Intelligenza Artificiale I

Logica formale

Automazione del calcolo

Marco Piastra

Calcolo automatico

- Una logica (p.es. L_P) è **decidibile**

  Se esiste una **procedura effettiva** per stabilire se: $\Gamma \models \varphi$

- Soddisfacibilità

  Stabilire se un certo insieme di fbf Σ è soddisfacibile

  (= se esiste un'interpretazione che soddisfa simultaneamente tutte le fbf di Σ)

  Ogni insieme di fbf Σ è logicamente equivalente ad una fbf $\bigwedge\Sigma$

  (chiusura congiuntiva di tutte le fbf dell'insieme)

  Esempio:

  $\Sigma \equiv \quad \{B \vee D \vee \neg(A \wedge C),\ B \vee C,\ A \vee D,\ \neg B\}$

  $\bigwedge\Sigma \equiv (B \vee D \vee \neg(A \wedge C)) \wedge (B \vee C) \wedge (A \vee D) \wedge \neg B$

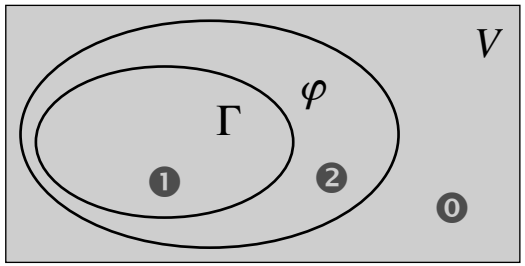
  Vale a dire che Σ è soddisfacibile sse $\bigwedge\Sigma$ lo è

Soddisfacibilità e conseguenza logica

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

  Perché:

  Si può dimostrare che $\Gamma \models \varphi$ sse $\Gamma \cup \{\neg\varphi\}$ è insoddisfacibile



(Nell'algebra dei sottoinsiemi di interpretazioni)

$\Gamma \models \varphi \Rightarrow v(\Gamma) \subseteq v(\{\varphi\})$ ❶ ⊆ ❶ + ❷

$v(\{\neg\varphi\}) = ❼$

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

$v(\Gamma \cup \{\neg\varphi\}) = \varnothing$ ❶ ∩ ❼ = ∅

Dato $\Gamma \cup \{\neg\varphi\}$, si consideri la fbf $\bigwedge(\Gamma \cup \{\neg\varphi\})$
(chiusura congiuntiva di tutte le fbf dell'insieme)

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

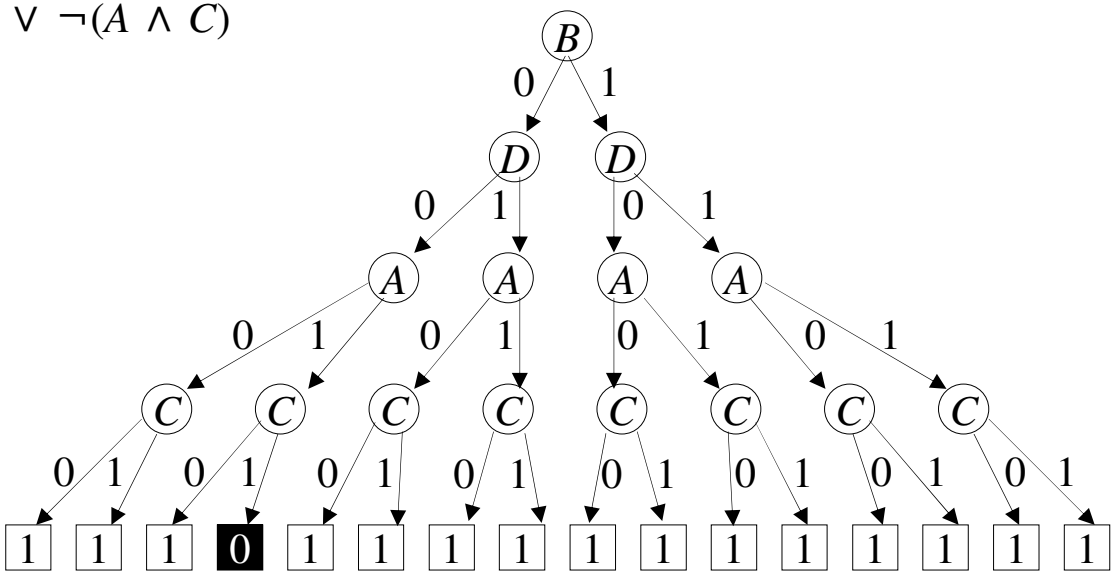
Soddisfacibilità (in L_P)

- Esiste una **procedura effettiva** per stabilire se una fbf è soddisfacibile

  Basta provare tutte le possibili assegnazioni di valore di verità

  Esempio:

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



  La procedura esaustiva ha complessità $O(2^n)$

- Il problema SAT è NP-completo

  Qualsiasi problema NP-completo è traducibile in SAT in tempo polinomiale

Derivazione automatica

- La definizione di derivabilità 'a la Hilbert' (derivazione da schemi di assioma) non è traducibile in una procedura effettiva

  Perché:

  Volendo stabilire se $\Gamma \vdash \varphi$

  Possiamo inserire una ad una nella dimostrazione le fbf $\psi \in \Gamma$ (OK)

  Possiamo applicare la regola MP non appena possibile (OK)

  NON possiamo inserire tutte le istanze degli assiomi Ax_n (KO)

  A rigor di teoria, tali istanze sono infinite

  Morale: *il problema sono gli assiomi*

- Esiste una procedura effettiva per la derivazione automatica?

  (Intesa come calcolo puramente simbolico)

  In caso contrario possiamo solo automatizzare il metodo delle tavole di verità

Regola di risoluzione

- Una nuova regola di inferenza (calcolo simbolico)

$$\varphi \vee \chi, \neg\chi \vee \psi \vdash \varphi \vee \psi$$

La fbf $\varphi \vee \psi$ viene anche detta *risolvente* di $\varphi \vee \chi$ e $\neg\chi \vee \psi$

La regola di risoluzione è corretta

φ	ψ	χ	$\varphi \vee \chi$	$\neg\chi \vee \psi$	$\varphi \vee \psi$
0	0	0	0	1	0
0	0	1	1	0	0
0	1	0	0	1	1
0	1	1	1	1	1
1	0	0	1	1	1
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	1

La regola del *modus ponens* è un caso particolare di *risoluzione*

$\chi \rightarrow \psi, \chi \vdash \psi$ può essere riscritta come $\chi, \neg\chi \vee \psi \vdash \psi$

Forme normali

- In generale
  Traduzione di una fbf qualsiasi in una forma logicamente equivalente

- **Forma normale congiuntiva** (FNC)
  Una fbf in cui $\wedge$ appare solo al livello più esterno

  $\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n$ α_i sono fbf in cui compare solo $\vee$ e $\neg$ (solo davanti alle lettere)

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

- **Forma normale disgiuntiva** (FND)
  Una fbf in cui $\vee$ appare solo al livello più esterno

  $\beta_1 \vee \beta_2 \vee \ldots \vee \beta_n$ β_i sono fbf in cui compare solo $\wedge$ e $\neg$ (solo davanti alle lettere)

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

Forma normale congiuntiva

- Traduzione automatica in FNC

  Algoritmo:

  Si eliminano $\rightarrow$ e $\leftrightarrow$ in base alle regole di riscrittura

  Si muove $\neg$ all'interno

  con le "leggi De Morgan": $\neg(\varphi \wedge \psi) \Leftrightarrow (\neg\varphi \vee \neg\psi)$
  $\neg(\varphi \vee \psi) \Leftrightarrow (\neg\varphi \wedge \neg\psi)$

  Si distribuisce $\vee$

  proprietà del connettivo: $((\varphi \wedge \psi) \vee \chi) \Leftrightarrow ((\varphi \vee \chi) \wedge (\psi \vee \chi))$

  Esempi:

  $(\neg B \rightarrow D) \vee \neg(A \wedge C)$
  $B \vee D \vee \neg(A \wedge C)$ (eliminazione di $\rightarrow$)
  $B \vee D \vee \neg A \vee \neg C$ (De Morgan)

  $\neg(B \rightarrow D) \vee \neg(A \wedge C)$
  $\neg(\neg B \vee D) \vee \neg(A \wedge C)$ (eliminazione di $\rightarrow$)
  $(B \wedge \neg D) \vee (\neg A \vee \neg C)$ (De Morgan)
  $(B \vee \neg A \vee \neg C) \wedge (\neg D \vee \neg A \vee \neg C)$ (distribuzione di $\vee$)

Forma a clausole

- In generale:

  Traduzione di un insieme di fbf in un insieme equivalente
  in cui occorrono solo $\vee$ e $\neg$

- Traduzione in FC

  Dato un insieme di fbf tradotte in FNC

  Si sostituisce ciascuna fbf della forma $\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n$
  con l'insieme di fbf $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ (le congiunzioni diventano implicite)

  Esempi:

  $B \vee D \vee \neg A \vee \neg C$ (FNC e FC)

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

  $\{(\neg B \vee \neg A \vee \neg C),\ (\neg D \vee \neg A \vee \neg C)\}$ (FC)

  Notazione:

  Le clausole si scrivono come insiemi di letterali (negativi e positivi)

  $\{B, D, \neg A, \neg C\}$

  $\{\{\neg B, \neg A, \neg C\},\ \{\neg D, \neg A, \neg C\}\}$

Derivazioni: risoluzione

- Il problema visto in precedenza ("Giorgio è contento")

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

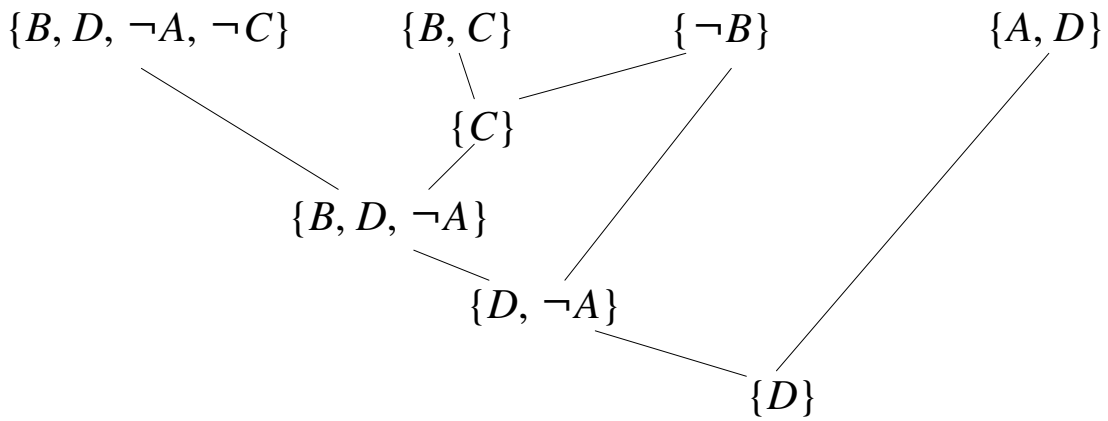
  Riscrivendo le formule dell'ipotesi in FNC:

  $B \vee D \vee \neg A \vee \neg C, B \vee C,\ A \vee D,\ \neg B$

  In FC:

  $\{B, D, \neg A, \neg C\}, \{B, C\}, \{A, D\},\ \{\neg B\}$

  Applicando la regola di risoluzione (in forma binaria):



Risoluzione: ipotesi di metodo

- Descrizione:

  Volendo stabilire se $\Gamma \vdash \varphi$

  Si traducono Γ e φ in FNC e quindi in FC

  Si applica la regola di risoluzione, in forma *binaria* (una coppia di letterali alla volta) ed in modo esaustivo

  Si ottiene un insieme Σ di fbf in FC, $\Sigma = \{\psi : \Gamma \vdash \psi, \textit{per risoluzione}\}$

  L'algoritmo termina con successo se $\varphi \in \Sigma$

  Vantaggi:

  Niente assiomi

  Una sola regola (risoluzione)

  Procedura effettiva (algoritmo)

  Svantaggi:

  Come metodo di derivazione, **non è completo**

  Contro-esempio:

  $B \vee D \vee \neg A \vee \neg C, B \vee C,\ A \vee D,\ \neg B \vdash D \vee E$

  Identico al caso precedente, con $D \vee E$ al posto di D, ma la derivazione fallisce

Risoluzione per refutazione

- Descrizione:
  - Volendo stabilire se $\Gamma \vdash \varphi$
  - Si trasforma il problema in $\Gamma \cup \{\neg\varphi\}$ (refutazione della tesi φ)
    - Se $\Gamma \vdash \varphi$, $\Gamma \cup \{\neg\varphi\}$ è insoddisfacibile
  - Si traduce $\Gamma \cup \{\neg\varphi\}$ in FNC e quindi in FC
  - Si applica la regola di risoluzione, in forma ristretta
    - (Forma ristretta: si risolve un solo simbolo proposizionale alla volta)
    - L'algoritmo termina con successo quando viene derivata una **clausola vuota** $\{\ \}$
    - Diversamente, l'algoritmo termina quando la risoluzione non è più applicabile (fallimento)
- Vantaggi:
  - Niente assiomi
  - Una sola operazione (applicazione della regola di risoluzione)
  - Procedura effettiva (algoritmo)
  - Come metodo di derivazione, è completo

Derivazioni: risoluzione per refutazione

- Il contro-esempio visto in precedenza

  $B \vee D \vee \neg A \vee \neg C, B \vee C, A \vee D, \neg B \vdash D \vee E$

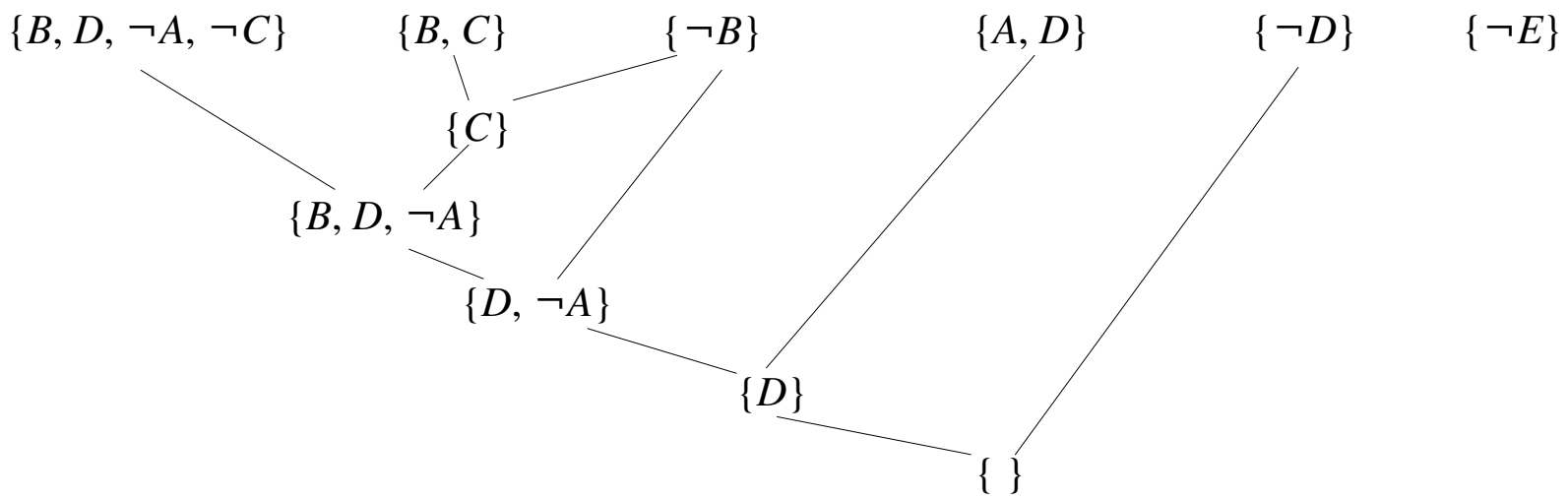
  Refutando la tesi e riscrivendo il tutto in FNC:

  $B \vee D \vee \neg A \vee \neg C, B \vee C, A \vee D, \neg B, \neg D, \neg E$

  In FC:

  $\{B, D, \neg A, \neg C\}, \{B, C\}, \{A, D\}, \{\neg B\}, \{\neg D\}, \{\neg E\}$

  Applicando la regola di risoluzione:



Metodo della risoluzione per refutazione

- Il metodo della risoluzione per refutazione in L_P

  Termina sempre

  E` corretto: $\Gamma \vdash \varphi \Rightarrow \Gamma \models \varphi$

  E` completo: $\Gamma \models \varphi \Rightarrow \Gamma \vdash \varphi$

- Complessità computazionale:

  Ha complessità $O(2^n)$ dove n è il numero delle *lettere*

Clausole di Horn

- Definizione

  Una **clausola di Horn** è una fbf in FC
  in cui si ha al massimo un letterale in forma positiva

- Tre tipi particolari:

  Regole: più letterali, uno solo positivo

  Esempi: $\{B, \neg D, \neg A, \neg C\}, \{A, \neg B\}$ (traduzione: $(D \wedge A \wedge C) \rightarrow B, B \rightarrow A$)

  Fatti: un solo letterale, positivo

  Esempi: $\{B\}, \{A\}$

  Goal: uno o più letterali, tutti negativi

  Esempi: $\{\neg B\}, \{\neg A, \neg B\}$ (refutazione di: $B,\ A \wedge B$)

  Ulteriore terminologia

  Regole e fatti si dicono anche clausole definite (*definite clauses*)

  I goal si dicono anche clausole negative (*negative clauses*)

Traduzione in clausole di Horn

- Non tutte le fbf sono traducibili in clausole di Horn

Esempi (negativi):

$(A \wedge \neg B) \rightarrow C$

$\neg(A \wedge \neg B) \vee C$ (eliminazione di $\rightarrow$)

$\neg A \vee B \vee C$ (De Morgan)

$A \rightarrow (B \vee C)$

$\neg A \vee B \vee C$ (eliminazione di $\rightarrow$)

Esempi (positivi):

$(A \wedge B) \rightarrow C$

$\neg(A \wedge B) \vee C$ (eliminazione di $\rightarrow$)

$\neg A \vee \neg B \vee C$ (De Morgan - FC - è una regola)

$A \rightarrow (B \wedge C)$

$\neg A \vee (B \wedge C)$ (eliminazione di $\rightarrow$)

$(\neg A \vee B) \wedge (\neg A \vee C)$ (distribuzione di $\vee$)

$(\neg A \vee B), (\neg A \vee C)$ (FC – si generano due regole)

$(A \vee B) \rightarrow C$

$\neg(A \vee B) \vee C$ (eliminazione di $\rightarrow$)

$(\neg A \wedge \neg B) \vee C$ (De Morgan)

$(\neg A \vee C) \wedge (\neg B \vee C)$ (distribuzione di $\vee$)

$(\neg A \vee C), (\neg B \vee C)$ (FC – si generano due regole)

Risoluzione SLD

Linear resolution with Selection function for Definite clauses

- Metodo particolare di risoluzione per refutazione

  Si applica ad un set di clausole definite + un goal

- Descrizione

  Applicazione sequenziale della regola di risoluzione (in forma ristretta)
  a partire dal goal usando una specifica *selection function*

  Esempio:

  Clausole definite: $\{C\}, \{D\}, \{B, \neg D\}, \{A, \neg B, \neg C\}$

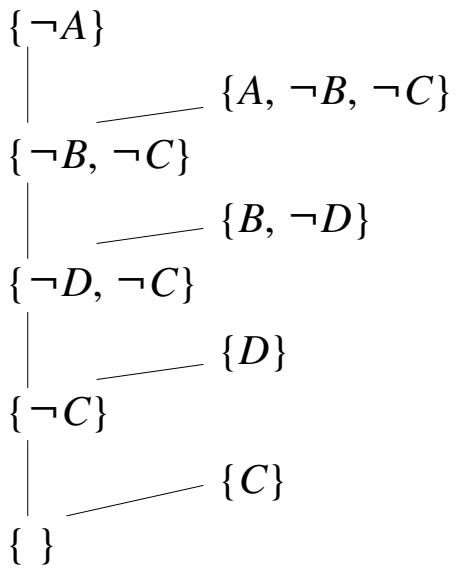
  Goal: $\{\neg A\}$

  Selection function

  Deve essere adeguata (*fair*):
  prima o poi tutti i subgoal devono essere selezionati

  In pratica si usa (quasi) sempre la
  leftmost subgoal first

  Equivale ad un *depth-first*:
  per essere *fair* si dovrebbe usare il *breadth-first*



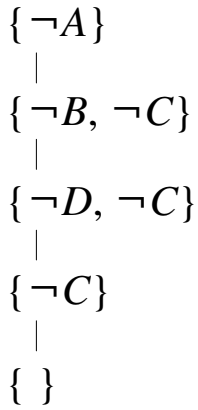
Alberi SLD

- Derivazioni SLD

  Ciascun letterale nel goal è risolto in un solo modo

  Esempio: $\{C\}, \{D\}, \{B, \neg D\}, \{A, \neg B, \neg C\}$ goal $\{\neg A\}$

Si traccia la progressione del goal

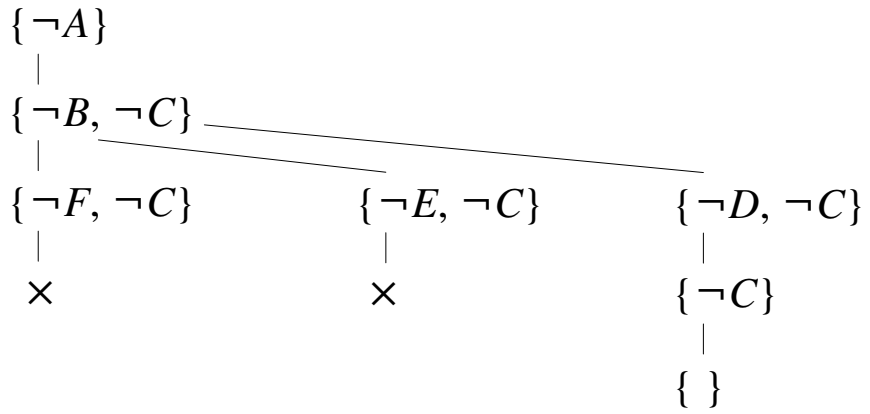


- Alberi SLD (=tutte le possibili derivazioni SLD)

  Una diramazione per ciascun modo di risolvere un letterale nel goal

  Esempio: $\{C\}, \{D\}, \{B, \neg F\}, \{B, \neg E\}, \{B, \neg D\}, \{A, \neg B, \neg C\}$ goal $\{\neg A\}$

Notare la strategia di ricerca *depth first*: i nuovi *subgoal* vengono inseriti all'inizio della lista



Metodo della risoluzione SLD

- Il metodo della risoluzione SLD in L_P
  - Si applica solo alle clausole di Horn (clausole definite)
  - Termina sempre
  - E` corretto (per le clausole di Horn): $\Gamma \vdash \varphi \Rightarrow \Gamma \models \varphi$
  - E` completo (per le clausole di Horn): $\Gamma \models \varphi \Rightarrow \Gamma \vdash \varphi$

- Complessità computazionale:
  - Ha complessità polinomiale (rispetto al numero di lettere ed alla dimensione delle fbf)
    - Rispetto ai metodi generali, limitandosi alle clausole di Horn si ottiene un'efficienza molto elevata
- Limitazioni
  - Non tutti i problemi sono traducibili in clausole di Horn
    - 'Giorgio è contento' non lo è
    - Solo combinazioni di regole e fatti (con fatti – negati – come goal)

Lo strano caso della negazione come fallimento

Negazione come fallimento (*Negation as Failure - NF*)

Come noto, non tutte le fbf sono traducibili in clausole di Horn

In particolare:

$(\neg A \wedge B) \rightarrow C$

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

$A \vee \neg B \vee C$ (clausola con due letterali positivi)

- Si legga il connettivo ¬ nella premessa della regola come 'non è noto'

  Esempio:

  $(\neg A \wedge B) \rightarrow C$

  Se A non è noto e B è vero, allora C

  Salvo *"diversa comunicazione"* e *"oggi è venerdì"*, allora *"c'è lezione di IA1"*

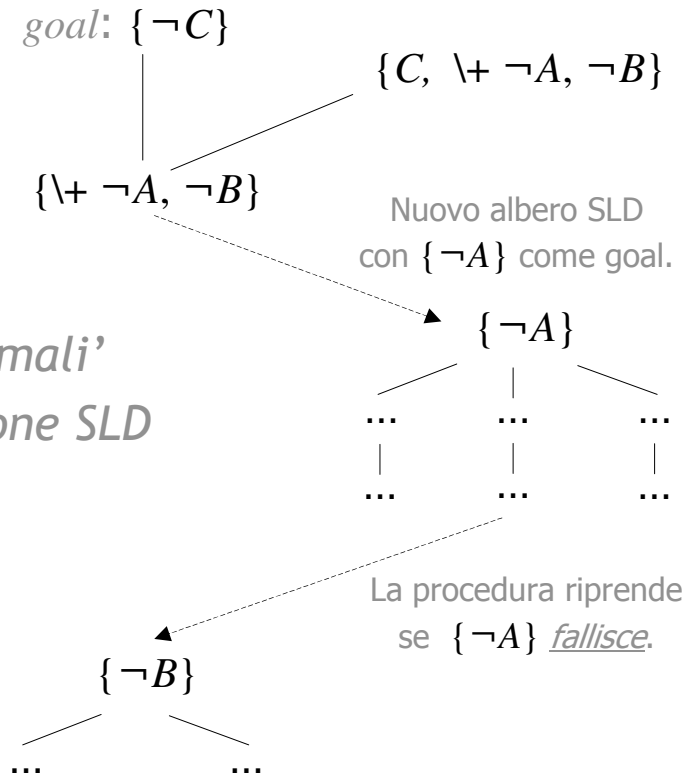
Risoluzione SLDNF

- Traduzione speciale di regole con premesse negative

  $(\neg A \wedge B) \rightarrow C$ si traduce come
  $\backslash+\ \neg A \vee \neg B \vee C$ (il simbolo \+ indica la *NF*)

- Metodo

  Le premesse negative sono interpretate come goal la cui derivazione SLD deve *fallire*



Il metodo SLDNF procede come SLD per i goal 'normali'

Incontrando un $\backslash+\ \neg A$ *, si apre una nuova derivazione SLD*

Il goal $\backslash+\ \neg A$ *ha successo se* $\neg A$ *fallisce*

Risoluzione SLDNF e completamento

- Completamento
  1) Dato un insieme Γ di clausole di Horn, si aggregano le regole in modo che ciascun atomo compaia al massimo in una implicazione

     Esempio: $\Gamma \equiv \{\{C\}, \{B, \neg F\}, \{B, \neg E\}, \{B, \neg D\}\}$

     Più comodo scrivere: $\{C, F \rightarrow B, E \rightarrow B, D \rightarrow B\}$

     $\Gamma' \equiv \{C, D \vee E \vee F \rightarrow B\}$
  2) Per ogni atomo φ non definito si aggiunge $false \rightarrow \varphi$ (*false* indica la *contraddizione*)

     Nell'esempio: $\Gamma'' \equiv \{C, D \vee E \vee F \rightarrow B, false \rightarrow D, false \rightarrow E, false \rightarrow F\}$
  3) Si sostituisce l'implicazione $\rightarrow$ con l'equivalenza $\leftrightarrow$

     $Comp(\Gamma) \equiv \{C, D \vee E \vee F \leftrightarrow B, false \leftrightarrow D, false \leftrightarrow E, false \leftrightarrow F\}$ (completamento di Γ)

- Correttezza di SLDNF (Clark, 1974)

  Il goal SLDNF φ ha successo per $\Gamma \Rightarrow Comp(\Gamma) \models \varphi$

  Esempio:

  $\Gamma \equiv \{\{C\}, \{B, \neg F\}, \{B, \neg E\}, \{B, \neg D\}\}$

  Il goal $\neg B$ ha successo SLDNF perchè $\neg B$ *fallisce* in SLD

  In effetti: $\{C, D \vee E \vee F \leftrightarrow B, false \leftrightarrow D, false \leftrightarrow E, false \leftrightarrow F\} \models \neg B$

Lo strano caso della SLDNF

- Non è completa

  $Comp(\Gamma) \models \neg\varphi \;\Rightarrow\; \Gamma \vdash_{SLDNF} \neg\varphi$

  Infatti la SLDNF può andare in loop

  Esempio:

  $\Gamma \equiv \{Q \to P,\ \neg Q \to P,\ Q \to Q\}$

  $Comp(\Gamma) \equiv \{Q \vee \neg Q \leftrightarrow P,\ Q \leftrightarrow Q\}$ $\qquad$ $Comp(\Gamma) \models P$

  Ma la SLDNF va in loop: si dovrebbe mostrare che la SLD *fallisce* per $\Gamma \cup \{\neg P\}$
  ma nessuna *selection function* che sia *fair* può evitare il ciclo infinito causato da $Q \to Q$

- Non è più logica classica

  Non vale più la proprietà di monotonia sintattica

  $\Gamma \vdash_{SLDNF} \neg\varphi \;\Rightarrow\; \Gamma \cup \Delta \vdash_{SLDNF} \neg\varphi$

  Esempio:

  $\{C, D \vee E \vee F \to B\} \vdash_{SLDNF} \neg B$

  $\{C, D \vee E \vee F \to B\} \cup \{D\} \nvdash_{SLDNF} \neg B$

  Aggiungendo informazione, il risultato della NF può cambiare

Altri metodi di calcolo automatico

Forward Chaining

Per le regole e fatti come clausole di Horn, anche il *Modus Ponens* può bastare

- *Modus Ponens* Generalizzato (GMP)

  Una regola di inferenza

  $\alpha_1,\ \alpha_2,\ \ldots\ \alpha_n, \alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n \rightarrow \beta \vdash \beta$ (MP originale: $\alpha, \alpha \rightarrow \beta \vdash \beta$)

- Descrizione:

  Per stabilire se $\Gamma \vdash \varphi$

  (Γ e φ in forma di clausole di Horn, Γ regole e fatti, φ fatto)

  Si applica la regola GMP a Γ in modo esaustivo

  Si ottiene un insieme Σ di fatti, $\Sigma = \{\psi : \Gamma \vdash \psi, per\ GMP\}$

  L'algoritmo termina con successo se $\varphi \in \Sigma$

- Caratteristiche:

  Niente assiomi

  Una sola regola (GMP)

  Niente refutazione o goal

Derivazioni: *Forward Chaining*

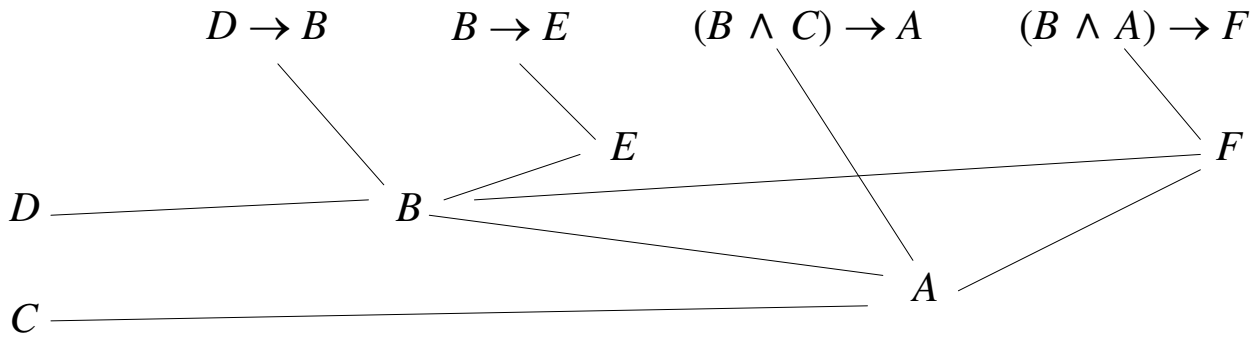
- Esempio

  $\{C\}, \{D\}, \{E, \neg B\}, \{B, \neg D\}, \{F, \neg B, \neg A\}, \{A, \neg B, \neg C\} \vdash \{A\}$

  Riscrivendo il tutto in termini di $\wedge$ e $\rightarrow$:

  $C,\ D,\ B \rightarrow E,\ D \rightarrow B, (B \wedge A) \rightarrow F, (B \wedge C) \rightarrow A \vdash A$

  Applicando il metodo del forward chaining:



Metodo del *Forward Chaining*

- Il metodo del forward chaining in L_P

  Si applica alle clausole di Horn (clausole definite)

  Termina sempre

  E` corretto (per le clausole di Horn): $\Gamma \vdash \varphi \Rightarrow \Gamma \models \varphi$

  E` completo (per le clausole di Horn): $\Gamma \vdash \varphi \Leftrightarrow \Gamma \models \varphi$

- Complessità computazionale:

  Ha complessità polinomiale (rispetto al numero di lettere ed alla dimensione delle fbf)

- Limitazioni

  Non tutti i problemi sono traducibili in clausole di Horn

Backward e Forward Chaining

I metodi della risoluzione SLD e del *forward chaining* a confronto

Esempio:

$\{C\}, \{D\}, \{E, \neg B\}, \{B, \neg D\}, \{F, \neg B, \neg A\}, \{A, \neg B, \neg C\} \vdash \{A\}$

equivale a $C,\ D,\ B \to E,\ D \to B, (B \wedge A) \to F, (B \wedge C) \to A \vdash A$

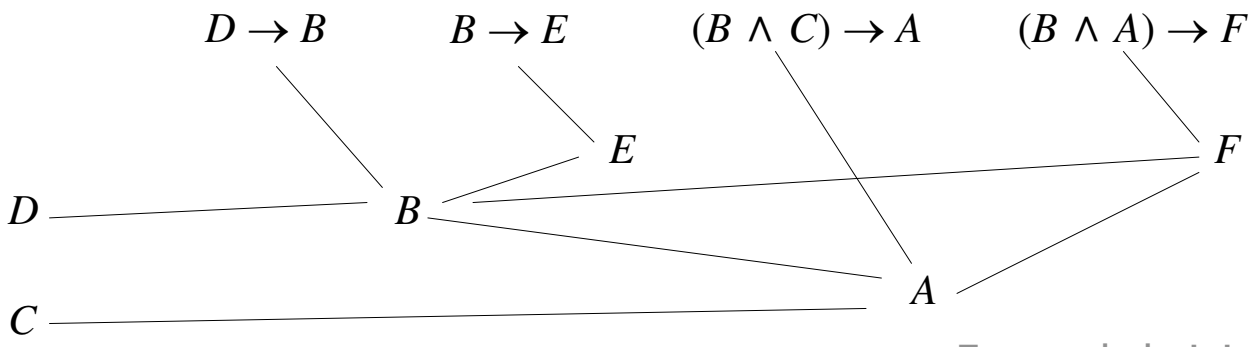
- Caratteristiche

Risoluzione SLD (*backward chaining*)

Si parte dal goal, si applicano regole e fatti

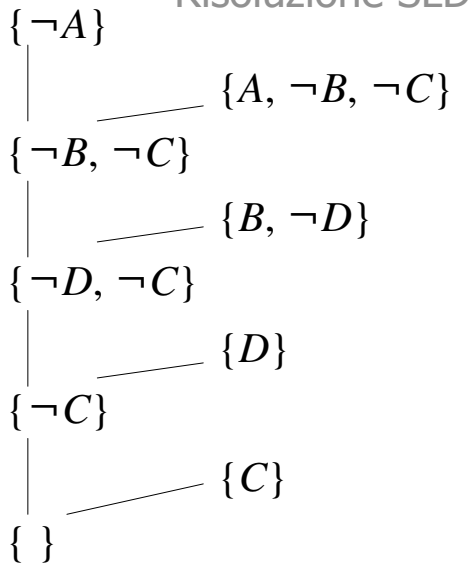
Forward Chaining

Si parte dai fatti e si applicano le regole



Forward chaining

Risoluzione SLD



Metodo a *tableau*

Anche 'tableau semantici' o a 'metodo a *tableaux*'

E` un metodo per refutazione:

Per stabilire che $\Gamma \vdash \varphi$ si tenta di mostrare che $\Gamma \cup \{\neg\varphi\}$ è inconsistente

- Descrizione:

L'insieme $\Gamma \cup \{\neg\varphi\}$ (fbf in forma qualsiasi) diventa il *tableau* radice di un albero

Ciascun nodo (tableau) dell'albero rappresenta una **congiunzione** di fbf

Si espande il nodo iniziale applicando diverse regole di inferenza (9 per L_P)

Alcune regole espandono, altre provocano una diramazione

Ciascuna diramazione rappresenta una **disgiunzione** di tableau

Un ramo si chiude se il tableau contiene una contraddizione $\{\varphi, \neg\varphi\}$

$\Gamma \cup \{\neg\varphi\}$ è inconsistente (o insoddisfacibile) se tutti i rami sono **chiusi**

$\Gamma \cup \{\neg\varphi\}$ non è inconsistente se almeno un ramo che non si chiude

Regole di inferenza per i *tableau*

- Regole alfa (o di espansione)

(a1) $\neg(\neg\varphi)$ / φ

(a2) $\varphi \wedge \psi$ / φ, ψ

(a3) $\neg(\varphi \vee \psi)$ / $\neg\varphi, \neg\psi$

(a4) $\neg(\varphi \rightarrow \psi)$ / $\varphi, \neg\psi$

- Regole beta (o di biforcazione)

(b1) $\varphi \vee \psi$ → φ | ψ

(b2) $\neg(\varphi \wedge \psi)$ → $\neg\varphi$ | $\neg\psi$

(b3) $\varphi \rightarrow \psi$ → $\neg\varphi$ | ψ

(b4) $\varphi \leftrightarrow \psi$ → $\neg\varphi, \neg\psi$ | φ, ψ

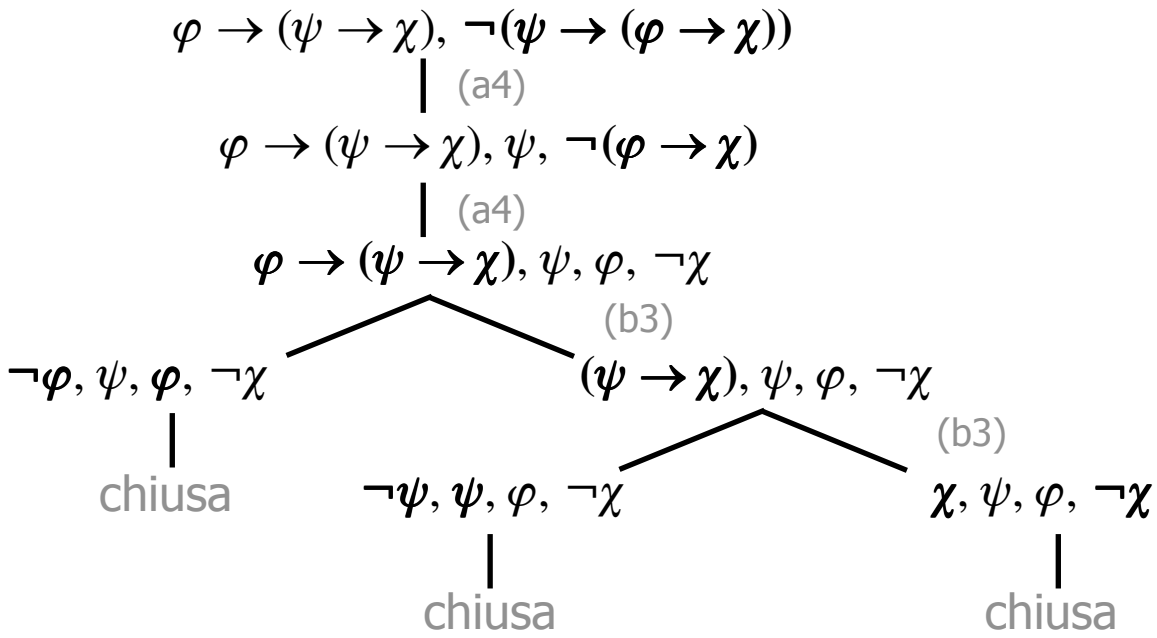
(b5) $\neg(\varphi \leftrightarrow \psi)$ → $\neg\varphi, \psi$ | $\varphi, \neg\psi$

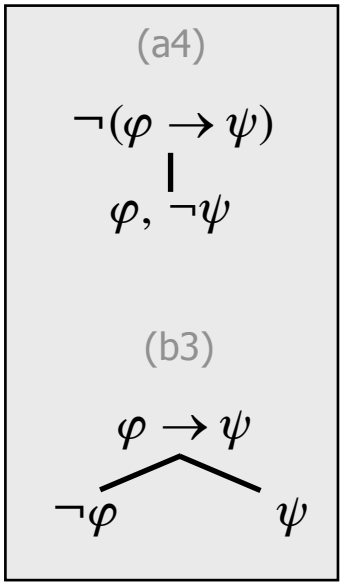
Derivazione: metodo a *tableau*

- Obiettivo

$$\varphi \to (\psi \to \chi) \vdash \psi \to (\varphi \to \chi)$$

- Espansione





Algoritmo del metodo a *tableau*

- Procedura:

  Dato il problema $\Gamma \vdash \varphi$

  Si assume $\Gamma \cup \{\neg\varphi\}$ come nodo *(tableau)* iniziale

  Su ciascun ramo, in modalità *depth-first*

  Se il nodo contiene solo letterali,

  se il nodo contiene una contraddizione, chiudere il ramo
  altrimenti terminare la procedura [fallimento]

  Se il nodo contiene formule composite:

  a) applicare le regole alfa
  b) applicare le regole beta

  Notare: le regole alfa vengono applicate prima delle regole beta

Caratteristiche del metodo a *tableau*

- Il metodo a tableau in L_P
  - Termina sempre
  - E` corretto: $\Gamma \vdash \varphi \Rightarrow \Gamma \models \varphi$
  - E` completo: $\Gamma \vdash \varphi \Leftrightarrow \Gamma \models \varphi$
- Complessità computazionale:
  - Ha complessità $O(2^n)$ dove n è il numero delle *lettere*
    - L'efficienza del metodo dipende in generale dalla struttura delle fbf, più che dal numero di letterali
    - In molti casi pratici, il metodo a tableau è estremamente efficace
- Leggibilità
  - Si applica a fbf in forma qualsiasi
- Generalità
  - Modificando opportunamente le regole di inferenza si possono costruire versioni del metodo anche per altre logiche (anche non classiche, come le logiche modali e multivalenti)