Rezoluție

Am văzut cum putem folosi raționamentul logic pentru a deschide noi căi într-o bază de cunoștințe (prin implicație logică). Raționamentul a fost făcut de mână, relativ informal.

Dată pentru o bază de cunoștințe dată KB și o propoziție α, dorim o procedură care să determine dacă $KB \models \alpha$ sau nu. De exemplu, dacă $\beta[x_1, \ldots, x_n]$ este o formulă care are variabile libere printre x_i, vrem o procedură care să găsească termenii t_i dacă există, astfel încât $KB \models \beta[t_1, \ldots, t_n]$.

Dar nu există o procedură automată care să satisfacă complet (în toate cazurile) această cerință.

Căutăm o procedură care să facă raționament deductiv într-o manieră cât mai <u>corectă</u> și <u>completă</u> și într-un limbaj cât mai expresiv de FOL.

[Raționamentul este <u>corect logic</u> dacă ori de câte ori produce α atunci este garantat că α este o consecință logică (exclude posibilitatea de a produce lucruri ce pot fi adevărate într-o interpretare dorită, dar să nu fie deduse logic).
Raționamentul este <u>logic complet</u> dacă garantează producerea lui α ori de câte ori α este o consecință logică (exclude posibilitatea de a rata unele consecințe de exemplu, când acestea ar fi dificil de determinat)]

Dacă KB este o mulțime finită de propoziții $\{\alpha_1, \ldots, \alpha_n\}$, raționamentul deductiv poate fi formulat în câteva feluri echivalente:

1. $KB \models \alpha$ dnd
2. $\models [(\alpha_1 \wedge \ldots \wedge \alpha_n) \supset \alpha]$ dnd
3. $KB \cup \{\neg\alpha\}$ nu poate fi satisfăcută dnd
4. $KB \cup \{\neg\alpha\} \models \neg TRUE$

unde TRUE este orice propoziție validă (de exemplu $\forall x.\ x=x$).

3 ⟺ 4
Dacă ∃ există $\mathcal{I}$ a.î. $\mathcal{I} \models KB \cup \{\neg\alpha\}$ atunci
$KB \cup \{\neg\alpha\} \models TRUE$ în $\mathcal{I}$ - contradicție cu $KB \cup \{\neg\alpha\} \models \neg TRUE$
deci $\nexists\, \mathcal{I}$ cu $\mathcal{I} \models KB \cup \{\neg\alpha\}$

Dacă avem o procedură pentru testarea validității unei propoziții sau pentru testarea satisfacerii unei propoziții sau pentru determinarea dacă $\neg TRUE$ este sau nu consecință logică atunci această procedură poate fi folosită pentru a găsi consecințele logice ale unei KB finite.

Procedura de <u>rezoluție</u> determină dacă anumite mulțimi de formule pot fi satisfăcute

<u>Cazul propozițional al rezoluției</u>

Logica propozițională - o formă restricționată pentru formule.
Orice propoziție α din logica propozițională poate fi transformată în α' o conjuncție de disjuncții de literali (adică atomi sau negațiile lor), astfel încât $\models (\alpha \equiv \alpha')$
α' este forma normală conjunctivă CNF
De exemplu $(p \vee q) \wedge (\neg r \vee p \vee \neg s) \wedge (\neg q \vee r)$
(folosim litere mici pentru simbolurile propoziționale)

Procedura de transformare a unei formule propoziționale în CNF:

1. se înlocuiesc $\supset$, $\equiv$ cu formulele pe care le reprezintă
2. se mută $\neg$ în interior și se operă înspre unul atom

$$\models (\neg\neg\alpha \equiv \alpha)$$
$$\models \neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$$
$$\models \neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$$

3. distribuirea $\wedge$ și $\vee$ folosind echivalențele

$$\models (\alpha \vee (\beta \wedge \gamma)) \equiv ((\beta \wedge \gamma) \vee \alpha) \equiv (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$$

4. restrângerea termenilor

$$\models (\alpha \vee \alpha) \equiv \alpha$$
$$\models (\alpha \wedge \alpha) \equiv \alpha$$

Obs Rezultatul este o formulă CNF echivalentă logic, care poate fi exponențial mai mare decât formula inițială

$((p \supset q) \supset r) \longrightarrow \neg(\neg p \vee q) \vee r$

$(p \wedge \neg q) \vee r$

$(p \vee r) \wedge (\neg q \vee r)$ CNF

Este convenabil să folosim o reprezentare prescurtată a CNF

clauză = mulțime finită de literali (interpretată ca disjuncție de literali)

formulă clauzală = mulțime finită de clauze (interpretată ca fiind conjuncție de clauze)

Notație: l literal atunci $\bar{l}$ este complementul lui

$\bar{l} = \neg l$ și $\neg\bar{l} = l$

{ mulțime de formule clauzale }

[mulțime de literali]

De exemplu, $[p, \neg q, r]$ reprezintă $(p \vee \neg q \vee r)$

$\{[p, \neg q, r], [q]\}$ reprezintă $(p \vee \neg q \vee r) \wedge q$

Clauză cu un singur atom sau clauză unitate

Obs $\{\} \neq \{[]\}$

$\{\}$ - formulă clauzală vidă = conjuncție de nicio constrângere reprezintă TRUE

$[]$ - disjuncție de nicio posibilitate - reprezintă $\neg$TRUE

deci $\{[]\}$ reprezintă $\neg$TRUE

Pentru a determina dacă $KB \models \alpha$ este suficient

1. să exprimăm propozițiile din KB și $\neg\alpha$ în CNF
2. să determinăm dacă mulțimea de clauze rezultată poate fi satisfăcută

Derivări de rezoluție

Regula de inferență numită rezoluție este următoarea:

Dată fiind clauza de forma $C_1 \cup \{l\}$, unde l este literal și o clauză de forma $C_2 \cup \{\bar{l}\}$, deducem clauza $C_1 \cup C_2$ (C_1 și C_2 pot fi vide)

Spunem că $C_1 \cup C_2$ este o rezolventă a celor două clauze de intrare în raport cu l.

De exemplu, din clauzele $[w, \underline{p}, q]$ și $[s, w, \neg \underline{p}]$ avem

$[w, q, s]$ rezolventă în raport cu p

$[p, q]$ și $[\neg p, \neg q]$ au două rezolvente:

$[q, \neg q]$ în raport cu p

și $[p, \neg p]$ în raport cu q.

Obs. Singurul mod de a obține $[\,]$ este din două clauze unitare complementare $[p]$ și $[\neg p]$

O derivare prin rezoluție a clauzei C dintr-un set de clauze S este o secvență de clauze $C_1, \ldots, C_n$, unde $C_n = C$ și fiecare C_i fie aparține lui S fie este o rezolventă a două clauze anterioare în derivare. Scriem $S \vdash C$ dacă există o derivare a lui C din S.

Derivările rezoluției sunt importante deoarece operează la nivel de simboluri pe mulțimi finite de literali și legătură directă cu interpretările logice la nivel de semnificație.

Obs Rezolventa este intotdeauna consecință logică a celor două clauze de intrare.

$$\{C_1 \cup \{p\}, C_2 \cup \{\neg p\}\} \models C_1 \cup C_2$$

Fie I o interpretare a.î. $I \models C_1 \cup \{p\}$ și $I \models C_2 \cup \{\neg p\}$

1. dacă $I \models p$ atunci $I \not\models \neg p$ } $\Rightarrow I \models C_2$ deci $I \models C_1 \cup C_2$
   dar $I \models C_2 \cup \{\neg p\}$ }
2. dacă $I \not\models p$ dar $I \models C_1 \cup \{p\} \Rightarrow I \models C_1$ deci $I \models C_1 \cup C_2$

Obs. Orice clauză derivabilă prin rezoluție din S este o consecință logică a lui S, adică dacă $S \vdash C$ atunci $S \models C$.

Demonstrație - prin inducție după lungimea derivării vrem ca pentru orice C_i avem $S \models C_i$

[$S \vdash C$ dacă $\exists C_1, \dots, C_n = C$ a.î. $C_i \in S$ sau C_i este rezolventa a două clauze anterioare din derivare

dacă $C_i \in S \Rightarrow S \models C_i$

dacă C_i rezolventa clauzelor C_j și $C_k \Rightarrow$

$\Rightarrow \{C_j, C_k\} \models C_i$

din ip de inducție $S \models C_j$, $S \models C_k$ $\Rightarrow S \models C_i$

Reciproca nu este adevărată - putem avea $S \models C$ fără ca $S \vdash C$.

De exemplu, fie $S = \{[\neg P]\}$ și $C = [\neg Q, Q]$

$S \models C$ dar $C \notin S$ și nu există rezolventă.

Obs. Derivările rezoluției nu sunt complete (adică nu garantează producerea lui α ori de câte ori $S \models \alpha$)

Dar rezoluția este atât corectă cât și completă când $C = [\,]$.

Adică $S \vdash [\,] \Leftrightarrow S \models [\,]$ (S nu poate fi satisfăcută)

Deci problema determinării dacă orice mulțime de clauze poate fi satisfăcută se reduce la căutarea unei derivări a clauzei vide.

Proceduri de deducere/implicație

Vrem să determinăm dacă $KB \models \alpha$ (echivalent cu $KB \cup \{\neg\alpha\}$ nu poate fi satisfăcută)

Fie S mulțimea de clauze obținută prin transformarea KB și $\{\neg\alpha\}$ în CNF.

Verificăm dacă S nu poate fi satisfăcută căutând o derivare a mulțimii vide.

Proceduri nedeterministe

Intrare S - o multime finită de clauze propoziționale

1. dacă $[] \in S$ return "nu poate fi satisfăcută"
2. altfel, verifică dacă există două clauze în S cărora se poate aplica rezoluția. Ieș rezolventa nu este în S - dacă nu, return "poate fi satisfăcută"
3. altfel, adaugă rezolventa în S și mergi la pasul 1.

Obs. Procedura se termină deoarece fiecare clauză adăugată este o rezolventă a clauzelor anterioare și conține două literali din multimea inițială S (sunt în număr finit) - la un moment dat nu se mai poate adăuga nimic nou.

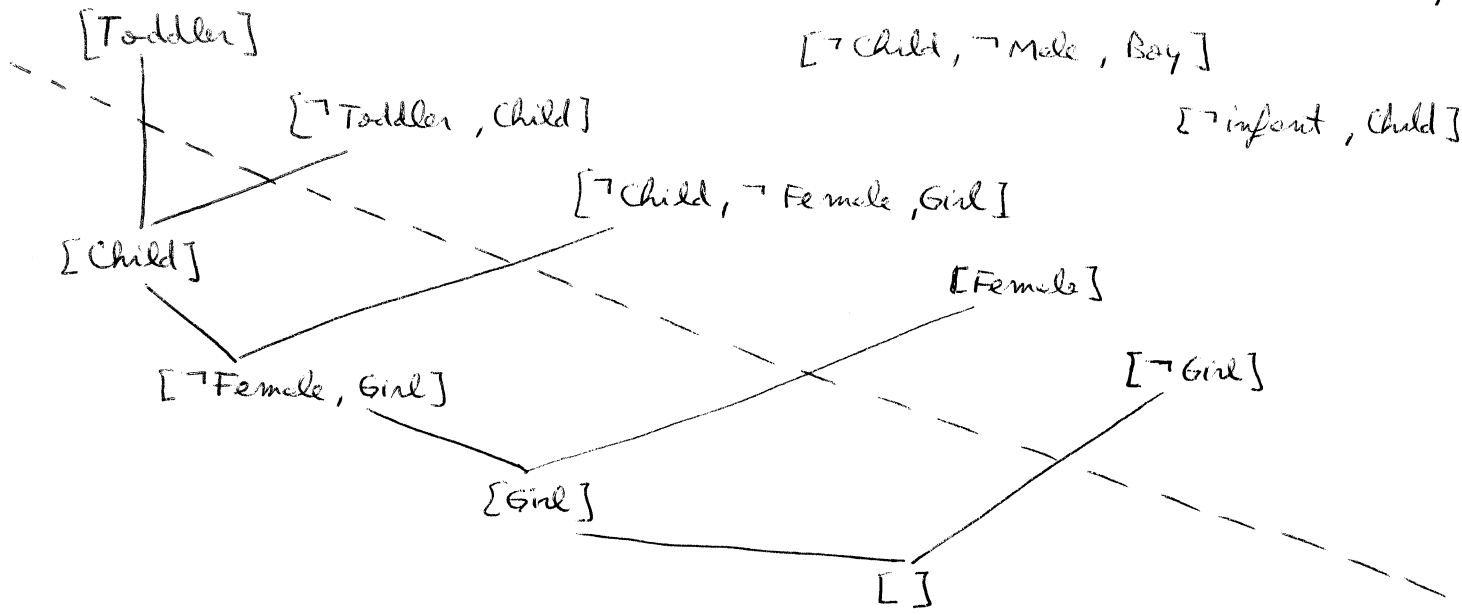
Putem face procedura deterministă - stabilim o strategie pentru alegerea perechilor de clauze -- putem alege prima pereche, sau pereche care produce cea mai scurtă rezolventă.
Dacă ne interesează să afișăm derivarea, cu fiecare rezolventă memorăm pointerii către clauzele de intrare.

Exemplul 1 Avem baza de cunoștințe:

$$KB \begin{cases} Toddler \\ Toddler \supset Child \\ Child \wedge Male \supset Boy \\ Infant \supset Child \\ Child \wedge Female \supset Girl \\ Female \end{cases}$$

Întrebare: Girl

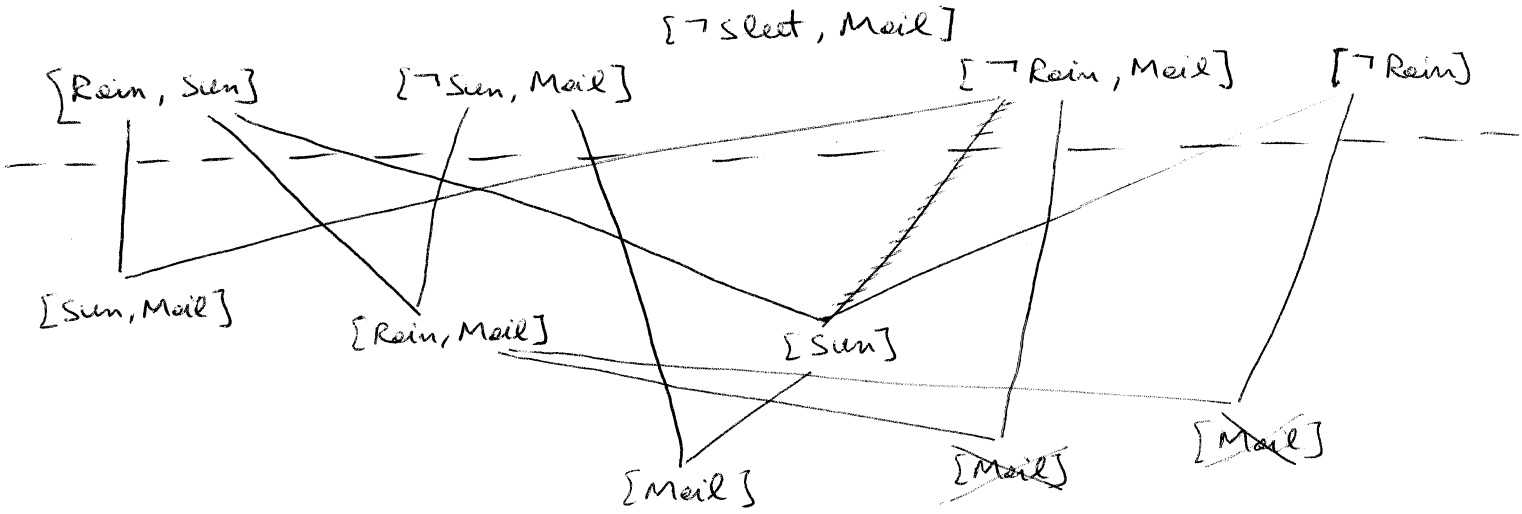
$KB \models Girl \iff KB \cup \{\neg Girl\}$ nu este satisfăcută



Exemplul 2

$$KB \left\{ \begin{array}{l} Sun \supset Mail \\ (Rain \vee Sleet) \supset Mail \\ Rain \vee Sun \end{array} \right.$$

$KB \not\models Rain$



Lucrul cu variabile și cuantificatori

Transformăm formulele în forme clauzale echivalente:

1. se înlocuiesc $\supset$, $\equiv$ cu formulele pe care le reprezintă
2. se mută $\neg$ în interior folosind în plus echivalențele

$$\models \neg \forall x.\alpha \equiv \exists x.\neg\alpha$$

$$\models \neg \exists x.\alpha \equiv \forall x.\neg\alpha$$

3. redenumirea variabilelor (dacă e necesară) ai variabilele din clauzele de intrare în rezoluție să fie distincte

4. eliminarea cuantificatorilor existentiali

5. mutarea cuantificatorilor in afara scopurilor conectorilor $\wedge$ si $\vee$

$$\models (\alpha \wedge \forall x . \beta) \equiv (\forall x . \beta \wedge \alpha) \equiv \forall x . (\alpha \wedge \beta)$$

$$\models (\alpha \vee \forall x . \beta) \equiv (\forall x . \beta \vee \alpha) \equiv \forall x (\alpha \vee \beta)$$

(cu conditia ca x sa nu fie liber in α)

6. distributivitatea $\wedge$ si $\vee$

$$\models (\alpha \vee (\beta \wedge \gamma)) \equiv ((\beta \wedge \gamma) \vee \alpha) \equiv (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$$

7. restrangerea termenilor

$$\models (\alpha \vee \alpha) \equiv \alpha$$

$$\models (\alpha \wedge \alpha) \equiv \alpha$$

Pentru inceput, consideram ca nu apare deloc cuantificatorul $\exists$.
Renuntam la scrierea cuantificatorilor $\forall$; variabilele sunt interpretate in sens universal.
Atomii pot fi de forma $P(t_1, \dots t_n)$ (nu consideram cazul $t_1 = t_2$)
De exemplu, formula clauzala

$$\{[P(x), \neg R(e, f(b, x))], [Q(x, y)]\}$$

reprezinta formula CNF

$$\forall x \forall y ([P(x) \vee \neg R(e, f(b, x))] \wedge Q(x, y))$$

<u>Def</u> Substitutie Θ este o multime finita de perechi $\{x_1/t_1, \dots, x_n/t_n\}$ unde x_i variabile, t_i termeni

Daca Θ substitutie, φ literal
$\varphi\Theta$ = literalul obtinut prin inlocuirea <u>simultana</u> a fiecarui x_i din φ cu t_i

De exemplu, $\Theta = \{x/a, y/g(x, b, z)\}$

$$\varphi = P(x, z, f(x, y))$$

$$\varphi\Theta = P(a, z, f(a, g(x, b, z)))$$

daca C este o clauza $C\Theta$ este clauza obtinuta prin substitutia pe fiecare literal.
φ este o instanta a lui φ' daca exista Θ a.i. $\varphi = \varphi'\Theta$.

Rezoluția de ordinul I

Deoarece clauzele cu variabile sunt cuantificate universal, dorim ca rezoluția să se aplice instanțelor acestor clauze.

De exemplu, $[P(x,e), \neg Q(x)]$ și $[\neg P(b,y), \neg R(b, f(y))]$

atunci avem $[P(b,e), \neg Q(b)]$ și $[\neg P(b,e), \neg R(b, f(e))]$

a căror rezolventă este $[\neg Q(b), \neg R(b, f(e))]$

Definim regula generală a rezoluției astfel:

Se dau clauzele $C_1 \cup \{\ell_1\}$ și $C_2 \cup \{\bar{\ell}_2\}$, cu ℓ_1 și ℓ_2 literali.

Redenumim variabilele din cele două clauze (dacă este necesar) astfel încât fiecare clauză să aibă variabile distincte.

Fie θ o substituție astfel încât $\ell_1\theta = \ell_2\theta$.

Atunci deducem clauza $(C_1 \cup C_2)\theta$.

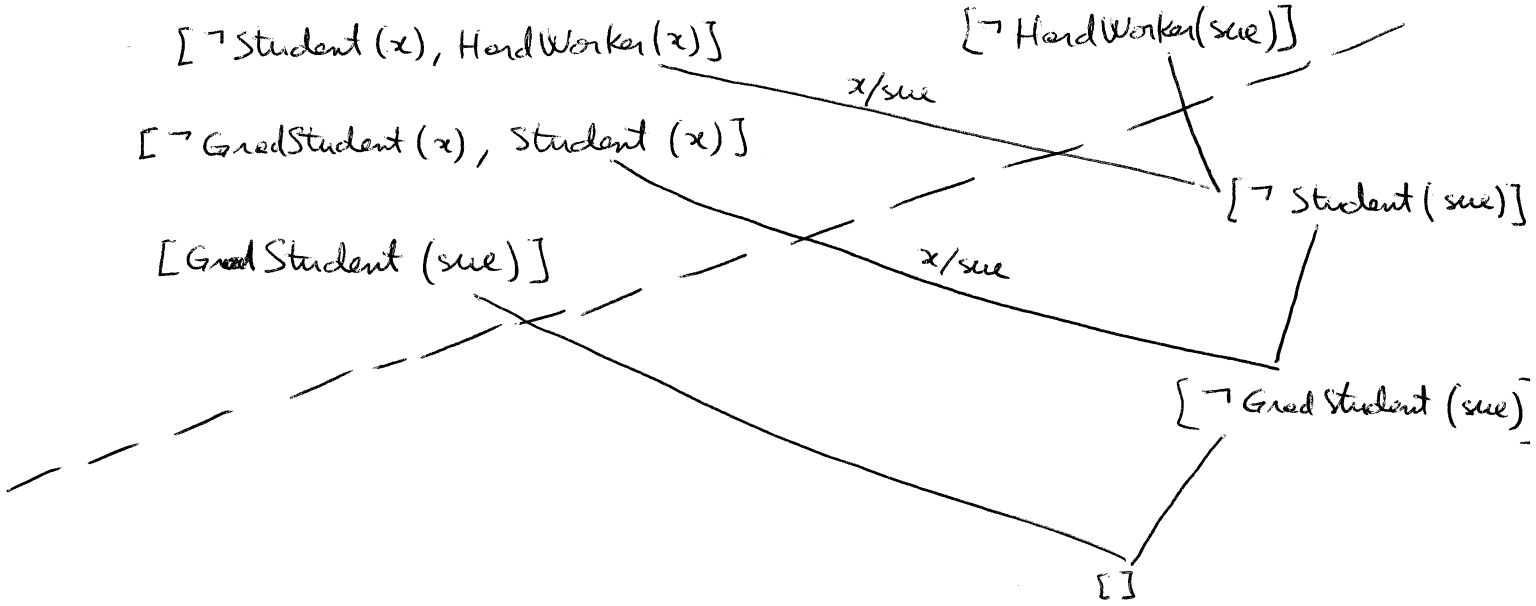
Spunem că θ unifică ℓ_1 și ℓ_2.

Avem că $S \vdash []$ dacă $S \models []$.

Exemplul 3

$$KB \begin{cases} \forall x.\ GradStudent(x) \supset Student(x) \\ \forall x.\ Student(x) \supset HardWorker(x) \\ GradStudent(sue) \end{cases}$$

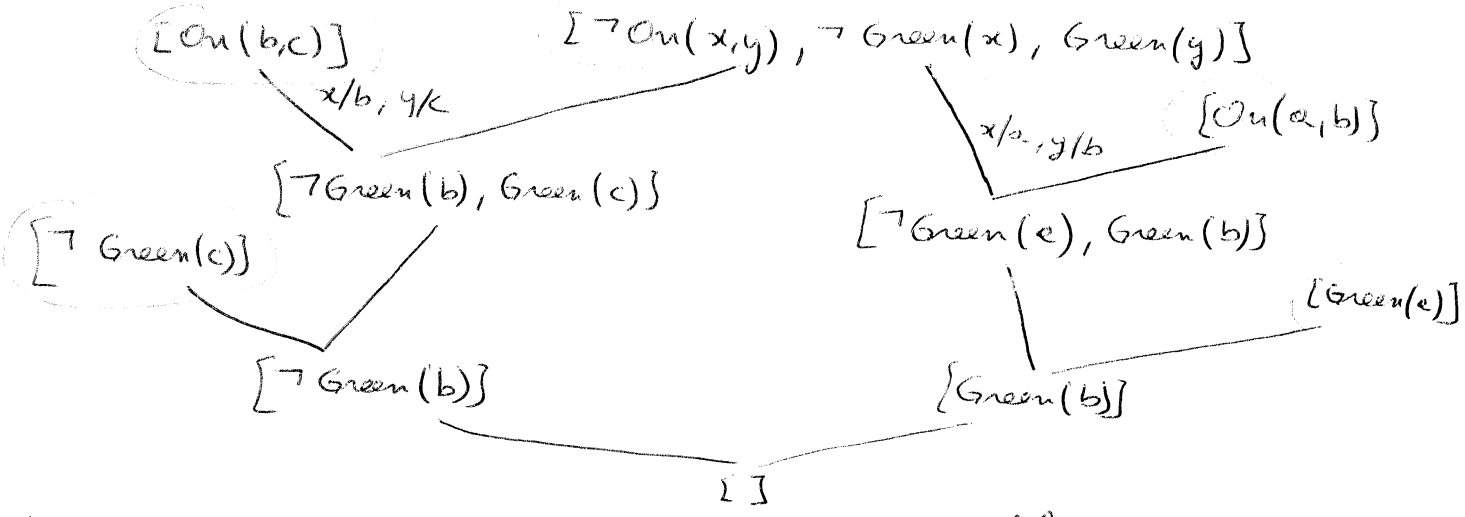
$KB \models HardWorker(sue)$



Exemplul 4 Problema celor 3 blocuri

KB: $On(a,b)$, $On(b,c)$, $Green(a)$, $\neg Green(c)$

Intrebare: $\exists x \exists y . On(x,y) \wedge Green(x) \wedge \neg Green(y)$

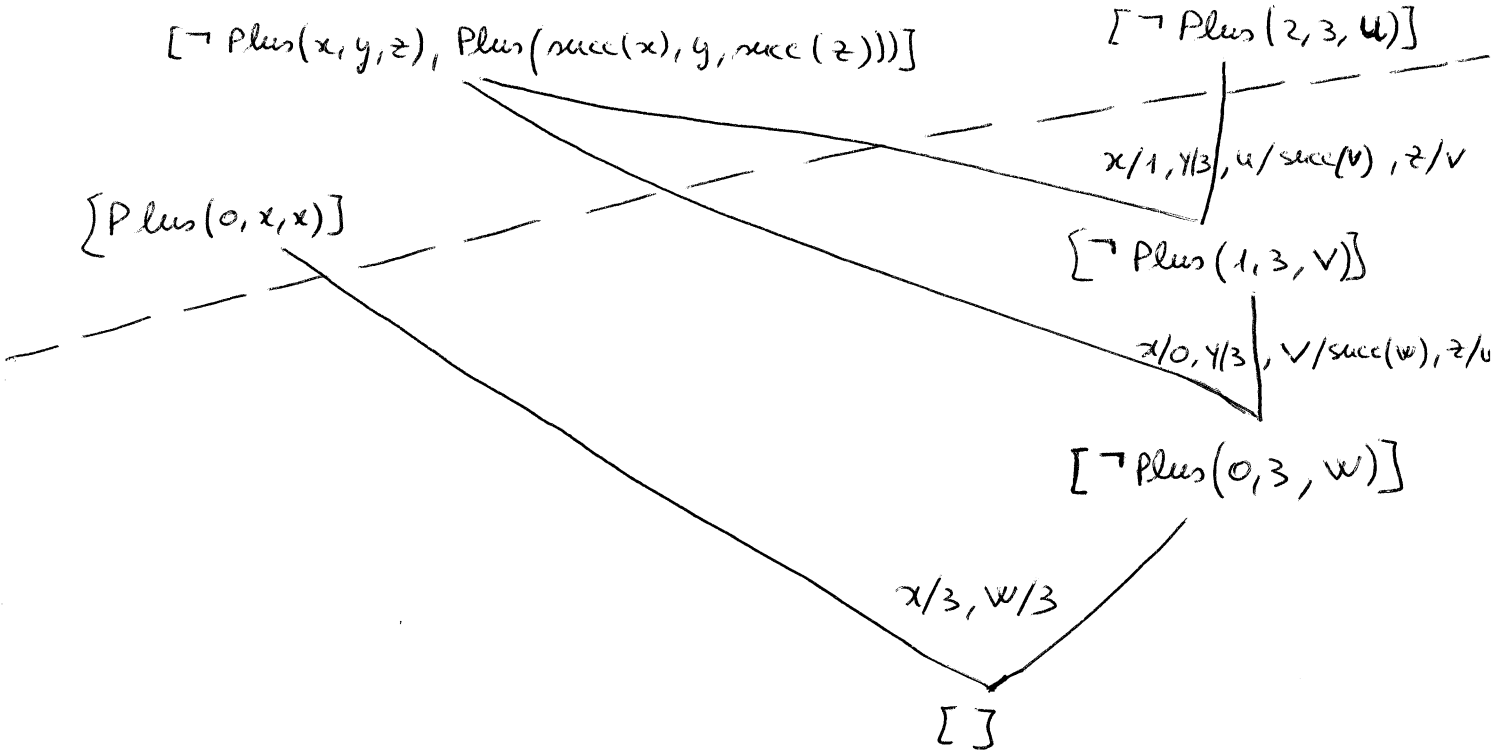


Exemplul 5 - Necesitatea redenumirii variabilelor

KB $\begin{cases} \forall x . Plus(zero, x, x) \\ \forall x \forall y \forall z . Plus(x,y,z) \supset Plus(succ(x), y, succ(z)) \end{cases}$

Intrebare $\exists u . Plus(2,3,u)$

$\begin{cases} Plus(x,y,z) - \text{reprezintă } x+y=z \\ succ(succ(succ(zero))) \text{ reprezintă } 3 \end{cases}$



Putem găsi valoarea lui u:

u legat de succ(v); v legat de succ(w); w legat de 3

adică $u = 5$

Extragerea răspunsului

Deseori este posibil să obținem răspunsurile la întrebări dacă ne uităm la legările variabilelor din derivarea unei clauze existențiale (Exemplul 5). Dar în FOL pot apărea situații mai complicate.

În Exemplul 4 știm că există un bloc ce satisface o condiție, dar nu știm ce bloc. Adică avem $KB \models \exists x.\, P(x)$ fără să implice $P(t)$ pentru un anume t.

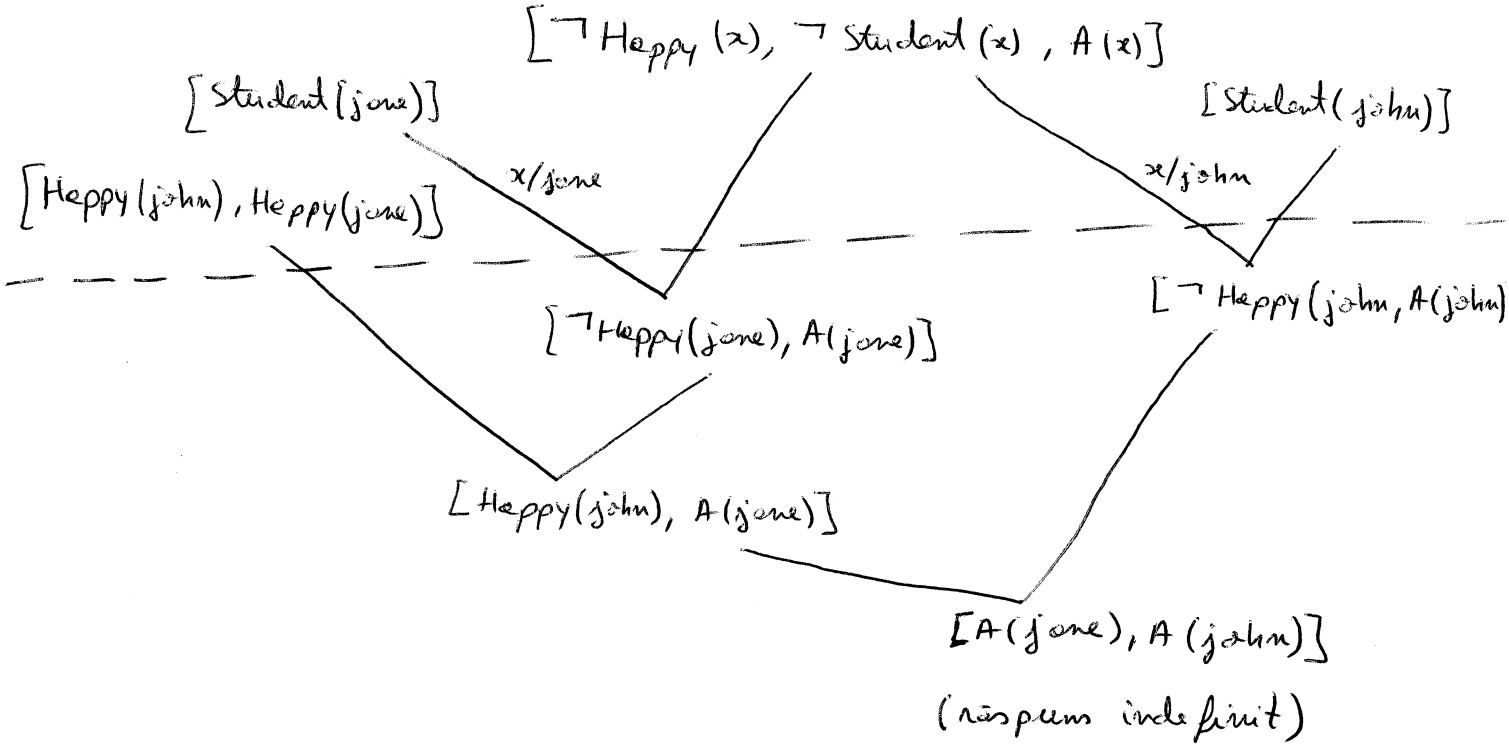
Idee: înlocuirea întrebării $\exists x.\, P(x)$, unde x este variabila ce ne interesează, cu $\exists x.\, P(x) \wedge \neg A(x)$, unde A este un predicat nou ce nu apare în altă parte. A s.n. predicat răspuns.

Deoarece A nu apare în altă parte, prin derivare nu putem ajunge la clauza vidă. Derivarea se termină când producem o clauză ce conține doar predicatul A.

Exemplul 6

KB
- Student (john)
- Student (jane)
- Happy (john) $\vee$ Happy (jane)

întrebare: $\exists x.\, Student(x) \wedge Happy(x)$



Obs. Putem obține răspunsuri ce conțin variabile.

De exemplu, dacă $KB \begin{cases} \forall x \; Student(f(x,a)) \\ \forall x \forall z . Happy(f(x,g(z))) \end{cases}$

întrebare: $\exists x . Student(x) \wedge Happy(x)$

prin derivare obținem clauza $[A(f(x,g(z)))]$, adică răspunsul este orice instanță a termenului $f(x,g(z))$

Skolemizare

- pentru manipularea cuantificatorului existențial

Ideea: introducem nume unice pentru fiecare variabilă cuantificată cu $\exists$

$\exists x \forall y \exists z . P(x,y,z)$

numim x a

numim z $f(y)$

vom folosi $\forall y \, P(a, y, f(y))$

a, f sunt simbolurile Skolem (nu apar în altă parte)

În general, Skolemizarea înlocuiește fiecare variabilă existențială printr-o funcție cu atâtea argumente câte variabile universale domină existențialul respectiv.

$\alpha \quad \forall x_1 (\ldots \forall x_2 (\ldots \forall x_3 (\ldots \exists y [\ldots y \ldots] \ldots) \ldots) \ldots)$

$\downarrow$ Skolemizare

$\alpha' \quad \forall x_1 (\ldots \forall x_2 (\ldots \forall x_3 (\ldots [f(x_1,x_2,x_3)] \ldots) \ldots) \ldots)$

Obs. Constantele reale (ce au semnificație în domeniul explicat) sunt diferite de constantele Skolem care sunt generate doar pentru a evita variabilele existențiale. Adică $\exists x . P(x)$ nu este echivalent logic cu $P(a)$, varianta Skolemizată.

Obs $\not\models (\alpha \equiv \alpha')$ dar se poate demonstra că α poate fi satisfăcută dacă α' poate fi satisfăcută.

Obs. Pentru corectitudine logică este importantă păstrarea corectă a dependenței variabilelor.

De exemplu, $\exists x \forall y\, R(x,y) \models \forall y \exists x\, R(x,y)$ dar reciproca nu este adevărată

$$\{\exists x \forall y\, R(x,y),\ \neg \forall y \exists x\, R(x,y)\}$$

$\downarrow$ CNF

$$\{[R(a,y)],\ [\neg R(x,b)]\} \qquad a,b \text{ constante Skolem}$$

$\downarrow$ $x/a,\ y/b$

$$[\,]$$

Reciproca $\{\neg \exists x \forall y\, R(x,y),\ \forall y \exists x\, R(x,y)\}$

$\downarrow$ CNF

$$\{[\neg R(x, g(x))],\ [R(f(y), y)]\} \qquad f,g \text{ funcții Skolem}$$

nu unifică

Egalitatea

Dacă ne considerăm $=$ ca un predicat (normal), nu am găsi că mulțimea $\{a=b,\ b=c,\ a \neq c\}$ nu poate fi satisfăcută.
De aceea, este necesară adăugarea versiunilor clauzale pentru axiomele egalității:

- reflexivitate: $\forall x.\ x=x$
- simetrie: $\forall x \forall y.\ x=y \supset y=x$
- tranzitivitate: $\forall x \forall y \forall z.\ ((x=y \wedge y=z) \supset x=z)$
- substituție pentru funcții: pentru orice simbol de funcție f de aritate n, avem:

$$\forall x_1, \forall y_1, \ldots \forall x_n \forall y_n.\ x_1=y_1 \wedge \ldots \wedge x_n=y_n \supset f(x_1,\ldots,x_n) = f(y_1,\ldots,y_n)$$

- substituție pentru predicate: pentru orice simbol de predicat P de aritate n, avem:

$$\forall x_1, \forall y_1, \ldots \forall x_n \forall y_n.\ x_1=y_1 \wedge \ldots \wedge x_n=y_n \supset P(x_1,\ldots x_n) \equiv P(y_1,\ldots,y_n)$$

Acum $=$ poate fi tratată ca un predicat binar

Exemplul 7

$KB \begin{cases} \forall x.\ Married(father(x), mother(x)) \\ father(john) = bill \end{cases}$

intrebare: $Married(bill, mother(john))$

$\forall x_1 \forall y_1 \forall x_2 \forall y_2.$

$x_1 = y_1 \wedge x_2 = y_2 \supset Married(x_1, x_2) \equiv Married(y_1, y_2)$

↓ CNF (înlocuind $\supset$ și $\equiv$ folosind distributivitatea $\wedge, \vee$ și rezultând clauze)

$\{Married(y_1, y_2), \neg Married(x_1, x_2), x_1 \neq y_1, x_2 \neq y_2\}$