Intelligenza Artificiale I

Logica formale

Automazione del calcolo logico

Marco Piastra

Problemi e decidibilità (automatica)

- Problema

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

  $K : \mathrm{I} \rightarrow \mathrm{S}$ (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

  Esempio: Fox, Goat & Cabbage, *istanze*: configurazioni iniziali, *soluzioni*: le strategie valide

  Problemi di ottimizzazione

  Esiste una funzione costo definita su S: $c : \mathrm{S} \rightarrow \boldsymbol{R}$ ($\boldsymbol{R}$ 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

  E' 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), 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 $< d$?

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?

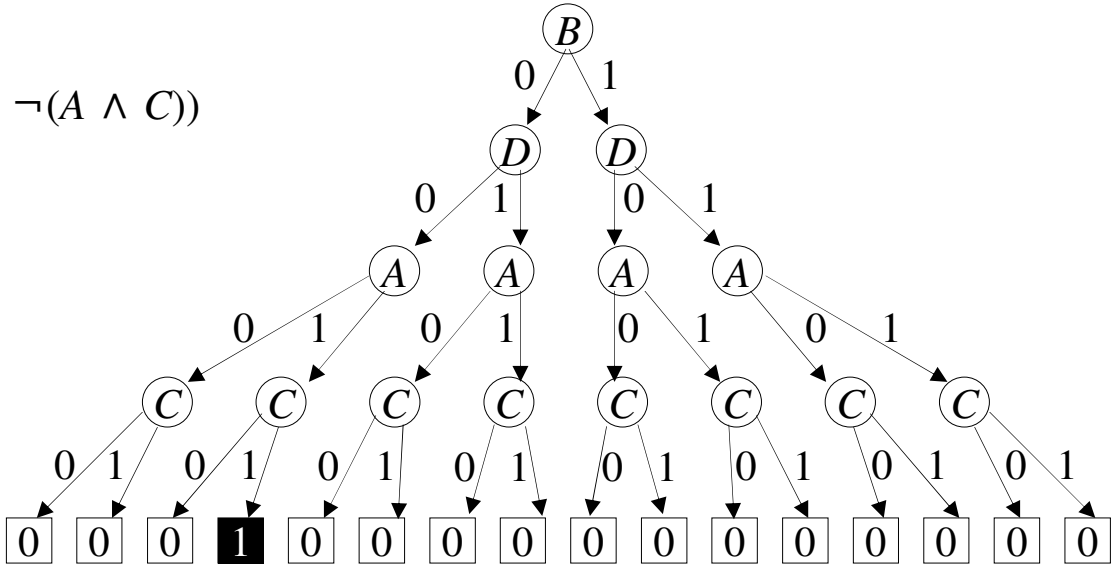
  L'insieme degli input è formato da tutte le possibili fbf φ

  In letteratura, il problema è denominato SAT

  E' decidibile: basta provare tutte le possibili interpretazioni

Esempio:

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



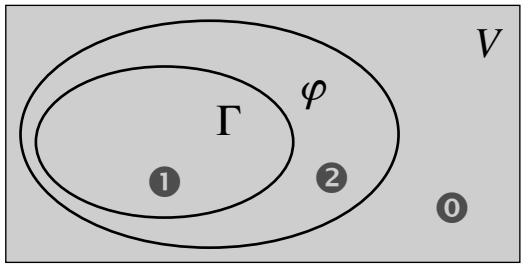
La procedura ha complessità $O(2^n)$, pari al numero di assegnazioni da provare

Conseguenza logica come soddisfacibilità

- 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)

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

$v(\{\neg\varphi\}) = ❼$

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

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

Morale:

Il problema $\Gamma \models \varphi$ può essere trasformato nel problema di stabilire se $\bigwedge(\Gamma \cup \{\neg\varphi\})$ (vale a dire una fbf) è soddisfacibile

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 procedure effettiva con complessità di *tempo* $O(\mathrm{P}(n))$

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

- Classe NP

  Tutti i problemi decidibili per cui esiste:

  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 $\mathrm{P} \subseteq \mathrm{NP}$)
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 (NP-complete $\subseteq$ 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

Automazione del calcolo simbolico

- Il problema $\Gamma \vdash \varphi$ è decidibile?

  Esiste una procedura effettiva per la derivazione automatica?

  (Intesa come calcolo puramente simbolico)

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

  Perché:

  Volendo costruire una dimostrazione di $\Gamma \vdash \varphi$

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

  Possiamo applicare la regola MP non appena possibile (OK)

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

  A rigor di teoria, tali istanze sono infinite

  Morale: *il problema sono gli assiomi, infiniti*

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 fbf logicamente equivalente

  Due fbf sono logicamente equivalenti quando sono soddisfatte dalle stesse interpretazioni

  $\varphi \equiv \psi$

  Le fbf in forma normale, in generale, hanno una struttura particolare

- **Forma normale congiuntiva** (FNC)

  Una fbf in cui $\wedge$ appare solo al livello più esterno

  $\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n$ $\quad$ α_i sono fbf in cui compare solo $\vee$ e $\neg$ (solo davanti ai simboli proposizionali)

  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

  $\beta_1 \vee \beta_2 \vee \ldots \vee \beta_n$ $\quad$ β_i sono fbf in cui compare solo $\wedge$ e $\neg$ (solo davanti ai simboli proposizionali)

  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":

$$\neg(\varphi \wedge \psi) \equiv (\neg\varphi \vee \neg\psi)$$
$$\neg(\varphi \vee \psi) \equiv (\neg\varphi \wedge \neg\psi)$$

Si distribuisce $\vee$

proprietà distributiva di $\vee$:

$$((\varphi \wedge \psi) \vee \chi) \equiv ((\varphi \vee \chi) \wedge (\psi \vee \chi))$$

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")

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

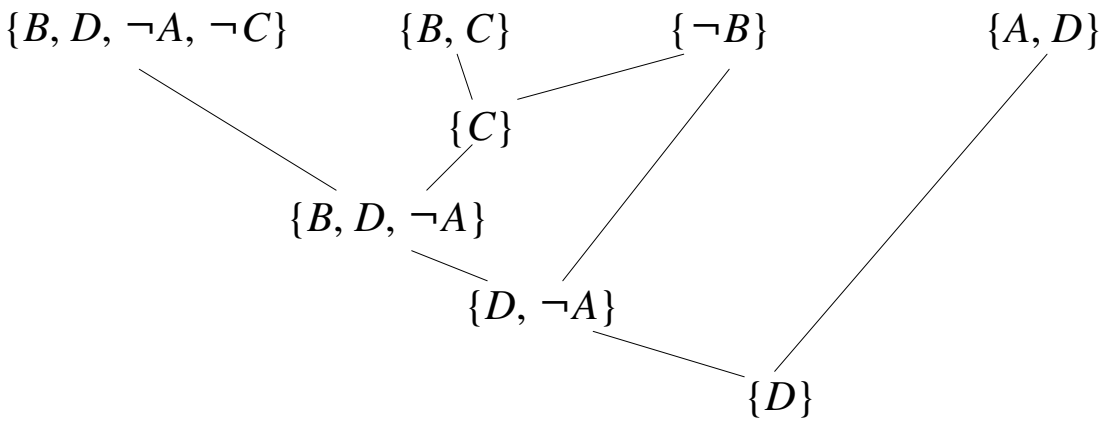
  Riscrivendo le formule dell'ipotesi in FNC:

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

  In FC:

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

  Applicando la regola di risoluzione (in forma binaria):



Derivazioni: risoluzione

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

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

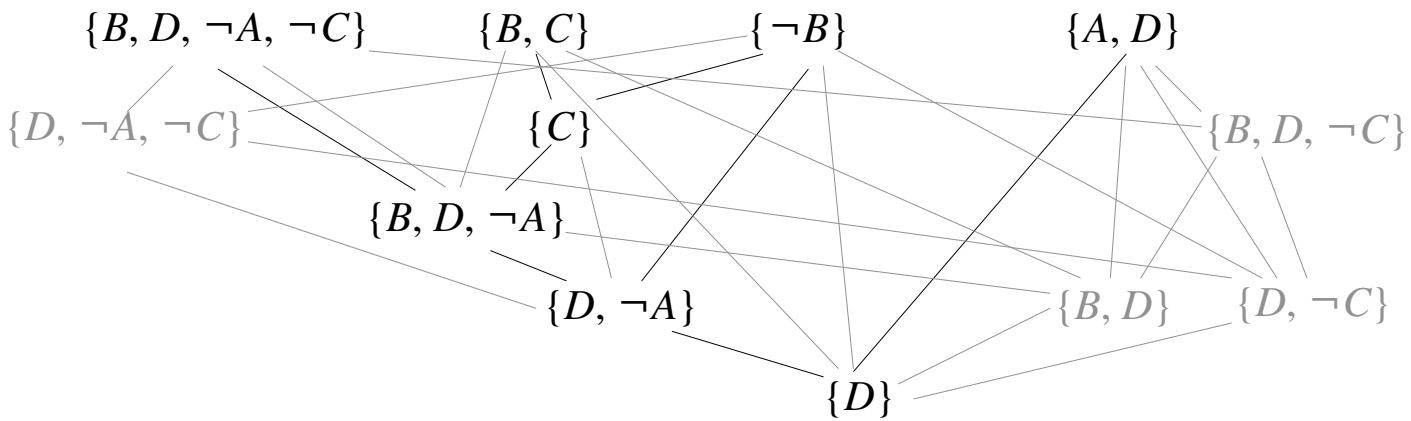
  Riscrivendo le formule dell'ipotesi in FNC:

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

  In FC:

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

  Applicando *esaustivamente* la regola di risoluzione (in forma binaria):



I *risolventi* ottenuti in modi diversi vengono considerati una volta sola

Risoluzione: ipotesi di metodo

- Descrizione:
  - Problema: $\Gamma \vdash \varphi$?
  - 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, $\Sigma = \{\psi : \Gamma \vdash \psi, \textit{per risoluzione}\}$
    - Il metodo 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!**
      - Esempio in negativo:

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

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

Risoluzione per refutazione

- Descrizione:

  Problema: $\Gamma \vdash \varphi$?

  Si trasforma il problema in $\Gamma \cup \{\neg\varphi\}$ (refutazione della tesi φ)

  Se $\Gamma \vdash \varphi$, $\Gamma \cup \{\neg\varphi\}$ è incoerente, quindi è possibile derivare una contraddizione

  Si traduce $\Gamma \cup \{\neg\varphi\}$ in FNC e quindi in FC

  Si applica esaustivamente la regola di risoluzione, in forma binaria

  L'algoritmo termina con successo quando viene derivata una **clausola vuota** { }

  Diversamente, l'algoritmo termina quando la regola non è più applicabile (fallimento)

- Vantaggi:

  Niente assiomi

  Una sola operazione (applicazione della regola di risoluzione)

  Procedura effettiva (algoritmo)

Derivazioni: risoluzione per refutazione

- Il contro-esempio visto in precedenza

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

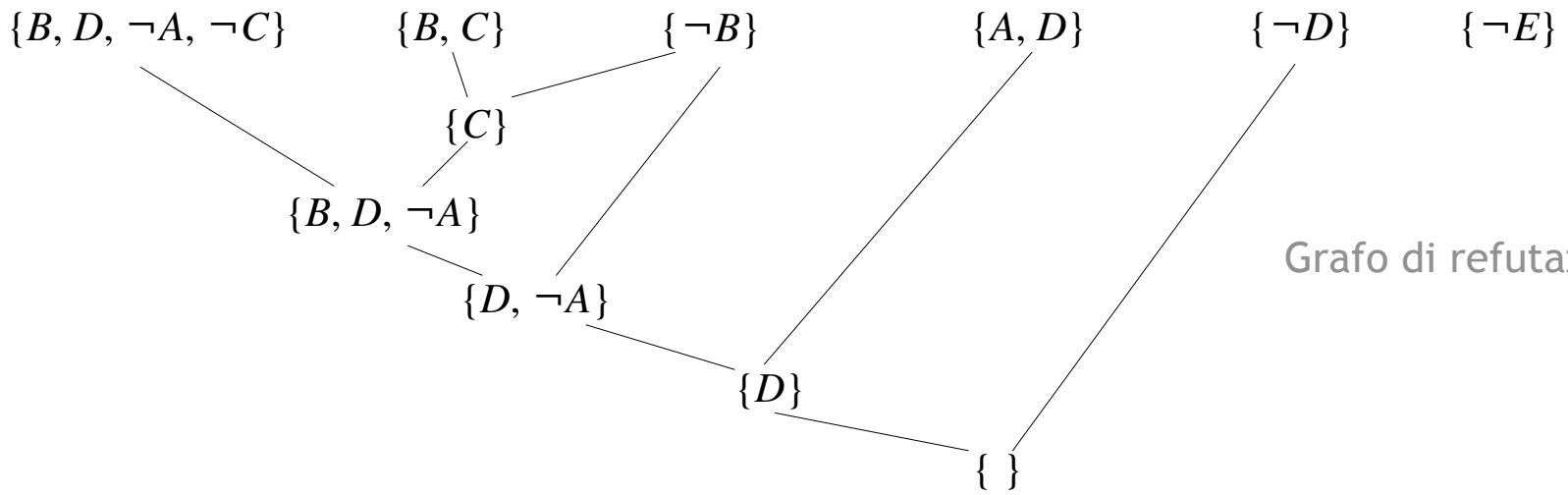
  Refutando la tesi e riscrivendo il tutto in FNC:

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

  In FC:

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

  Applicando la regola di risoluzione:



Grafo di refutazione

Risoluzione per refutazione

- Il metodo della risoluzione per refutazione in L_P

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

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

  Si intende: se $\Gamma \models \varphi$ allora esiste un grafo di refutazione

- Algoritmo di risoluzione per refutazione in L_P

  Il metodo della risoluzione inteso come procedura effettiva

  E' una procedura di decisione per $\Gamma \vdash \varphi$

  Se esiste un grafo di refutazione, l'algoritmo lo trova

  Ha complessità di tempo $O(2^n)$

  Dove n è il numero dei simboli proposizionali che compaiono in $\Gamma \cup \{\neg\varphi\}$

Minore potenza espressiva, maggiori possibilità di calcolo: clausole di Horn

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

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)

- 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 $\rightarrow$)

Risoluzione SLD

Linear resolution with Selection function for Definite clauses

Metodo particolare di risoluzione per refutazione (per le clausole di Horn)

Si applica ad un set di clausole definite + un *goal* (una lista di *subgoal* atomici)

- Descrizione

  Applicazione della regola di risoluzione (in forma binaria) con una specifica *regola di calcolo* per la scelta del *subgoal* atomico da risolvere e della regola da applicare

  Più precisamente, a partire da un insieme di *goal* correnti:

  1) Seleziona un *subgoal* atomico (*selection function*)
  2) Seleziona una clausola definita da applicare al *subgoal*
  3) Applica la regola di risoluzione, sostituendo al *subgoal* risolto i *subgoal* risolventi della clausola applicata

  Si procede finché si ottiene $\{\ \}$ oppure il passo 2) fallisce.

  Nota:

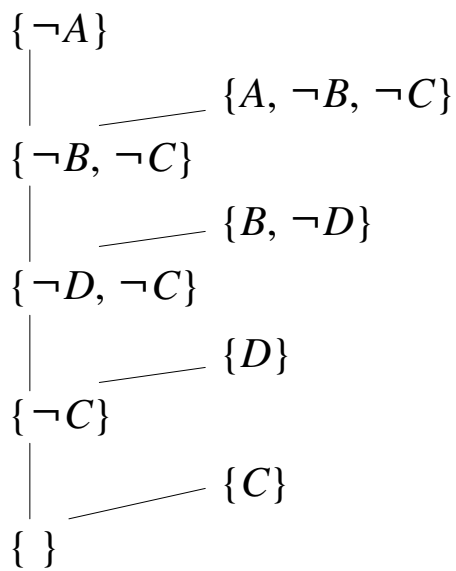
  La scelta del *subgoal* atomico è vincolante: se non c'è una regola, il passo 2) fallisce e la procedura termina

  Esempio:

  Regola di calcolo: ***leftmost subgoal first***

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

  Goal: $\{\neg A\}$



Alberi SLD

Derivazioni SLD

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

In questo caso, c'è al massimo un modo per risolvere ciascun *subgoal*

In generale non è così

Progressione *lineare* →

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

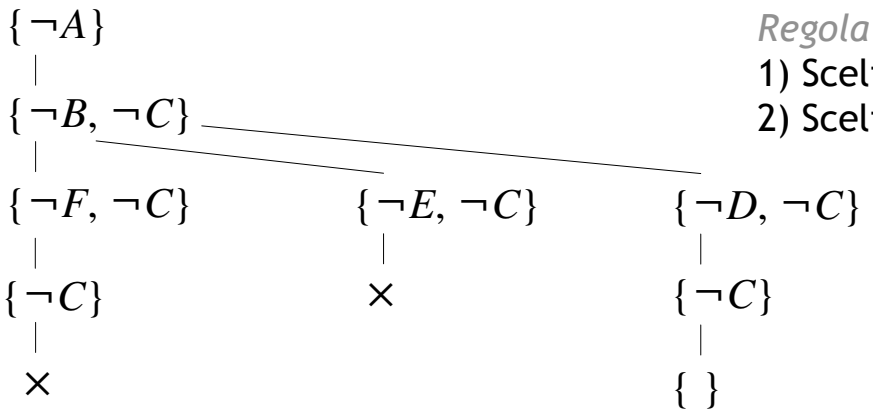
- Alberi SLD (=tutte le possibili derivazioni SLD)

Una diramazione per ciascun modo di risolvere un *subgoal*

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

Sono state aggiunte nuove regole: esistono ora diverse possibilità

Albero SLD



Regola di calcolo:
1) Scelta del *subgoal*
2) Scelta della clausola da applicare

Regola di calcolo *fair*

La regola di calcolo deve considerare tutte le possibilità

Altrimenti non è completa

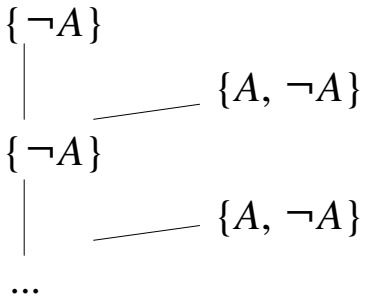
Controesempio: *leftmost subgoal first, leftmost rule first*

- Loop checking

  Esempio 1:

  Regola: *leftmost subgoal first, leftmost rule first*
  Clausole definite: $\{A, \neg A\}$
  Goal: $\{\neg A\}$

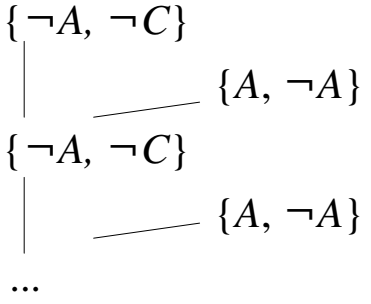
  L'esecuzione produce un loop infinito
  (facilmente identificabile: il *goal* è sempre lo stesso)



  Esempio 2:

  Regola: *leftmost subgoal first, leftmost rule first*
  Clausole definite: $\{A, \neg A\}, \{A, \neg C\}, \{C\}$
  Goal: $\{\neg A, \neg C\}$

  Notare che $\{A, \neg A\}, \{A, \neg C\}, \{C\} \models \{A\}, \{C\}$
  L'esecuzione produce un loop infinito
  e quindi il metodo non ha successo
  (anche se il *loop* rimane facilmente identificabile)



Regola di calcolo *fair*

Una regola di calcolo è *fair* se garantisce che qualsiasi risoluzione possibile nell'albero SLD viene effettivamente applicata entro un numero finito di passi

- In altri termini, non vi sono subgoal "dimenticati" o possibili risoluzioni trascurate

- Esempio di regola *fair*
  - Scelta del *subgoal*: ***least recent subgoal first***
  - Scelta della clausola definita: ***least recent instantiation first***

- Concetto di *istanziazione* (*instantiation*):
  - Un'istanziazione è una combinazione <*subgoal*, *clausola definita*> cui *si potrebbe* applicare la regola di risoluzione

Metodo della risoluzione SLD

- Il metodo della risoluzione SLD in L_P

  Si applica solo alle clausole di Horn (clausole definite + *goal*)

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

  Con una regola di calcolo *fair*:

  Termina sempre

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

- 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)

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*

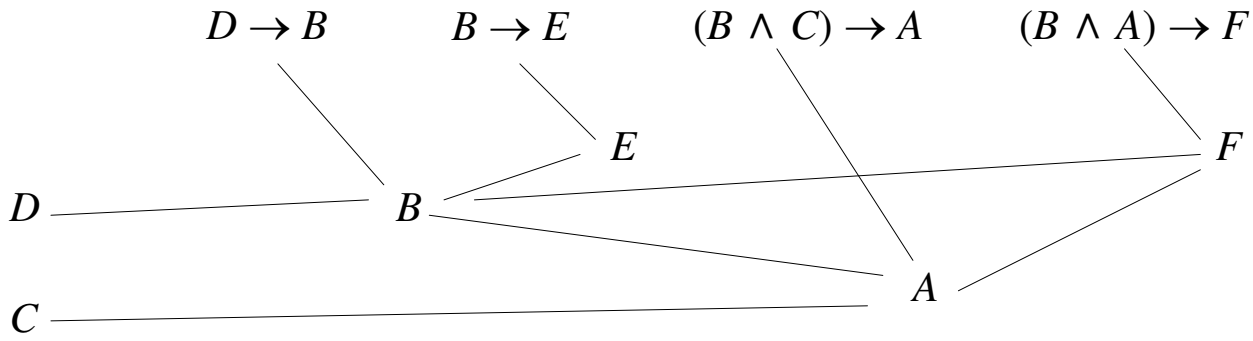
- Esempio

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

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

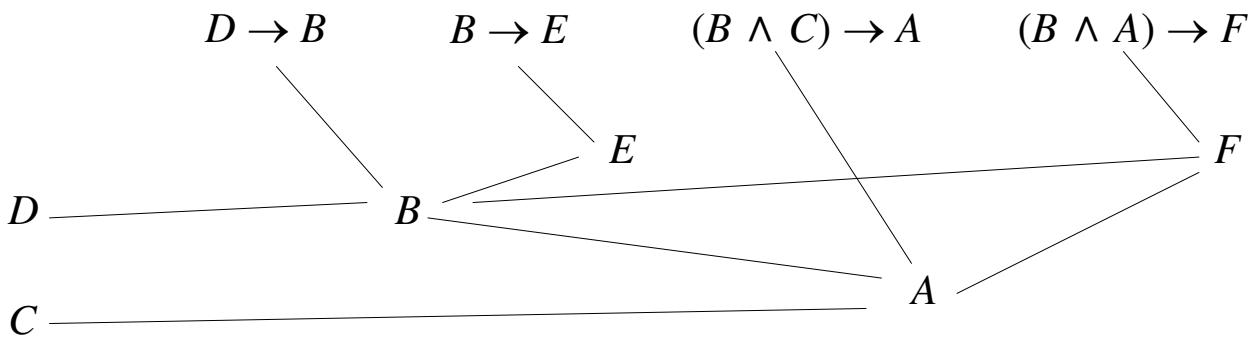
- Caratteristiche

  Risoluzione SLD (*backward chaining*)

  Si parte dal goal, si applicano regole e fatti

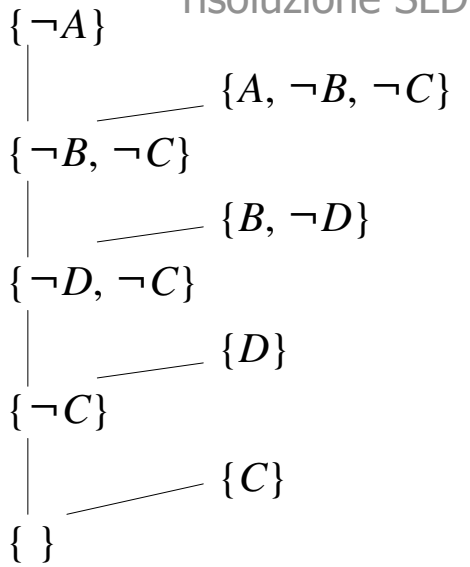
  Forward Chaining

  Si parte dai fatti e si applicano le regole



Forward chaining

Refutazione e 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 derivazione 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

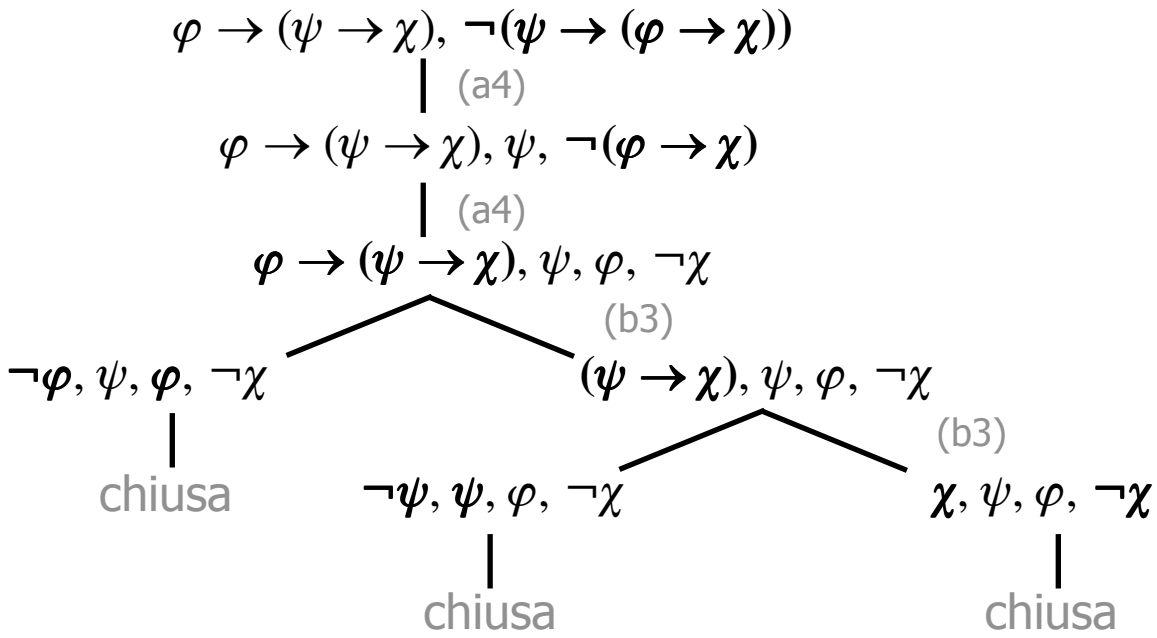
(b5) $\neg(\varphi \leftrightarrow \psi)$ → $\neg\varphi, \psi$ | $\varphi, \neg\psi$

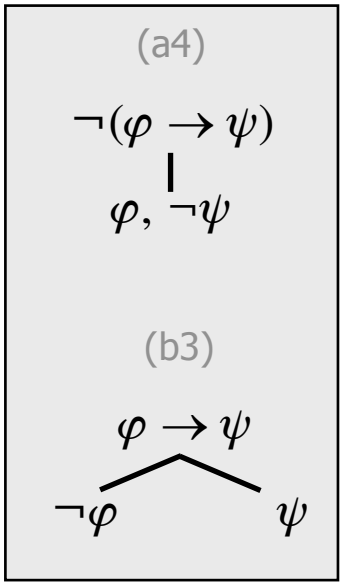
Derivazione: metodo a *tableau*

- 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)

Appendice:
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 $\neg$ 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"*

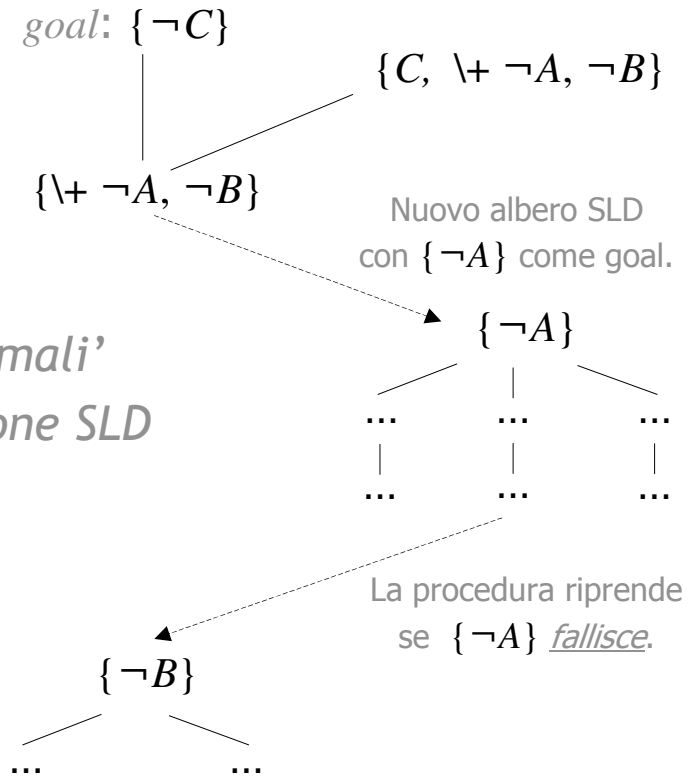
Risoluzione SLDNF

- 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 $\backslash+$ 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:

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

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

  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 $Q \rightarrow Q$

- 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:

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

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

  Aggiungendo informazione, il risultato della NF può cambiare