

**Soddisfacibilità
e *Semantic Tableau***

Marco Piastra

Problemi e decidibilità (automatica)

■ Problema

In forma rigorosa, un problema è una **relazione** tra *istanze* (input) e *soluzioni*

$K : I \rightarrow S$ (K è la relazione, I è lo spazio delle istanze, S quello delle soluzioni)

■ Problema di ricerca

La relazione K associa ciascuna *istanza* a un numero qualsiasi di soluzioni

Problemi di ottimizzazione

Esiste una funzione costo definita su S : $c : S \rightarrow \mathbf{R}$ (\mathbf{R} è l'insieme dei numeri reali)

Si tratta di trovare la soluzione valida a costo minimo o massimo

■ Problema di decisione

L'insieme delle soluzioni S coincide con $\{0, 1\}$ e la soluzione è unica

Esempio: $\Gamma \models \varphi$?

L'insieme I è formato da tutti i possibili insiemi di fbf Γ e tutte le possibili fbf φ

■ Problema decidibile

È un problema di decisione per cui esiste una *procedura effettiva* o *algoritmo* che calcola la risposta corretta (per qualsiasi input ammissibile)

- Vale a dire se esiste una macchina di Turing in grado di calcolare la risposta

(Vi sono altri modi di definire una procedura effettiva, tutti equivalenti)

Problemi e decidibilità (automatica)

Esempio di problema *indecidibile*

Data una macchina di Turing e lo stato iniziale del nastro, l'esecuzione è terminante?

In altri termini, esiste una macchina di Turing che, data la tabella di transizione e l'input di un'altra macchina di Turing, è in grado di dire se l'esecuzione termina?

La risposta è no. (non esiste una simile macchina di Turing)

■ Problemi di decisione come problemi di ricerca

Un problema di decisione può essere definito come un problema di ricerca

Esempio: esiste in un grafo un cammino da un nodo a a un nodo b ?

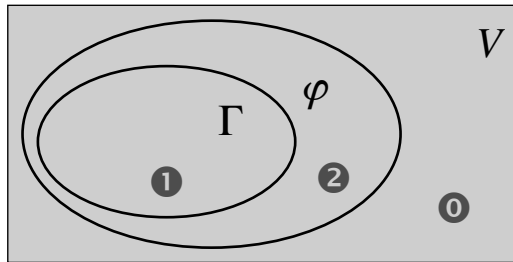
Problemi di decisione come problemi di ottimizzazione

Esempio: esiste in un grafo un cammino da un nodo a a un nodo b che abbia lunghezza $< d$?

Conseguenza logica come soddisfacibilità

- Il problema $\Gamma \models \varphi$ può essere trasformato in un problema di **soddisfacibilità**

E' facile vedere che $\Gamma \models \varphi$ sse $\Gamma \cup \{\neg\varphi\}$ è insoddisfacibile



(Nell'algebra dei sottoinsiemi di interpretazioni)

$$\Gamma \models \varphi \Rightarrow v(\Gamma) \subseteq v(\{\varphi\})$$

$$\mathbf{1} \subseteq \{\mathbf{1}, \mathbf{2}\}$$

$$v(\{\neg\varphi\}) = \mathbf{0}$$

$$v(\Gamma \cup \{\neg\varphi\}) = v(\Gamma) \cap v(\{\neg\varphi\})$$

$$\mathbf{1} \cap \mathbf{0} = \emptyset$$

$$v(\Gamma \cup \{\neg\varphi\}) = \emptyset$$

Inoltre, l'insieme $\Gamma \cup \{\neg\varphi\}$ è soddisfacibile sse $\bigwedge (\Gamma \cup \{\neg\varphi\})$ lo è

Morale:

Il problema $\Gamma \models \varphi$ può essere trasformato

nel problema di stabilire se $\bigwedge (\Gamma \cup \{\neg\varphi\})$ è soddisfacibile

(chiusura congiuntiva di tutte le fbf dell'insieme)

Soddisfacibilità (in L_P)

- La soddisfacibilità di una fbf φ è decidibile

E' un problema di decisione definito come problema di ricerca:

Esiste un'interpretazione che soddisfa φ ?

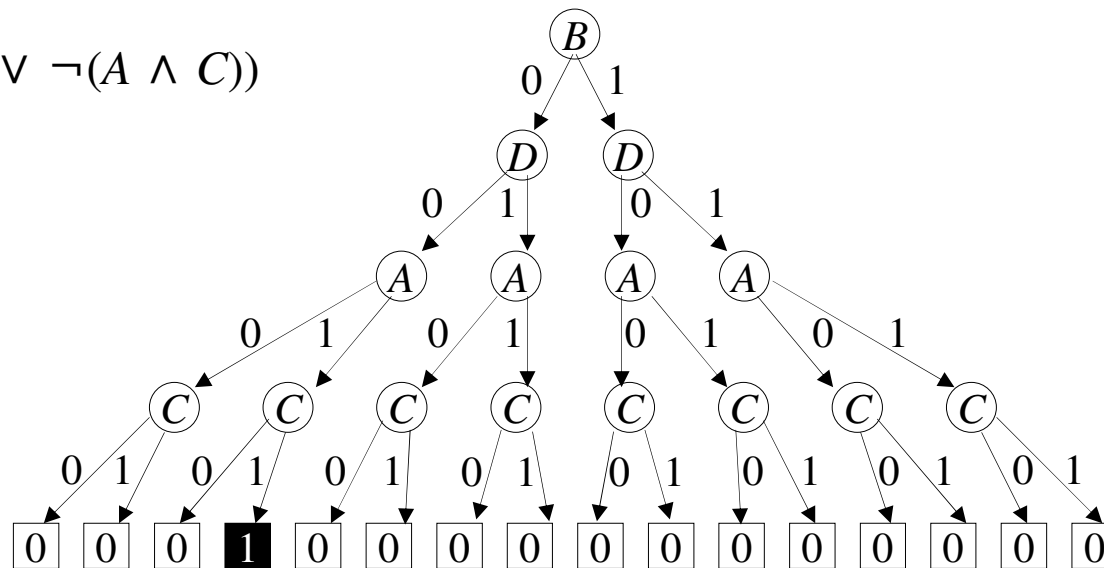
L'insieme degli input è formato da tutte le possibili fbf φ

In letteratura, il problema è denominato SAT

E' *decidibile*: basta provare tutte i possibili valori di verità dei simboli proposizionali in φ

Esempio:

$$\neg(B \vee D \vee \neg(A \wedge C))$$



La procedura ha complessità $O(2^n)$, pari al numero di assegnazioni da provare

Complessità di calcolo, classi P e NP

Il concetto di applica solo ai *problemi decidibili*

Si considera la *miglior* macchina di Turing (conosciuta) che calcola la risposta

■ Complessità di tempo

Il numero di *passi* necessari alla macchina di Turing per calcolare la risposta, in funzione delle dimensioni dell'input

■ Complessità di memoria

Il numero di *celle del nastro* necessarie alla macchina di Turing per calcolare la risposta, in funzione delle dimensioni dell'input

■ Classe P

Tutti i problemi per cui è nota una procedura effettiva con complessità di *tempo* $O(P(n))$

Dove $P()$ è un polinomio di grado qualsiasi (finito) e n è la dimensione dell'input

■ Classe NP

Tutti i problemi decidibili per cui esiste (almeno):

a) Una procedura che enumera tutte le possibili risposte (enumerabilità ricorsiva)

b) Una procedura con complessità P che **verifica** se una risposta è una soluzione

Include tutti i problemi della classe P (vale a dire $P \subseteq NP$)

Include anche problemi per cui non è nota una procedura con complessità P

La classe NP-complete ed il problema SAT

- Classe NP-complete

E' una sottoclasse di NP ($\text{NP-complete} \subseteq \text{NP}$)

Un problema K è NP-complete se tutti i problemi di NP sono **riducibili** a K

- Riducibilità

Nel caso della classe NP-complete

Si consideri un problema K e si supponga di avere una procedura di decisione $p(K)$

Un problema J è **riducibile** a K se esiste una procedura $p(J)$ di decisione per J che:

a) Chiama una volta sola, alla fine, $p(K)$ come "subroutine"

b) Ha una complessità di tempo polinomiale (trascurando la complessità di $p(K)$)

- Il problema SAT

E' NP-complete

Morale: se esistesse una procedura per SAT con complessità polinomiale, si avrebbe

$P = NP$

E' un fatto non noto, si suppone di no: $P \neq NP$

Semantic Tableau, regole alfa e beta

- Un tableau è un'insieme di fbf

Le regole (alfa e beta) trasformano un tableau in uno o due nuovi tableau

Si parte da un tableau iniziale

- Regole alfa (o di espansione)

(a1)	(a2)	(a3)	(a4)
$\neg(\neg\varphi)$	$\varphi \wedge \psi$	$\neg(\varphi \vee \psi)$	$\neg(\varphi \rightarrow \psi)$
φ	φ, ψ	$\neg\varphi, \neg\psi$	$\varphi, \neg\psi$

- Regole beta (o di biforcazione)

(b1)	(b2)	(b3)	(b4)	(b5)
$\varphi \vee \psi$	$\neg(\varphi \wedge \psi)$	$\varphi \rightarrow \psi$	$\varphi \leftrightarrow \psi$	$\neg(\varphi \leftrightarrow \psi)$
φ ψ	$\neg\varphi$ $\neg\psi$	$\neg\varphi$ ψ	$\neg\varphi, \neg\psi$ φ, ψ	$\neg\varphi, \psi$ $\varphi, \neg\psi$

Metodo dei *Semantic Tableau*

- Procedura:

Dato il problema $\Gamma \models \varphi$

E' un metodo per **refutazione**:

- si assume $\Gamma \cup \{\neg\varphi\}$ come nodo (*tableau*) iniziale
- lo scopo è provare che $\Gamma \cup \{\neg\varphi\}$ è **insoddisfacibile**

Per ciascun tableau attivo:

- Se il tableau contiene solo letterali,
 - se il tableau contiene una contraddizione, chiudere
 - altrimenti terminare la procedura [tableau aperto = fallimento]
- Se il tableau contiene formule composite:
 - a) applicare le regole alfa
 - b) applicare le regole beta

Notare: le regole alfa vengono applicate prima delle regole beta

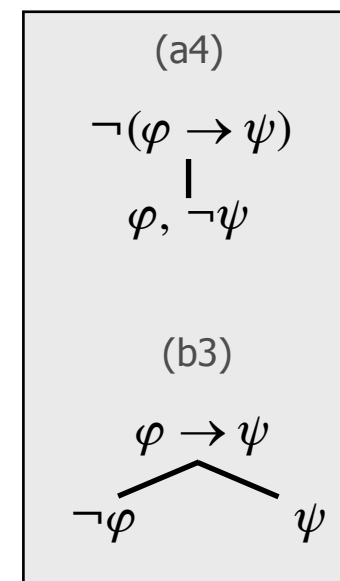
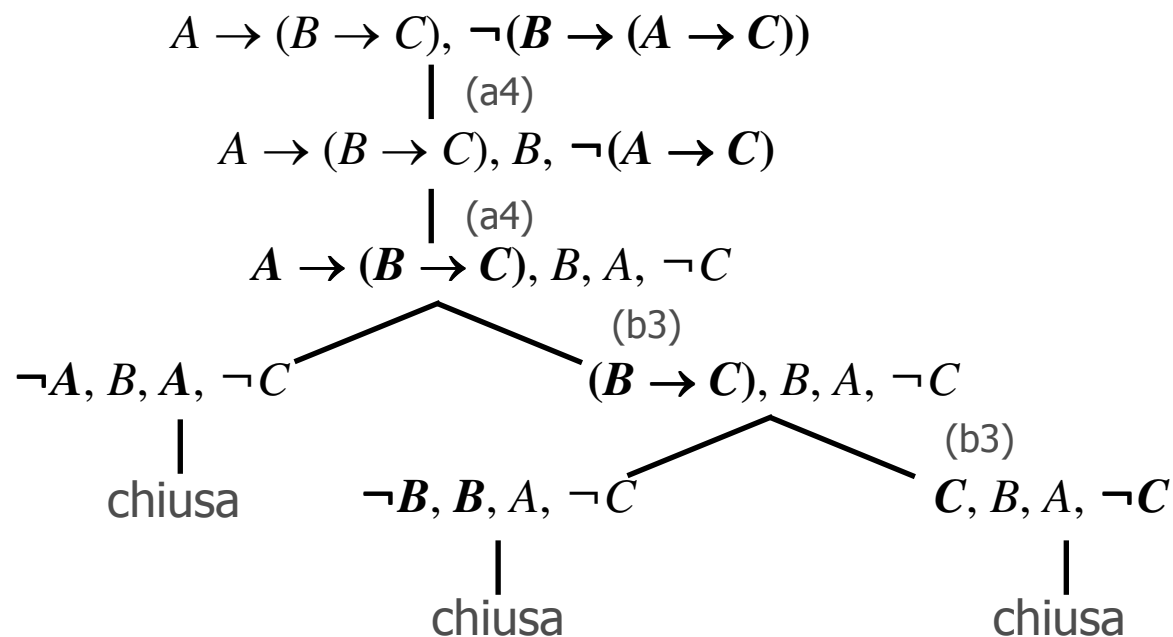
Esempio del metodo dei *Semantic Tableau*

- Obiettivo: decidere se $\Gamma \models \varphi$

Esempio: mostrare che $A \rightarrow (B \rightarrow C) \models B \rightarrow (A \rightarrow C)$

- Espansione

Per definizione, il tableau iniziale deve essere $\Gamma \cup \{\neg\varphi\}$



Proprietà del metodo dei *Semantic Tableau*

- **Terminante**

Le regole alfa e beta semplificano progressivamente le formule contenute nei tableau
Ad un certo punto restano solo letterali in forma positiva o negata (del tipo A oppure $\neg A$)

- **Corretto (*Sound*)**

Ogni tableau $\Gamma \cup \{\neg\varphi\}$ per cui il metodo termina con successo
(tutti i tableau terminali sono *chiusi*) è effettivamente *insoddisfacibile*

- **Completo**

Per qualsiasi tableau $\Gamma \cup \{\neg\varphi\}$ *insoddisfacibile*, il metodo termina con successo
(tutti i tableau terminali sono *chiusi*)

- **Correttezza + Completezza = Procedura di Decisione**

Il metodo a tableau è una procedura di decisione per la logica proposizionale

Quale metodo è più veloce?

- Complessità asintotica

E' la stessa: $O(2^n)$

- In pratica?

Dipende.

Esempio 1:

$$A \wedge B \wedge C \wedge \neg A$$

Con il metodo dei valori di verità si effettuano $2^3=8$ prove

Con il metodo dei tableau si applicano 3 regole alfa e si ottiene un tableau chiuso

Esempio 2:

$$(A \vee B) \wedge (A \vee \neg B) \wedge (\neg A \vee B) \wedge (\neg A \vee \neg B)$$

Con il metodo dei valori di verità si effettuano $2^2=4$ prove

Con il metodo dei tableau si applicano 3 regole alfa
poi l'applicazione delle regole beta produce un albero a 4 livelli,
ciascuno con un fattore di branching 2

Si ottengono così $2^4=16$ tableau, tutti chiusi