Appunti di Logica

Sul corso di Logica Matematica @ DISI,
Università degli Studi di Trento

A.A. 2018/2019

— 1 —

PL - Introduzione

La logica delle proposizioni (Propositional Logic -**PL**) è una logica che permette di rappresentare fatti (affermazioni), che possono essere vere o false.

PL si compone di **simboli logici** e **variabili proposizionali**. Una proposizione (formula) è composta quindi di variabili proposizionali uniti da simboli logici. La formula può essere *vera* o *falsa* a seconda dell'assegnazione delle singole variabili.

Definizione 1 (Linguaggio della PL)
Logical symbols: *(1)* $\neg$ *(2)* $\wedge$ *(3)* $\vee$ *(4)* $\supset$ *(5)* $\equiv$
PL formulas and sub-formulas

- *every logical variable* $P \in \mathrm{P}$ *is an atomic formula*
- *every atomic formula is a formula*
- *if* A *and* B *are formulas, then* $\neg A, A \wedge B, A \vee B, A \supset B, A \equiv B$ *are formulas*

Una **funzione di interpretazione** $I : P \to \{\top, \bot\}$ assegna un valore vero o falso a ciascuna variabile $P \in \mathrm{P}$.

Una funzione di interpretazione è detta **modello** di una funzione φ se le sue assegnazioni rendono il valore della funzione vero. In simboli: $I \models \varphi$.

SAT, UNSAT, VAL

- Una formula A è soddisfacibile (**SAT**) se $\exists I$ funzione di interpretazione t.c. $I \models$ A.
- Una formula A è insoddisfacibile (**UNSAT**) se $\nexists I$ funzione di interpretazione t.c. $I \models$ A.
- Una formula A è valida (**VALID**) se $\forall I, I \models$ A

Osservazione:
Se A è **VALID**, $\neg$A è **UNSAT**.
Se A è **SAT**, $\neg$A non è valida.
Se A non è valida, $\neg$A è **SAT**.
Se A è **UNSAT**, $\neg$A è **VALID**.

Conseguenza e equivalenza logica

- Una formula A è una **conseguenza logica** di un insieme di formule Γ, in simboli $\Gamma \models \mathrm{A}$ sse per ogni funzione di interpretazione I che soddisfa tutte le formule di Γ, I soddisfa A.
- Due formule A, B sono **equivalenti**, in simboli $\mathrm{A} \equiv \mathrm{B}$ sse per ogni funzione di interpretazione I, $I(\mathrm{A}) = I(\mathrm{B})$.

Procedure di decisione

Model checking (I, φ): $I \overset{?}{\models} \varphi$ (I soddisfa φ?)

Satisfiability (φ): $\overset{?}{\exists} I | I \models \varphi$ (Esiste un modello che soddisfi φ?)

Validity (φ): $\overset{?}{\models} \varphi$ (φ è soddisfatta da qualsiasi modello?)

Logical consequence (Γ, φ): $\Gamma \overset{?}{\models} \varphi$ (Ogni modello che soddisfa Γ soddisfa anche φ?)

Formalizzazione del linguaggio naturale

- A: "It is the case that A"
- $\neg A$: "It is not the case that A"
- $A \wedge B$: "A and B", "A but B", "Although A, B", "Both A and B"
- $A \vee B$: "A or B", "Either A or B"
- $A \rightarrow B$: "If A, then B", "B if A"
- $\neg(A \vee B)$: "Neither A nor B"
- $\neg(A \wedge B)$: "It is not the case that both A and B"

— 2 —

PL - CNF & DPLL

Definizione 2 (Conjunctive Normal Form)

- *A **literal** is a propositional variable A or the negation of a propositional variable $\neg A$*
- *A **clause** is a disjunction of literals $\bigvee_{j=1}^{m} A_j$*
- *A **formula** is in **CNF** if it is a conjunction of clauses $\bigwedge_{i=1}^{n} (\bigvee_{j=1}^{m} I_{ij})$*

Proposizioni:

1. Ogni PL formula può essere ridotta in CNF
2. $\models$ CNF(ϕ) $\equiv$ ϕ (Ogni Pl formula è equivalente alla sua riduzione in CNF)

Notazione insiemistica

Una formula in CNF può essere rappresentata come un insieme di *clauses*, o un insieme di insiemi di *literals*. Le operazioni sono implicite. La generica formula CNF $\bigwedge_{i=1}^{n} (\bigvee_{j=1}^{m} I_{ij})$ può essere rappresentata così: $\{\{I_{1,1}, ..., I_{1,m_1}\}, ..., \{I_{n,1}, ..., I_{n,m_n}\}\}$.

Riduzione in CNF

- CNF(p) $= p$ if p is a literal
- CNF($\neg p$) if $\neg p$ is a literal
- CNF($\neg\neg p$) $=$ CNF(p)

- CNF($\phi \to \psi$) $=$ CNF($\neg\phi$) $\oplus$ CNF(ψ)
- CNF($\phi \wedge \psi$) $=$ CNF(ϕ) $\wedge$ CNF(ψ)
- CNF($\phi \vee \psi$) $=$ CNF(ϕ) $\oplus$ CNF(ψ)
- CNF($\phi \equiv \psi$) $=$ CNF($\phi \to \psi$) $\wedge$ CNF($\psi \to \phi$)

- CNF($\neg(\phi \to \psi)$) $=$ CNF(ϕ) $\wedge$ CNF($\neg\psi$)

- $\mathrm{CNF}(\neg(\phi \wedge \psi)) = \mathrm{CNF}(\neg\phi) \otimes \mathrm{CNF}(\neg\psi)$
- $\mathrm{CNF}(\neg(\phi \vee \psi)) = \mathrm{CNF}(\neg\phi) \wedge \mathrm{CNF}(\neg\psi)$
- $\mathrm{CNF}(\neg(\phi \equiv \psi)) = \mathrm{CNF}(\phi \wedge \neg\psi) \otimes \mathrm{CNF}(\psi \wedge \neg\phi)$

Dove $\otimes$ è definito come segue:

$$(C_1, ..., C_n) \otimes (D_1, ..., D_n) := (C_1 \vee D_1) \wedge ... \wedge (C_1 \vee D_n) \wedge ... \wedge (C_n \vee D_1) \wedge ... \wedge (C_n \vee D_n)$$

DPLL

DPLL è un algoritmo per calcolare la soddisfacibilità di una PL formula ridotta in **CNF**.

Definizione 3 (Partial evaluation)
Osservazioni:
Sia $F := C_0, ..., C_n = CNF(\varphi)$, I funzione di interpretazione. Vale quanto segue:

1. $I \models \varphi \iff I \models C_i \forall i = 0, ..., n$ *(φ è soddisfatta se tutte le sue clauses sono soddisfatte)*
2. $I \models C_i \iff \exists l \in C_i : I \models l$ *(Una clause è soddisfatta se almeno uno dei literals che la compongono è soddisfatto)*

Proposizione: *per verificare se I soddisfa F non è necessario conoscere le assegnazioni di ogni literal che compare in F.*

Definizione 4 (Unit propagation)
Sia φ una PL formula, I funzione di interpretazione; sia u una unit clause $\{u\} \in \varphi$ (clause composta di un solo literal). Vale: $I \models \varphi \iff I : u \mapsto \top$. (segue dalla proprietà (2) sopra esposta: una unit clause non può essere soddisfatta se la sua unica componente non è valutata vera).

L'algoritmo **DPLL** calcola un possibile modello che soddisfa la PL formula φ, se esiste. La costruzione del modello I avviene partendo da un insieme vuoto di assegnazioni, via via aumentato.

Ad ogni passo dell'algoritmo le *clauses* di φ possono essere in uno dei seguenti stati rispertto al modello parziale I', che va via via costruendosi:

1. una *clause* $c \in \varphi$ è **vera** se $\exists l \in c | I : l \mapsto \top$ (se il modello parziale assegna il valore di verità ad uno dei *literals* che la compongono)
2. una *clause* $c \in \varphi$ è **falsa** se $\forall l \in c | I : l \mapsto \bot$
3. una *clause* $c \in \varphi$ è **indecidibile** se non è né vera, né falsa

Ad ogni passo l'algoritmo effettua un'operazione di assegnazione ad un *literal* di una *clause* ancora in uno stato indecidibile. Data la formula φ e un literal p, indichiamo con $\varphi_{|p}$ la formula ottenuta sostituendo ad ogni occorrenza di p il valore di verità $\top$, analogamente $\varphi_{|\neg p}$ la formula ottenuta sostiutendo $\bot$ a p. Dalle osservazioni sulla *partial evaluation* segue:

- tutte le *clauses* contenenti almeno un *literal* valutato $\top$ possono ora essere rimosse
- tutte le occorrenze di *literal* valutati $\bot$ possono essere rimosse

L'algoritmo termina appena φ contiene una *clause* alla quale sono stati rimossi tutti i *literals* (detta **empty clause**), in tal caso la formula è **insoddisfacibile**; oppure quando φ non contiene più *clauses*, in tal caso la formula è **soddisfacibile** e $I' = I$.

Data: φ PL formula ridotta in CNF, I' modello parziale
Result: **SAT** se soddisfacibile, **UNSAT** altrimenti

DPLL(φ, I'):
UnitPropagation(φ, I')
if $\{\} \in \varphi$ **then**
  return *UNSAT*
if $\varphi = \{\}$ **then**
  return *(SAT, I')*
$l \leftarrow C \in \varphi$
DPLL($\varphi_{|l}$, $I' \cup \{I'(l) = \top\}$)
DPLL($\varphi_{|l}$, $I' \cup \{I'(l) = \bot\}$)

— 3 —

PL - Tableaux

Regole di riduzione

α rules

$$\frac{\phi \land \psi}{\begin{array}{c}\phi\\ \psi\end{array}} \qquad \frac{\phi \lor \psi}{\begin{array}{c}\neg\phi\\ \neg\psi\end{array}} \qquad \frac{\neg(\phi \supset \psi)}{\begin{array}{c}\phi\\ \neg\psi\end{array}}$$

$\neg\neg$ elimination

$$\frac{\neg\neg\phi}{\phi}$$

β rules

$$\frac{\phi \lor \psi}{\phi \mid \psi} \qquad \frac{\neg(\phi \land \psi)}{\neg\phi \mid \neg\psi} \qquad \frac{\phi \supset \psi}{\neg\phi \mid \psi}$$

Branch closure

$$\frac{\begin{array}{c}\phi\\ \neg\phi\end{array}}{\text{X}}$$

L'equivalenza può essere riscritta come doppia implicazione.

$$\phi \equiv \psi \iff (\phi \supset \psi) \land (\psi \supset \phi)$$

Osservazione: le α- e β rules del tableaux sono analoghe a quelle di riduzione in *CNF*:

- una α rule è equivalente a and logico $\land$ delle formule da ridurre;
- una β rule è equivalente a or logico (nella forma $\otimes$) fra tutte le formule da ridurre, prese a due a due.

Metodo del tableaux

Il **tableaux** è un metodo per provare se un insieme di formule dato è **insoddisfacibile**. Di conseguenza, è possibile dimostrare anche la **validità** dell'insieme di formule (dimostrando l'insoddisfacibilità della negazione dell'insieme di formule).

Il **tableaux** costruisce un albero binario, la cui radice è la congiunzione dell'insieme di formule di cui si vuole verificare l'insoddisfacibilità. Nuove foglie sono aggiunte applicando α rules (*deterministic rules*) o β rules (*branch splitting*)a una qualsiasi formula che appare in un nodo *ancestor*.

Un ramo dell'albero è **chiuso** se il cammino fra la foglia e la radice contiene formule contraddittorie (es. p, $\neg p$). Se tutti i rami possono essere chiusi, allora la formula di partenza è insoddisfacibile.

Osservazione: è conveniente applicare α rules anziché β rules, laddove possibile, in modo da non aumentare il numero di rami dell'albero.

Interpretazione dal tableaux

Si può dimostrare che un tableaux in PL termina sempre (dopo un numero finito di passi tutti i rami sono chiusi oppure tutte le formule che compaiono nel tableaux sono state valutate). Pertanto, se una formula genera un tableaux che non si chiude, la formula è **soddisfacibile**.

I modelli che rendono il tableaux soddisfacibile possono essere ricavati dai rami rimasti aperti. Per ogni ramo e per ogni variabile proposizionale p, vale $I(p) = \top$ se nel cammino dalla foglia alla radice compare p; $I(p) = \bot$ se nel cammino dalla foglia alla radice compare $\neg p$. Se né p né $\neg p$ compaiono, $I(p)$ può essere definito arbitrariamente (entrambe le definizioni renderanno la formula soddisfacibile).

— 4 —

FOL - Introduzione

La logica del primo ordine (First-Order Logic - **FOL**) è un'estensione della propositional logic.

Mentre la PL prevede solamente valori di verità o falsità, FOL prevede *variabili* che rappresentano oggetti del mondo da descrivere, inoltre, questi oggetti possono essere quantificati (si possono descrivere *tutti* gli oggetti o *alcuni* oggetti, senza nominare ciascuno esplicitamente, come sarebbe necessario in PL).

Definizione 5 (Sintassi della FOL)
Logical symbols: *Comprende gli stessi simboli della PL, e in più:*

1. *quantificatori* ($\forall$, $\exists$)
2. *variabili* $x_1, x_2, ...$
3. *simbolo di uguaglianza (opzionale)* $=$

Non-logical symbols:

- *costanti* $c_1, c_2, ...$
- *funzioni* $f_1, f_2, ...$ *alle quali è associata una arità*
- *relazioni* $P_1, P_2, ...$ *alle quali è associata una arità*

Terms e formule

Un **term** è l'elemento sintattico che rappresenta un oggetto del mondo.

Ci sono tre possibili tipologie di **term**:

1. **costanti**: descrivono sempre uno specifico oggetto (p. es. "Mario", "Giappone");
2. **variabili**: possono descrivere un qualsiasi oggetto, oppure essere associate ad un quantificatore;
3. **funzioni**: un simbolo applicato a zero, uno o più *terms* - il numero di *terms* è definito dalla *arità* del simbolo funzionale (oss: una funzione con *arità* 0 è equivalente ad una costante).

Un **predicate** o **relation** costituisce una "frase" in FOL. Un simbolo relazionale è applicato a zero, uno o più *terms* - il numero di *terms* è definito dalla *arità* del simbolo relazionale. I **predicate** rappresentano delle relazioni fra oggetti del mondo.

Il simbolo di uguaglianza = può essere visto come una relazione con arità uguale a 2.

Una **formula** è definita come segue.

1. Siano $t_1, ..., t_n$ *terms*, P relazione di arità n, allora $P(t_1, ..., t_n)$ è una formula. Vale anche: t_1, t_2 *terms*, $t_1 = t_2$ è una formula.
2. Siano A, B formule, allora $A \wedge B$, $A \vee B$, $A \supset B$, $A \equiv B$, $\neg A$ sono formule.
3. Sia A una formula, x una variabile, allora sono formule $\forall x.A$ e $\exists x.A$.

Interpretazione in FOL

In PL, un'interpretazione consiste nell'associazione di variabili proposizionali al valore vero o falso. In FOL l'interpretazione è definita in modo più complesso; è composta da:

1. un dominio di interpretazione Δ. Questo contiene tutti gli oggetti che vogliamo descrivere
2. una funzione di interpretazione I che mappa i simboli non logici in elementi del dominio:
   - $I(c_i) \in \Delta$ (mappa le costanti in elementi del dominio)
   - $I(P_i) \subset \Delta^n$ (mappa le relazioni di arità n in n-tuple)
   - $I(f_i) : \Delta^n \rightarrow \Delta$ (mappa le funzioni di arità n in elementi del dominio)

Osservazione (Relazioni e funzioni): esattamente come definite in algebra, le funzioni possono essere viste come una specializzazione delle relazioni. In altre parole, ogni funzione di arità n può essere equivalentemente rappresentata come una relazione di arità $n+1$. Esempio: $Mary$ è madre di Joe, $Jill$ e $Bill$. È possibile definire:

- $motherOf$ come una relazione di Δ^2:

$$motherOf := \{\langle Joe, Mary\rangle, \langle Jill, Mary\rangle, \langle Bill, Mary\rangle\}$$

- $motherOf$ come una funzione $\Delta \rightarrow \Delta$:

$$motherOf(Joe) = Mary, motherOf(Jill) = Mary, motherOf(Bill) = Mary$$

- $brotherOf$ come una relazione di Δ^2:

$$brotherOf := \{\langle Joe, Jill\rangle, \langle Jill, Joe\rangle, \langle Joe, Bill\rangle, \langle Bill, Joe\rangle, \langle Bill, Jill\rangle, \langle Jill, Bill\rangle\}$$

- ...ma non è possibile definire brotherOf come una funzione $\Delta \rightarrow \Delta$: $brotherOf(Jill) = ?$

Assignments

Formalmente, si definisce un'**assignment** $a[x/d]$ come una funzione (a) che mappa una variabile (x) in un elemento del dominio di interpretazione $d \in \Delta$. La funzione è interpretata come segue:

- se è applicata ad una costante, non ha alcun effetto:

$$I(c)[a[x/d]] = c$$

- se è applicata ad una variabile, avviene la sostituzione

$$I(x)[a[x/d]] = d$$

- se è applicata ad una funzione, l'assignment viene applicato ricorsivamente ai parametri della funzione

$$I(f(t_1, ..., t_n))[a[x/d]] = I(f)(I(t_1)[a[x/d]], ..., I(t_n)[a[x/d]])$$

Free variables

Una **occorrenza libera** (*free occourence*) di una variabile x in una formula φ è un'occorrenza di x che non è legata ad un quantificatore $(\forall, \exists)$.

Una **variabile** x è **libera** in φ se esiste almeno una occorrenza di x in φ che è libera.

Una **formula** φ è **ground** se non contiene nessuna variabile.

Una formula φ è **closed** se non contiene alcuna variabile libera.
Esempio: $\varphi := P(x) \supset \forall x.Q(x)$; x è una variabile libera, infatti la prima occorrenza di x è libera (la seconda non lo è).

Un *term* t è libero per una certa variabile x in una formula φ se tutte le occorrenze di x in φ non sono nello scope di un quantificatore di una variabile che ha occorrenze in t. In altre parole: t è libero per x in φ se si può sostituire t al posto di x senza che la formula cambi significato.
Esempio: $\varphi := \exists x.brotherOf(x, y)$; $t := z$. Il term t è libero per y: $\varphi' := \exists x.brotherOf(x, z)$ ha lo stesso significato di φ. Il term t non è però libero per x: $\varphi'' := \exists x.brotherOf(x, x)$ ha un significato diverso da φ.

Procedure di decisione in FOL

Innanzitutto è necessario definire quando una interpretazione è modello di una formula in FOL. Si osservi che, in presenza di variabili libere, il significato della variabile non è definito finché non viene effettuato un *assignment*. Scelte diverse degli *assignment* possono determinare se un'interpretazione è o non è modello di una formula φ.

Un'interpretazione I soddisfa (è un **modello** per) una formula φ rispetto ad un *assignment* $a[x/d]$ secondo le seguenti regole:

1. ("Caso base") Se la formula è una relazione P di arità n, allora

$$I \models P(t_1, ..., t_n)[a] \iff \langle I(t_1)[a], ..., I(t_n)[a] \rangle \in I(P)$$

   Ovvero: affiché φ sia soddisfatta deve esistere nell'interpretazione della relazione P una relazione che contenga i *terms* $t_1, ..., t_n$ a cui è stata applicata l'associazione a.
   Lo stesso è valido per la relazione di uguaglianza:

$$I \models (t_1 = t_2)[a] \iff I(t_1)[a] = I(t_2)[a]$$

2. (Formule composte) Siano φ, ψ formule, allora (al solito):
   - $I \models (\neg\varphi)[a] \iff I \not\models \varphi[a]$
   - $I \models (\varphi \wedge \psi)[a] \iff (I \models \varphi[a]) \wedge (I \models \psi[a])$
   - $I \models (\varphi \vee \psi)[a] \iff (I \models \varphi[a]) \vee (I \models \psi[a])$
   - $I \models (\varphi \rightarrow \psi)[a] \iff (I \not\models \varphi[a]) \vee (I \models \psi[a])$
   - $I \models (\varphi \equiv \psi)[a] \iff (I \models \varphi[a]) \equiv (I \models \psi[a])$

3. (Quantificatori) Sia φ una formula, allora:
   - $I \models (\exists q.\varphi)[a] \iff \exists z \in \Delta | I \models (\varphi[q/z])[a]$
   - $I \models (\forall q.\varphi)[a] \iff \forall z \in \Delta | I \models (\varphi[q/z])[a]$

Una formula φ è **soddisfacibile** se esiste una *interpretazione* I e un *assignment* a tali per cui $I \models \varphi[a]$.

Una formula φ è **insoddisfacibile** se non è soddisfacibile.

Una formula φ è **valida** se per ogni *interpretazione* I e per ogni *assignment* a vale $I \models \varphi[a]$.

Osservazione: Se una formula è **chiusa**, la sua validità o (in)soddisfacibilità non dipende dall'assegnazione a (l'assegnazione non ha alcun effetto sulla formula perché la formula non ha variabili libere sulle quali effettuare l'assegnazione).

Analogia con i DB

Un database relazionale può essere rappresentato in First-Order Logic; l'equivalenza è così descritta:

- le **relazioni** in FOL corrispondono alle **tabelle** del database;
- il dominio di interpretazione Δ in FOL corrisponde all'insieme di tutti i valori che compaiono nei dati contenuti nelle tabelle;
- la funzione di interpretazione I in FOL corrisponde alle tuple contenute nelle tabelle;

- formule FOL corrispono a query del DB; le interpretazioni che rendono la formula vera corrispondono alla risposta alla query.

Esempio: Si consideri il database composto dalla tabella

$$Employee(ename, dep, manager, age)$$

- Elencare tutti i dipartimenti nei quali *Joe* lavora per *Jill*:

  $$\exists y.Employee(Joe, x, Jill, y)$$

  Spiegazione: Employee è una relazione FOL; Joe, Jill sono costanti del dominio di interpretazione; x è variabile libera, y è bound dal quantificatore esistenziale. Sono modello della query tutte le interpretazioni che assegnano $ename = Joe$, un qualunque valore di dep e age, $manager = Jill$. Sono resituite le assegnazioni delle variabili libere (dunque il quantificatore esistenziale può essere usato come operatore di proiezione; le variabili che sono legate al quantificatore non sono restituite).
- Elencare i nomi e i dipartimenti di tutti i lavoratori che hanno età uguale a 40 anni.
  $$\exists m, a.Employee(x, y, m, a) \wedge a = 40$$
- Elencare i dati di tutti gli impiegati che sono manager di loro stessi (wat?):
  $$Employee(x, y, x, z)$$

Unique Name Assumption

La **UNA** (Unique Name Assumption) è un'assunzione che prevede che ogni elemento del dominio sia rappresentato da una e una sola costante. Tale assunzione si esprime in FOL con la seguente formula: siano $c_1, ..., c_n \subset \Delta$ tutte e sole le costanti del dominio, allora

$$\varphi_{UNA} \coloneqq (\bigwedge_{i=1}^{n} \bigwedge_{j=1, j \neq i}^{n} c_i \neq c_j) \wedge (\forall x \bigvee_{i=1}^{n} c_i = x)$$

Grounding

Se Δ è finito e vale la UNA, formule FOL possono essere proposizionalizzate (**grounded**, nel senso di rese *ground*) riformulando i quantificatori come segue:

- $\forall x.\varphi(x) \equiv \bigwedge_{1}^{n} \varphi(c_i)$
- $\exists x.\varphi(x) \equiv \bigvee_{1}^{n} \varphi(c_i)$

— 5 —

FOL - Tableaux