Rezoluția

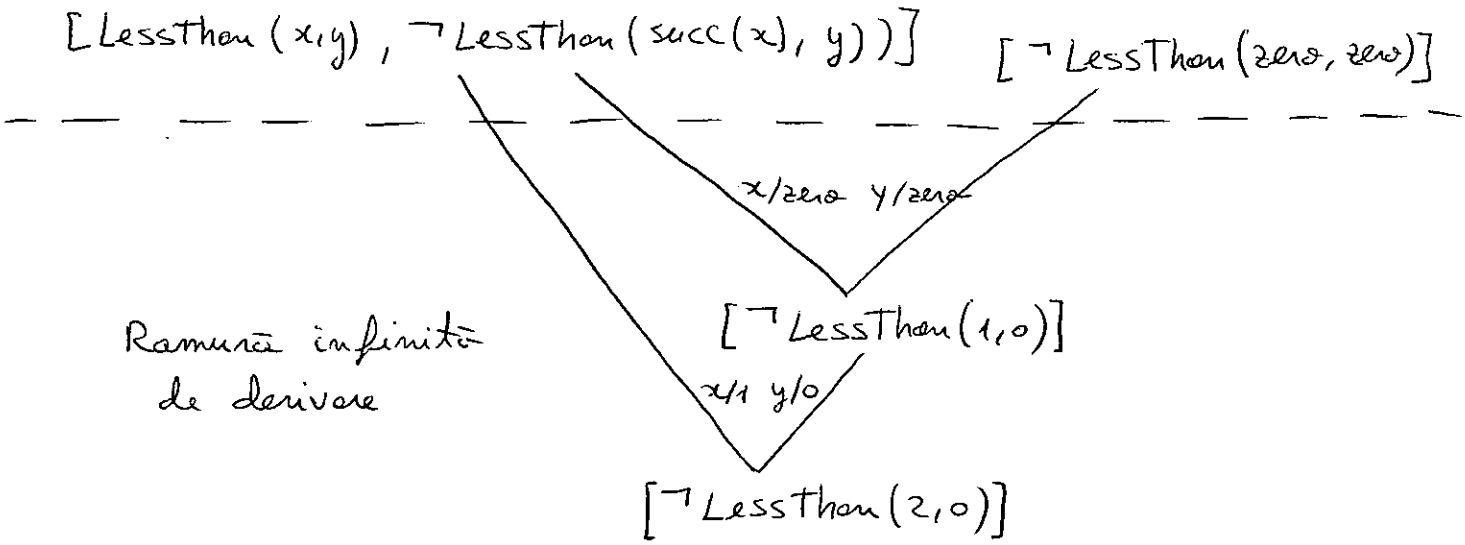
Dificultatea/imposibilitatea rezolvării computaționale

Rezoluția nu oferă o soluție generală eficientă pentru problemele raționamentului automat.

Cazul FOL

Considerăm KB: $\forall x \forall y . \text{LessThan}(\text{succ}(x), y) \supset \text{LessThan}(x, y)$.

Întrebare: $\text{LessThan}(\text{zero}, \text{zero})$



Nu putem aplica o procedură depth-first pentru a căuta clauza vidă deoarece ne putem bloca pe o ramură infinită.

Nu se poate detecta automat dacă o ramură va continua la nesfârșit

Dar știm că $S \models []$ dacă $S \vdash []$, adică dacă o mulțime de propoziții nu poate fi satisfăcută atunci o ramură a derivării va conține clauza vidă. Deci o căutare de tip breadth-first garantează raportarea cazului când clauzele nu pot fi satisfăcute. Dar dacă clauzele pot fi satisfăcute, căutarea poate sau nu să se oprească.

Teorema Herbrand

În cazul propozițional, procedura de rezoluție se termină întotdeauna (în mulțimea inițială avem un număr finit de literali).

Uneori, rezoluția în FOL se poate reduce la cazul propozițional.

Def Dacă S este o mulțime de clauze, universul Herbrand al lui S, notat H_S, este mulțimea tuturor termenilor de bază (adică fără variabile) formați din constantele și simbolurile de funcții din S (dacă S nu are constante sau simboluri de funcții, folosim o singură constantă a)

De exemplu, dacă $S = \{[\neg P(x, f(x,a)), \neg Q(x,a), R(x,b)]\}$

atunci $H_S = \{a, b, f(a,a), f(a,b), f(b,a), f(b,b), f(a,f(a,a)) \ldots\}$

Bază Herbrand a lui S, notată $H_S(S)$ este mulțimea tuturor clauzelor de bază $c\theta$, unde $c \in S$ și θ asignează variabilele din c termenilor din universul Herbrand.

Pentru S de mai sus avem:

$$H_S(S) = \{[\neg P(a, f(a,a)), \neg Q(a,a), R(a,b)],$$
$$[\neg P(b, f(b,a)), \neg Q(b,a), R(b,b)],$$
$$[\neg P(f(a,a), f(f(a,a),a)), \neg Q(f(a,a),a), R(f(a,a),b)],$$
$$[\neg P(f(b,a), f(f(b,a),a)), \neg Q(f(b,a),a), R(f(b,a),b)], \ldots\}$$

Teoreme Herbrand O mulțime de clauze poate fi satisfăcută dacă baza ei Herbrand poate fi satisfăcută.

Rezultatul este important deoarece baza Herbrand este o mulțime de clauze fără variabile (adică în esență este propozițională).

Însă dificultatea cu baza Herbrand este că tipic aceasta este o mulțime infinită de clauze propoziționale.

Dar este finită când universul Herbrand este finit (fără simboluri de funcții și cu număr finit de constante în S)

Uneori putem menține universul Herbrand finit considerând "tipul" argumentelor și valorilor funcțiilor, includând termeni ca $f(t)$ doar dacă tipul t este potrivit funcției f.

De exemplu, putem exclude din universul Herbrand birthday(birthday(john)).

Cazul propozitional

Știm că pentru o mulțime finită de clauze propoziționale, procedura de rezoluție se termină. Dar cât durează?

În 1985, Armin Haken a demonstrat că există clauze propoziționale ce nu pot fi satisfăcute $c_1, \dots c_n$, astfel încât cea mai scurtă derivare a clauzei vide are lungimea de ordinul 2^n.

Deci nu contează cât de bine alegem derivările, rezoluția necesită un timp exponențial.

Există o modalitate mai bună de a determina dacă o mulțime de clauze propoziționale poate fi satisfăcută? – este una dintre cele mai dificile întrebări din informatică.

În 1972, Stephen Cook a demonstrat că problema satisfacerii este NP-completă. Aproape orice problemă de căutare în care căutăm un element cu o anumită proprietate și în care putem testa în timp polinomial dacă un element candidat satisface proprietatea respectivă, poate fi transformată într-o problemă de satisfacere propozițională. Astfel, un timp polinomial pentru problema satisfacerii (ceea ce nu este cazul rezoluției) ar implica un timp polinomial pentru toate aceste probleme de căutare (în planificare, rutare).

Rezoluția nu este o soluție universală. Pentru ecuații imediate putem produce implicații ale KB, dar determinarea satisfacerii clauzelor poate fi prea dificilă dpv computațional.

Trebuie luate în calcul alte opțiuni:

- să dăm mai mult control utilizatorului asupra procesului de raționament (reprezentare procedurală)
- utilizarea limbajelor de reprezentare mai puțin expresive decât FOL sau logica propozițională (limbaje descriptive)

Cercetarea în domeniul reprezentării cunoștințelor abordează ambele direcții.

În unele aplicații ale rezoluției merită să așteptăm (îndelung) răspunsurile. Un domeniu al IA numit "demonstrare automată a teoremelor" folosește rezoluția în acest scop. De exemplu, să determinăm dacă conjectura Goldbach se deduce din axiomele teoriei numerelor.
Există aplicații ale rezoluției în care nu dorim neapărat o garanție a eficienței sau terminarea procedurii, ci o modalitate de a căuta derivări ce elimină cât de mult posibil pașii nenecesari.

Proceduri de rezolvare SAT

Sunt proceduri mai eficiente ca rezoluția, care determină dacă o mulțime de clauze poate fi satisfăcută.
Procedurile caută o interpretare care arată că clauzele sunt satisfăcute.
Se aplică de obicei clauzelor ce pot fi satisfăcute, dar nu se cunoaște interpretarea care le satisface.

Dacă C este o mulțime de clauze și m este un literal

$$C \bullet m = \{c \mid c \in C,\ m \notin c,\ \bar{m} \notin c\} \cup \{(c \setminus \bar{m}) \mid c \in C,\ m \notin C,\ \bar{m} \in C\}$$

Dacă $C = \{[p,q], [\bar{p}, a, b], [\bar{p}, c], [d, e]\}$

atunci $C \bullet p = \{[a,b], [c], [d,e]\}$

$$C \bullet \bar{p} = \{[q], [d,e]\}$$

$$(C \bullet \bar{p}) \bullet \bar{q} = \{[\,], [d,e]\}$$

$$(C \bullet \bar{p}) \bullet q = \{[d,e]\}$$

$$((C \bullet \bar{p}) \bullet q) \bullet d = \{\}$$

Dacă în interpretarea I
- p este adevărat atunci C satisfăcută dacă $C \bullet p$ satisfăcută
- p este fals atunci C satisfăcută dacă $C \bullet \bar{p}$ satisfăcută

Procedura D.P (Davis, Putnam)

intrare: multimea de clauze C

ieșire: sunt satisfăcute clauzele: YES sau NO

```
procedure DP(C)
    if C este vidă then return YES
    if C conține [] then return NO
    alegem p un atom din C
    if DP(C•p) = YES then return YES
                     else return DP(C•p̄)
end
```

Strategii pentru alegerea lui p:

- p apare în cele mai multe clauze din C
- p apare în cele mai puține clauze
- p este cel mai "echilibrat" atom (numărul aparițiilor pozitive este cel mai apropiat de numărul aparițiilor negative)
- p este cel mai puțin echilibrat atom
- p apare în cea mai scurtă clauză din C

În diverse variante optimizate, procedura DP este în practică cea mai rapidă dintre toate procedurile SAT cunoscute (pe cazul propozițional). Astfel, se pot aborda probleme cu zeci de milioane de variabile. Procedurile SAT au revoluționat domenii precum verificarea hardware și verificarea protocoalelor de securitate.

Unificatorii cei mai generali

O modalitate de a evita căutările nenecesare în derivările de ordin I este să menținem căutarea cât mai generală posibil.

De exemplu clauza C_1 conține literalul $P(g(x), f(x), z)$

C_2 conține literalul $\neg P(y, f(w), a)$

Pentru unificare, avem $\theta_1 = \{x/b, y/g(b), z/a, w/b\}$

sau

$\theta_2 = \{x/f(z), y/g(f(z)), z/a, w/f(z)\}$

sau ...

Putem încerca să derivăm clauze vidă folosind θ_1; dacă nu merge încercăm cu θ_2; ș.a.m.d

θ_1 și θ_2 sunt mai specifică decât ar trebui. (nu este necesar să dăm o valoare pentru x).

Substituția $\theta_3 = \{y/g(x), z/a, w/x\}$ unifică cei 2 literali fără a face alegeri arbitrare nenecesare, care ar putea exclude calea către clauze vidă.

θ_3 este un cel mai general unificator. Pot exista mai mulți cei mai generali unificatori: $\theta_4 = \{y/g(w), z/a, x/w\}$

Un cel mai general unificator θ al literalilor S_1 și S_2 are proprietatea că pentru orice alt unificator θ', există un alt unificator θ^* cu $\theta' = \theta \cdot \theta^*$ (adică $S(\theta \cdot \theta^*) = (S\theta)\theta^*$)

De exemplu, din θ_3 putem ajunge la θ_1 aplicând x/b și la θ_2 aplicând $x/f(z)$

Putem limita rezoluția la cei mai generali unificatori, fără pierderea completitudinii. Căutarea este îmbunătățită prin reducerea dramatică a numărului de rezolvente ce pot deriva din două clauze.

Un cel mai general unificator al literalilor S_1 și S_2 poate fi calculat eficient astfel:

1. $\theta = \{\}$
2. dacă $S_1\theta = S_2\theta$ atunci stop.
3. determină mulțimea de neconcordanță MN, adică perechea termenilor primei neconcordanțe (de la stânga spre dreapta) a celor doi literali

$S_1\theta = P(a, f(a, \underline{g(z)}, \ldots))$ atunci $MN = \{u, g(z)\}$

$S_2\theta = P(a, f(a, \underline{u}, \ldots))$

4. găsește variabila $V \in MN$ și termenul $t \in MN$ ce nu conține V
   dacă nu există atunci stop (S_1 și S_2 nu unifică)
5. altfel, $\theta = \theta \cdot \{V/t\}$ și mergi la pasul 2.

Procedura este foarte eficientă în practică. Toate sistemele bazate pe rezoluție folosesc unificatorii cei mai generali.

Alte optimizări ale rezoluției pentru îmbunătățirea căutării

Eliminarea de clauze

Se dorește menținerea unui cât mai mic număr de clauze, fără a afecta completitudinea. Există clauze care nu participă în derivarea (cea mai scurtă) către clauza vidă:

- clauze pure: clauzele ce conțin un literal ℓ astfel încât $\overline{\ell}$ nu apare nicăieri;
- tautologiile: clauzele ce conțin ℓ și $\overline{\ell}$
- clauze subsumate: clauze pentru care există deja în KB o clauză cu un subset al literalilor (adică clauze mai specifice decât o clauză în KB, de exemplu, dacă $[P(x)] \in KB$ atunci nu mai adăugăm $[P(a)]$ sau $[P(a), Q(b)]$; dacă avem $[p, r]$ nu mai ținem cont de $[p, q, r]$

Strategii de ordonare

- alegerea unei ordini de executare a rezoluției care să maximizeze șansa derivării clauzei vide.

Cea mai bună strategie de până acum este "unit preference" adică folosirea întâi a clauzelor unitate (i.e. clauze cu un singur literal)

O clauză unitate + o clauză cu k literali $\Rightarrow$ o clauză cu $k-1$ literali (scurtând clauzele se speră că se ajunge mai repede la clauza vidă)

Multimea de suport

Într-o aplicație, este de așteptat ca KB să fie satisfăcută, de aceea nu are rost să aplicăm rezoluția doar pe clauze din KB. Rezoluția este permisă doar dacă una dintre clauzele de intrare are un strămoș în negarea întrebării.

Cazul special al egalității

Utilizarea explicită a axiomelor egalității poate genera foarte multe rezolvante. Din acest motiv se introduce o a doua regulă de inferență numită Paramodulare:

Avem două clauze $C_1 \cup \{t = s\}$ unde t și s sunt termeni
și $C_2 \cup \{S[t']\}$ ce conține un termen t'

Redenumim (dacă e necesar) variabilele din cele două clauze astfel încât să fie distincte și presupunem că avem substituția θ cu $t\theta = t'\theta$.
Atunci putem deduce clauza $(\{C_1 \cup C_2 \cup S[s]\})\theta$ ce elimină atomul de egalitate, înlocuiește t' cu s și apoi aplică substituția θ.

De exemplu $[\underbrace{father(john)}_{t} = \underbrace{bill}_{s}]$ $[\underbrace{Married}_{S}(\underbrace{father(x)}_{t'}, mother(x))]$

$C_1 = \{\}$ $C_2 = \{\}$
$\theta = \{x/john\}$

$\Rightarrow$ ~~Married(father(bill), mother(john))~~ $[Married(bill, mother(john))]$
se obține într-un singur pas al Paramodulării

Conexiuni direcționale

O clauză de tipul $[\neg p, q]$ (reprezentând $p \Rightarrow q$) poate fi folosită în două direcții:

- forward - dacă derivăm o clauză ce conține p, derivăm apoi clauza cu q
- backward - dacă derivăm o clauză ce conține $\neg q$, derivăm apoi clauza cu $\neg p$

Putem marca anumite clauze ca să fie utilizate doar într-o direcție

De exemplu, dacă în KB avem

$$\forall x.\ Battleship(x) \Rightarrow Gray(x)$$

ar putea fi folosită doar în direcția forward
(ar putea să nu fie o idee ex de bună să demonstrăm că ceva nu este nevă de război dacă nu este gri)

În această strategie trebuie să ne asigurăm că nu se pierde completitudinea.