Appunti di Logica

Sul corso di Logica Matematica del Prof. Giunchiglia,
Università degli Studi di Trento

A.A. 2018/2019

Giacomo Fabris

— 1 —

PL - Introduzione

La logica delle proposizioni (Propositional Logics -**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** $I : P \rightarrow \{\top, \bot\}$ 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 (**SAT**) 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**, ¬A è **UNSAT**.
Se A è **SAT**, ¬A non è valida.
Se A non è valida, ¬A è **SAT**.

Se A è **UNSAT**, ¬A è **VALID**.

Conseguenza e equivalenza logica

- Una formula A è una **conseguenza logica** di un insieme di formule Γ, in simboli $\Gamma \models$ 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 A $\equiv$ 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"
- $A \land B$: "A and B", "A but B", "Although A, B", "Both A and B"
- $A \lor B$: "A or B", "Either A or B"
- $A \to B$: "If A, then B", "B if A"
- $\neg(A \lor B)$: "Neither A nor B"
- $\neg(A \land B)$: "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($\phi \land \psi$) $=$ CNF(ϕ) $\land$ CNF(ψ)

- $\text{CNF}(\phi \vee \psi) = \text{CNF}(\phi) \oplus \text{CNF}(\psi)$
- $\text{CNF}(\phi \equiv \psi) = \text{CNF}(\phi \to \psi) \wedge \text{CNF}(\psi \to \phi)$

- $\text{CNF}(\neg(\phi \to \psi)) = \text{CNF}(\phi) \wedge \text{CNF}(\neg\psi)$
- $\text{CNF}(\neg(\phi \wedge \psi)) = \text{CNF}(\neg\phi) \oplus \text{CNF}(\neg\psi)$
- $\text{CNF}(\neg(\phi \vee \psi)) = \text{CNF}(\neg\phi) \wedge \text{CNF}(\neg\psi)$
- $\text{CNF}(\neg(\phi \equiv \psi)) = \text{CNF}(\phi \wedge \neg\psi) \oplus \text{CNF}(\psi \wedge \neg\phi)$

Dove $\oplus$ è definito come segue:

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

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

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.

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