alla variabile s tramite la sostituzione, quello che otteniamo è soddisfatto da A.

Se per ogni sostituzione s, A è vera in I, allora l'interpretazione viene detta *modello*. Viceversa, se nessuna sostituzione esiste per le associazioni, allora I viene detta *contromodello*. Considerando una *teoria* (di primo ordine) T, formata da un insieme di proposizioni, una proposizione A si dice *conseguenza logica* $T \models A$, se per ogni interpretazione I e sostituzione s che rendono vera la teoria T, vale anche $(I, s) \models A$.

	
Thoralf Albert Skolem (1887-1963)	Alfred Horn (1918-2001)

Quello che facciamo ora è la stessa cosa fatta nei casi proposizionali, ovvero costruire un algoritmo per il calcolo del tableau, usando la strada asfaltata dal matematico norvegese Thoralf Skolem, che scrisse molto sulla logica matematica e lattici. Una **clausola di Horn**, dal nome del matematico americano Alfred Horn, è una proposizione in cui:

- non esistono quantificatori esistenziali, ma solo quelli universali sono ammessi;
- i quantificatori universali, se ci sono, sono unicamente nella parte iniziale, ovvero una testa formata unicamente da quantificatori che potenzialmente, può essere vuota;
- ci sono unicamente letterali disgiunti, affermati o negati;
- non più di un letterale è affermato, il che significa che stiamo fortemente restringendo le nostre possibilità (1/2 quantificatori).

$$\forall x_1.\forall x_2.\cdots\forall x_n.(L_1 \vee L_2 \vee \cdots \vee L_m)$$

Viste in questo modo, il tipo di formule componibili è ristretto considerevolmente. Tuttavia, le formule che vogliamo scrivere hanno essenza in queste regole e un qualsiasi problema di soddisfacibilità di una formula all'interno di una teoria, può essere ridotto a problemi tipici, che è esattamente la proprietà di queste clausole. Una clausola di Horn si dice *definita* se il letterale positivo L è presente, cioè clausola del tipo "$\text{se}\ldots\text{allora}\ldots$", dove il "$se$" viene scritto come congiunzione di letterali positivi. Se la clausola di Horn non ammette letterali positivi, allora questa si chiama *clausola goal* (che sono esattamente i goal che cerchiamo di verificare su SWISH). La clausola di Horn *vuota* è quella che restituisce $\bot$, ovvero `false`.

Horn clause

Una clausola di Horn è una clausola disgiunzione di letterali con al massimo un'affermazione, ovvero un letterale non negato. Una clausola con queste proprietà, svolge ruoli importanti nei concetti di logica costruttiva e computazionale, soprattutto perché, il risolutore di 2 clausole di Horn è anch'esso una clausola di Horn e risolutore della clausola di goal definita (esattamente, quello che scriviamo nella casella interattiva in basso a destra di SWISH).

Tipo di clausola di Horn	Forma disgiunta	Forma implicata	Intuitivamente letta
Clausola definita	$\neg p \vee \neg q \vee \ldots \neg t \vee u$	$u \leftarrow p \wedge q \wedge \cdots \wedge t$	assumendo che p e q e $\ldots$ e t reggano, allora anche u regge
Fatto	u	$u \leftarrow \texttt{TRUE}$	assumendo che u regga
Clasuola goal	$\neg p \vee \neg q \vee \cdots \vee \neg t$	$\texttt{FALSE} \leftarrow p \wedge q \wedge \cdots \wedge t$	mostrare che p e q e $\ldots$ e t reggano tutti

Horn clause

Resolution rule

Nella logica matematica e dimostrazione automatizzata di teoremi, con "risoluzione" s'intende una regola d'inferenza che porta a una dimostrazione di un teorema completo in base alle affermazioni, tecnica presente nella logica proposizionale e di primo ordine. Una clausola prodotta da una regola di risoluzione è a volte chiamata "risolvente", o "clausola risolvente".

Per esempio:

$$\frac{\overbrace{a \vee b}^{\texttt{TRUE}} \quad \overbrace{\neg a \vee c}^{\texttt{TRUE}}}{\underbrace{b \vee c}_{\texttt{TRUE}}}$$

supponiamo che a sia falso. Per fare sì che la premessa $a \vee b$ sia vera, b deve essere vero. Alternativamente, supponiamo che a sia vero. Per far sì che la premessa $\neg a \vee c$ sia vera, c deve essere vero. Quindi, qualsiasi sia la veridicità o falsità di a, se entrambe le premesse reggono, allora la conclusione $b \vee c$ è vera.

Resolution (logic)

Nel mondo di queste clausole, andiamo a costruire un metodo che ci permetta di trovare nuove verità, partendo dalle ipotesi che le verità siano vere. Da un certo punto di vista, possiamo quindi riprendere la menzione ai tableau vista sopra: si parte da 2 formule, le si combina e si genera una nuova formula; se le 2 iniziali sono vere, combinate generano qualcosa a sua volta vero. In un tempo finito, se riusciamo a generare solo verità, siamo soddisfatti, altrimenti c'è un'inconsistenza.

Partendo da 2 clausole di Horn,

$$\forall x_1.\forall x_2.\cdots\forall x_n.(L_1 \vee L_2 \vee \cdots \vee L_m)$$

con n variabili chiamate x, e m letterali chiamati L e

$$\forall y_1.\forall y_2.\cdots\forall y_r.(M_1 \vee M_2 \vee \cdots \vee M_s)$$

con r variabili chiamate y, e s letterali chiamati M, se troviamo una sostituzione tale per cui riusciamo a rendere vero uno dei letterali della prima clausola con la sostituzione alla negazione di uno dei letterali della seconda clausola con la sostituzione applicata $L_1\theta = \neg M_j\theta$, allora si può costruire una terza clausola che chiamiamo *clausola risolvente* R. La sostituzione viene trovata costruendo l'unificatore comune tra la clausola L_i e la clausola M_j che andiamo a prendere, $\theta = \texttt{mgu}(L_i, M_j)$. Guardiamo tutti i letterali della prima e della seconda clausola e se ne troviamo 2 che è possibile unificare con un unificatore θ, che rende una affermata e l'altra negata, allora quello che possiamo costruire è una 3ª clausola.

La clausola risolvente R, che avrà numero di clausole inferiore, è stata costruita con la *regola di risoluzione* e si può dimostrare che R è conseguenza logica di T, se la clausola risolvente non è vuota $\perp$. Se stiamo facendo una dimostrazione di questo tipo, stiamo dicendo di avere dimostrato il teorema.

Reductio ad absurdum

La dimostrazione per assurdo è un tipo di argomentazione logica nella quale, muovendo dalla negazione della tesi che si intende sostenere e facendone seguire una sequenza di passaggi logico-deduttivi, si giunge a una conclusione incoerente o contraddittoria. Tale risultato, nella logica argomentativa, confermerebbe l'ipotesi iniziale, per mezzo della falsificazione della sua negazione. È una delle principali forme di dimostrazione matematica.

Esempio di applicazione della regola di risoluzione

Considerando una signature del linguaggio di primo ordine formato da un unico predicato p, 3 simboli di costante a, b, e e 1 simbolo di funzione f. La teoria viene descritta come segue:

1. $p(e, X, X)$
2. $p(f(X, Y), Z, f(X, W)) \vee \neg p(Y, Z, W)$

Dimostrare che esiste un T tale che $p(f(a, e), T, f(a, f(b, e)))$.

Essendo le variabili tutte quantificate universalmente in testa, il $\forall x.$ precedente ogni regola, viene omesso. Da queste affermazioni che cosa possiamo dedurre? Per dedurre qualcosa utilizzando il metodo di risoluzione diretto (forward chaining), cercando un letterale affermato e un letterale negato in 2 clausole diverse e se questi due unificano, generiamo la clausola risolvente.

Nella clausola #1 c'è un unico letterale affermato (p), un unico negato c'è nella clausola #2 ($\neg p$): questi possono unificare. `mgu` è quello che sostituisce Y con e, Z con S e W con S. Nel caso non fosse chiaro: il nuovo nome che viene assegnato alla X della prima clausola, è appunto S.

3. $p(f(X,e),S,f(X,S)) \quad \texttt{RES}(1,2) \quad \{S/W,e/Y,S/Z\}$

Al risultato aspettato non ci siamo ancora arrivati, però la nostra teoria ha ora 3 clausole, siccome le prime due sono state trasformate: cerchiamo di usare quest'ultima. La soluzione alla clausola #3 è raggiunta assegnando nuovi nomi alle variabili S e X che ci erano rimaste con rispettivamente $f(b,e)$ e a.

4. $p(f(a,e),T,f(a,f(b,e))) \quad \texttt{RES}(3) \quad \{f(b,e)/S,a/X\}$

Alla proposizione goal che volevasi dimostrare, la sostituzione $\{f(b,e)/T\}$ può essere applicata.

Proviamo ora il metodo indiretto (backward chaining), negando il goal per cercare di ottenere $\perp$. Per prima cosa appunto, siccome ragioniamo per confutazione, neghiamo il goal

3. $\neg p(f(a,e),T,f(a,f(b,e)))$

Procediamo usando le sostituzioni. La clausola #1 non può essere utilizzata siccome ha e come primo argomento e viene detto che questa è vera (mentre nel nostro goal è chiaramente falsa). La regola #2 si addice di più alla nostra casistica, siccome non presenta contraddizioni.

4. $\neg p(e,U,f(b,e)) \quad \texttt{RES}(2,3) \quad \{e/Y,U/Z,f(b,e)/W\}$
5. $\perp \quad \texttt{RES}(1,4)$

Procedendo a ritroso con le sostituzioni, la conseguenza logica della teoria iniziale si ottiene applicando la sostituzione $\{f(b,e)/T\}$ alla proposizione goal che si vuole dimostrare.

25/05/2023