Capitolul 3

Demonstrarea teoremelor în logica cu predicate

Demonstrarea teoremelor în logica propozitionala si în logica cu predicate de ordinul întîi consta, în esenta, în gasirea unei proceduri de decizie pentru a verifica validitatea sau inconsistenta unei formule. Aceasta problema a fost abordata întîi de Leibnitz (1646-1716) si a fost apoi reluata de Peano la începutul secolului XX si de scoala lui Hilbert în jurul anilor '20. Abia Church (1936) si Turing (1936) au demonstrat ca acest lucru, posibil în calculul propozitional, este imposibil în calculul cu predicate de ordinul întîi. Ei au demonstrat ca nu exista o procedura efectiva de verificare a validitatii unei formule, dar exista proceduri efective ce pot demonstra ca o formula este valida daca aceasta este într-adevar valida. Deci sistemul formal al logicii cu predicate nu este decidabil, dar este semidecidabil.

Primul mare pas în demonstrarea automata a teoremelor a fost facut de Herbrand (1930). Prin definitie, o formula este valida daca formula este adevarata în toate interpretarile. Herbrand a dezvoltat un algoritm pentru a gasi o interpretare ce poate falsifica o formula. Daca formula este într-adevar valida, nici o astfel de interpretare nu poate exista si algoritmul se opreste dupa un numar finit de pasi. Teorema lui Herbrand este baza celor mai multe metode moderne de demonstrare automata a teoremelor [Chang,Lee,1973].

Gilmore [1960] a fost unul dintre primii cercetatori care a implementat procedura lui Herbrand pe calculator. Programul lui determina inconsistenta negarii formulei care se demonstreaza a fi valida. Programul a putut fi aplicat efectiv numai pentru dimensiuni reduse ale intrarii. La putin timp dupa el, Davis si Putnam au îmbunatatit metoda, dar nu suficient, deoarece multe formule nu puteau fi demonstrate într-un timp rezonabil.

În 1965, plecînd de la rezultatele lui Herbrand, Robinson a propus o metoda mult mai eficienta de stabilire a inconsistentei unei formule: rezolutia [Robinson,1965]. Aceasta metoda, cu diversele ei rafinari ulterioare, a devenit abordarea preferentiala a celor mai multe demonstratoare automate de teoreme dezvoltate pîna în prezent.

Metodele propuse de Hilbert, Gilmore, Davis si Putnam si Robinson sînt *metode sintactice* de demonstrare a teoremelor. Metoda axiomatica prezentata în Sectiunea 2.3 este, de asemenea, o metoda sintactica. Metodele sintactice se bazeaza pe procedee mecanice de aplicare a regulilor de inferenta si sînt independente de domeniul de interpretare a formulei. Exista si *metode semantice* pentru demonstrarea teoremelor, metode care se bazeaza pe utilizarea sistematica a valorilor de adevar date de functia de valorizare a formulei.

Metodele semantice de demonstrare a teoremelor utilizeaza doua strategii pentru demonstrarea validitatii, respectiv inconsistentei unei formule:

- se atribuie valori de adevar propozitiilor elementare, apoi se stabileste valoarea de adevar pentru formula
- se acorda o valoare ipotetica propozitiei complexe asupra careia trebuie sa se decida, apoi, pas cu pas, se ajunge sa se atribuie valori de adevar inconsistente propozitiilor elementare.

Principalele metode semantice sînt: metoda matriceala, numita si metoda tabelelor de adevar, metoda lui Quine, metoda tablourilor semantice, metoda arborilor de decizie. O prezentare detaliata a acestor metode poate fi gasita în Popa [1992].

Metodele sintactice de demonstrare a teoremelor sînt cele mai indicate pentru automatizarea procesului de demonstrare si ele stau la baza celor mai multe demonstratoare automate de teoreme [Gabbay,s.a.,1993]. În continuare se vor prezenta cele mai importante metode sintactice care au stat la baza construirii programelor de demonstrare automata a teoremelor în inteligenta artificiala.

3.1 Logica cu predicate de ordinul I

Logica propozitiilor este un caz particular al logicii cu predicate de ordinul I. La fel ca si în cazul logicii propozitiilor, definirea logicii cu predicate de ordinul I începe prin a fixa alfabetul limbajului. Alfabetul contine simboluri pentru reprezentarea constantelor, notate prin conventie cu litere mici de la începutul alfabetului $(a, b, c, \ldots)$, variabilelor, notate prin conventie cu litere mici de la sfîrsitul alfabetului $(x, y, z, \ldots)$, functiilor, notate cu $f, g, \ldots$, predicatelor, notate cu $P, Q, R, \ldots$, a conectorilor si a cuantificatorilor logici. Conectorii logici folositi în logica cu predicate de ordinul I sînt: $\sim, \wedge, \vee, \rightarrow$ si $\leftrightarrow$, iar cuantificatorii sînt cuantificatorul existential ($\exists$) si cuantificatorul universal ($\forall$).

În cazul logicii cu predicate de ordinul I, predicatele sînt functii logice de mai multe argumente, argumentele predicatelor numindu-se termeni.

Definitie. Fie D un domeniu de valori. Un *termen* se defineste astfel:

(1) O constanta este un termen cu valoare fixa apartinînd domeniului D.

(2) O variabila este un termen ce poate primi valori diferite din domeniul D.

(3) Daca f este o functie de n argumente $(f: D^n \rightarrow D)$ si $t_1, \ldots, t_n$ sînt termeni, atunci $f(t_1, \ldots, t_n)$ este termen.

(4) Toti termenii sînt generati prin aplicarea regulilor (1)÷(3).

Definitie. Se numeste *predicat de aritate n* o functie P de n argumente cu valori adevarat sau fals, $P: D^n \to \{\mathbf{a}, \mathbf{f}\}$. Un predicat de aritate 0 este o propozitie, numita si *predicat constant*.

Definitie. Daca P este un predicat de aritate n si $t_1, \ldots, t_n$ sînt termeni, atunci $P(t_1, \ldots, t_n)$ se numeste *atom* sau *formula atomica.* Nici o alta expresie nu poate fi atom.

Definitie. Se numeste *literal* un atom sau un atom negat. Literalul reprezentat de un atom se numeste *literal pozitiv*, iar literalul reprezentat de un atom negat se numeste *literal negativ*.

Definitie. O *formula bine formata* în logica cu predicate de ordinul I se defineste astfel:

(1) Un atom este o formula bine formata.

(2) Daca P este o formula bine formata atunci: $\sim P$, $(\exists x)P(x)$, $(\forall x)P(x)$ sînt formule bine formate.

(3) Daca P si Q sînt formule bine formate atunci: $P \wedge Q$, $P \vee Q$, $P \to Q$, $P \leftrightarrow Q$ sînt formule bine formate.

(4) Orice formula bine formata este generata prin aplicarea de un numar finit de ori a regulilor (1)÷(3).

Interpretarea unei formule F în logica cu predicate de ordinul I consta în fixarea unui domeniu de valori nevid D si a unei atribuiri de valori pentru fiecare constanta, functie si predicat ce apare în F astfel:

(1) Fiecarei constante i se asociaza un element din D.

(2) Fiecarei functii f, de aritate n, i se asociaza o corespondenta $D^n \to D$, unde $D^n = \{(x_1, \ldots, x_n) | x_1 \in D, \ldots, x_n \in D\}$.

(3) Fiecarui predicat de aritate n, i se asociaza o corespondenta $P: D^n \to \{\mathbf{a}, \mathbf{f}\}$.

Într-o formula variabilele pot fi variabile libere sau legate. O variabila este legata într-o formula daca exista un cuantificator ce o refera. În caz contrar variabila este libera. O formula care contine variabile libere nu poate fi evaluata.

O formula bine formata poate fi: consistenta (realizabila), inconsistenta (nerealizabila), valida sau consecinta logica a unei multimi de formule, asa cum s-a definit în Capitolul 2.

Metodele de demonstrare a teoremelor ce vor fi studiate în acest capitol au la baza cele doua teoreme ale echivalentei logice prezentate în Sectiunea 2.1.4. Procedura lui Herbrand si rezolutia sînt metode de demonstrare prin respingere sau de demonstrare prin reducere la absurd. În loc de

a arata ca o formula este valida, ele arata ca negatia aceleiasi formule este inconsistenta, de unde si denumirea de demonstrare prin "respingere". Aceste metode se aplica însa unei forme standard a formulelor, numita forma clauzala, forma introdusa de Davis si Putnam.

Definitie. Se numeste *clauza* o disjunctie între un numar de literali. Se numeste *clauza de baza* o clauza fara variabile. Se numeste *clauza Horn* o clauza care contine cel mult un literal pozitiv.

Definitie. Se numeste *clauza vida* o clauza fara nici un literal; clauza vida se noteaza, prin conventie, cu simbolul □. Se numeste *clauza unitara* o clauza ce contine un singur literal.

Definitie. O *clauza Horn* poate avea una din urmatoarele patru forme: o clauza unitara pozitiva ce consta într-un singur literal pozitiv; o clauza negativa formata numai din literali negati; o clauza formata dintr-un literal pozitiv si cel putin un literal negat (*clauza Horn mixta*) sau clauza vida. Se numeste *clauza* (Horn) *distincta* o clauza ce are exact un literal pozitiv, ea fiind fie o clauza unitara pozitiva, fie o clauza Horn mixta.

Transformarea unei formule bine formate în forma clauzala se face pe baza regulilor prezentate în continuare.

Etapa 1. Transformarea formulei în forma normala prenex.

Definitie. O formula F în calculul cu predicate de ordinul I este în *forma normala prenex* daca si numai daca formula F este de forma $(Q_1x_1)\ldots(Q_nx_n)M$, unde fiecare $(Q_ix_i), i = 1,\ldots,n$, este fie $(\forall x_i)$, fie $(\exists x_i)$, iar M este o formula ce nu contine cuantificatori. $(Q_1x_1)\ldots(Q_nx_n)$ se numeste *prefixul* formulei F, iar M se numeste *matricea* formulei F.

Pentru transformarea unei formule bine formate în forma normala prenex se folosesc urmatoarele legi de echivalenta:

(1.1) $(Qx)F[x] \vee G = (Qx)(F[x] \vee G)$

(1.2) $(Qx)F[x] \wedge G = (Qx)(F[x] \wedge G)$

(2.1) $\sim((\forall x)F[x]) = (\exists x)(\sim F[x])$

(2.2) $\sim((\exists x)F[x]) = (\forall x)(\sim F[x])$

(3.1) $(\forall x)F[x] \wedge (\forall x)H[x] = (\forall x)(F[x] \wedge H[x])$

(3.2) $(\exists x)F[x] \vee (\exists x)H[x] = (\exists x)(F[x] \vee H[x])$

(4.1) $(Q_1x)F[x] \wedge (Q_2x)H[x] = (Q_1x)(Q_2z)(F[x] \wedge H[z])$

(4.2) $(Q_1x)\ F[x] \vee (Q_2x)\ H[x] = (Q_1x)(Q_2z)(F[x] \vee H[z])$

Transformarea unei formule în forma normala prenex parcurge urmatorii pasi:

Pas 1. Se elimina din formula conectorii logici de implicatie si echivalenta, prin utilizarea legilor

$$F \leftrightarrow G = (F \rightarrow G) \wedge (G \rightarrow F)$$

$$F \rightarrow G = \sim F \vee G$$

Pas 2. Se muta toate negatiile din formula astfel încît sa preceada atomii, utilizînd repetat legea $\sim (\sim F) = F$, legile lui de Morgan si formulele de echivalenta (2.1) si (2.2).

Pas 3. Se redenumesc, daca este cazul, variabilele legate din formula astfel încît toti cuantificatorii sa se refere la variabile diferite (variabilele referite de un cuantificator nu trebuie sa fie referite de alt cuantificator).

Pas 4. Se folosesc formulele de echivalenta (1.1), (1.2), (3.1), (3.2), (4.1) si (4.2).

Etapa 2. Transformarea matricei formei normale prenex în forma normala conjunctiva, deci în forma $F_1 \wedge F_2 \wedge \ldots \wedge F_n$.

Etapa 3. Eliminarea tuturor cuantificatorilor existentiali din prefixul formei normale prenex.

Fara a afecta proprietatile de inconsistenta, cuantificatorii existentiali din prefixul unei forme normale prenex pot fi eliminati prin utilizarea substitutiilor de skolemnizare: înlocuirea variabilelor cuantificate existential cu functii Skolemn, adica functii arbitrare ce pot lua întotdeauna valoarea ceruta de cuantificatorul existential.

Pasii executati în aceasta etapa sînt urmatorii:

Pas 1. Daca primul (cel mai din stînga) cuantificator este un cuantificator existential, se înlocuiesc toate aparitiile variabilei pe care acesta o cuantifica cu o constanta arbitrara ce nu apare în prefixul formei normale prenex si se elimina cuantificatorul. Acest proces se aplica pentru toti cuantificatorii existentiali care nu sînt precedati de cuantificatori universali, folosind constante diferite în procesele de substitutie.

Pas 2. Pentru fiecare cuantificator existential care este precedat de unul sau mai multi cuantificatori universali, se înlocuiesc toate aparitiile variabilei cuantificate printr-o functie (distincta de toate functiile si variabilele ce apar în prefixul formei normale prenex) care are ca argumente toate variabilele cuantificate universal ce preced cuantificatorul existential si se elimina cuantificatorul existential. Procesul se repeta pentru fiecare cuantificator existential, folosind un simbol de functie diferit si alegînd ca variabile ale functiei argumentele ce corespund tuturor variabilelor cuantificate universal care preced cuantificatorul existential.

Etapa 4. Eliminarea tuturor cuantificatorilor universali din prefixul formei normale prenex si a tuturor conjunctiilor din matricea formei normale prenex.

Prin eliminarea cuantificatorilor existentiali, prefixul formei normale prenex dispare. Multimea de formule care rezulta dupa eliminarea conjunctiilor din matricea formei normale prenex formeaza un set de clauze ce nu este echivalent cu forma originala a formulei bine formate, dar a carui realizabilitate, respectiv inconsistenta, este pastrata, conform urmatoarei teoreme.

Teorema. Fie S un set de clauze reprezentînd o forma standard a unei formule F. Formula F este inconsistenta daca si numai daca S este inconsistent.

Pentru o prezentare mai detaliata si exemple de transformare a unei formule bine formate în forma clauzala se poate consulta Chang si Lee [1973] si Florea [1993].

3.2 Universul Herbrand

Prin definitie, un set de clauze S este nerealizabil daca si numai daca este fals în toate interpretarile posibile. O astfel de abordare, posibila în calculul propozitiilor, este ineficienta sau chiar imposibila în calculul cu predicate de ordinul I, unde domeniul de interpretare poate fi foarte mare sau chiar infinit. Din acest motiv s-a cautat stabilirea unui domeniu de interpretare special, astfel încît un set de clauze S este nerealizabil daca si numai daca S este fals în toate interpretarile pe acest domeniu. Din fericire un astfel de domeniu exista. El a fost propus de Herbrand si se numeste *universul Herbrand al multimii de clauze* S.

3.2.1 Universul Herbrand al unei multimi de clauze.

Definitie. Fie S o multime de clauze ce corespund unei formule. Fie H_0 multimea constantelor care apar în S. Daca în S nu apare nici o constanta, atunci H_0 se initializeaza cu o singura constanta, $H_0 = \{a\}$. Pentru $i = 0,1,2,\ldots$, fie $H_{i+1} = H_i \cup \{f(t_1,\ldots,t_n) | t_j \in H_i, 1 \le j \le n\}$, unde f sînt functii de aritate n ce apar în S. H_i se numeste *multimea constanta de nivel i* a lui S. *Universul Herbrand al setului de clauze* S este $\lim_{i\to\infty} H_i$.

Observatie. Constanta de initializare a multimii H_0, ce se alege arbitrar în cazul în care nu exista constante în formulele din S, nu trebuie confundata cu valoarea de adevar **a**.

Exemple:

1. Fie setul de clauze $S = \{P(a), \sim P(x) \vee P(f(x))\}$. Multimile constante de nivel $i = 0,1,2,\ldots$ sînt: $H_0 = \{a\}$, $H_1 = \{a, f(a)\}$, $H_2 = \{a, f(a), f(f(a))\}$,... iar universul Herbrand al setului de clauze S este $H = H_\infty = \{a, f(a), f(f(a)), f(f(f(a))), \ldots\}$.

2. Fie setul de clauze $S = \{P(x) \vee Q(x), R(z), T(y) \vee \sim W(y)\}$. Atunci $H_0 = \{a\}$ (deoarece nu exista constante, se alege o constanta arbitrara) si deoarece nu exista functii în formulele din S universul Herbrand este $H = H_0 = H_1 = \ldots = \{a\}$.

3. Fie setul de clauze $S = \{P(f(x), a, g(y), b)\}$. Multimile constante de nivel $i = 0,1,2,...$ sînt:

$H_0 = \{a, b\}$,

$H_1 = \{a, b, f(a), f(b), g(a), g(b)\}$,

$H_2 = \{a, b, f(a), f(b), g(a), g(b), f(f(a)), f(f(b)), f(g(a)), f(g(b)), g(g(a)), g(g(b)), g(f(a)), g(f(b))\}$

...

Definitie. Se numeste *expresie* un termen, o multime de termeni, un atom, o multime de atomi, un literal, o clauza sau o multime de clauze. Daca într-o expresie nu apar variabile, ea se numeste *expresie de baza.*

Se pot utiliza deci notiunile de termen de baza, atom de baza, literal de baza si clauza de baza. O subexpresie a unei expresii E este o expresie care apare în E.

Definitie. Fie S un set de clauze. Multimea $A = \{P(t_1,...,t_n) | t_i \in H, 1 \le i \le n, P \text{ apare în } S\}$ a atomilor de baza, deci multimea tuturor predicatelor ce apar în S, avînd ca termeni elemente ale universului Herbrand, se numeste *baza Herbrand* sau *multime atom* a lui S.

Definitie. O *instanta de baza* a unei clauze $C \in S$, unde S este o multime de clauze, este o clauza obtinuta prin înlocuirea variabilelor din C cu membrii ai universului Herbrand al lui S.

Exemplu. Fie $S = \{P(x), Q(f(y)) \vee R(y)\}$ o multime de clauze, avînd universul Herbrand $H = \{a, f(a), f(f(a)), \ldots\}$. Fie clauza $C = P(x)$. Atunci $P(a)$ si $P(f(f(a)))$ sînt ambele instante de baza ale lui $C \in S$. Daca se considera clauza $C_1 = Q(f(y)) \vee R(y)$ atunci $Q(f(a)) \vee R(a)$ este o instanta de baza a lui $C_1 \in S$.

3.2.2 H-interpretare

Se considera acum interpretarile unui set de clauze pe universul Herbrand.

Definitie. Fie S o multime de clauze, H universul Herbrand al lui S si I o interpretare a lui S pe H. I se numeste *H- interpretare* a lui S daca satisface urmatoarele conditii:

(1) I pune în corespondenta toate constantele din S cu ele însele (a a a, pentru orice constanta $a \in S$);

(2) Fie f o functie de aritate n din S. Se asociaza lui f în I functia care stabileste corespondenta $(h_1, \ldots, h_n) \in H^n$ a $f(h_1, \ldots, h_n) \in H$.

Fie $A = \{A_1, \ldots, A_m, \ldots\}$ baza Herbrand a lui S. O H-interpretare I poate fi reprezentata convenabil prin multimea $I = \{m_1, \ldots, m_n, \ldots\}$ în care m_j este fie A_j, fie $\sim A_j$, pentru $j = 1, 2, \ldots$.

Semnificatia acestei multimi este aceea ca daca m_j este A_j, atunci lui A_j i se asociaza valoarea adevarat (**a**), altfel lui A_j i se asociaza valoarea fals (**f**).

Exemplu. Fie multimea de clauze $S = \{P(x) \vee Q(x), R(f(y))\}$ cu universul si baza Herbrand $H = \{a, f(a), f(f(a)), \ldots\}$, respectiv $A = \{P(a), Q(a), R(a), P(f(a)), Q(f(a)), R(f(a)), \ldots\}$. Urmatoarele sînt H-interpretari ale lui S:

(1) $I_1 = \{P(a), Q(a), R(a), P(f(a)), Q(f(a)), R(f(a)), \ldots\}$

(2) $I_2 = \{\sim P(a), \sim Q(a), \sim R(a), \sim P(f(a)), \sim Q(f(a)), \sim R(f(a)), \ldots\}$

(3) $I_3 = \{P(a), Q(a), \sim R(a), P(f(a)), Q(f(a)), \sim R(f(a)), \ldots\}$

Observatie. O interpretare a unei multimi de clauze S nu trebuie neaparat sa fie definita pe universul Herbrand al lui S. Deci o interpretare ar putea sa nu fie o H-interpretare.

Exemplu. Fiind dat un set de clauze $S = \{P(x), Q(y, f(y, a))\}$ si domeniul de interpretare $D = \{1, 2\}$, se considera urmatoarea interpretare I

a	f(1,1)	f(1,2)	f(2,1)	f(2,2)
2	1	2	2	1

P(1)	P(2)	Q(1,1)	Q(1,2)	Q(2,1)	Q(2,2)
a	**f**	**f**	**a**	**f**	**a**

Aceasta interpretare nu este o H-interpretare a lui S.

Pentru orice interpretare I a unei multimi de clauze se poate defini H-interpretarea I^*, corespunzatoare lui I.

Definitie. Fiind data o interpretare I peste un domeniu D, o *H-interpretare* I^* *corespunzatoare lui* I este o H-interpretare care satisface urmatoarele:

(1) $(h_1, \ldots, h_n) \in H^n$, unde H este universul Herbrand al lui S;

(2) h_i a d_i, $d_i \in D$, $\forall i\ 1 \le i \le n$;

(3) daca $P(d_1, \ldots, d_n)$ este adevarat (respectiv fals) în I, atunci $P(h_1, \ldots, h_n)$ este tot adevarat (respectiv fals) în I^*.

Exemplu. Considerînd multimea de clauze S si interpretarea I din exemplul anterior, $A = \{P(a), Q(a,a), P(f(a,a)), Q(a, f(a,a)), Q(f(a,a), a), Q(f(a,a), f(a,a))\}$ este baza Herbrand a

lui S. Pentru a obtine o H-interpretare I^* corespunzatoare lui I se evalueaza fiecare membru al lui A, utilizînd valorile interpretarii I.

$P(a) = P(2) =$ **f**

$Q(a,a) = Q(2,2) =$ **a**

$P(f(a,a)) = P(f(2,2)) = P(1) =$ **a**

$Q(a,f(a,a)) = Q(2,f(2,2)) = Q(2,1) =$ **f**

$Q(f(a,a),a) = Q(f(2,2),2) = Q(1,2) =$ **a**

$Q(f(a,a),f(a,a)) = Q(f(2,2),f(2,2)) = Q(1,1) =$ **f**

Pe baza acestor evaluari se construieste H-interpretarea I^* corespunzatoare lui I

$I^* = \{\sim P(a), Q(a,a), P(f(a,a)), \sim Q(a,f(a,a)), Q(f(a,a),a), \sim Q(f(a,a),f(a,a)), ...\}$

Lema. Daca o interpretare I peste un domeniu D satisface o multime de clauze S, i.e. multimea de clauze este realizabila în acea interpretare, atunci orice H-interpretare I^* corespunzatoare lui I satisface, de asemenea, S.

Se propune cititorului gasirea demonstratiei acestei leme.

Teorema. O multime de clauze S este inconsistenta (nerealizabila) daca si numai daca S este falsa pentru toate H-interpretarile lui S.

Demonstratie.

($\Rightarrow$) Aceasta implicatie este evidenta deoarece daca S este nerealizabila atunci S este falsa în toate interpretarile peste orice domeniu, deci si pentru H-interpretarile lui S.

($\Leftarrow$) Fie multimea de clauze S falsa pentru toate H-interpretarile si sa presupunem S realizabila, deci S nu este inconsistenta. Atunci exista o interpretare I peste un domeniu D astfel încît S este adevarata în I. Fie I^* H-interpretarea corespunzatoare lui I. Conform lemei, S este adevarata în I^*. Aceasta contrazice presupunerea ca S este falsa pentru toate H-interpretarile.

În concluzie s-a atins obiectivul propus la începutul acestui capitol. Este nevoie sa se considere numai interpretarile peste universul Herbrand, în particular H-interpretari, pentru a verifica daca o multime de clauze este sau nu contradictie (nerealizabila). În continuare, din cauza teoremei de mai sus, de cîte ori se va vorbi de o "interpretare", se va face referire la o H-interpretare.

3.2.3 Arbori semantici

Arborii semantici reprezinta structuri conceptuale care permit organizarea sistematica a H-interpretarilor unei formule. Ei stau la baza primei formulari a criteriului de nerealizabilitate a unei multimi de clauze propus de Herbrand. În plus, arborii semantici pot fi utilizati pentru stabilirea legaturii între metoda lui Herbrand de demonstrare a inconsistentei unui set de clauze si metoda lui Robinson, rezolutia.

Definitie. Daca A este un atom, atunci cei doi literali A si ~A se spun a fi unul *complementul* celuilalt si multimea $\{A, \sim A\}$ este numita *pereche complementara.*

Definitie. Fie S o multime de clauze si fie A baza Herbrand a lui S. Un *arbore semantic* al lui S este un arbore T construit astfel:

(1) Pentru fiecare nod N exista un numar finit de succesori $L_1, \ldots, L_n$ pentru care, pe arcele $(N, L_i), 1 \le i \le n$, se ataseaza o multime finita de atomi sau atomi negati din A (literali din A).

(2) Fie Q_i conjunctia tuturor literalilor atasati legaturii $(N, L_i), 1 \le i \le n$. Atunci $Q_1 \vee Q_2 \vee \ldots \vee Q_n$ este o formula propozitionala valida.

(3) Pentru fiecare nod N, daca I(N) este reuniunea tuturor multimilor de literali atasate legaturilor de-a lungul unei cai de la radacina la N, atunci I(N) nu contine o pereche complementara.

Observatie. $Q_1 \vee Q_2 \vee \ldots \vee Q_n$ este o formula propozitionala deoarece ea nu contine variabile, fiind construita numai cu elemente din A.

Definitie. Fie $A = \{A_1, \ldots, A_k, \ldots\}$ baza Herbrand a multimii de clauze S. Un arbore semantic T al lui S este *arbore semantic complet* daca si numai daca pentru fiecare nod frunza al lui T, I(N) contine fie A_i, fie $\sim A_i$, $i = 1, 2, \ldots$, i.e. fiecare cale de la radacina la o frunza contine toate elementele din baza Herbrand, fie sub forma directa, fie sub forma negata.

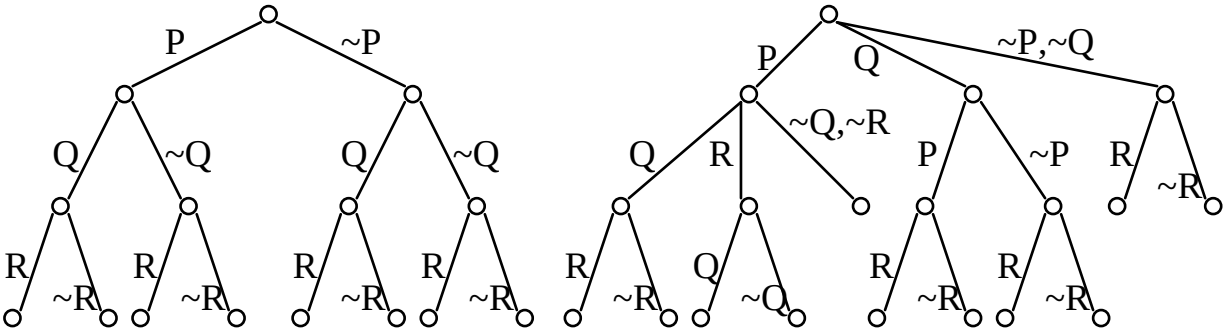
Observatii:

- Pentru orice nod $N \in T$, unde T este un arbore semantic al lui S, I(N) este o submultime a unei interpretari a lui S si se numeste *interpretare partiala* a lui S.
- Într-un arbore semantic complet, pentru orice nod frunza $N \in T$, I(N) descrie o interpretare completa a lui S.
- Daca baza Herbrand A a unei multimi de clauze S este infinita, atunci orice arbore semantic complet va fi infinit.

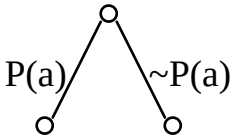
- Un arbore semantic complet corespunde testarii complete a tuturor interpretarilor posibile ale lui S.

Exemple:

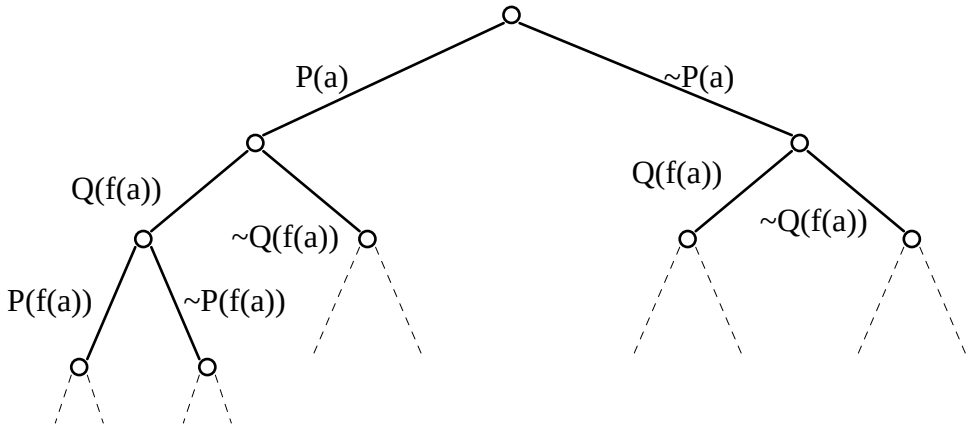
1. Fie un set de clauze S si $A = \{P, Q, R\}$ baza Herbrand a lui S. Cei doi arbori prezentati mai jos sînt arbori semantici completi ai multimii de clauze S.



2. Fie setul de clauze $S = \{P(x), P(a)\}$ avînd baza Herbrand $A = \{P(a)\}$. Arborele semantic complet al setului de clauze S este



3. Fie multimea de clauze $S = \{P(x), Q(f(x))\}$ avînd baza Herbrand infinita $A = \{P(a), Q(a), P(f(a)), Q(f(a)), P(f(f(a))), Q(f(f(a))), ...\}$. Deoarece baza Herbrand A este infinita, si arborele semantic complet este infinit. O portiune a acestui arbore semantic este



Pornind de la ultima observatie facuta si stiind ca o formula este nerealizabila daca este falsa în toate interpretarile, se poate opri expandarea unui nod N în constructia arborelui semantic daca I(N) falsifica S.

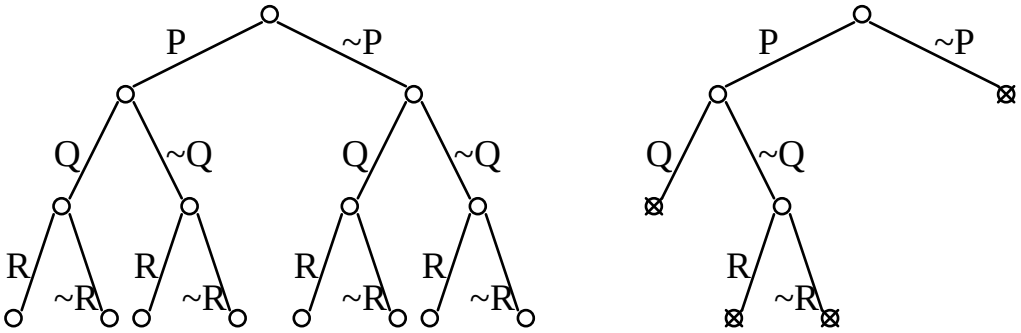
Definitie. Într-un arbore semantic, un nod N se numeste *nod de infirmare* daca I(N) falsifica o instanta de baza a unei clauze din S, dar I(N') nu falsifica nici o instanta de baza a unei clauze C din S, pentru orice nod N' predecesor al lui N.

Definitie. Un arbore semantic T se numeste *arbore semantic închis* daca si numai daca orice ramura a lui T se termina cu un nod de infirmare.

Definitie. Un nod N, într-un arbore semantic închis, se numeste *nod de inferenta* daca toti succesorii imediati ai lui N sînt noduri de infirmare.

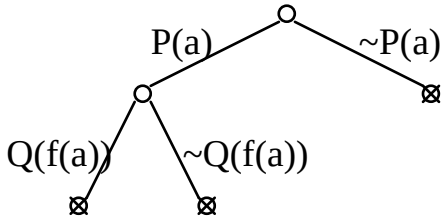
Exemple:

1. Fie multimea de clauze $S = \{P, Q \vee R, \sim P \vee \sim Q, \sim P \vee \sim R\}$, baza Herbrand a lui S $A = \{P, Q, R\}$ si urmatorii doi arbori semantici ai lui S:



   Primul arbore este un arbore semantic complet al lui S, iar cel de-al doilea este un arbore semantic închis al lui S.

2. Fie multimea de clauze $S = \{P(x), \sim P(x) \vee Q(f(x)), \sim Q(f(a))\}$ si baza Herbrand $A = \{P(a), Q(a), P(f(a)), Q(f(a)), P(f(f(a))), Q(f(f(a))), ...\}$. Un arbore semantic închis al lui S este



3.3 Teorema lui Herbrand

Ideea de baza a teoremei lui Herbrand este urmatoarea: pentru a testa daca o multime de clauze S este nerealizabila, trebuie sa se testeze daca S este nerealizabila numai pentru interpretari pe universul Herbrand al lui S. Daca S este falsa pentru toate interpretarile pe universul Herbrand, atunci se poate concluziona ca S este nerealizabila.

Exista doua versiuni ale teoremei lui Herbrand, prima bazata pe modelul arborilor semantici si a doua pornind de la baza Herbrand a unei multimi de clauze. A doua versiune a teoremei a stat la baza metodei de demonstrare prin respingere propusa de Gilmore, si a fost îmbunatatita ulterior de Davis si Putnam. Prima versiune a teoremei a fost cea care i-a inspirat lui Robinson rezolutia, metoda alternativa de demonstrare prin respingere. Din aceste motive, teorema lui Herbrand se considera a fi baza tuturor metodelor automate de demonstrare a teoremelor.

3.3.1 Prima formulare a teoremei lui Herbrand

Teorema. *Teorema lui Herbrand versiunea I.* O multime de clauze S este nerealizabila daca si numai daca pentru fiecare arbore semantic complet al lui S, exista un arbore semantic închis finit. Altfel spus, orice arbore semantic complet al lui S este arbore semantic închis finit.

Demonstratie.

($\Rightarrow$) Fie S nerealizabila si fie T un arbore semantic complet pentru S. Pentru fiecare cale B (radacina-frunza) a lui T, fie I_B multimea tuturor literalilor atasati legaturii caii B. Rezulta ca I_B este o interpretare a lui S. Deoarece S este nerealizabila, atunci I_B trebuie sa falsifice o instanta de baza C' a unei clauze $C \in S$. Dar, deoarece C' este finita, trebuie sa existe un nod de infirmare N, la distanta numar finit de arce de nodul radacina, pe calea B. Deoarece fiecare cale B a lui T are un nod de infirmare rezulta ca exista un arbore închis T' pentru S. În plus, deoarece fiecare nod are un numar finit de succesori, T' este finit, i.e. numarul de noduri din T' este finit, deoarece, în caz contrar, pe baza lemei lui Konig [Knuth,1968] s-ar putea gasi o cale infinita care nu contine nici un nod de infirmare.

($\Leftarrow$) Orice arbore semantic complet al lui S este un arbore închis finit. Rezulta ca fiecare cale în T contine un nod de infirmare, deci orice interpretare falsifica S. Rezulta ca S este nerealizabila.

3.3.2 A doua formulare a teoremei lui Herbrand

Teorema. *Teorema lui Herbrand versiunea II.* O multime de clauze S este nerealizabila daca si numai daca exista o multime finita S' de instante de baza ale clauzelor din S care este nerealizabila (sau baza Herbrand a lui S este nerealizabila).

Demonstratie.

($\Rightarrow$) Fie S nerealizabila si T un arbore semantic complet al lui S. Atunci, pe baza teoremei lui Herbrand versiunea I, rezulta ca exista un arbore semantic închis finit T', corespunzator lui T. Fie S' multimea tuturor instantelor de baza ale clauzelor care sînt falsificate de toate nodurile de infirmare din T'. S' este finita deoarece exista un numar finit de noduri de infirmare în T'. Deoarece S' este falsa în orice interpretare a lui S, S' este nerealizabila.

(⇐) Fie S' o multime finita de instante de baza ale clauzelor din S. Deoarece orice interpretare I a lui S contine o interpretare I' a lui S', daca I' falsifica S', atunci I falsifica, de asemenea, S'. Dar S' este falsificata de orice interpretare I'. În consecinta, S' este falsificata de orice interpretare I a lui S. Rezulta ca S este falsificata de orice interpretare, deci S este nerealizabila.

Exemple:

1. Fie multimea de clauze $S = \{P(x), \sim P(f(a))\}$. S este nerealizabila. Prin teorema lui Herbrand, exista o multime finita S' de instante de baza ale clauzelor din S. Una din aceste multimi este $S' = \{P(f(a)), \sim P(f(a))\}$.

2. Fie multimea de clauze $S = \{\sim P(x) \vee Q(f(x), x), P(g(b)), \sim Q(y, z)\}$. S este nerealizabila. $S' = \{\sim P(g(b)) \vee Q(f(g(b)), g(b)), P(g(b)), \sim Q(f(g(b), g(b))\}$ este una din multimile de clauze de baza, nerealizabila.

A doua versiune a teoremei lui Herbrand sugereaza o procedura de demonstrare prin respingere. Pentru a demonstra ca o multime S de clauze este nerealizabila, daca exista o metoda mecanica de generare a multimilor instantelor de baza din S: $S'_1, \ldots, S'_n, \ldots$, se poate testa succesiv realizabilitatea multimilor deduse $S'_1, S'_2, \ldots$. Conform teoremei lui Herbrand, daca S este nerealizabila, aceasta procedura poate detecta un N finit, astfel încît S'_N este nerealizabila.

Gilmore a fost cel care a propus si a implementat pentru prima oara aceasta metoda. Pentru a demonstra nerealizabilitatea unui set de clauze S, el a scris un program care a generat succesiv $S'_0, S'_1, \ldots$, unde S'_i este multimea de constante de baza obtinuta prin înlocuirea variabilelor din S prin constante din multimea constanta de nivel i, H_i, a lui S. Deoarece S'_i este o conjunctie de clauze de baza, orice metoda din logica propozitionala poate fi utilizata pentru a demonstra nerealizabilitatea lui S'_i. Gilmore a utilizat metoda multiplicarii, i.e. fiecare S'_i produs este multiplicat într-o forma normal disjunctiva. Orice conjunctie din forma normal disjunctiva care contine o pereche complementara este eliminata. Daca o multime S'_i devine vida, rezulta imediat ca S'_i este nerealizabila si deci s-a gasit o demonstratie a faptului ca S este nerealizabila.

Exemple:

1. Fie $S = \{P(x), \sim P(a)\}$, $H_0 = \{a\}$ si $S'_0 = P(a) \wedge \sim P(a) = \square$. Rezulta deci ca S este nerealizabila.

2. Fie $S = \{P(a), \sim P(x) \vee Q(f(x)), \sim Q(f(a))\}$, $H_0 = \{a\}$ si

   $S'_0 = P(a) \wedge (\sim P(a) \vee Q(f(a))) \wedge \sim Q(f(a))$ deci

   $S'_0 = (P(a) \wedge \sim P(a) \wedge \sim Q(f(a))) \vee (P(a) \wedge Q(f(a)) \wedge \sim Q(f(a))) = \square$ ### $\square = \square$. Rezulta deci ca S este nerealizabila.

Metoda utilizata de Gilmore este ineficienta din punct de vedere computational, deci greu de aplicat pentru multimi de clauze complexe. În 1960, Davis si Putnam au dezvoltat o metoda mai eficienta, o parte din regulile metodei fiind sursa de inspiratie a unor strategii de rezolutie care vor fi prezentate în Sectiunea 3.6.

prin constante din multimea constanta de nivel i, H_i, a lui S. Deoarece S'_i este o conjunctie de clauze de baza, orice metoda din logica propozitionala poate fi utilizata pentru a demonstra nerealizabilitatea lui S'_i. Gilmore a utilizat metoda multiplicarii, i.e. fiecare S'_i produs este multiplicat într-o forma normal disjunctiva. Orice conjunctie din forma normal disjunctiva care contine o pereche complementara este eliminata. Daca o multime S'_i devine vida, rezulta imediat ca S'_i este nerealizabila si deci s-a gasit o demonstratie a faptului ca S este nerealizabila.

Exemple:

1. Fie $S = \{P(x), \sim P(a)\}$, $H_0 = \{a\}$ si $S'_0 = P(a) \wedge \sim P(a) = \square$. Rezulta deci ca S este nerealizabila.

2. Fie $S = \{P(a), \sim P(x) \vee Q(f(x)), \sim Q(f(a))\}$, $H_0 = \{a\}$ si

   $S'_0 = P(a) \wedge (\sim P(a) \vee Q(f(a))) \wedge \sim Q(f(a))$ deci

   $S'_0 = (P(a) \wedge \sim P(a) \wedge \sim Q(f(a))) \vee (P(a) \wedge Q(f(a)) \wedge \sim Q(f(a))) = \square \vee \square = \square$. Rezulta deci ca S este nerealizabila.

Metoda utilizata de Gilmore este ineficienta din punct de vedere computational, deci greu de aplicat pentru multimi de clauze complexe. În 1960, Davis si Putnam au dezvoltat o metoda mai eficienta, o parte din regulile metodei fiind sursa de inspiratie a unor strategii de rezolutie care vor fi prezentate în Sectiunea 3.6.

3.3.3 Metoda lui Davis si Putnam

Metoda multiplicarii utilizata de Gilmore este ineficienta. De exemplu, se poate vedea cu usurinta ca pentru o multime de zece clauze de baza, fiecare clauza avînd doi literali, exista 2^{10} conjunctii. Pentru a reduce numarul de clauze generate, Davis si Putnam au dezvoltat o metoda mai eficienta pentru a testa nerealizabilitatea unei multimi de clauze de baza. Aceasta metoda se bazeaza pe aplicarea a patru reguli de transformare a clauzelor, reguli care pastreaza proprietatea de nerealizabilitate a multimii de clauze. Se poate demonstra ca regulile de transformare ale lui Davis si Putnam sînt consistente, cu alte cuvinte multimea de clauze S' obtinuta din multimea de clauze S prin aplicarea acestor reguli este nerealizabila daca si numai daca S este nerealizabila. Demonstratia este lasata pe seama cititorului.

Fie S o multime de clauze de baza. Metoda lui Davis si Putnam consta în aplicarea urmatoarelor patru reguli:

I. Regula tautologiei.

Daca se elimina din S toate clauzele de baza care sînt tautologii, atunci multimea ramasa S' este nerealizabila daca si numai daca S este nerealizabila.

II. Regula literalului mic.

Fie L o clauza de baza unitara în S. Daca S' se obtine prin eliminarea din S a acelor clauze de baza care contin clauza L, atunci, daca S' este vida, S este realizabila. În caz contrar, daca S'' se obtine prin eliminarea lui ~L din S', atunci S'' este nerealizabila daca si numai daca S este nerealizabila. Trebuie remarcat ca daca ~L este o clauza de baza unitara, atunci eliminarea lui ~L produce clauza vida.

III. Regula literalului pur.

Un literal L dintr-o clauza de baza din S se spune o fi *pur* în S daca ~L nu apare în nici o clauza de baza în S. Fie L un literal pur în S. Daca S' se obtine prin eliminarea din S a tuturor clauzelor de baza care contin literalul L, atunci multimea ramasa S' este nerealizabila daca si numai daca S este nerealizabila.

IV. Regula scindarii.

Daca multimea S poate fi pusa sub forma:

$$(A_1 \vee L) \wedge ... \wedge (A_m \vee L) \wedge (B_1 \vee \sim L) \wedge ... \wedge (B_n \vee \sim L) \wedge R,$$

unde A_i, B_i si R nu contin L si ~L, atunci se construiesc multimile $S_1 = A_1 \wedge ... \wedge A_m \wedge R$ si $S_2 = B_1 \wedge ... \wedge B_n \wedge R$. Multimea S este nerealizabila daca si numai daca $(S_1 \vee S_2)$ este nerealizabila, adica atît S_1 cît si S_2 sînt nerealizabile.

Aceste reguli sînt importante deoarece reprezinta o implementare a teoremei lui Herbrand. În acelasi timp regulile de mai sus sînt importante datorita generalizarii lor în logica cu predicate de ordinul I si utilizarea lor în diverse strategii rezolutive, asa cum se va prezenta în Sectiunile 3.5 si 3.6.

Exemple:

1. Sa se arate ca $S = (P \vee Q \vee \sim R) \wedge (P \vee \sim Q) \wedge \sim P \wedge R \wedge U$ este nerealizabila.

   (1) $(P \vee Q \vee \sim R) \wedge (P \vee \sim Q) \wedge \sim P \wedge R \wedge U$

   (2) $(Q \vee \sim R) \wedge (\sim Q) \wedge R \wedge U$ Regula II pentru ~P

   (3) $\sim R \wedge R \wedge U$ Regula II pentru ~Q

   (4) $\square \wedge U$ Regula II pentru ~R.

   Deoarece ultima formula contine clauza vida, S este nerealizabila.

2. Sa se arate ca $S = (P \vee Q) \wedge \sim Q \wedge (\sim P \vee Q \vee \sim R)$ este realizabila.

(1) $(P \vee Q) \wedge \sim Q \wedge (\sim P \vee Q \vee \sim R)$

(2) $P \wedge (\sim P \vee \sim R)$ Regula II pentru ~Q

(3) ~R Regula II pentru P

(4) □ Regula II pentru ~R.

Ultima multime este multimea vida, deci S este realizabila.

3. Sa se arate ca $S = (P \vee \sim Q) \wedge (\sim P \vee Q) \wedge (Q \vee \sim R) \wedge (\sim Q \vee \sim R)$ este realizabila.

(1) $(P \vee \sim Q) \wedge (\sim P \vee Q) \wedge (Q \vee \sim R) \wedge (\sim Q \vee \sim R)$

(2) $(\sim Q \wedge (Q \vee \sim R) \wedge (\sim Q \vee \sim R)) \vee (Q \wedge (Q \vee \sim R) \wedge (\sim Q \vee \sim R))$

Regula IV pentru P

(3) $\sim R \vee \sim R$ Regula II pentru ~Q si Q

(4) □ ∨ □ Regula II pentru ~R.

Deoarece ambele multimi scindate sînt realizabile, S este realizabila.

3.4 Principiul rezolutiei

Utilizarea teoremei lui Herbrand ca metoda de demonstrare prin respingere are un dezavantaj esential: necesita generarea multimilor instantelor de baza $S'_0, S'_1, ...$, a clauzelor din S, multimea de clauze care trebuie dovedita a fi nerealizabila. Deoarece, de cele mai multe ori, dimensiunea lui S'_i creste exponential, metodele de demonstrare a teoremelor derivate direct din teorema lui Herbrand sînt complex computationale si aproape impracticabile pentru dimensiuni chiar medii ale intrarii.

În 1965, Robinson propune principiul rezolutiei, care reprezinta baza tuturor actualelor demonstratoare automate de teoreme. Rezolutia este o regula de inferenta care permite obtinerea de noi clauze din multimea initiala de clauze S. Principiul rezolutiei poate fi aplicat pe orice multime de clauze S, nu neaparat pe o multime de instante de baza a clauzelor din S, pentru a demonstra nerealizabilitatea lui S.

Ideea esentiala a principiului rezolutiei este aceea de a verifica daca S contine clauza vida sau daca clauza vida poate fi derivata din S. În acest caz S este o multime de clauze nerealizabila. În Sectiunea 3.4.4 se va arata ca, pe baza teoremei lui Herbrand, versiunea I, verificarea apartenentei sau derivarii clauzei vide din S este echivalenta cu determinarea numarului de noduri ale unui arbore semantic închis al lui S.

Principiul rezolutiei este tot o metoda de demonstrare prin respingere, care corespunde în general unei demonstrari prin reducere la absurd. De aceea, utilizarea principiului rezolutiei în demonstrarea teoremelor se mai numeste si *metoda respingerii prin rezolutie* sau *respingere rezolutiva.* În aceasta sectiune se va prezenta principiul rezolutiei în calculul propozitional, extinderea lui în logica cu predicate de ordinul I prin unificarea literalilor si justificarea formala a corectitudinii acestui principiu prin punerea lui în corespondenta cu teorema lui Herbrand. În sectiunile urmatoare se vor prezenta diverse strategii de aplicare a acestei unice reguli de inferenta, rezolutia, pentru a ajunge cît mai repede la derivarea clauzei vide.

3.4.1 Rezolutia în logica propozitionala

Principiul rezolutiei în logica propozitionala este urmatorul. Pentru orice doua clauze C_1 si C_2, daca exista un literal L_1 în C_1 care este complementar cu un literal L_2 în C_2 ($L_1 =\sim L_2$) atunci disjunctia între C_1 din care s-a eliminat L_1 si C_2 din care s-a eliminat L_2 este *rezolventul* clauzelor C_1 si C_2.

Definitie. Fie clauzele:

(C_1) $P_1 \vee P_2 \vee ... \vee P_i \vee ... \vee P_n$

(C_2) $Q_1 \vee Q_2 \vee ... \vee \sim Q_j \vee ... \vee Q_m$

cu $P_i = Q_j = L$. *Rezolventul* clauzelor C_1 si C_2 este $C = rez(C_1, C_2) = (C_1 - \{L\}) \vee (C_2 - \{\sim L\})$ deci

(C) $P_1 \vee P_2 \vee ... \vee P_{i-1} \vee P_{i+1} \vee ... \vee P_n \vee Q_1 \vee ... \vee Q_{j-1} \vee Q_{j+1} \vee ... \vee Q_m$

Teorema. Fiind date doua clauze, C_1 si C_2, un rezolvent C al clauzelor C_1 si C_2 este o consecinta logica a clauzelor C_1 si C_2.

Pentru a demonstra ca S este o teorema derivata dintr-un set de axiome A utilizînd principiul rezolutiei, se aplica algoritmul prezentat în continuare. O prezentare mai detaliata si exemple pot fi gasite în Chang si Lee [1976] si Florea [1993].

Algoritm: Respingerea prin rezolutie în logica propozitionala

1. Converteste setul de axiome A în forma clauzala si obtine multimea de clauze S_0
2. Neaga teorema, transforma teorema negata în forma clauzala si adauga rezultatul la S_0
   $S \leftarrow S_0$
3. **repeta**
   - 3.1. Selecteaza o pereche de clauze C_1 si C_2 din S
   - 3.2. Determina $C = rez(C_1, C_2)$
   - 3.3. **daca** $C \neq \square$

atunci $S \leftarrow S \cup \{C\}$

pîna s-a obtinut clauza vida **sau**

nu mai exista nici o pereche de clauze care rezolva

4. **daca** s-a obtinut clauza vida

atunci teorema este adevarata (este demonstrata)

5. **altfel** teorema este falsa

sfîrsit.

3.4.2 Substitutie si unificare

Pentru a putea extinde principiul rezolutiei în logica cu predicate de ordinul I, trebuie gasit un mecanism de identificare a literalilor complementari. Acest mecanism este dat de unificarea termenilor cu ajutorul unor transformari numite substitutii.

Definitie. O *substitutie* este o multime finita, posibil vida, de perechi de forma t_i / v_i, $i = 0, n$, unde v_i este un termen variabil (variabila) iar t_i este un termen, cu urmatoarele restrictii:

(1) O variabila apare într-o singura pereche a substitutiei ($v_i \neq v_j \ \ \forall i, j \ \ i \neq j, \ i, j = 1, n$)

(2) Nici una din variabile nu apare în termenii substitutiei (v_i nu apare în $t_j \ \forall i,j \ \ i,j=1,n$)

O substitutie se noteaza $\alpha = \{t_1 / v_1, t_2 / v_2, ..., t_n / v_n\}, \ n \geq 1$.

În continuare se vor folosi litere grecesti pentru reprezentarea substitutiilor.

Rezultatul aplicarii unei substitutii α asupra unei expresii E este notat $E\alpha$ si este expresia obtinuta prin înlocuirea tuturor aparitiilor variabilei v_i cu termenul t_i în expresia E, pentru toate perechile t_i/v_i din substitutia α. Se reaminteste ca o expresie poate fi un termen, un literal, un atom sau o formula bine formata sau o multime de termeni, literali, atomi sau formule bine formate.

Exemplu. Fie expresia $E = P(f(x, g(x, b, y), h(z)))$ si substitutia $\alpha = \{a / x, f(a, b, c) / y, c / z\}$. Expresia obtinuta prin aplicarea substitutiei este $E\alpha = P(f(a, g(a, b, f(a, b, c)), h(c)))$.

Definitie. Expresia E_1 este o *instanta* a expresiei E_2 daca exista o substitutie α astfel încît $E_1 = E_2 \ \alpha$.

Definitie. O expresie E este o *instanta comuna* a doua expresii E_1 si E_2 daca exista substitutiile α_1 si α_2 astfel încît $E = E_1 \ \alpha_1 = E_2 \ \alpha_2$.

Se observa ca definitia instantei unei expresii este compatibila cu definitia instantei de baza a unei clauze data în Sectiunea 3.2.1 pentru cazul în care expresia este o clauza.

Exemplu. Expresia $E = P(f(a,b(a,w),b,g(u,c)))$ este o instanta comuna a expresiilor $E_1 = P(f(x,b(x,y),z,g(u,v)))$ si $E_2 = P(f(a,b(x',w),b,v'))$, substitutiile care realizeaza aceasta fiind $\alpha_1 = \{a/x, w/y, b/z, c/v\}$ si $\alpha_2 = \{a/x', g(u,c)/v'\}$.

Definitie. O expresie E este *mai generala* decît expresia E' daca E' este o instanta a lui E dar E nu este o instanta a lui E'. Cu alte cuvinte exista o substitutie α astfel încît $E' = E\alpha$, dar nu exista nici o substitutie α' astfel încît $E = E'\alpha'$.

Exemple:

1. Pentru termenii $t = x$ si $t' = a$ exista $\alpha = \{a/x\}$ astfel încît $t' = t\alpha = a$, dar nu exista α' astfel încît $t = t'\alpha'$ din cauza faptului ca t' nu contine variabile.
2. Analog pentru expresiile $E = P(x)$ si $E' = P(f(y))$ exista $\alpha = \{f(y)/x\}$ astfel încît $E' = E\alpha = f(y)$, dar nu exista α' astfel încît $E = E'\alpha'$ deoarece f(y) este un termen functional.

Definitie. Se numeste *unificator* al unei multimi de expresii $\{E_1, E_2, ..., E_n\}$, o substitutie α care face ca expresiile sa devina identice, adica $E_1\alpha = E_2\alpha = ... = E_n\alpha$. Multimea $\{E_1, E_2, ..., E_n\}$ se numeste *multime de expresii unificabila*, daca exista un unificator pentru aceasta multime. Se mai spune ca multimea de expresii *unifica*.

Exista o relatie evidenta între unificatori si instantele comune. Un unificator determina întotdeauna o instanta comuna si invers, o instanta comuna determina un unificator.

Definitie. *Cel mai general unificator* al unei multimi de expresii, cu prescurtarea din limba engleza mgu si notat β, este un unificator astfel încît instanta comuna asociata este cea mai generala.

Se poate da si urmatoarea definitie alternativa.

Definitie. Un unificator β al unei multimi de expresii $\{E_1, E_2, ..., E_n\}$ este cel mai general unificator daca si numai daca pentru orice alt unificator α al multimii exista o substitutie α' astfel încît $E_i\alpha = E_i\beta\alpha'$, $\forall i = 1,n$. Altfel spus, orice unificator α al multimii $\{E_1, E_2, ..., E_n\}$ este o instanta a lui β.

Exemplu. Fie literalii $L_1 = P(x,b,f(y))$ si $L_2 = P(a,u,f(z))$. Un unificator al multimii $\{L_1, L_2\}$ este $\alpha = \{a/x, b/u, c/y, c/z\}$, iar cel mai general unificator al multimii $\{L_1, L_2\}$ este $\beta = \{a/x, b/u, z/y\}$.

Observatie. Daca doua expresii unifica, atunci exista un *unic* cel mai general unificator.

Aplicarea principiului rezolutiei în logica cu predicate de ordinul I necesita identificarea a doi literali complementari care unifica si determinarea celui mai general unificator (mgu) al

acestora. Conform definitiei unificarii expresiilor si a celui mai general unificator, doi literali unifica daca simbolurile predicative ale literalilor sînt identice, daca cele doua predicate au acelasi numar de argumente (termeni) si daca cele doua expresii formate cu argumentele fiecarui predicat unifica. Daca aceste doua expresii admit un mgu β, atunci aplicarea lui β asupra celor doi literali îi face complementar identici, deci ei vor putea rezolva în rezolutie.

Algoritm: Unificarea literalilor

1. Fie $L_1 = P_1(t_{11}, t_{12}, ..., t_{1n})$ si $L_2 = P_2(t_{21}, t_{22}, ..., t_{2m})$
2. **daca** $P_1 \neq P_2$
   atunci întoarce INSUCCES
3. **daca** $n \neq m$
   atunci întoarce INSUCCES
4. $MGU \leftarrow \{\}$
5. **pentru** fiecare t_{1i}, t_{2i}, $i = 1, n$ **executa**
   5.1. $M \leftarrow$ Unificã(t_{1i}, t_{2i}, MGU)
   5.2. **daca** M = INSUCCES
   atunci întoarce INSUCCES
   5.3. Aplica substitutia M asupra termenilor t_{1j}, $j = i, n$ si t_{2j}, $j = i, n$
   5.4. $MGU \leftarrow MGU \cup M$
6. **întoarce** SUCCES

sfîrsit.

Pasul 5.1 al algoritmului construieste cel mai general unificator a doi termeni, daca acesta exista, si îl întoarce ca rezultat sau raporteaza insucces daca termenii nu unifica. Algoritmul se bazeaza pe definitia termenilor si a substitutiei luînd în considerare urmatoarele cazuri:

1. doi termeni constanti identici unifica întotdeauna cu mgu = ϕ.
2. doi termeni, unul variabil v si unul constant c, unifica întotdeauna cu mgu = $\{c / v\}$.
3. doi termeni variabili identici unifica întotdeauna cu mgu = ϕ.
4. doi termeni variabili diferiti v si v', unifica întotdeauna cu mgu = $\{v / v'\}$.
5. doi termeni, unul variabil v si unul functional $f(t_1, ..., t_n)$ unifica daca v nu se afla printre termenii $t_1, ..., t_n$ si nici printre subtermenii acestora la orice nivel, rezultînd mgu = $\{f(t_1, ..., t_n) / v\}$.

6. doi termeni functionali avînd acelasi simbol functional f si aceeasi aritate n, unifica daca toti subtermenii lor de acelasi rang unifica rezultînd $mgu = mgu_1 \cup ... \cup mgu_n$, unde mgu_i este unificatorul subtermenilor de rang i al celor doi termeni functionali.

Tinînd cont de prima restrictie din definitia substitutiei, precautii suplimentare trebuie luate în cazurile 2, 4 si 5, în sensul ca daca termenul variabil v apare deja în substitutie (exista o pereche t/v), unificarea trebuie facuta între termenul pereche t al termenului variabil v si respectiv termenul constant c, termenul variabil v' sau termenul functional $f(t_1,...,t_n)$. Orice alta combinatie, în afara celor 6 enumerate mai sus, duce la esecul unificarii termenilor.

Algoritm: Unificarea termenilor

Intrare: Doi termeni si un unificator partial

Iesire: Unificatorul celor doi termeni sau INSUCCES

Unifica (TA, TB, MGUP)

1. **daca** TA = TB /* cazurile 1 si 3 */
   atunci întoarce MGUP
2. **daca** TA este termen variabil
   atunci întoarce UnificaVariabila (TA, TB, MGUP)
3. **daca** TB este termen variabil
   atunci întoarce UnificaVariabila (TB, TA, MGUP)
4. Identifica $TA = f_A(t_{A1},...,t_{An})$ si $TB = f_B(t_{B1},...,t_{Bm})$
5. **daca** $f_A \neq f_B$ **sau** $n \neq m$
   atunci întoarce INSUCCES
6. **altfel** $MGUP_i \leftarrow MGUP$
7. **pentru** fiecare pereche t_{Ai}, t_{Bi} **executa**
   7.1. $MGUP_i \leftarrow$ Unificã $(t_{Ai}, t_{Bi}, MGUP_i)$
   7.2. **daca** $MGUP_i$ = INSUCCES
   atunci întoarce INSUCCES
8. **întoarce** $MGUP_i$

sfîrsit.

UnificaVariabila (V, T, MGUP)

1. **daca** exista o pereche T'/V în MGUP
   atunci întoarce Unificã(T', T, MGUP)
2. **daca** T este un termen variabil **si** exista o pereche T'/T în MGUP
   atunci întoarce Unifica (V, T', MGUP)

3. **daca** variabila V nu apare în termenul T
   atunci întoarce concat (MGUP, {T / V})
4. **întoarce** INSUCCES

sfîrsit.

Observatii:

- Caracterul recursiv al algoritmului este determinat de definitia recursiva a termenilor.
- Unificatorul este construit în variabila MGUP, unificatorul partial.

3.4.3 Implementarea algoritmului de unificare a termenilor

În continuare este prezentata o solutie posibila de implementre în limbajul Lisp a algoritmului de unificare a termenilor. În cadrul programului termenii constanti vor fi reprezentati prin atomi Lisp, iar termenii functionali vor fi reprezentati prin liste în care primul element va fi numele functiei, reprezentat printr-un atom Lisp, restul elementelor fiind la rîndul lor termeni. Reprezentarea interna a termenilor variabili se face printr-o lista continînd doua elemente (*VAR * < nume – variabilã >) în care < nume – variabilã > este, de asemenea, un atom Lisp. Reprezentarea externa a termenilor variabili va fi ? < nume – variabilã >. Pentru aceasta se va defini macrocaracterul ?, astfel încît acesta sa transforme reprezentarea externa a termenilor variabili în reprezentarea lor interna. Substitutiile vor fi reprezentate prin liste de perechi cu punct (variabilã.valoare), structura variabilei fiind cea descrisa mai sus.

```
;; se defineste macrocaracterul ? astfel incit ?<simbol> se
;; transforma intr-o lista (*var* <simbol>)
(set-macro-character #\?
  #'(lambda (stream char)
      (list '*var* (read stream t nil t))))

(defun variabila-p (var)
  "Intoarce t daca argumentul este o variabila si nil in caz contrar."
  (and (listp var) (eq (first var) '*var*)))

(defun extrage-substitutie (variabila mgu)
  "Intoarce substitutia variabilei in unificatorul mgu."
  (cdr (assoc variabila mgu :test #'equal)))

(defun adauga-substitutie (variabila termen mgu)
  "Adauga o substitutie variabilei in unificatorul mgu."
  (cons (cons variabila termen) mgu))
```

```
(defun apare-in-p (variabila termen mgu)
  "Intoarce t daca termenul contine variabila."
  (cond ((atom termen) nil)                    ;; termen constant deci nu contine variabila
        ((variabila-p termen)
         (or (equal variabila termen)
             (apare-in-p variabila (extrage-substitutie termen mgu) mgu)))
        (t (or (apare-in-p variabila (first termen) mgu)
               (apare-in-p variabila (rest termen) mgu)))))

(defun unifica (TA TB &optional mgu)
  "Verifica daca termenii TA si TB unifica si construieste cel mai general unificator mgu."
  (cond ((variabila-p TA)
         (unifica-variabila TA TB mgu))
        ((variabila-p TB)
         (unifica-variabila TB TA mgu))
        ((atom TA)
         (when (eq TA TB)                      ;; constante identice
           (values t mgu)))                    ;; nu se adauga o substitutie noua
        ((atom TB) nil)                        ;; termenii nu unifica
        ((and (eq (first TA) (first TB))       ;; termeni functionali
              (= (length TA) (length TB)))     ;; au acelasi functor si aritate?
         (let (succes)
           (when (every #'(lambda (TAi TBi)
                            (multiple-value-setq
                              (succes mgu)
                              (unifica TAi TBi mgu)))
                        (rest TA) (rest TB))
             (values t mgu))))))

(defun unifica-variabila (var term mgu)
  "Unifica variabila cu termenul term in prezenta unificatorului partial mgu."
  (if (equal var term)                         ;; variabila unifica cu ea insasi?
    (values t mgu)                             ;; daca da, nu adauga substitutia
    (let ((S (extrage-substitutie var mgu)))
      (cond (S  (unifica S term mgu))          ;; variabila are deja o substitutie
            ((not (apare-in-p var term mgu))
             (values t (adauga-substitutie var term mgu)))))))

(defun substituie-variabile (termen mgu)
```

```
"Intoarce termenul in care variabilele au fost substituite."
(cond ((atom termen) termen)                    ;; termen constant
      ((variabila-p termen)                     ;; termen variabil
       (let ((S (extrage-substitutie termen mgu)))
        (if S                                   ;; daca are o substitutie,
          (substituie-variabile S mgu)          ;; verifica substitutia
          termen)))                             ;; altfel intoarce termenul
      (t (cons (substituie-variabile (first termen) mgu)
               (substituie-variabile (rest termen) mgu)))))
```

3.4.4 Rezolutia în logica cu predicate de ordinul I

Aplicarea principiului rezolutiei în logica cu predicate de ordinul I implica construirea rezolventului a doi literali complementari, care fie sînt identici, fie au fost facuti identici prin aplicarea substitutiei mgu asupra ambelor clauze ce contin acesti doi literali.

Definitie. Daca doi sau mai multi literali, de acelasi semn, ai unei clauze C au un mgu β, atunci clauza $C\beta$ este numita *factor* al clauzei C. Daca $C\beta$ este o clauza unitara atunci este numita *factor unitar*.

Exemplu. Pentru clauza $C = P(x) \vee P(f(y)) \vee \sim Q(x)$, literalii $P(x)$ si $P(f(y))$ au $mgu = \beta = \{f(y)/x\}$ si $C\beta = P(f(y)) \vee \sim Q(f(y))$ este un factor al lui C.

Definitie. Fie clauzele:

(C_1) $P_1 \vee P_2 \vee ... \vee P_i \vee ... \vee P_n$

(C_2) $Q_1 \vee Q_2 \vee ... \vee \sim Q_j \vee ... \vee Q_m$

numite clauze parinte si β cel mai general unificator al lui P_i si Q_j, $P_i\beta = Q_j\beta$, atunci $C = rez(C_1, C_2) = (C_1\beta - \{P_i\beta\}) \cup (C_2\beta - \{\sim Q_j\beta\})$ este un *rezolvent binar* al clauzelor C_1 si C_2.

Exemplu. Fie clauzele $C_1 = P(x, f(y)) \vee Q(x)$ si $C_2 = \sim P(a, f(z)) \vee R(z)$ si cel mai general unificator $\beta = \{a/x, y/z\}$, atunci rezolventul binar al clauzelor C_1 si C_2 este clauza $C = rez(C_1, C_2) = (C_1\beta - \{P(x, f(y))\beta\}) \cup (C_2\beta - \{P(a, f(z))\beta\}) = Q(a) \vee R(z)$.

Definitie. Un *rezolvent* al perechii de clauze C_1, C_2 este unul dintre urmatorii posibili rezolventi:

(1) un rezolvent binar al clauzelor C_1 si C_2

(2) un rezolvent binar al clauzei C_1 si al unui factor al clauzei C_2

(3) un rezolvent binar al unui factor al clauzei C_1 si al clauzei C_2

(4) un rezolvent binar al unui factor al clauzei C_1 si al unui factor al clauzei C_2.

Exemplu. Fie clauzele $C_1 = P(x) \vee P(f(y)) \vee R(g(y))$ si $C_2 = \sim P(f(g(a))) \vee Q(b)$. Un factor al lui C_1 este $C_1' = P(f(y)) \vee R(g(y))$. Un rezolvent binar al lui C_1' si C_2 este clauza $R(g(g(a))) \vee Q(b)$. Deci $R(g(g(a))) \vee Q(b)$ este în acelasi timp un rezolvent al clauzelor C_1 si C_2.

Pentru a demonstra, prin metoda respingerii prin rezolutie, ca S este o teorema derivata dintr-un set de axiome A în logica cu predicate de ordinul I, se aplica algoritmul prezentat în continuare.

Algoritm: Respingerea prin rezolutie în logica cu predicate de ordinul I

1. Converteste setul de axiome A în forma clauzala si obtine multimea de clauze S_0
2. Neaga teorema de demonstrat, transforma teorema negata în forma clauzala si adauga rezultatul obtinut la S_0

   $S \leftarrow S_0$
3. **repeta**
   - 3.1. Selecteaza o pereche de clauze C_1 si C_2 din S
   - 3.2. Fie $L_1 \in C_1$ si $\sim L_2 \in C_2$
   - 3.3. Aplica unificarea si calculeaza $\beta = mgu(L_1, L_2)$
   - 3.4. **daca** $\beta \neq \phi$

     atunci

     3.4.1. Determina $C = rez(C_1\beta, C_2\beta)$

     3.4.2. **daca** $C \neq \square$

     atunci $S \leftarrow S \cup \{C\}$

   pîna s-a obtinut clauza vida **sau**

   nu mai exista nici o pereche de clauze care rezolva **sau**

   o cantitate predefinita de efort a fost epuizata
4. **daca** s-a obtinut clauza vida

   atunci teorema este adevarata (este demonstrata)
5. **altfel**
   - 5.1. **daca** nu mai exista nici o pereche de clauze care rezolva

     atunci teorema este falsa
   - 5.2. **altfel** nu se poate spune nimic despre adevarul teoremei

sfîrsit.

Observatii:

- Se observa ca algoritmul respingerii prin rezolutie în logica cu predicate de ordinul I, ca si cel din calculul propozitional de altfel, contine o etapa nedeterminista, pasul 3.1. În acest pas al algoritmului nu se spune nimic despre modul în care trebuie selectate cele doua clauze care rezolva. Este rolul strategiei rezolutive de a transforma acest pas într-un pas determinist. În cazul în care exista mai multi literali care pot rezolva în cele doua clauze, si pasul 3.2 este nedeterminist. O discutie detaliata a strategiilor rezolutive va fi facuta în Sectiunile 3.5, 3.6 si 3.7.

- În cazul în care s-a obtinut clauza vida, metoda respingerii prin rezolutie garanteaza faptul ca teorema este adevarata, deci este demonstrabila pe baza setului de axiome A.

- Reciproc, daca teorema este adevarata, se poate obtine clauza vida dupa un numar finit de executii ale pasului 3, cu conditia ca strategia de rezolutie sa fie completa. Sectiunile urmatoare vor discuta completitudinea diverselor strategii de rezolutie.

- Cele doua observatii anterioare se refera de fapt la completitudinea principiului rezolutiei, problema care este prezentata în Sectiunea 3.4.4.

- Conditia de oprire a ciclului, "o cantitate predefinita de efort a fost epuizata", absenta în cazul algoritmului respingerii prin rezolutie în calculul cu propozitii, a fost introdusa în acest caz deoarece metoda demonstrarii teoremelor prin respingere rezolutiva este semidecidabila în logica cu predicate de ordinul I. În cazul în care concluzia T de demonstrat este falsa, deci nu este teorema, este posibil sa se ajunga în situatia în care, daca avem noroc, "nu mai exista nici o pereche de clauze care rezolva". Atunci se poate concluziona ca teorema este falsa. Dar este de asemenea posibil ca pasul 3 sa se execute la infinit daca T nu este teorema. Din acest motiv se introduce o cantitate predefinita de efort (resurse de timp sau spatiu) la epuizarea careia algoritmul se opreste. În acest caz s-ar putea ca teorema sa fie adevarata dar efortul predefinit impus sa fie prea mic, sau se poate ca T sa nu fie teorema. Rezulta deci ca nu se poate spune nimic despre adevarul teoremei.

Exemplu. Fie urmatoarea multime de formule:

A1. $(\forall x)\ (C(x) \rightarrow (W(x) \wedge R(x)))$

A2. $(\exists x)\ (C(x) \wedge Q(x))$

T. $(\exists x)\ (Q(x) \wedge R(x))$

Sa se arate ca T este concluzia multimii de axiome $A = \{A_1, A_2\}$. Pentru aceasta se transforma A în forma clauzala, se neaga T si se adauga rezultatul la A, obtinînd multimea de clauze S:

C1. $\sim C(x) \vee W(x)$

C2. $\sim C(x) \vee R(x)$

C3. $C(a)$

C4. $Q(a)$

C5. $\sim Q(x) \vee \sim R(x)$

C1 si C2 au fost obtinute din prima teorema, C3 si C4 din a doua, iar C5 a fost obtinuta din concluzia T negata.

Pentru a demonstra T trebuie sa se aplice metoda rezolutiei si sa se obtina clauza vida din S. Aplicînd rezolutia se obtine:

C6. $R(a)$ rezolvent al perechii C2 si C3 cu substitutia $\{a / x\}$

C7. $\sim R(a)$ rezolvent al perechii C4 si C5 cu substitutia $\{a / x\}$

C8. $\square$ rezolvent al perechii C6 si C7.

Deci T este o teorema adevarata, demonstrabila pe baza axiomelor A_1 si A_2.

3.4.5 Completitudinea principiului rezolutiei

În Sectiunea 3.2.3 s-a prezentat notiunea de arbore semantic care a fost folosita ulterior pentru demonstrarea teoremei lui Herbrand. În aceasta sectiune se vor folosi arborii semantici pentru a demonstra completitudinea principiului rezolutiei.

Între arborii semantici si respingerea prin rezolutie exista o legatura strînsa, asa cum se poate observa din urmatorul exemplu.

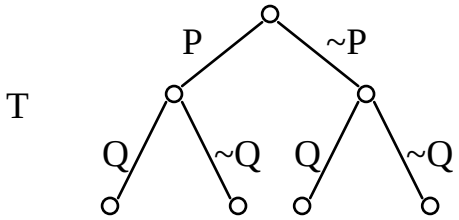
Exemplu. Fie setul de clauze S cuprinzînd:

(1) P

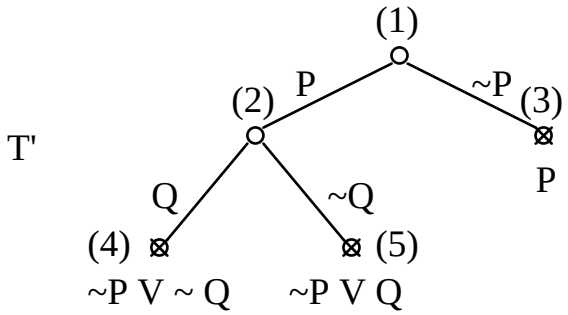
(2) $\sim P \vee Q$

(3) $\sim P \vee \sim Q$

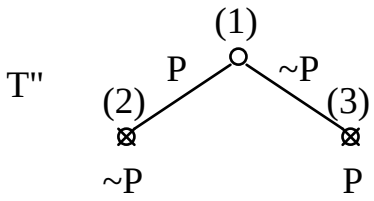
Baza Herbrand a lui S este $A = \{P, Q\}$. Un arbore semantic complet T este:



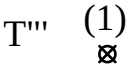
T are un arbore semantic închis T':



În arborele T', nodul (2) este un nod de inferenta deoarece toate nodurile succesoare sînt noduri de infirmare. Clauza falsificata în nodul (4) este $\sim P \vee \sim Q$, iar $\sim P \vee Q$ este clauza falsificata în nodul (5). Dupa cum se poate vedea, aceste doua clauze au o pereche complementara de literali, deci pot rezolva. Rezolventul este ~P. Se observa ca ~P este falsificat de interpretarea partiala corespunzatoare nodului (2). Daca se adauga ~P în S, atunci pentru $S \cup \{\sim P\}$ se obtine arborele semantic închis T'':



În arborele T'', nodul (1) este un nod de inferenta. De data aceasta se poate obtine clauza vida, prin rezolvarea lui P si ~P. Adaugînd clauza vida în $S \cup \{\sim P\}$, se obtine arborele semantic închis T''' pentru $S \cup \{\sim P\} \cup \{\square\}$.



Observatie. Orice set de clauze S care contine clauza vida are un arbore semantic închis redus la un singur nod, nod de infirmare.

Procesul de reducere a arborelui semantic prezentat mai sus corespunde de fapt urmatoarei demonstratii prin metoda respingerii prin rezolutie:

(4) ~P rezolvent al clauzelor (2) si (3)

(5) $\square$ rezolvent al clauzelor (4) si (1).

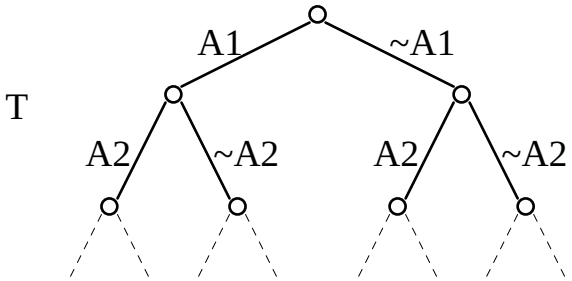
Aceste observatii vor fi utilizate pentru a demonstra completitudinea principiului rezolutiei. Se construieste un arbore semantic pentru o multime de clauze nerealizabile, apoi, treptat, se poate forta reducerea arborelui, în acelasi timp cu obtinerea demonstratiei prin rezolutie.

Lema. *Lema ascensiunii*. Daca clauzele C_1' si C_2' sînt instante ale clauzelor C_1 si C_2 si $C' = rez(C_1', C_2')$, atunci exista clauza $C = rez(C_1, C_2)$ astfel încît C' este o instanta a lui C.

Teorema.*Completitudinea principiului rezolutiei*. O multime de clauze S este nerealizabila daca si numai daca exista o deductie a clauzei vide din S.

Demonstratie.

($\Rightarrow$) Fie S nerealizabila si $A = \{A_1, A_2, ...\}$ baza Herbrand a lui S. Fie T un arbore semantic complet:



Conform teoremei lui Herbrand, versiunea I, T are un arbore semantic închis T'.

Daca T' este numai nodul radacina, atunci clauza vida trebuie sa fie în S, deoarece nici o alta clauza nu poate fi falsificata la radacina arborelui semantic. În acest caz teorema este evident adevarata.

Sa presupunem ca T' are mai mult de un nod. Atunci T' are cel putin un nod de inferenta. Daca T' nu ar avea un nod de inferenta, atunci orice nod ar avea cel putin un nod care nu este nod de infirmare, deci s-ar putea gasi o cale infinita în T', infirmînd astfel faptul ca T' este un arbore semantic închis.

Fie N un nod de inferenta în T'. Fie N_1 si N_2 noduri de infirmare descendenti imediati ai lui N si fie interpretarile $I(N) = \{m_1, m_2, ..., m_n\}$, $I(N_1) = \{m_1, m_2, ..., m_n, m_{n+1}\}$ si $I(N_2) = \{m_1, m_2, ..., m_n, \sim m_{n+1}\}$. Deoarece N_1 si N_2 sînt noduri de infirmare dar N nu este un nod de infirmare, trebuie sa existe doua instante de baza C_1' si C_2' ale clauzelor C_1 si C_2 astfel încît C_1' si C_2' sînt false în interpretarile $I(N_1)$ si respectiv $I(N_2)$, dar atît C_1' cît si C_2' nu sînt falsificate de interpretarea I(N). Rezulta ca C_1' trebuie sa contina $\sim m_{n+1}$ si C_2' trebuie sa contina m_{n+1}.

Fie literalii $L_1' = \sim m_{n+1}$ si $L_2' = m_{n+1}$. Rezolvînd perechea de literali L_1' si L_2' obtinem clauza $C' = (C_1' - L_1') \cup (C_2' - L_2')$. Clauza C' trebuie sa fie falsa în interpretarea I(N) deoarece atît $(C_1' - L_1')$ cît si $(C_2' - L_2')$ sînt false în interpretarea I(N).

Conform lemei, exista clauza $C = rez(C_1, C_2)$ astfel încît clauza C' este o instanta de baza a lui C. Fie T" arborele semantic închis al setului de clauze $S \cup \{C\}$ obtinut din T' prin eliminarea tuturor nodurilor si legaturilor ce urmeaza primului nod în care clauza C' este falsa. Evident, numarul de noduri din T" este mai mic decît numarul de noduri din T'.

Aplicînd procesul de mai sus asupra lui T", se poate obtine un alt rezolvent al clauzelor din $S \cup \{C\}$. Adaugînd acest rezolvent la $S \cup \{C\}$, se obtine un alt arbore semantic închis si mai mic. Acest proces este aplicat repetat pîna în momentul în care se obtine un arbore semantic închis constînd numai din nodul radacina. Aceasta este posibil numai cînd s-a derivat clauza vida, deci exista o deductie a clauzei vide din S.

($\Leftarrow$) Exista o deductie a clauzei vide din S. Fie $R_1, R_2, ..., R_k$ rezolventii succesivi ai acestei deductii. Sa presupunem ca S este realizabila. Atunci exista o interpretare care satisface S. Dar daca o interpretare satisface clauzele C_1 si C_2, atunci aceasta satisface si orice rezolvent al clauzelor C_1 si C_2. Atunci aceasta interpretare satisface si $R_1, R_2, ..., R_k$. Dar acest lucru este imposibil deoarece unul dintre rezolventii R_i este clauza vida, deci S este nerealizabil.

Fie N un nod de inferenta în T'. Fie N_1 si N_2 noduri de infirmare descendenti imediati ai lui N si fie interpretarile $I(N) = \{m_1, m_2, ..., m_n\}$, $I(N_1) = \{m_1, m_2, ..., m_n, m_{n+1}\}$ si $I(N_2) = \{m_1, m_2, ..., m_n, \sim m_{n+1}\}$. Deoarece N_1 si N_2 sînt noduri de infirmare dar N nu este un nod de infirmare, trebuie sa existe doua instante de baza C_1' si C_2' ale clauzelor C_1 si C_2 astfel încît C_1' si C_2' sînt false în interpretarile $I(N_1)$ si respectiv $I(N_2)$, dar atît C_1' cît si C_2' nu sînt falsificate de interpretarea I(N). Rezulta ca C_1' trebuie sa contina $\sim m_{n+1}$ si C_2' trebuie sa contina m_{n+1}.

Fie literalii $L_1' = \sim m_{n+1}$ si $L_2' = m_{n+1}$. Rezolvînd perechea de literali L_1' si L_2' obtinem clauza $C' = (C_1' - L_1') \cup (C_2' - L_2')$. Clauza C' trebuie sa fie falsa în interpretarea I(N) deoarece atît $(C_1' - L_1')$ cît si $(C_2' - L_2')$ sînt false în interpretarea I(N).

Conform lemei, exista clauza $C = rez(C_1, C_2)$ astfel încît clauza C' este o instanta de baza a lui C. Fie T" arborele semantic închis al setului de clauze $S \cup \{C\}$ obtinut din T' prin eliminarea tuturor nodurilor si legaturilor ce urmeaza primului nod în care clauza C' este falsa. Evident, numarul de noduri din T" este mai mic decît numarul de noduri din T'.

Aplicînd procesul de mai sus asupra lui T", se poate obtine un alt rezolvent al clauzelor din $S \cup \{C\}$. Adaugînd acest rezolvent la $S \cup \{C\}$, se obtine un alt arbore semantic închis si mai mic. Acest proces este aplicat repetat pîna în momentul în care se obtine un arbore semantic închis constînd numai din nodul radacina. Aceasta este posibil numai cînd s-a derivat clauza vida, deci exista o deductie a clauzei vide din S.

($\Leftarrow$) Exista o deductie a clauzei vide din S. Fie $R_1, R_2, ..., R_k$ rezolventii succesivi ai acestei deductii. Sa presupunem ca S este realizabila. Atunci exista o interpretare care satisface S. Dar daca o interpretare satisface clauzele C_1 si C_2, atunci aceasta satisface si orice rezolvent al

clauzelor C_1 si C_2. Atunci aceasta interpretare satisface si $R_1, R_2, ..., R_k$. Dar acest lucru este imposibil deoarece unul dintre rezolventii R_i este clauza vida, deci S este nerealizabil.

3.5 Strategii rezolutive

Rezolutia ofera un mijloc convenabil de a gasi demonstrarea prin respingere a unei teoreme fara a încerca toate substitutiile pe care le implica teorema lui Herbrand. Cu toate acestea apar dificultati de implementare directa a acestei metode. Algoritmii de respingere prin rezolutie prezentati în Sectiunile 3.4.1 si 3.4.3 sînt nedeterministi deoarece nu specifica cum trebuie selectate cele doua clauze care rezolva în aplicarea rezolutiei.

S-a aratat ca rezolutia este o metoda de inferenta, deci o procedura de obtinere la un moment dat, din elemente particulare ale reprezentarii, de noi date implicate în mod direct. Pentru rezolvarea oricarei probleme, deci în particular pentru demonstrarea teoremelor utilizînd rezolutia, metoda de inferenta trebuie aplicata repetat, în mod sistematic, conform unei strategii de control. O strategie de control este modalitatea de aplicare repetata a metodei de inferenta pentru a ajunge, de preferinta cît mai repede, la solutie. În consecinta, realizarea unui program de demonstrare automata a teoremelor prin metoda respingerii prin rezolutie necesita stabilirea unei strategii de control în efectuarea repetata a rezolutiei.

Dupa ce în 1965 Robinson a propus metoda rezolutiei, s-au dezvoltat o serie de strategii de control pentru implementarea respingerii prin rezolutie [Chang,Lee,1973]. Diversele strategii rezolutive propuse au urmarit completitudinea si eficienta procedeului. O strategie rezolutiva este completa daca asigura întotdeauna deducerea clauzei vide din multimea de axiome si teorema negata, în cazul în care teorema se poate demonstra. Din punct de vedere al eficientei, toate strategiile propuse sînt în principiu complexe computational, adica de complexitate $O(e^n)$, dar realizeaza, printr-o modalitate sau alta, o reducere a timpului de calcul. Timpul poate fi îmbunatatit si prin adaugarea unor criterii euristice, asa cum se va prezenta în Sectiunea 3.7.4.

3.5.1 Clasificarea strategiilor rezolutive

În continuare se vor prezenta pe scurt cele mai importante strategii rezolutive propuse. O mare parte dintre acestea vor fi reluate în sectiunile urmatoare si discutate pe larg, cu un accent deosebit pus pe strategia rezolutiei semantice si cea a rezolutiei liniare.

(1) *Strategia dezvoltarii pe nivel*, numita si strategia epuizarii rezolventilor imediati sau metoda saturarii nivelului, are la baza urmatoarea idee: se calculeaza toti rezolventii posibili de pe un nivel, acesti rezolventi se adauga la acest nivel pentru a forma nivelul urmator si se reia procesul pentru nivelul urmator.

Aceasta strategie suporta urmatoarea îmbunatatire: la fiecare nivel se elimina tautologiile deduse si clauzele subsumate de clauze deduse anterior, aceste modificari procedurale reprezentînd *strategia eliminarii*. Aceasta strategie este o strategie completa.

(2) *Strategia rezolutiei semantice* [Slagle,1967], combina strategia hiperrezolutiei [Robinson,1965] cu strategia multimii suport [Wos,s.a.,1965] avînd la baza urmatoarea idee: setul de clauze S se împarte în doua submultimi S_1 si S_2 astfel încît pentru orice interpretare I, I satisface S_1 si I falsifica S_2. Se aplica rezolutia între $C_1 \in S_1$ si $C_2 \in S_2$. În plus se încearca eliminarea obtinerii de clauze inutile prin rezolvarea simultana a unui grup de clauze, ordonarea predicatelor si ordonarea clauzelor. Strategia rezolutiei semantice este, de asemenea, completa.

(3) *Strategia rezolutiei blocarii* [Boyer,1971] este o strategie completa care foloseste ordonarea clauzelor si indici pentru a marca ordinea literalilor în clauze. Aceasta strategie este completa dar nu este compatibila cu majoritatea celorlalte strategii. De exemplu, rezolutia blocarii împreuna cu rezolutia eliminarii nu este completa, si nici combinarea rezolutiei blocarii cu rezolutia multimii suport nu este completa, dar poate fi uneori foarte eficienta.

(4) *Strategia rezolutiei liniare* [Loveland,1970;Luckham,1970;Kovalski,1970] este o strategie completa, usor de implementat, care are la baza urmatoarea idee: orice rezolvent C_i obtinut este utilizat pentru a obtine rezolventul C_{i+1}, $i = 1,2,...,n-1$. Exista doua cazuri particulare ale strategiei rezolutiei liniare:

- *Rezolutia de intrare liniara* în care una din clauzele ce rezolva apartine setului initial de axiome.
- *Rezolutia unitara* în care una din clauzele ce rezolva este fie o clauza unitara, fie un factor unitar al unei clauze.

Ambele cazuri particulare sînt incomplete dar eficiente. Din aceasta cauza ele sînt utilizate deseori. Un exemplu este limbajul Prolog [Clocksin,Melish,1981;Bratko,1986] care foloseste strategia rezolutiei de intrare liniara.

(5) *Strategia rezolutiei teleologice* este o strategie utilizata în derivarea rezolutiva, în care teorema este derivata si nu demonstrata prin respingere [Popa,1992].

3.5.2 Strategia dezvoltarii pe nivel

Fie S multimea de clauze pentru care trebuie aratat ca este nerealizabila (inconsistenta). Se calculeaza secventa $S^0, S^1, S^2, ...$, unde $S^0 = S$ si $S^k = \{rez(C_1, C_2) | C_1 \in S^0 \cup ... \cup S^{k-1}, C_2 \in S^{k-1}\}$, $k = 1,2,...$ pîna se obtine clauza vida, daca teorema este adevarata. Altfel spus, se calculeaza rezolventii tuturor perechilor de clauze din S, acestia se adauga la S si se repeta procesul pîna se obtine clauza vida. O posibila metoda sistematica de aplicare a acestei strategii este data de urmatorul algoritm.

Algoritm: Strategia dezvoltarii pe nivel

1. Fie L multimea clauzelor din S, corespunzatoare lui S^0
2. $Lc \leftarrow L,\ Lc' \leftarrow \{\ \}$;
3. **pentru** fiecare $C_{1i} \in L$ si $C_{2j} \in Lc,\ j > i$ **executa** /* deci C_{2j} dupa C_{1i} */
   - 3.1. **daca** $C = rez(C_{1i}, C_{2j}) = \square$
     atunci întoarce SUCCES /* teorema demonstrata */
   - 3.2. $Lc' \leftarrow Lc' \cup C$ /* introduce C în multimea Lc' */
4. $L \leftarrow L \cup Lc',\ Lc \leftarrow Lc',\ Lc' \leftarrow \{\ \}$
5. **repeta de la** 2

sfîrsit.

Observatii:

- S-au folosit notatiile { } pentru lista vida si $\cup$ pentru operatorul de reuniune de multimi.
- În algoritmul de mai sus $L = S^0 ... S^{k-1},\ Lc = S^{k-1},\ Lc' = S^k$
- Pasul 3 este formulat pentru cazul în care teorema este adevarata, deci se va obtine întotdeauna clauza vida. Pentru cazul general, în care acest lucru nu se stie, algoritmul trebuie modificat conform conditiilor generale de oprire a procesului rezolutiv prezentate în Sectiunea 3.4.3.
- Multimile L, Lc si Lc' pot fi reprezentate, în cazul unei implementari a algoritmului, prin liste, iar operatorul de reuniune de multimi printr-o operatie de concatenare a listelor.

Exemplu. Fie $S = S^0 = \{P \vee Q\ \ (1), \sim P \vee Q\ \ (2), P \vee \sim Q\ \ (3), \sim P \vee \sim Q\ \ (4)\}$. Atunci S^1 contine:

(5)	Q	1+2	(13)	$P \vee Q$	1+7 subsumat de 5
(6)	P	1+3	(14)	$P \vee Q$	1+8 subsumat de 5
(7)	$Q \vee \sim Q$	1+4 tautologie	(15)	$P \vee Q$	1+9 subsumat de 5
(8)	$P \vee \sim P$	1+4 tautologie	(16)	$P \vee Q$	1+10 subsumat de 5
(9)	$Q \vee \sim Q$	2+3 tautologie	(17)	Q	1+11 subsumat de 5
(10)	$P \vee \sim P$	2+3 tautologie	(18)	P	1+12 subsumat de 6
(11)	~P	2+4	(19)	Q	2+6 subsumat de 5
(12)	~Q	3+4		...	

(37) Q 5+7 subsumat de 5

(38) Q 5+9 subsumat de 5

(39) □ 5+12

Se observa ca s-au generat multe clauze redundante sau irelevante. De exemplu 7, 8, 9 si 10 sînt tautologii. Deoarece o tautologie este adevarata în orice interpretare, daca se elimina o tautologie dintr-un set de clauze nerealizabile, multimea rezultata este tot nerealizabila. Deci aceste clauze ar putea fi eliminate, altfel vor genera alte clauze irelevante si asa mai departe. În plus, se pot elimina si clauzele subsumate, asa cum se prezinta în sectiunea urmatoare. Din aceste motive strategia dezvoltarii pe nivel este frecvent combinata cu strategia eliminarii.

3.5.3 Strategia eliminarii

Strategia eliminarii împiedica adaugarea clauzelor irelevante la multimea de clauze, crescînd astfel eficienta strategiei dezvoltarii pe nivel.

Definitie. O clauza C *subsumeaza* o clauza D daca si numai daca exista o substitutie α astfel încît $C\alpha \subseteq D$. D se numeste *clauza subsumata.* Subsumarea este o regula de inferenta.

Observatie. Daca D este identic cu C, sau D este o instanta a clauzei C, atunci D este, evident, subsumata de C.

Exemplu. Fie $C = P(x)$, $D = P(a) \vee Q(a)$ si $\alpha = \{a/x\}$. Atunci $C\alpha = P(a)$, deci $C\alpha \subseteq D$. Rezulta ca C subsumeaza D. Exemplul din sectiunea anterioara a pus în evidenta, de asemenea, clauze subsumate.

Strategia eliminarii realizeaza eliminarea tautologiilor si a clauzelor subsumate. Eliminarea tautologiilor este o generalizare a regulii I a lui Davis si Putnam din calculul propozitional. Completitudinea strategiei eliminarii depinde de modul în care se elimina tautologiile si clauzele subsumate. Utilizata împreuna cu strategia dezvoltarii pe nivel, strategia eliminarii este completa daca se aplica algoritmul prezentat anterior, modificînd pasul 3.2 astfel:

3.2' **daca nu** (C este tautologie **sau** C este subsumata de $C_k \in L$)

atunci $Lc' \leftarrow Lc' \cup C$

Aplicînd strategia eliminarii pe exemplul din sectiunea anterioara, se obtin urmatoarele rezultate:

$S = S^0 = \{P \vee Q \quad (1), \quad \sim P \vee Q \quad (2), \quad P \vee \sim Q \quad (3), \quad \sim P \vee \sim Q \quad (4)\}$.

si S^1 va contine de aceasta data

(5) Q 1+2

(6) P 1+3

(7) ~P 2+4

(8) ~Q 3+4

(9) □ 5+8

Eliminarea clauzelor irelevante se face prin verificarea tautologiilor si a clauzelor subsumate si eliminarea acestora, conform pasului 3.2'. O clauza este tautologie daca contine o pereche complementara. Pentru a verifica daca o clauza subsumeaza o alta clauza se utilizeaza urmatorul algoritm.

Fie $\alpha = \{a_1 / x_1, ..., a_n / x_n\}$ o substitutie, unde $x_1, \ldots, x_n$ sînt variabilele care apar în D si $a_1, \ldots, a_n$ constante distincte ce nu apar nici în C, nici în D. D este de forma $L_1 \vee L_2 \vee ... \vee L_m$ si exista relatiile: $D\alpha = L_1\alpha \vee L_2\alpha \vee ... \vee L_m\alpha$ si $\sim D\alpha = \sim L_1\alpha \vee \sim L_2\alpha \vee ... \vee \sim L_m\alpha$.

Algoritm: Verificare daca C subsumeaza D

1. $W \leftarrow \{\sim L_1\alpha, \sim L_2\alpha, ..., \sim L_m\}$
2. $k \leftarrow 0,\ U^0 \leftarrow \{C\}$
3. **daca** $\square \in U^k$
   atunci întoarce SUCCES /* C subsumeaza D */
4. $U^{k+1} \leftarrow \{rez(C_1, C_2) \mid C_1 \in U^k, C_2 \in W\}$
5. **daca** $U^{k+1} = \phi$
   atunci întoarce INSUCCES /* C nu subsumeaza D */
6. $k \leftarrow k + 1$
7. **repeta de la** 3

sfîrsit.

Observatie. În acest algoritm se observa ca fiecare clauza din U^{k+1} este cu un literal mai mica decît clauza din U^k din care a provenit. Deci $U^0, U^1, \ldots$ fie va contine clauza vida, fie va genera o multime de rezolventi vida. Algoritmul se termina întotdeauna si este corect.

Exemple:

1. Fie $C = \sim P(x) \vee Q(f(x), a)$ si $D = \sim P(h(y)) \vee Q(f(h(y)), a) \vee \sim P(z)$ doua clauze si fie $\alpha = \{b / y, c / z\}$, $b, c \notin C$, $b, c \notin D$ o substitutie. Aplicarea substitutiei asupra clauzei D negate conduce la $\sim D\alpha = P(h(b)) \wedge \sim Q(f(h(b)), a) \wedge P(c)$. Rezulta deci multimile:

$W = \{P(h(b)), \sim Q(f(h(b)), a), P(c)\}$,

$U^0 = \{\sim P(x) \vee Q(f(x), a)\}$, $U^1 = \{Q(f(h(b)), a), \sim P(h(b)), Q(f(c), a)\}$ si $U^2 = \{\square\}$

deci C subsumeaza D.

2. Fie $C = P(x, x)$ si $D = P(f(x), y) \vee P(y, f(x))$ doua clauze si $\alpha = \{a / x, b / y\}$ o substitutie. Prin aplicarea substitutiei asupra clauzei D negate se obtine clauza $\sim D\alpha = \sim P(f(a), b) \wedge \sim P(b, f(a))$ si multimile: $W = \{\sim P(f(a), b), \sim P(b, f(a))\}$, $U^0 = \{P(x, x)\}$ si $U^1 = \phi$, deci C nu subsumeaza D.

3.6 Strategia rezolutiei semantice

Strategia dezvoltarii pe nivel este o strategie complex computationala. Desi are avantajul simplitatii si completitudinii, aceasta strategie, chiar îmbunatatita cu strategia eliminarii, genereaza un numar foarte mare de clauze inutile. În plus, multimea de clauze din care se pot alege cele doua clauze care rezolva este toata multimea de clauze obtinuta pe un anumit nivel. Strategia rezolutiei semantice propune modalitati de reducere a numarului de clauze inutile si a numarului de combinatii posibile de doua clauze ce rezolva.

3.6.1 Introducere informala

Se introduc urmatoarele restrictii în aplicarea principiului rezolutiei: se împarte multimea de clauze în doua submultimi S1 si S2 si nu se permit rezolvari de clauze ce apartin aceleiasi submultimi. Aceste restrictii implica reducerea numarului de clauze generate în aplicarea rezolutiei.

Exemplu: Fie multimea de clauze S:

(1) $\sim P \vee \sim Q \vee R$

(2) $P \vee R$

(3) $Q \vee R$

(4) $\sim R$

Se împarte multimea S în submultimile S1, care contine clauzele (2) si (3), si S2, care contine clauzele (1) si (4). În acest fel se reuseste blocarea rezolutiei între clauzele (1) si (4).

În rezolutia semantica, împartirea multimii de clauze în doua submultimi se face pornind de la o interpretare ce satisface S1 si nu satisface S2. Daca S este nerealizabila, exista întotdeauna

cel putin o astfel de interpretare. Pentru exemplul de mai sus, $I = \{\sim P, \sim Q, \sim R\}$ falsifica clauzele (2) si (3) si satisface clauzele (1) si (4).

Cum se poate bloca în continuare rezolutia altor clauze care ar duce la generarea de rezolventi inutili? În exemplul de mai sus, pentru S1 si S2 se observa ca se mai pot efectua rezolutii între clauzele (2) si (3) cu clauza (4). Solutia consta în fixarea unei ordonari a simbolurilor predicative din S si adaugarea urmatoarei conditii: la rezolvarea clauzelor $C_1 \in S1$ si $C_2 \in S2$, literalul din C_1 care rezolva trebuie sa contina (sa fie) cel mai mare predicat existent în aceasta clauza. Pentru exemplul de mai sus se considera ordonarea predicatelor $P > Q > R$. În aceste conditii, clauzele (2) si (4) nu mai pot rezolva si nici clauzele (3) si (4), deoarece R nu este literalul cu cel mai mare predicat nici în clauza (2), nici în clauza (3).

S-au introdus pîna în acest moment doua concepte: o interpretare ce împarte multimea de clauze în doua submultimi si o ordonare a predicatelor din S împreuna cu conditia de rezolvare. Se introduce acum un al treilea concept, numit *multime de coincidenta*.

Se considera multimea de clauze S, din exemplul precedent si interpretarea $I = \{\sim P, \sim Q, \sim R\}$, care împarte S în cele doua submultimi S1 si S2; S1 contine clauzele (2) si (3), iar S2 contine clauzele (1) si (4). Submultimea S1 este falsificata de I, iar S2 este adevarata în I. Se considera ordonarea predicatelor $P > Q > R$. În aceste conditii, singurii rezolventi care se pot produce sînt:

(5) $\sim Q \vee R$ $\quad 1\ (\in S2) + 2\ (\in S1)$

(6) $\sim P \vee R$ $\quad 1\ (\in S2) + 3\ (\in S1)$

Ambii rezolventi sînt satisfacuti de interpretarea I, deci se adauga la S2. În acest moment S2 contine clauzele (1), (4), (5) si (6). Aplicînd rezolutia în continuare se obtine

(7) R $\quad 3\ (\in S1) + 5\ (\in S2)$

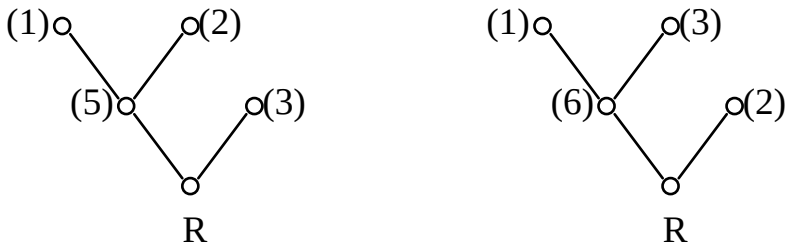
(8) R $\quad 2\ (\in S1) + 6\ (\in S2)$

R este fals în interpretarea I, deci R se adauga la S1. În acest moment S1 contine clauzele (2), (3) si (7).

Deoarece $\sim R \in S2$ si $R \in S1$ se obtine

(9) □ $\quad 7\ (\in S1) + 4\ (\in S2)$

Dupa cum se observa, derivarea lui R s-a putut face în doua moduri:



Evident, este inutila generarea lui R de doua ori. R se poate genera direct din multimea de clauze (1), (2) si (3). Aceasta multime de clauze este o multime de coincidenta semantica. Pe baza ei se pot elimina clauzele intermediare (5) si (6).

3.6.2 Definirea formala a rezolutiei semantice

Definitie. Fie I o interpretare si P o ordonare a simbolurilor predicative. O multime finita de clauze $\{E_1, E_2, ..., E_q, N\}$, $q \geq 1$, se numeste *multime de coincidenta semantica* fata de P si I, pe scurt PI-clash, daca si numai daca $E_1, E_2, \ldots, E_q$, numiti *electroni*, si N, numit *nucleu*, satisfac urmatoarele conditii:

(1) Electronii $E_1, E_2, \ldots, E_q$ sînt falsificati de interpretarea I

(2) Fie rezolventul $R_1 = N$. Pentru fiecare $i = 1, \ldots, q$ exista rezolventul $R_{i+1} = rez(R_i, E_i)$

(3) Literalul din E_i ce rezolva contine cel mai mare simbol predicativ din E_i, $i = 1, \ldots, q$

(4) Rezolventul R_{q+1} este falsificat de interpretarea I.

R_{q+1} se numeste *PI-rezolvent* al multimii de coincidenta semantica $\{E_1, E_2, ..., E_q, N\}$, $q \geq 1$.

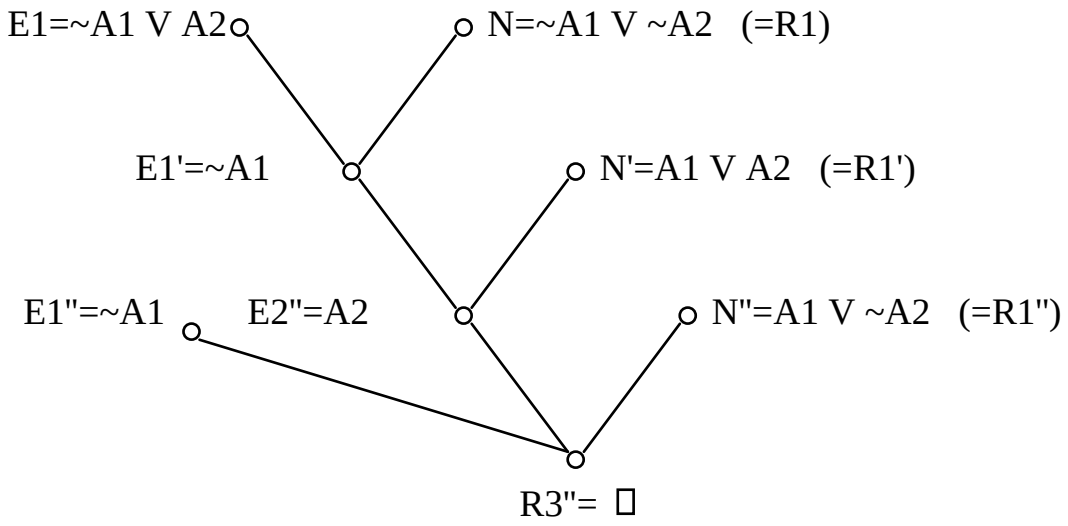
Observatie. Ordonarea $E_1, E_2, \ldots, E_q$ este nesemnificativa. Oricare din primele q clauze dintr-o multime de coincidenta semantica poate sta pe postul oricarui electron deoarece pentru multimi de acest tip întotdeauna se obtine R_{q+1}.

Exemplu. Fie multimea de clauze $S = \{A_1 \vee A_3, A_2 \vee A_3, \sim A_1 \vee \sim A_2 \vee A_3\}$. Considerînd interpretarea $I = \{\sim A_1, \sim A_2, \sim A_3\}$ si ordonarea simbolurilor predicative $A_1 > A_2 > A_3$, se observa ca S este o multime de coincidenta semantica cu $E_1 = A_1 \vee A_3$, $E_2 = A_2 \vee A_3$ si $N = \sim A_1 \vee \sim A_2 \vee A_3$, $R_1 = N$, $R_2 = rez(E_1, R_1) = \sim A_2 \vee A_3$, $R_3 = rez(R_2, E_2) = A_3$, deci PI-rezolventul multimii de coincidenta semantica este A_3. Se observa ca E_1, E_2 si A_3 sînt falsificate de interpretarea I.

Utilizarea conceptului de multime de coincidenta semantica sta la baza strategiei rezolutiei semantice.

Definitie. Fie I o interpretare a unei multimi de clauze S si P o ordonare a predicatelor din S. O deductie din S este numita *PI-deductie* daca si numai daca orice clauza obtinuta în rezolutie fie apartine lui S, fie este un PI-rezolvent.

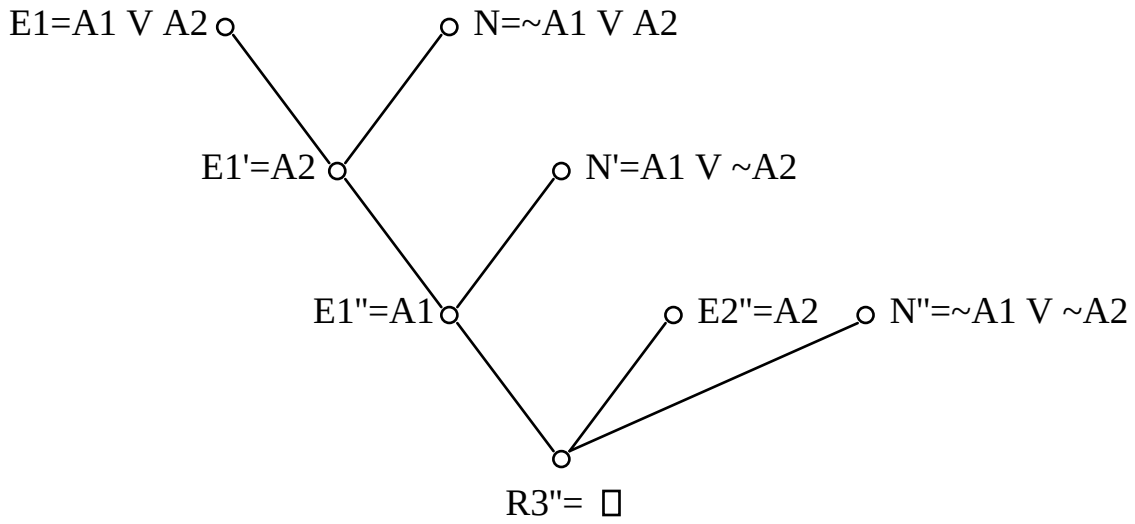
Exemplu. Fie setul de clauze $S = \{A_1 \vee A_2, A_1 \vee \sim A_2, \sim A_1 \vee A_2, \sim A_1 \vee \sim A_2\}$, interpretarea $I = \{A_1, \sim A_2\}$ si ordonarea predicatelor $A_1 < A_2$. În aceste conditii se obtine urmatoarea PI-deductie:



Observatii:

- În exemplul de mai sus exista trei PI-rezolventi, respectiv $\sim A_1$, A_2 si □. Ultimul PI-rezolvent este obtinut pe baza multimii de coincidenta semantica $\{\sim A_1, A_2, A_1 \vee \sim A_2\}$ unde $E_1" = \sim A_1$, $E_2" = A_2$, $N" = A_1 \vee \sim A_2$, $R_1" = N"$, $R_2" = rez(E_1", N") = \sim A_2$ si $R_3" = rez(R_2", E_2") = \square$.

- Se poate utiliza orice ordonare a predicatelor si orice interpretare în rezolutia semantica. Pentru exemplul anterior, interpretarea $I = \{\sim A_1, \sim A_2\}$ si $A_2 < A_1$, se obtine urmatoarea PI-deductie a clauzei vide din S.



Teorema. *Completitudinea PI-rezolutiei*. Daca P este o ordonare a simbolurilor predicative dintr-o multime de clauze finita si nerealizabila S, si daca I este o interpretare a lui S, atunci exista o PI-deductie a clauzei vide din S.

Observatie. Fara a se pierde completitudinea, se pot combina strategiile de eliminare si PI-rezolutie.

Se introduc în continuare doua tipuri de interpretari ce vor fi utilizate în rezolutia semantica. O interpretare duce la hiperrezolutie, iar cealalta la strategia multimii suport.

3.6.3 Hiperrezolutia

Definitie. O clauza se numeste *pozitiva* daca nu are nici un literal negat. O clauza se numeste *negativa* daca toti literalii din clauza sînt negati. O clauza se numeste *mixta* daca nu este nici pozitiva, nici negativa.

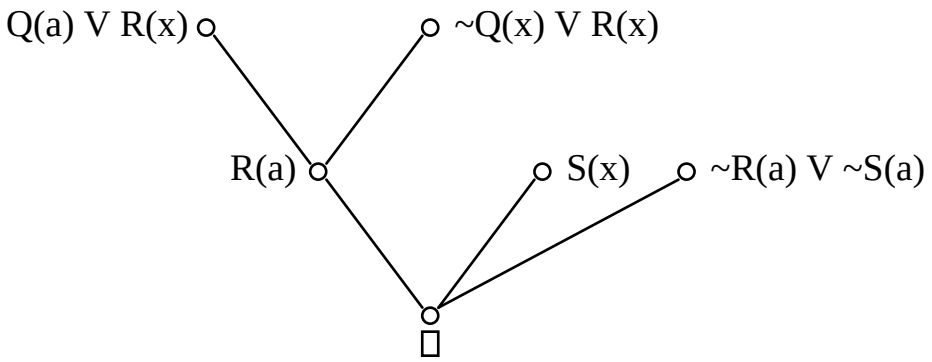
Definitie. Se numeste *hiperrezolutie pozitiva* PI-rezolutia pentru care toti literalii din interpretarea I sînt negati ($I = \{\sim A_1, \sim A_2, ..., \sim A_n\}$)

Se pune problema de ce aceasta strategie se numeste hiperrezolutie pozitiva daca toti literalii sînt negati. Numele de hiperrezolutie pozitiva este dat deoarece pentru o astfel de interpretare toti electronii $E_1, E_2, ..., E_q$ din toate multimile de coincidenta semantica utilizate si toti PI-rezolventii R_{q+1} obtinuti sînt clauze pozitive.

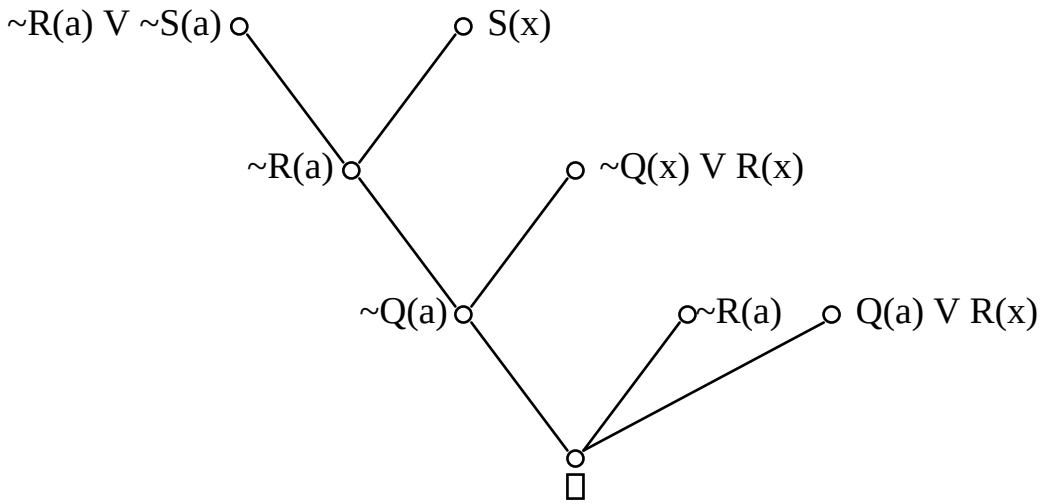
Definitie. Se numeste *hiperrezolutie negativa* PI-rezolutia pentru care nici un literal din interpretarea I nu este negat.

Aceeasi justificare se poate da si pentru denumirea de hiperrezolutie negativa. Pentru o interpretare în care nici un literal nu este negat, toti electronii $E_1, E_2, ..., E_q$ si toti PI-rezolventii R_{q+1} sînt clauze negative.

Exemplu. Fie setul de clauze $S = \{Q(a) \vee R(x), \sim Q(x) \vee R(x), \sim R(a) \vee \sim S(a), S(x)\}$ si ordonarea predicatelor $R < Q < S$. Aplicînd hiperrezolutia pozitiva în interpretarea $I = \{\sim Q(a), \sim R(a), \sim S(a)\}$, se obtine



Aplicînd hiperrezolutia negativa în interpretarea $I = \{Q(a), R(a), S(a)\}$, se obtine



Atît hiperrezolutia pozitiva, cît si hiperrezolutia negativa sînt strategii rezolutive complete. În concluzie se poate spune ca s-au stabilit niste interpretari ce reusesc sa separe multimea de clauze în asa fel încît sa reduca numarul de rezolventi posibili.

3.6.4 Strategia multimii suport

Aceasta strategie are la baza urmatoarea idee: o teorema T se demonstreaza pe baza unei multimi de axiome $A_1,...,A_n$. Pentru a demonstra teorema, trebuie sa se arate ca $A_1 \wedge A_2 ... \wedge A_n \wedge \sim T$ este nerealizabila. Cum $A_1,...,A_n$ este, de obicei, o multime de clauze realizabile, ar fi bine sa se evite rezolvarea clauzelor din $A_1,...,A_n$ între ele.

Definitie. O submultime $T \subset S$, unde S este o multime de clauze, se numeste *multime suport* a lui S daca $S - T$ este realizabila. O *rezolutie bazata pe multimea suport* este o rezolutie a doua clauze astfel încît $C_1 \in S - T$ si $C_2 \in T$.

Teorema. *Completitudinea strategiei multimii suport*. Daca S este o multime de clauze finita si nerealizabila si $T \subset S$ astfel încît $S - T$ este realizabila, atunci exista o deductie bazata pe rezolutia multimii suport a clauzei vide din S, unde T este multimea suport.

3.6.5 Rezolutia semantica cu utilizarea clauzelor ordonate

Ordonarea simbolurilor predicative este eficienta în calculul propozitional, dar mai putin eficienta în logica cu predicate de ordinul I. De exemplu, un electron dintr-o multime de coincidenta semantica poate contine mai multe aparitii ale predicatului maxim conform relatiei de ordonare a predicatelor. Într-o astfel de situatie, oricare din literalii ce contin acest predicat poate fi utilizat în rezolutie, deci avantajul introdus de ordonarea simbolurilor predicative se reduce semnificativ. De exemplu, ordonarea simbolurilor predicative $Q < R < S$ dintr-o multime de coincidenta

semantica ce contine un electron $E = Q(a) \vee S(x) \vee S(f(a))$ permite selectarea atît a lui S(x), cît si a lui S(f(a)) ca literal ce rezolva din E.

Fie un alt exemplu în care electronul este $E = Q(a) \vee Q(b) \vee Q(c) \vee Q(d)$ si nucleul este $N = \sim Q(x)$. Ordonarea predicatelor este Q si fie interpretarea $I = \{\sim Q(a), \sim Q(b), \sim Q(d)\}$. $\{E, N\}$ este o multime de coincidenta semantica, dar exista patru rezolventi ce pot fi obtinuti: $Q(b) \vee Q(c) \vee Q(d)$, $Q(a) \vee Q(c) \vee Q(d)$, $Q(a) \vee Q(b) \vee Q(d)$ si $Q(a) \vee Q(b) \vee Q(c)$. Deci în logica cu predicate de ordinul I se pot genera mai multi PI-rezolventi dintr-o multime de coincidenta semantica. Pentru eliminarea acestui inconvenient se introduce conceptul de clauza ordonata.

Definitie. O *clauza ordonata* este o secventa de literali distincti, în care relatia de ordine între literali este data de pozitia lor în clauza. Primul literal este cel mai mic, iar ultimul cel mai mare. Daca $C = L_1 \vee L_2 \vee ... \vee L_n$ atunci ordonarea literalilor este $L_1 < L_2 < ... < L_n$.

Definitie. Daca doi sau mai multi literali de acelasi semn (pozitivi sau negativi) ai unei clauze ordonate C au un cel mai general unificator β, atunci $C\beta$ este numit *factor ordonat* al lui C, daca $C\beta$ este o clauza obtinuta prin aplicarea substitutiei β asupra lui C si eliminarea fiecarui literal identic cu literali mai mici decît el.

Pentru a determina factorul unei clauze în general, daca $L_1, L_2 \in C$ si $L_1\beta = L_2\beta$, atunci în $C\beta$ trebuie retinut fie $L_1\beta$, fie $L_2\beta$. Spre deosebire de definitia factorului simplu, cum $C\beta$ trebuie sa fie tot o clauza ordonata, de aceasta data este importanta pozitia pe care va ramîne $L_1\beta = L_2\beta$ în clauza, adica în locul lui L_1 sau în locul lui L_2. Se va pastra aparitia celui mai mic literal, deci a celui mai din stînga. Daca L_1 apare înaintea lui L_2, deci $L_1 < L2$, atunci se va pastra $L_1\beta$ si se va elimina $L_2\beta$.

Exemplu. Fie clauza $C = P(x) \vee Q(x) \vee P(a)$ si cel mai general unificator $\beta = \{a / x\}$. Atunci $C\beta = P(a) \vee Q(a) \vee P(a) = P(a) \vee Q(a)$.

Definitie. Fie C_1 si C_2 doua clauze ordonate ce nu au variabile comune, $L_1 \in C_1$, $\sim L_1 \in C_2$ doi literali si β cel mai general unificator al literalilor L_1 si L_2. *Rezolventul binar ordonat* al clauzelor C_1 si C_2, notat cu rezord(C_1,C_2), se defineste astfel:

$$C = \text{rezord}(C_1, C_2) = \text{concat}(C_1\beta, C_2\beta) - L_1\beta - \sim L_2\beta,$$

dupa care se elimina din C orice literal identic cu literali mai mici decît el în secventa.

Observatie. Din definitie se observa ca $\text{rezord}(C_1, C_2) \neq \text{rezord}(C_2, C_1)$

Exemplu. Fie clauzele $C_1 = P(x) \vee Q(x) \vee R(x)$ si $C_2 = \sim P(a) \vee Q(a)$ si cel mai general unificator $\beta = \{a / x\}$. Atunci

$$\text{rezord}(C_1, C_2) = P(a) \vee Q(a) \vee R(a) \vee \sim P(a) \vee Q(a) = Q(a) \vee R(a) \vee Q(a) = Q(a) \vee R(a)$$

Similar cu definitia rezolventului data în Sectiunea 3.4.3 se defineste notiunea de *rezolvent ordonat* pe baza notiunilor de rezolvent binar ordonat si factor ordonat.

Definitie. O r*ezolutie ordonata* între clauzele C_1 si C_2 se obtine prin calculul unui rezolvent ordonat.

Observatie. Rezolutia ordonata este o regula de inferenta. Se poate demonstra ca rezolutia ordonata este completa.

Considerînd conceptul de rezolutie ordonata, notiunea de multime de coincidenta semantica se modifica la notiunea de multime ordonata de coincidenta semantica în raport cu I.

Definitie. Fie I o interpretare. O multime finita de clauze ordonate $\{E_1, E_2, ..., E_q, N\}$, $q \geq 1$, se numeste *multime ordonata de coincidenta semantica* fata de I, pe scurt OI-clash, daca si numai daca $E_1, E_2, \ldots, E_q$, numiti *electroni ordonati*, si N, numit *nucleu ordonat*, satisfac urmatoarele conditii:

(1) Electronii ordonati $E_1, E_2, \ldots, E_q$ sînt falsificati de interpretarea I.

(2) Fie $R_0 = N$. Pentru fiecare $i = 1, \ldots, q$ exista $R_{i+1} = \text{rezord}(R_i, E_i)$.

(3) Literalul din E_i ce rezolva este *ultimul literal* (cel mai mare) din E_i, $i = 1, \ldots, q$.

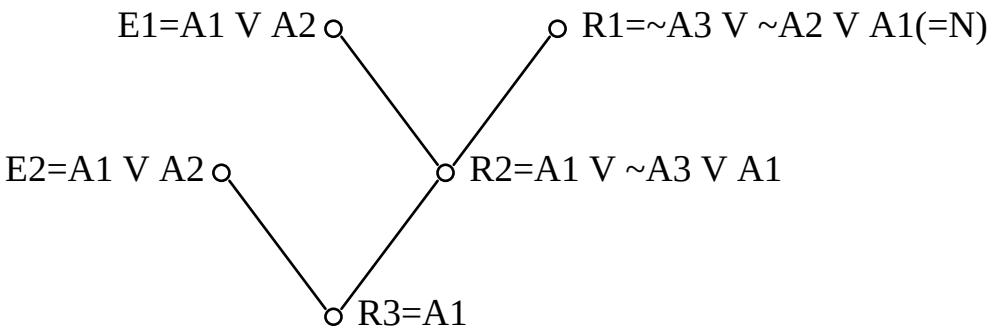
(4) Literalul din R_i ce rezolva este *cel mai mare* literal din R_i care are o instanta adevarata în interpretarea I.

(5) Rezolventul R_{q+1} este falsificat de interpretarea I.

R_{q+1} se numeste *OI-rezolvent* al multimii ordonate de coincidenta semantica $\{E_1, E_2, ..., E_q, N\}$, $q \geq 1$.

Exemple:

1. Fie multimea ordonata de clauze: $S = \{A_1 \vee A_2 \ (1), A_1 \vee A_3 \ (2), \sim A_3 \vee \sim A_2 \vee A_1 \ (3)\}$ si interpretarea $I = \{\sim A_1, \sim A_2, \sim A_3\}$. Clauzele (1) si (2) sînt falsificate de interpretarea I, deci E_1 si E_2 sînt atomi ordonati, iar clauza (3) are un literal adevarat în interpretarea I, deci poate fi considerata un nucleu ordonat.



Rezulta ca multimea clauzelor {(2), (1), (3)} este o multime ordonata de coincidenta semantica fata de I, iar A_1 este un OI-rezolvent.

Dar, daca se încearca întîi rezolvarea între electronul E_2 si nucleu, acest lucru nu este posibil deoarece literalul $\sim A_3$, care poate rezolva, nu este cel mai mare literal adevarat (în interpretarea I) din $R_1 = N$. Deci {(1), (2), (3)} nu este o multime ordonata de coincidenta semantica.

2. Fie multimea ordonata de clauze $S = \{Q(a) \vee Q(b) \vee Q(c) \vee Q(d)\ (1), \sim Q(x)\ (2)\}$, cu electronul $E = Q(a) \vee Q(b) \vee Q(c) \vee Q(d)$ si nucleul $N = \sim Q(x)$, exemplu discutat la începutul acestei sectiuni. Ordonarea predicatelor este Q si interpretarea este $I = \{\sim Q(a), \sim Q(b), \sim Q(d)\}$. Daca se considera $E_1 = (1)$ si $R_1 = N = (2)$, atunci $R_2 = Q(a) \vee Q(b) \vee Q(c)$ este OI-rezolvent al multimii de coincidenta semantica {(1), (2)}. Se observa ca pentru acest exemplu pot exista patru PI-rezolventi dar un singur OI-rezolvent.

Definitie. Fie I o interpretare a unei multimi de clauze ordonate S. O deductie din S se numeste *OI-deductie* daca si numai daca fiecare clauza ordonata din aceasta deductie fie apartine lui S, fie este un OI-rezolvent.

Observatie trista. OI-rezolutia este foarte eficienta si relativ simplu de implementat dar, din nefericire, OI-rezolutia nu este completa.

Exemplu. Fie setul de clauze:

(1) $P \vee Q$ (4) $\sim R \vee \sim P$

(2) $Q \vee R$ (5) $\sim W \vee \sim Q$

(3) $R \vee W$ (6) $\sim Q \vee \sim R$

si interpretarea $I = \{\sim P, \sim Q, \sim R, \sim W\}$. Clauzele (1)÷(3) pot fi utilizate ca electroni ordonati, iar clauzele (4)÷(6) pot fi utilizate ca nuclee ordonate. Din clauzele (1)÷(6) se pot obtine urmatorii OI-rezolventi:

(7) $R \vee P$ din multimea de coincidenta semantica {(3), (1), (5)}

(8) $P \vee Q$ din multimea de coincidenta semantica {(1), (2), (6)}

Din clauzele (1)÷(8) se obtine urmatorul OI-rezolvent:

(9) $Q \vee R$ din multimea de coincidenta semantica {(2), (7), (4)}

Se observa ca (8) si (9) sînt deja în S, deci din clauzele (1)÷(9) nu se mai pot produce noi OI-rezolventi. Cu toate acestea, S este nerealizabila. Deci OI-rezolutia nu este completa.

(7) $R \vee P$ din multimea de coincidenta semantica {(3), (1), (5)}

(8) $P \vee Q$ din multimea de coincidenta semantica {(1), (2), (6)}

Din clauzele (1)÷(8) se obtine urmatorul OI-rezolvent:

(9) $Q \vee R$ din multimea de coincidenta semantica {(2), (7), (4)}

Se observa ca (8) si (9) sînt deja în S, deci din clauzele (1)÷(9) nu se mai pot produce noi OI-rezolventi. Cu toate acestea, S este nerealizabila. Deci OI-rezolutia nu este completa.

3.6.6 Implementarea rezolutiei semantice

Deoarece OI-rezolutia nu este completa, se utilizeaza PI-rezolutia, dar se foloseste conceptul de OI-rezolutie în scopul determinarii literalilor care rezolva. O clauza ordonata pozitiva este o clauza cu toti literalii pozitivi. O clauza ordonata nepozitiva este o clauza care nu este clauza pozitiva. În continuare se vor considera numai hiperrezolutii pozitive deoarece hiperrezolutiile negative pot fi tratate într-o maniera similara. Se considera numai clauze pozitive pentru electroni si clauze nepozitive pentru nuclee. Se face urmatoarea conventie: pentru orice clauza ordonata nepozitiva, literalii negativi se pun dupa cei pozitivi, ceea ce înseamna ca într-un nucleu (clauza nepozitiva) cel mai mare literal adevarat în interpretare va fi ultimul literal care apare în clauza.

Fie S o multime de clauze ordonate si P o ordonare a simbolurilor predicative din S. Urmatorul algoritm calculeaza hiperrezolventii pozitivi ai lui S, utilizînd strategia rezolutiei semantice.

Algoritm: Strategia rezolutiei semantice.

1. Fie M = multimea clauzelor pozitive din S si N = multimea clauzelor nepozitive din S
2. $A_0 \leftarrow \phi$, $B_0 \leftarrow N$, $i \leftarrow 0$, $W_0 \leftarrow \phi$
3. **daca** $\square \in A_i$

   atunci întoarce SUCCES /*S este nerealizabila */
4. **daca** $B_i = \phi$

   atunci executa pasul 6
5. **altfel**

5.1. Calculeaza multimea

$$W_{i+1} \leftarrow \left\{ \text{rezord}(C_1, C_2) \;\middle|\; \begin{array}{l} C_1 \in M, C_2 \in B_i, L_1 \in C_1, \sim L_2 \in C_2 \\ L_1 \text{ este cel mai mare simbol predicativ din } C_1 \\ \sim L_2 \text{ este ultimul literal din } C_2 \end{array} \right\}$$

5.2. $A_{i+1} \leftarrow$ multimea clauzelor pozitive din W_{i+1}

$B_{i+1} \leftarrow$ multimea clauzelor nepozitive din W_{i+1}

5.3. $i \leftarrow i + 1$

5.4. **repeta de la** 3

6. Fie $T \leftarrow A_0 \cup ... \cup A_i$, $M \leftarrow T \cup M$

7. Relaxeaza conditia de clauze ordonate

7.1. Calculeaza multimea

$$R \leftarrow \left\{ \text{rezord}(C_1, C_2) \;\middle|\; \begin{array}{l} C_1 \in T, C_2 \in N, L_1 \in C_1, \sim L_2 \in C_2 \\ L_1 \text{ este cel mai mare simbol predicativ din } C_1 \\ \sim L_2 \text{ este orice literal din } C_2 \text{ (nu neapărat ultimul)} \end{array} \right\}$$

7.2. $A_0 \leftarrow$ multimea clauzelor pozitive din R

$B_0 \leftarrow$ multimea clauzelor nepozitive din R

7.3. $i \leftarrow 0$

7.4. **executa** pasul 3

sfîrsit.

Observatii:

- În algoritmul de mai sus, dupa un anumit numar de iteratii, B_i va ajunge (eventual) vid deoarece numarul maxim de literali negati în orice clauza ordonata a lui B_i scade cu o unitate pe masura ce i creste cu o unitate. În acest moment, daca nu s-a dedus clauza vida, se comuta pe o conditie mai putin "tare".

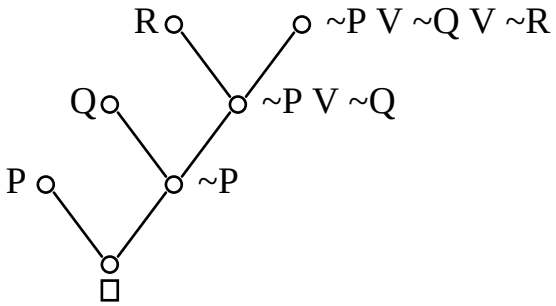
- Toate clauzele din A_i sînt hiperrezolventi pozitivi.

- Daca S este nerealizabil, algoritmul de mai sus va produce întotdeauna clauza vida.

- În algoritm se poate include si strategia eliminarii, strategia combinata ramînînd completa. Pentru T si M obtinuti în pasul 6, orice clauza din T si M subsumata de clauze din T sau M poate fi eliminata. Nu este nevoie sa se verifice prezenta tautologiilor deoarece, clauzele din T si M fiind pozitive, nu exista tautologii în T sau M.

- Algoritmul poate cicla la infinit daca S este realizabila. Pentru a surprinde acest caz, trebuie introduse cele doua conditii suplimentare de oprire prezente în algoritmul general al metodei rezolutiei în logica cu predicate de ordinul I, prezentat în Sectiunea 3.4.3.

Exemplu. Fie multimea de clauze $S = \{\sim P \vee \sim Q \vee \sim R, P, Q, R\}$, cu ordonarea simbolurilor predicative $P < Q < R$. Aplicînd algoritmul rezolutiei semantice pentru multimea de clauze S, multimile M si N vor fi $M = \{P, Q, R\}$, $N = \{\sim P \vee \sim Q \vee \sim R\}$, iar multimile initiale A_0, B_0 si W_0 vor fi $A_0 = \phi$, $B_0 = \{\sim P \vee \sim Q \vee \sim R\}$, $W_0 = \phi$. Dupa primul pas, se obtine $W_1 = \{\sim P \vee \sim Q\}$ deci $A_1 = \phi$ si $B_1 = \{\sim P \vee \sim Q\}$. Dupa pasul urmator, se obtine $W_2 = \{\sim P\}$ deci $A_2 = \phi$ si $B_2 = \{\sim P\}$ si în final, dupa înca un pas, se obtine $W_3 = \{\square\}$, deci $A_3 = \{\square\}$ si $B_3 = \phi$. Arborele de derivare este:



Daca se aplica din nou algoritmul, dar fara a utiliza conceptul de clauza ordonata, deci fara restrictia de alegere a celui mai mare literal din C_2 în pasul 5.1, atunci se obtine: $M = \{P, Q, R\}$, $N = \{\sim P \vee \sim Q \vee \sim R\}$, $A_0 = \phi$ si $B_0 = \{\sim P \vee \sim Q \vee \sim R\}$, $W_1 = \{\sim Q \vee \sim R, \sim P \vee \sim R, \sim P \vee \sim Q\}$, $A_1 = \phi$, $B_1 = \{\sim Q \vee \sim R, \sim P \vee \sim R, \sim P \vee \sim Q\}$, $W_2 = \{\sim R, \sim Q, \sim R, \sim P, \sim Q, \sim P\}$, $A_2 = \phi$, $B_2 = \{\sim R, \sim Q, \sim R, \sim P, \sim Q, \sim P\}$, si în final $W_3 = \{\square\}$, deci $A_3 = \{\square\}$, $B_3 = \phi$. Se observa ca în acest caz s-au generat mai multi rezolventi decît în cazul în care algoritmul a utilizat si conceptul de clauza ordonata.

3.7 Strategia rezolutiei liniare

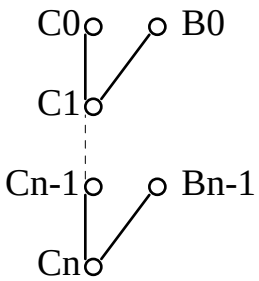
Strategia rezolutiei liniare este o metoda simpla si relativ eficienta pentru demonstrarea teoremelor. Aceasta strategie a fost utilizata în numeroase implementari si o varianta a ei, strategia liniara de intrare, sta la baza functionarii oricarei implementari a limbajului Prolog [Clocksin,Melish,1981].

Definitie. Fie S o multime de clauze si o clauza $C_0 \in S$. O *deductie liniara* a unei clauze C_n din S pornind de la clauza C_0, este o deductie pentru care:

(1) $C_{i+1} = rez(C_i, B_i)$, $i = 0,1,...,n-1$. C_i se numeste *clauza centrala*, iar B_i *clauza laterala*

(2) B_i fie apartine lui S, fie este egal cu C_j, $j < i$ sau este egal cu o instanta a lui C_j, $j < i$.

Deductia liniara se poate reprezenta grafic intuitiv astfel:



Din aceasta reprezentare grafica se observa de ce $C_0, C_1, ..., C_n$ se numesc clauze centrale, iar $B_0, B_1, ..., B_{n-1}$ clauze laterale. Clauza C_0 se numeste clauza de pornire.

3.7.1 Rezolutia de intrare si rezolutia unitara

În momentul alegerii unei strategii, se doreste, evident, ca aceasta sa fie completa dar si eficienta. Cîteodata se renunta la completitudine pentru eficienta. Daca o rafinare a unei strategii este eficienta si suficient de puternica pentru a demonstra o clasa mare de teoreme, chiar daca aceasta strategie nu este completa, ea poate fi utila.

Rezolutia de intrare si rezolutia unitara sînt mult mai eficiente ca rezolutia liniara, dar nu sînt complete. Rezolutia de intrare este echivalenta cu rezolutia unitara, iar teoremele ce pot fi demonstrate utilizînd rezolutia de intrare pot fi demonstrate si pe baza rezolutiei unitare. Prin conventie, fiecare clauza din multimea initiala de clauze S se numeste clauza de intrare.

Definitie. O *rezolutie de intrare* este o rezolutie în care una din cele doua clauze care rezolva este o clauza de intrare. O *deductie de intrare* este o deductie în care toate rezolutiile sînt rezolutii de intrare.

Observatie. O deductie de intrare este o deductie liniara în care fiecare clauza laterala este o clauza de intrare.

Definitie. O *rezolutie unitara* este o rezolutie în care un rezolvent este obtinut utilizînd cel putin o clauza unitara sau un factor unitar al unei clauze. O *deductie unitara* este o deductie în care toate rezolutiile sînt rezolutii unitare.

Observatie. Rezolutia unitara este în esenta o extindere a regulii literalului unic a lui Davis si Putnam din logica propozitiilor. Aceasta strategie rezolutiva este eficienta deoarece se obtin clauze din ce în ce mai mici.

Teorema. *Echivalenta între rezolutia unitara si rezolutia de intrare*. Exista o deductie a clauzei vide prin deductie unitara daca si numai daca exista o deductie a clauzei vide prin deductie de intrare.

Exemplu. Fie setul de clauze S:

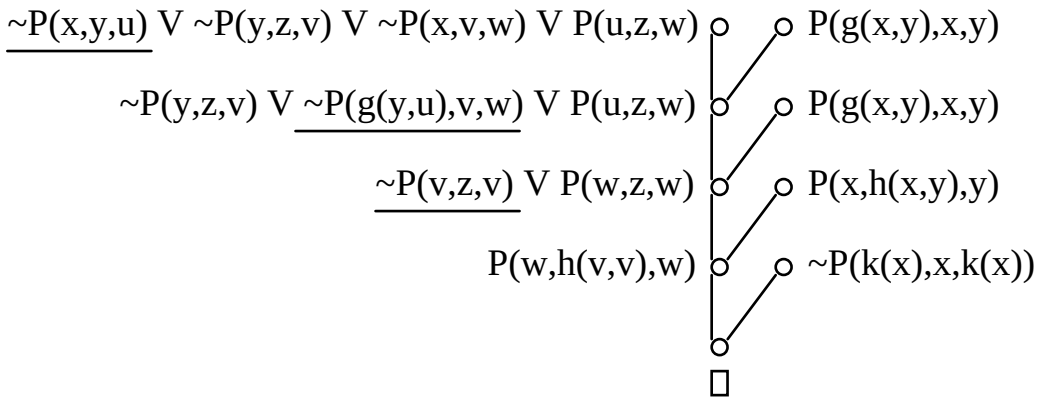
(1) $\sim P(x,y,u) \vee \sim P(y,z,v) \vee \sim P(x,v,w) \vee P(u,z,w)$

(2) $P(g(x,y),x,y)$

(3) $P(x,h(x,y),y)$

(4) $\sim P(k(x),x,k(x))$.

Urmatoarea demonstratie este o deductie de intrare a clauzei vide care arata ca S este o multime de clauze inconsistente. Literalii din clauzele centrale care rezolva au fost subliniati.



3.7.2 Rezolutia liniara cu utilizarea clauzelor ordonate

Introducerea clauzelor ordonate creste eficienta rezolutiei liniare. Spre deosebire de rezolutia semantica, introducerea conceptului de clauze ordonate în strategia rezolutiei liniare mentine completitudinea strategiei.

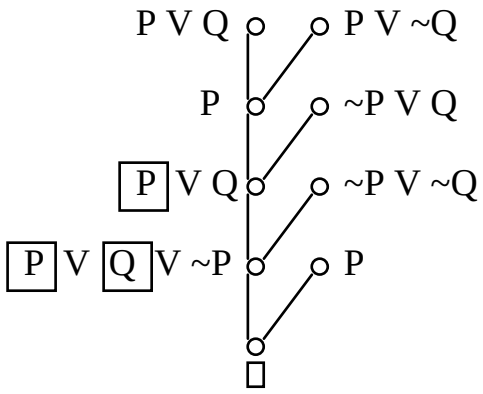
Un alt concept util în cresterea eficientei rezolutiei liniare este acela al utilizarii informatiei continute de literalii care au rezolvat. În procesul rezolutiv acesti literali sînt eliminati, dar daca ei nu se elimina, informatia literalilor care au rezolvat poate fi utilizata pentru a îmbunatati performantele rezolutiei liniare. Aceasta informatie poate fi folosita drept criteriu de alegere a clauzei care rezolva fie din multimea clauzelor de intrare, fie dintre clauzele centrale produse anterior. Informatia literalilor care rezolva este memorata prin pastrarea literalului din clauza centrala care a rezolvat si încadrarea acestuia, considerînd clauzele ordonate.

Exemplu. Fie clauzele $C_1 = P \vee Q$ si $C_2 = \sim Q \vee R$. Rezolventul celor doua clauze este $C = rez(C_1, C_2) = P \vee [Q] \vee R$, unde [Q] este literalul încadrat care a rezolvat.

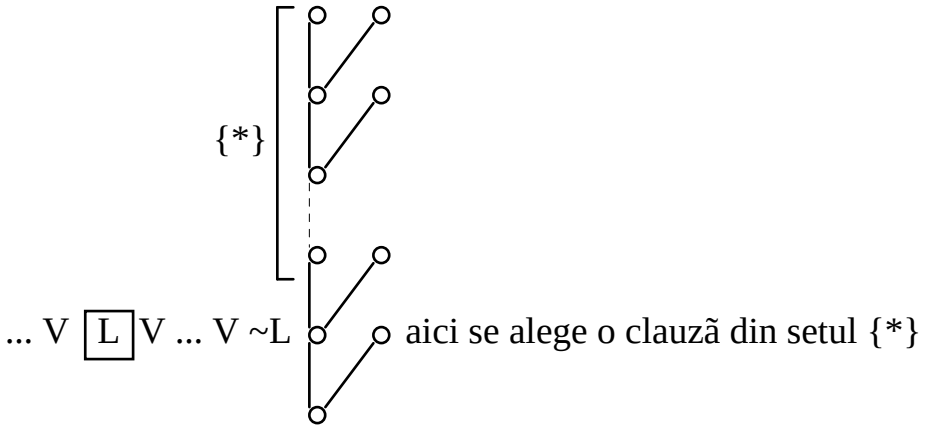
Asa cum se observa din exemplu, literalul încadrat este lasat la locul lui în rezolvent, deoarece clauzele sînt ordonate. Literalii încadrati vor fi pastrati în toate cazurile în care exista cel putin un literal neîncadrat dupa cel încadrat. Dupa ce au rezolvat, literalii încadrati nu mai participa la rezolutie în continuare. Ei sînt utilizati numai pentru a ghida selectia ulterioara a literalilor care rezolva.

Exemplu. Fie setul de clauze $S = \{P \vee Q, P \vee \sim Q, \sim P \vee Q, \sim P \vee \sim Q\}$. Sa se demonstreze nerealizabilitatea acestui set de clauze utilizînd rezolutia liniara cu clauze ordonate. Arborele de

derivare a clauzei vide, cu clauza de pornire $P \vee Q$, este prezentat în continuare. Literalii care rezolva si care sînt urmati de literali neîncadrati nu mai sînt eliminati din clauzele centrale si se încadreaza.



Clauza $[P] \vee [Q] \vee \sim P$ are o proprietate interesanta: ultimul literal este complementar cu un literal încadrat. În acest caz, clauza laterala a pasului urmator este întotdeauna o clauza centrala obtinuta anterior în rezolutie. Rezolvînd $[P] \vee [Q] \vee \sim P$ cu P, se obtine $[P] \vee [Q]$ si cum nu mai exista literali neîncadrati dupa acestia, [P] si [Q] dispar, deci rezulta clauza vida. Din acest exemplu se poate sintetiza criteriul de alegere a clauzei laterale ce rezolva în rezolutia liniara. De fiecare data cînd se genereaza o clauza centrala în care ultimul literal este complementar cu unul din literalii încadrati din clauza, se foloseste o clauza centrala generata anterior drept clauza laterala, în conditiile în care se utilizeaza clauze ordonate si rezolventi ordonati. Acest criteriu se poate prezenta grafic intuitiv astfel.



Definitie. Se numeste *clauza ordonata reductibila*, o clauza ordonata în care ultimul literal unifica cu negarea unui literal încadrat din clauza.

Definitie. Fie C o clauza ordonata reductibila. Fie L ultimul literal din C care unifica cu negarea unui literal încadrat [L'] din C, avînd cel mai general unificator β. Se numeste *clauza ordonata redusa*, clauza ordonata obtinuta din clauza $C\beta$ prin eliminarea literalului $L\beta$ din $C\beta$ si eliminarea tuturor literalilor încadrati ce nu mai sînt urmati de literali neîncadrati în clauza $C\beta$.

Definitie. Fie S o multime de clauze ordonate si $C_0 \in S$ o clauza ordonata din S. O *OL-deductie* a clauzei C_n din S, pornind de la clauza C_0, este o deductie ce satisface urmatoarele proprietati:

(1) $C_{i+1} = \text{rezord}(C_i, B_i)$, $i = 0,1,\ldots,n-1$. C_i se numeste *clauza centrala ordonata*, iar B_i *clauza laterala ordonata*. Literalul $L_i \in C_i$ care rezolva este ultimul literal din C_i.

(2) B_i fie apartine lui S, fie este o instanta a unei clauze C_j, $j < i$. B_i este o instanta a lui C_j, $j < i$ daca si numai daca C_i este o clauza ordonata reductibila. În acest caz C_{i+1} este clauza ordonata redusa a clauzei C_i.

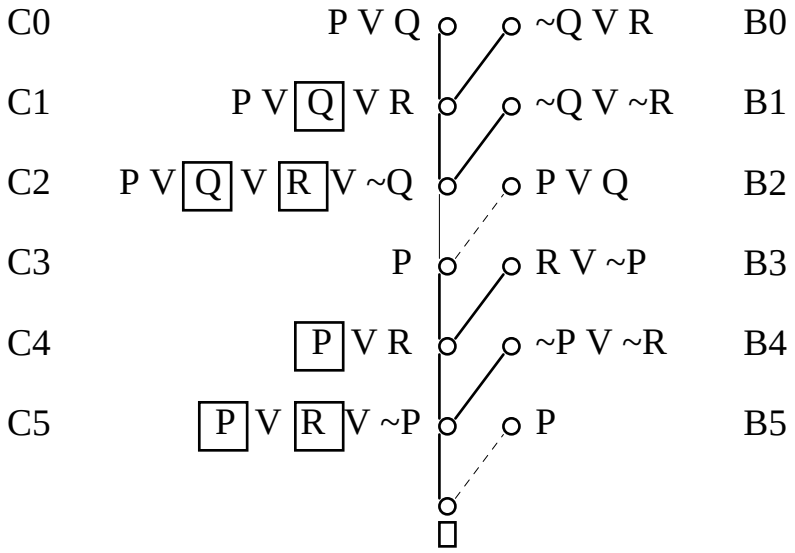
(3) nu exista nici o tautologie în deductie.

Lema. Într-o OL-deductie, daca C_i este o clauza ordonata reductibila, atunci exista o clauza ordonata centrala C_j, $j < i$, astfel încît clauza ordonata redusa C_{i+1} a lui C_i este $C_{i+1} = \text{rezord}(C_i, \text{inst}(C_j))$.

Demonstrarea acestei leme, bazata pe notiunea de clauza ordonata, este propusa cititorului.

Utilizarea clauzelor ordonate si a informatiei literalilor încadrati în rezolutia liniara are o consecinta importanta din punct de vedere al implementarii rezolutiei liniare. Nu mai este necesara memorarea clauzelor intermediare, fiind suficient sa se memoreze clauza curenta si setul initial de clauze S. În plus, introducerea clauzelor ordonate reductibile reduce numarul de rezolventi posibili, deoarece, odata detectata o astfel de clauza, nu se mai aplica rezolutia, ci se trece direct la clauza ordonata redusa.

Exemplu. Fie setul de clauze $S = \{P \vee Q, \sim Q \vee R, R \vee \sim P, \sim Q \vee \sim R, \sim P \vee \sim R\}$.



Clauzele C_2 si C_5 sînt clauze ordonate reductibile. Pentru aceste clauze nu mai este necesara aplicarea rezolutiei, deoarece se poate obtine direct clauza ordonata redusa pentru fiecare dintre

cele doua clauze. Din aceasta cauza, clauzele B_2 si B_5 egale cu C_0, respectiv C_3, sînt trecute cu linii punctate, pentru a pune în evidenta faptul ca ele sînt clauze centrale anterioare.

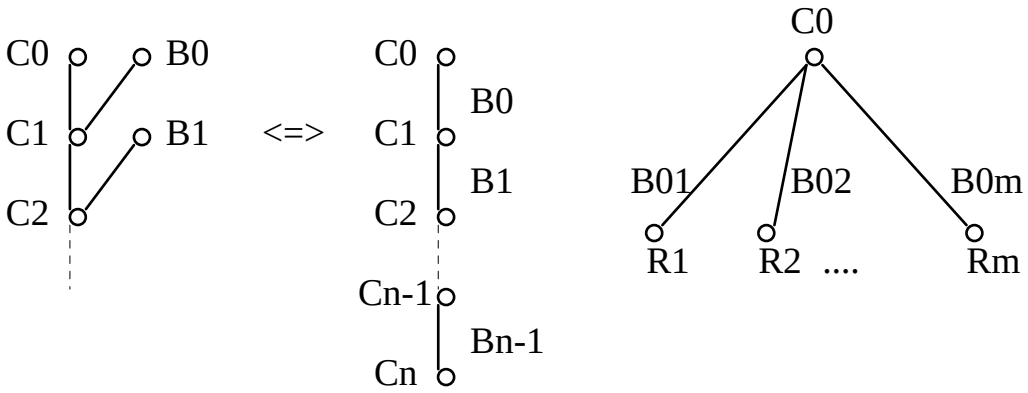
Teorema. *Completitudinea OL-rezolutiei*. Fie S o multime de clauze ordonate nerealizabile si o clauza $C \in S$. Daca $S - \{C\}$ este realizabila, atunci exista o OL-deductie a clauzei vide din S pornind de la clauza C.

Observatii:

- Se poate demonstra usor ca rezolutia bazata pe OL-deductie implica si strategia multimii suport. Deci teorema completitudinii OL-rezolutiei implica teorema completitudinii strategiei multimii suport.
- Desi rezolutia liniara este simplu de implementat, în cele mai multe cazuri exista diverse posibilitati de alegere a clauzelor ordonate laterale care rezolva.

3.7.3 Implementarea rezolutiei liniare

Odata ce s-a ales o clauza de pornire C_0, se pot obtine diversi rezolventi $R_1,...,R_m$, corespunzatori posibilelor clauze laterale ce rezolva cu C_0: $B_{01},...,B_{0m}$. Fiecare rezolvent R_i $(1 \le i \le m)$ este o posibila clauza centrala ce poate duce la deductia clauzei vide. Pentru fiecare R_i, trebuie deci obtinuti toti rezolventii posibili R_{i+1} si asa mai departe. Acest proces se poate reprezenta grafic convenabil astfel:



Din punct de vedere conceptual, procesul generarii tuturor rezolventilor posibili în rezolutia liniara ordonata utilizînd informatia literalilor care rezolva, si al gasirii clauzei vide poate fi vazut ca dezvoltarea unui spatiu de cautare în care starea initiala este clauza de pornire C_0, operatorii care realizeaza tranzitiile între stari sînt toate clauzele laterale ce rezolva cu clauza centrala curenta (starea curenta), iar starile sînt rezolventii astfel obtinuti. Starea finala cautata este rezolventul clauza vida.

În aceste conditii se pot aplica algoritmii de cautare a solutiei în spatiul starilor. Daca solutia exista, atunci ea este reprezentata de drumul rezolutiv spre clauza vida. În continuare se prezinta adaptarea algoritmului de cautare a solutiei pentru solutia problemei reprezentata prin spatiul

starilor, în cazul implementarii strategiei rezolutiei liniare. Algoritmul utilizeaza doua liste: TERITORIU, reprezentînd lista starilor expandate si FRONTIERA, reprezentînd lista starilor explorate. Se indica ambele variante de cautare neinformata, respectiv cautarea pe nivel, corespunzatoare pasului iv al algoritmului si cautarea în adîncime corespunzatoare pasului iv'. O prezentare detaliata a acestor algoritmi se poate gasi în Barr,s.a. [1982], Pearl [1984] si Florea [1993].

Algoritm: Implementarea rezolutiei liniare. Versiunea I.

1. Creeaza listele FRONTIERÃ $\leftarrow \{C_0\}$ si TERITORIU $\leftarrow \{\ \}$
2. **daca** FRONTIERÃ = { }
   atunci întoarce INSUCCES /* nu exista demonstratie */
3. Elimina prima clauza C din FRONTIERA si insereaz-o în TERITORIU
4. Expandeaza clauza C
   4.1. Identifica toate clauzele $B_1,...,B_m$ din S care pot fi clauze laterale pentru C
   4.2. **pentru** fiecare B_j $(1 \le j \le m)$ **executa**
      4.2.1. Stabileste legatura $B_j \rightarrow C$
      4.2.2. **pentru** fiecare R_i = rezord(C, B_j) **executa**
         i. **daca** R_i este reductibila
            atunci $R_i \leftarrow R_i$ redusã
         ii. Stabileste legatura $R_i \rightarrow C$
         iii. **daca** $R_i = \square$
            atunci întoarce SUCCES /* teorema e demonstrata */
         iv. Insereaza R_i în FRONTIERA, *la sfîrsit* /* parcurgere pe nivel */
      (iv'. Insereaza R_i în FRONTIERA, *la început* /* parcurgere în adîncime */
)
5. **repeta de la** 2

sfîrsit.

Observatii:

- Parcurgerea pe nivel este garantata sa gaseasca demonstratia minimala din S cu clauza de pornire C, daca o astfel de demonstratie exista. O demonstratie este minimala, daca contine numarul minim necesar de rezolutii pentru a deduce clauza vida.
- Evident trebuie pusa si o conditie de oprire pentru cazul în care nu se poate demonstra teorema. Asa cum s-a subliniat de mai multe ori, logica cu predicate de ordinul I nu este decidabila, ci semidecidabila, deci pentru formulele care nu sînt teoreme, algoritmul poate cicla.

- Pentru cazul în care nu exista o demonstratie sau pentru cazul în care parcurgerea se face în adîncime, trebuie introdusa o adîncime maxima de cautare AdMax. Aceasta implica introducerea pasului 3' dupa pasul 3.

  3' **daca** ad(C) > AdMax

  atunci repeta de la 2

  Adîncimea unei clauze C, ad(C), se defineste recursiv astfel:

  (1) $ad(C_0) = 0$

  (2) **daca** ad(C) = k **si** R = rezord (C, B)

  atunci ad(R) = k + 1

  Se poate deduce de aici ca lungimea unei demonstratii prin respingere rezolutiva este adîncimea clauzei vide.

- Daca exista o deductie a clauzei vide în cel mult AdMax pasi, atunci algoritmul de parcurgere în adîncime, în care se adauga pasul 3', va gasi o astfel de deductie.

- În algoritm s-a considerat ipoteza spatiului de cautare arbore si nu graf. Prezenta listei TERITORIU în algoritm, în acest caz, este utila numai daca intereseaza refacerea drumului rezolutiv. În cazul în care intereseaza numai obtinerea clauzei vide, deoarece rezolventii generati drept clauze centrale nu mai trebuie memorati, aplicîndu-se notiunea de clauze reductibile, lista TERITORIU devine inutila si poate fi eliminata. De asemenea, se pot elimina, în acest caz, si pasii de stabilire a legaturilor între rezolventi si clauzele din care acestia au provenit, deci pasii 4.2.1 si ii. În cazul în care spatiul de cautare este graf, situatie perfect posibila în respingerea rezolutiva, se pastreaza lista TERITORIU si se verifica, dupa obtinerea fiecarui rezolvent, daca acesta a mai aparut sau nu în TERITORIU sau FRONTIERA. [a]i de aceasta data, cu penalizari de eficienta, se poate renunta la lista TERITORIU daca se introduce testul de nedepasire a adîncimii maxime AdMax. Acest test va împiedica ciclurile infinite între rezolventi identici.

Algoritmul se poate modifica, memorînd, în lista FRONTIERA, perechile posibile de rezolventi (C, B_j). Adîncimea maxima de cautare este oricum binevenita pentru cazul în care nu exista demonstratie. Respingerea prin rezolutie este un caz tipic în care spatiul de cautare poate fi infinit. Aceste modificari sînt incluse în urmatoarea varianta de algoritm.

Algoritm: Implementarea rezolutiei liniare. Versiunea II.

1. Considera C_0

   Gaseste toate clauzele laterale din S ce rezolva cu C_0: $B_{01},...B_{0r}$

   Creeaza lista FRONTIERÃ $\leftarrow \{(C_0, B_{01}),(C_0, B_{02}),...,(C_0, B_{0r})\}$

2. **daca** FRONTIERÃ = { }

   atunci întoarce INSUCCES /* nu exista demonstratie sau demonstratia nu poate fi gasita în AdMax pasi */

3. Elimina prima pereche (C, B) din FRONTIERA

4. **daca** ad(C) > AdMax

   atunci repeta de la 2

5. Genereaza rezolventii perechii (C,B)

   5.1. Gaseste toti rezolventii $R_1,...,R_m$ ai perechii (C,B)

   5.2. **pentru** fiecare R_i = rezord (C,B), $1 \le i \le m$ **executa**

      5.2.1. **daca** R_i este reductibila

         atunci $R_i \leftarrow R_i$ redusã

      5.2.2. **daca** $R_i = \square$

         atunci întoarce SUCCES /* teorema este demonstrata */

      5.2.3. Gaseste toate clauzele laterale din S care rezolva cu R_i: $B_{i1},...,B_{ik}$

      5.2.4. Introduce perechile (R_i, B_{ij}), $1 \le j \le k$ în FRONTIERA *la sfîrsit*

         /* parcurgere pe nivel */

   (5.2.4'. Introduce perechile (R_i, B_{ij}), $1 \le j \le k$ în FRONTIERA *la început*

         /* parcurgere în adîncime */)

6. **repeta de la** 2

sfîrsit.

3.7.4 Cresterea performantelor rezolutiei liniare

Cresterea performantelor strategiei rezolutiei liniare, în sensul reducerii numarului de rezolventi posibili, se poate face prin includerea strategiei eliminarii, strategia rezultata fiind în continuare completa, sau prin transformarea cautarii drumului spre clauza vida într-o cautare informata prin utilizarea functiilor euristice.

(a) Includerea strategiei eliminarii

Conform principiilor strategiei eliminarii, prezentate în Sectiunea 3.5.3, includerea acestei strategii în strategia rezolutiei liniare se poate face adaugînd la algoritmul prezentat în sectiunea anterioara, dupa pasul 5.2.2,pasul 5.2.2'.

5.2.2'. **daca** R_i este tautologie

atunci ignora R_i si pasii 5.2.3 si 5.2.4

daca R_i este subsumata de C, pentru un C dintr-o pereche $(C, B) \in$ FRONTIERÃ

atunci ignora R_i si pasii 5.2.3 si 5.2.4

(b) Utilizarea functiilor euristice

Reducerea numarului de rezolventi generati în demonstratie se poate face prin utilizarea unei strategii de cautare informata. În cazul demonstrarii teoremelor, deoarece se presupune costul fiecarei rezolutii constant, nu intereseaza calitatea solutiei, ci drumul cel mai scurt, în termeni de numar minim de stari generate, pina la solutie, în particular starea finala identificata de obtinerea clauzei vide. Din acest motiv se va utiliza un algoritm de cautare "best-first", cu o functie euristica de evaluare a celei mai promitatoare perechi (C,B) din punct de vedere al avansului spre solutie.

Pornind de la algoritmul prezentat în sectiunea anterioara, se asociaza fiecarei perechi (C,B) din FRONTIERA o valoare a functiei euristice corespunzatoare acestei perechi, iar în pasul 3 al algoritmului nu se alege prima pereche (C,B) din FRONTIERA, ci perechea (C, B, f_{min}) pentru care valoarea functiei euristice este minima.

Exista diverse definitii posibile ale functiei euristice. O prima functie de evaluare conduce la o strategie euristica numita *strategia preferintei clauzelor cu cei mai putini literali*. Se defineste lungimea unei clauze ordonate C ca fiind numarul de literali neîncadrati din C, notata cu lung(C). În aceste conditii, daca $R = \text{rezord}(C, B)$, atunci lung(R) indica numarul minim de rezolutii necesare pentru a obtine din R clauza vida si reprezinta o buna estimare a starii (C,B) din punct de vedere al avansului spre solutie.

În realitate nu trebuie sa se calculeze lung(R), ceea ce ar implica deja calculul tuturor rezolventilor posibili. Se poate face o estimare a lungimii lui R tinînd cont de faptul ca $\text{lung}(R) \leq \text{lung}(C) + \text{lung}(B) - 2$. Deci, în loc de a calcula lung(R), se poate folosi (C, B, f), unde $f = \text{lung}(C) + \text{lung}(B)$ si se selecteaza perechea cu valoarea f minima.

O alta posibilitate de construire a functiei euristice este urmatoarea: pentru o pereche (C,B), se defineste $h^*(C, B)$ ca fiind numarul de rezolutii dintr-o demonstratie minima care are C drept clauza de pornire si B clauza laterala. Cum se estimeaza $h^*(C, B)$?

Fie $h^*(C, B)$ o estimare a lui $h(C, B)$. Se presupune ca $h^*(C, B)$ se exprima astfel:

$$h^*(C, B) = w_0 + w_1 f_1(C, B) + \ldots + w_n f_n(C, B),$$

unde fiecare $f_i(C, B)$, $i = 1, n$, este o functie care depinde de C si B, numita *functie caracteristica* a clauzelor C si B, iar $w_i > 0$, $i = 1, n$, sînt ponderi (constante pozitive) asociate valorilor f_i. Se alege din FRONTIERA perechea $(C, B, h_{min}(C, B))$.

Functiile f_i, $i = 1,n$, deci caracteristicile clauzei centrale si clauzei laterale, sînt selectate "euristic" de cel care implementeaza strategia, în functie de relevanta lor în estimarea functiei $h(C,B)$. Unele dintre cele mai folosite functii se definesc astfel:

- numarul de literali neîncadrati din C
- numarul de literali încadrati din C
- numarul de clauze laterale ale clauzei C
- numarul de constante din ultimul literal al clauzei C
- numarul de simboluri functionale din ultimul literal din C
- numarul de variabile distincte din B si C
- lung(C) + lung(B) – 2
- ad(C)
- numarul de literali din B care au un literal complementar încadrat în C
- numarul de simboluri predicative distincte în C si B

În continuare se prezinta un posibil mod de calcul al constantelor w_i, $i = 1,n$. Se presupune ca, pentru un anumit exemplu, s-a colectat deja un set $(C_1,B_1),...,(C_q,B_q)$ pentru care se cunosc valorile reale $h(C_1,B_1),...,h(C_q,B_q)$. Atunci problema revine la a determina $w_0,w_1,...,w_n$ astfel încît suma

$$\sum_{i=1}^{q}[h^*(C_i,B_i) - (w_0 + w_1 f_1(C_i,B_i) + ... + w_n f_n(C_i,B_i))]^2 \qquad (1)$$

sa fie minima. Aceasta problema este de fapt problema estimarii celor mai mici patrate.

Fie matricele:

$$F = \begin{bmatrix} 1 & f_1(C_1,B_1) & ... & f_n(C_1,B_1) \\ 1 & f_1(C_2,B_2) & ... & f_n(C_2,B_2) \\ ... & & & \\ 1 & f_1(C_q,B_q) & ... & f_n(C_q,B_q) \end{bmatrix} \quad H = \begin{bmatrix} h^*(C_1,B_1 \\ h^*(C_2,B_2 \\ ... \\ h^*(C_q,B_q \end{bmatrix} \quad W = \begin{bmatrix} w_0 \\ w_1 \\ ... \\ w_n \end{bmatrix}$$

Atunci matricea W, care minimizeaza expresia (1), este data de relatia: $W = (F'*F)^{-1}F'*H$, unde F' este matricea transpusa a matricii F, iar $(F'*F)^{-1}$ este matricea inversa a matricii produs F'*F.

De obicei se fac calcule pentru una sau mai multe instante de baza ale unei probleme rezolvate, deci pentru care se cunosc valorile functiei h, apoi se utilizeaza coeficientii w_i calculati pentru cazul general al problemei sau pentru alte probleme cu o euristica similara.

3.7.5 Solutii de implementare

Se prezinta în continuare o solutie posibila de implementre în limbajul Lisp a algoritmului de respingere prin rezolutie, utilizînd rezolutia liniara cu clauze ordonate si literali încadrati.

În cadrul programului clauzele apar sub forma unor liste de literali, fiecare literal fiind o celula Lisp ce memoreaza în cîmpul CAR "semnul" literalului, iar în cîmpul CDR un predicat. Semnul unui literal este reprezentat de unul din simbolurile poz, neg sau inc, corespunzator unui literal pozitiv, negativ sau încadrat. Un predicat atomic este reprezentat în program printr-un atom Lisp, iar un predicat neatomic este reprezentat printr-o lista în care primul element este numele predicatului, un atom Lisp, urmatoarele elemente ale listei fiind termeni. S-a utilizat o structura a termenilor similara celei descrise în Sectiunea 3.4.3 pentru a se putea folosi functiile de unificare a termenilor prezentate în acea sectiune.

Programul utilizeaza predicatele *pozitiv-p*, *negativ-p* si *incadrat-p* pentru a distinge tipul unui literal si functiile *literal*, respectiv *predicat*, pentru a construi, respectiv a adresa, literalii. Functia *rezolvent-ordonat* calculeaza rezolventul ordonat al unei perechi de clauze ordonate, utilizînd pentru aceasta functiile:

- *sterge-literal* care sterge prima aparitie dintr-o clauza a unui literal specificat,
- *sterge-duplicate* care sterge dintr-o clauza specificata literalii duplicati si
- *elimina* care elimina dintr-o clauza specificata literalii încadrati care nu sînt urmati de literali neîncadrati.

Functia *rezolvent-redus* calculeaza rezolventul redus pentru un rezolvent specificat si este apelata împreuna cu predicatul *rezolva-p*, care verifica daca doua clauze ordonate rezolva, în cadrul functiei *respingere*. Aceasta din urma reprezinta efectiv implementarea în Lisp a algoritmului prezentat în Sectiunea 3.7.3. Functia *respingere* are ca parametri o clauza centrala, un set de clauze si o adîncime maxima admisa pentru o clauza. Functia utilizeaza lista FRONTIERA pentru a memora perechile de clauze care pot rezolva, perechi ce apar în timpul procesului de respingere. Împreuna cu fiecare pereche se memoreaza si adîncimea clauzei centrale pentru a putea fi comparata cu adîncimea maxima a unei clauze centrale, aceasta adîncime functionînd astfel ca un criteriu de eliminare a perechilor de clauze care "par sa nu duca spre solutie".

Functia *pereche-clauze* construieste un element al listei FRONTIERA pe baza parametrilor primiti: doua clauze si adîncimea clauzei centrale, iar functia *adincime-clauze* extrage adîncimea

clauzei centrale dintr-un element al listei FRONTIERA. Implementarea utilizeaza functiile de unificare a termenilor prezentate în Sectiunea 3.4.3.

```
(load "termunif")

(defun pozitiv-p (literal)
  "Intoarce t daca literal este un literal pozitiv."
  (and (consp literal) (eq (first literal) 'poz)))

(defun negativ-p (literal)
  "Intoarce t daca literal este un literal negativ."
  (and (consp literal) (eq (first literal) 'neg)))

(defun incadrat-p (literal)
  "Intoarce t daca literal este un literal incadrat."
  (and (consp literal) (eq (first literal) 'inc)))

(defun literal (predicat &optional (semn 'poz))
  "Intoarce literalul (semn predicat)."
  (cons semn predicat))

(defun semn (literal)
  "Intoarce semnul unui literal."
  (first literal))

(defun semn-complementar (literal)
  "Intoarce semnul complementar sau nil daca literalul este incadrat."
  (cond ((eq (semn literal) 'poz) 'neg)
        ((eq (semn literal) 'neg) 'poz)))

(defun predicat (literal)
  "Intoarce predicatul unui literal."
  (rest literal))

(defun sterge-duplicate (clauza)
  "Intoarce clauza fara literali duplicati."
  (let (clauza1)
    (dolist (literal clauza (reverse clauza1))
      (unless (member literal clauza1 :test #'equal)
        (push literal clauza1)))))
```

```
(defun elimina (clauza)
  "Intoarce clauza din care au fost eliminati toti
  literalii incadrati neurmati de literali neincadrati."
  (do ((clauza1 (reverse clauza) (rest clauza1)))
      ((not (incadrat-p (first clauza1))) (reverse clauza1))))

(defun sterge-literal (literal clauza)
  "Intoarce clauza din care a fost eliminat literalul."
  (do* ((C clauza (rest C))
        (L (first C) (first C))
        clauza-rezultat)
       ((or (equal L literal) (null C)) (append clauza-rezultat (rest C)))
    (push L clauza-rezultat)))

(defun rezolvent-ordonat (C B)
  "Intoarce rezolventul ordonat al perechii de clauze C B."
  (let* ((CR (reverse C))
         (ultim-C (first CR))
         (literal-B (literal (predicat ultim-C) (semn-complementar ultim-C))))
    (elimina
      (sterge-duplicate
        (append (reverse (cons (literal (predicat ultim-C) 'inc) (rest CR)))
                (sterge-literal literal-B B))))))

(defun rezolvent-redus (rezolvent)
  "Intoarce rezolventul redus al clauzei rezolvent
  sau nil in cazul in care acesta nu exista."
  (let ((ultim (first (last rezolvent)))
        literal)
    (elimina
      (when (and (negativ-p ultim)
                 (member (setf literal (literal (predicat ultim) 'inc)) rezolvent
                         :test #'equal))
        (reverse (rest (reverse (sterge-literal literal rezolvent))))))))

(defun rezolva-ordonat-p (C B)
  "Intoarce o substitutie daca clauzele ordonate C si B pot rezolva,
  t daca rezolva fara substitutie si nil in caz contrar."
  (do* ((ultim-C (first (last C)))
        (B-aux B (rest B-aux))
```

```
          (literal-B (first B-aux) (first B-aux))
          succes mgu)
         ((or (null B-aux) succes) (if succes (if mgu mgu t) nil))
    (when (eq (semn-complementar ultim-C) (semn literal-B))
      (multiple-value-setq
        (succes mgu)
        (unifica (predicat ultim-C) (predicat literal-B))))))

(defvar *AD-MAX* 0
  "Adincimea maxima admisa a clauzei vide.")

(defun pereche-clauze (C B adincime-C)
  "Intoarce o celula reprezentind o pereche de
  clauze impreuna cu adincimea clauzei centrale."
  (cons (cons C adincime-C) B))

(defun adincime-clauze (pereche)
  "Intoarce adincimea unei perechi de clauze."
  (cdar pereche))

(defun substituie-variabile-in-clauza (clauza mgu)
  "Intoarce clauza in care variabilele au fost substituite."
  (mapcar #'(lambda (L)
              (literal (substituie-variabile (predicat L) mgu) (semn L)))
            clauza))

(defun respingere (C0 S &optional (*AD-MAX* 10))
  "Intoarce t daca clauza vida rezulta din clauza C0 si setul de clauze S,
  nil in caz contrar."
  (let ((FRONTIERA
         (delete nil
          (mapcar #'(lambda (B0i)
                      (let ((mgu (rezolva-ordonat-p C0 B0i)))
                        (when mgu
                          (if (eq mgu t)
                            (pereche-clauze C0 B0i 0)
                            (pereche-clauze (substituie-variabile-in-clauza C0 mgu)
                                            (substituie-variabile-in-clauza B0i mgu)
                                            0)))))
                  S))))
```

```
(when FRONTIERA
  (do* ((FRONT FRONTIERA)
        (pereche (first FRONT) (first FRONT))
        Ri Ri-redus adincime SUCCES)
       ((or (null FRONT) SUCCES) SUCCES)
    (setf FRONT (rest FRONT))
    (when (< (setf adincime (adincime-clauze pereche)) *AD-MAX*)
      (setf Ri (rezolvent-ordonat (caar pereche) (rest pereche)))
      (setf Ri-redus (rezolvent-redus Ri))
      (when Ri-redus (setf Ri Ri-redus))
      (unless (when (null Ri) (setf SUCCES t))
        (mapc #'(lambda (Bik)
                  (let ((mgu (rezolva-ordonat-p Ri Bik)))
                    (when mgu
                      (if (eq mgu t)
                        (push (pereche-clauze Ri Bik (1+ adincime)) FRONT)
                        (push (pereche-clauze
                                (substituie-variabile-in-clauza Ri mgu)
                                (substituie-variabile-in-clauza Bik mgu)
                                (1+ adincime))
                              FRONT)))))
                S)))))))
```

3.8 Complexitatea calculului logic

Problema realizabilitatii unei formule logice, numita în literatura de specialitate SAT, are un statut special din punct de vedere al complexitatii calculului. În 1971, S.A. Cook a dat o demonstratie directa a faptului ca SAT este o problema NP-completa: daca exista un algoritm polinomial pentru problema SAT, atunci toate problemele din clasa NP pot fi rezolvate în timp polinomial. Prezentarea demonstratiei facute de Cook depaseste scopul acestei lucrari, dar ideea generala a demonstratiei poate fi enuntata. Cook a pornit de la modelul unei masini Turing ce poate executa orice procedura efectiva, asa cum s-a discutat în Sectiunea 1.2, într-un timp egal (în limitele unui factor polinomial) cu timpul de executie al procedurii pe calculator. Extinzînd modelul masinii Turing la masina Turing nedeterminista, modelul extins poate rezolva orice problema din clasa problemelor NP-complete. În continuare, Cook a descris fiecare caracteristica a masinii Turing nedeterministe în termenii formulelor logice care apar în problema SAT. În acest fel s-a stabilit o corespondenta între orice problema NP-completa (ce poate fi reprezentata ca un program pe o masina Turing nedeterminista) si o instanta a problemei SAT. O solutie a problemei SAT corespunde în acest moment simularii functionarii masinii.

În consecinta, orice problema SAT este NP-completa, deci exponentiala în raport cu n, numarul de variabile din formula. Acest rezultat este valabil atît pentru logica propozitionala, în care SAT este problema decidabila, cît si pentru logica cu predicate de ordinul I, în care SAT este problema semidecidabila.

Problema realizabilitatii unei formule ce contine clauze cu cel mult doi literali, în logica propozitionala, numita si problema 2SAT, este polinomiala. Problema SAT pentru clauze Horn în logica propozitionala este tot polinomiala, de timp liniar în raport cu n. Problema realizabilitatii unei formule ce contine clauze cu cel mult trei literali, în logica propozitionala, numita si 3SAT, este NP-completa.

3.9 Exercitii si probleme

1. Fie multimea de clauze $S = \{P(x), \sim Q(x) \vee R(f(y))\}$. Se cere:

   1) Sa se genereze H_0, H_1, H_2 si forma generala a multimii H, universul Herbrand al lui S.
   2) Sa se indice baza Herbrand a lui S.
   3) Sa se indice doua instante de baza ale clauzelor din S.

2. Fie clauza $C = \sim P(x) \vee Q(f(x))$ si fie urmatoarele trei interpretari:

   $I_1 = \{\sim P(a), \sim Q(a), \sim P(f(a)), \sim Q(f(a)), \sim P(f(f(a))), \sim Q(f(f(a)))...\}$

   $I_2 = \{P(a), Q(a), P(f(a)), Q(f(a)), P(f(f(a))), Q(f(f(a)))...\}$

   $I_3 = \{P(a), \sim Q(a), P(f(a)), \sim Q(f(a)), P(f(f(a))), \sim Q(f(f(a)))...\}$

   Indicati care din aceste interpretari satisface, respectiv falsifica, clauza C.

3. Fie multimea de clauze $S = \{P, \sim P \vee Q, \sim Q\}$. Se cere:

   1) Sa se construiasca un arbore semantic complet al lui S si sa se indice multimile de interpretari partiale, I(N), pentru toate nodurile N din arbore.
   2) Sa se construiasca un arbore semantic închis al lui S si sa se indice nodurile de infirmare si nodurile de inferenta.

4. Fie multimea de clauze $S = \{P(x), \sim P(x) \vee Q(x,a), \sim Q(y,a)\}$. Se cere:

   1) Sa se specifice H, universul Herbrand al multimii de clauze S.
   2) Sa se specifice baza Herbrand a multimii S.
   3) Sa se construiasca un arbore semantic complet al lui S.
   4) Sa se construiasca un arbore semantic închis al lui S.

5) Sa se specifice o interpretare ce falsifica S.

5. Fie multimea de clauze $S = \{P(x,a,g(x,b)), \sim P(f(y),z,g(f(a),b))\}$. Sa se gaseasca o multime S' de instante de baza ale lui S astfel încît S' sa fie nerealizabila.

6. Aceeasi problema pentru setul de clauze $S = \{P(x), Q(x,f(x)) \vee \sim P(x), \sim Q(g(y),z)\}$.

7. Sa se exprime în logica cu predicate de ordinul I urmatoarea problema definita prin premisele:

   1) Vamesii au controlat toti turistii care au intrat în tara si care nu erau VIP.

   2) Anumiti traficanti de droguri au intrat în tara si au fost controlati de traficanti de droguri.

   si concluzia

   3) Nici un traficant de droguri nu este VIP.

8. Se considera urmatoarele perechi de clauze:

   (a) $C = P(x,y) \vee Q(z)$ $D = Q(a) \vee P(b,b) \vee R(u)$

   Sa se determine daca C subsumeaza D.

   (b) $C = P(x,y) \vee R(y,x)$ $D = P(a,y) \vee R(z,b)$

   Sa se arate ca C nu subsumeaza D.

9. Fie clauzele $E_1 = A_1 \vee A_3$, $E_2 = A_2 \vee A_3$ si $N = \sim A_1 \vee \sim A_2 \vee A_3$, interpretarea $I = \{\sim A_1, \sim A_2, \sim A_3\}$ si ordonarea simbolurilor predicative $A_1 > A_2 > A_3$. Se cere:

   1) Sa se arate ca $\{E_1, E_2, N\}$ este o multime de coincidenta semantica.

   2) Sa se indice PI-rezolventul.

   3) Sa se arate ca $\{E_1, N\}$ si $\{E_2, N\}$ nu sînt multimi de coincidenta semantica.

10. Fie multimea de clauze $\{E_1, E_2, N\}$, unde $E_1 = \sim Q(z) \vee \sim Q(a)$, $E_2 = R(b) \vee S(c)$ si $N = Q(x) \vee Q(a) \vee \sim R(y) \vee \sim R(b) \vee S(c)$, ordonarea simbolurilor predicative $Q > R > S$ si interpretarea $I = \{Q(a), Q(b), Q(c), \sim R(a), \sim R(b), \sim R(c), \sim S(a), \sim S(b), \sim S(c)\}$. Sa se determine daca $\{E_1, E_2, N\}$ este multime de coincidenta semantica si sa se indice PI-rezolventul.

11. Sa se arate ca S este o teorema pe baza multimii de clauze $\{Q, R, \sim Q \vee \sim R \vee S\}$, utilizînd ordonarea $Q > R > S$ si hiperrezolutia pozitiva sau negativa .

12. Sa se demonstreze, utilizînd algoritmul rezolutiei semantice, ca plecînd de la premisele:

    (1) Anumiti pacienti apreciaza toti doctorii.

    (2) Nici un pacient nu apreciaza sarlatanii.

    se poate concluziona ca:

    (3) Nici un doctor nu este sarlatan.

13. Sa se demonstreze concluzia problemei 7, plecînd de la premisele specificate în problema, si utilizînd

    (1) algoritmul rezolutiei semantice;

    (2) rezolutia liniara cu clauze ordonate si informatia literalilor ce rezolva.

14. Fie urmatoarele premise:

    (1) Alin este corect.

    (2) Bob sau Constantin este corect.

    (3) Constantin nu este partener cu Alin.

    (4) Daca doi indivizi sînt corecti atunci ei sînt parteneri.

    Sa se arate ca Bob este partener cu Alin, prin metodele rezolutive indicate în problema precedenta.

15. Sa se scrie un program care implementeaza respingerea prin rezolutie, utilizînd rezolutia liniara cu clauze ordonate si literali încadrati, îmbunatatita prin introducerea strategiei eliminarii si a functiilor euristice, utilizînd o strategie de cautare informata, cu functia de evaluare $\text{lung}(C) + \text{lung}(B) - 2$.

3.10 Rezolvari

2. Se cunoaste ca o clauza C este satisfacuta de o interpretare I daca si numai daca este realizabila în acea interpretare. În plus, se poate demonstra ca:

    (a) O clauza C este satisfacuta de o interpretare I daca si numai daca toate instantele de baza ale lui C sînt satisfacute de interpretarea I.

    (b) O instanta de baza C' a unei clauze C este satisfacuta de o interpretare I daca si numai daca exista un literal de baza L' în C' astfel încît L este si în interpretarea I, deci $C' \cap I \neq \phi$.

(c) O clauza C este falsificata de o interpretare I daca si numai daca exista cel putin o instanta de baza C' a lui C astfel încît C' nu este satisfacuta de interpretarea I.

(d) O multime de clauze S este nerealizabila daca si numai daca pentru fiecare interpretare I, exista cel putin o instanta de baza C' a unei clauze C apartinînd lui S ($C \in S$) astfel încît C' nu este satisfacuta de interpretarea I.

Conform (a) si (b) interpretarea I_1 satisface clauza C deoarece orice instanta de baza a clauzei C $(\sim P(a) \vee Q(f(a)), \sim P(f(a)) \vee Q(f(f(a))),...)$ are un literal comun cu interpretarea I_1, respectiv literalii de forma $\sim P(a), \sim P(f(a)),...$.

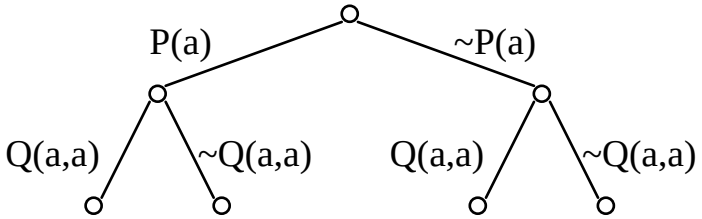
Analog, interpretarea I_2 satisface clauza C deoarece orice instanta de baza a clauzei C are un literal comun cu interpretarea I_2, respectiv literalii de forma $Q(f(a)), Q(f(f(a))),...$.

Din (c) rezulta ca interpretarea I_3 falsifica clauza C deoarece instanta de baza $C' = \sim P(a) \vee Q(f(a))$ nu este satisfacuta de interpretarea I_3, neavînd nici un literal comun cu interpretarea I_3.

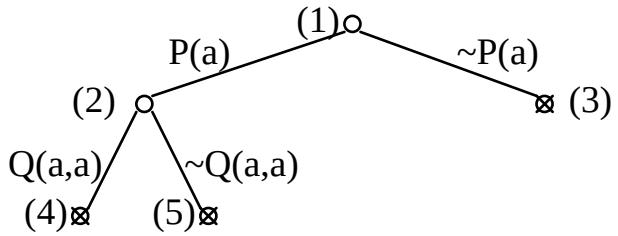
4. 1) Multimea constanta de nivel 0 a setului S este $H_0 = \{a\}$ si este egala cu universul Herbrand al setului de clauze S, deoarece clauzele componente nu contin functii.

2) Baza Herbrand a setului de clauze S este multimea atomilor de baza, i.e. multimea tuturor predicatelor ce apar în S, avînd ca termeni elemente ale universului Herbrand. Deci $A = \{P(a), Q(a,a)\}$.

3) Un arbore semantic al lui S este complet daca si numai daca pentru orice nod frunza al arborelui, calea de la radacina la el contine toate elementele din baza Herbrand, fie în forma directa, fie în forma negata. Deci urmatorul arbore este un arbore semantic complet al lui S.



4) Urmatorul arbore semantic este un arbore semantic închis al setului de clauze S deoarece toate ramurile sale sfîrsesc în noduri de infirmare.



Nodurile 3, 4 si 5 sînt noduri de infirmare deoarece:

i. pentru nodul 3, ~P(a) falsifica instanta de baza P(a) a clauzei P(x),

ii. pentru nodul 3, Q(a,a) falsifica instanta de baza ~Q(a,a) a clauzei Q(y,a) si

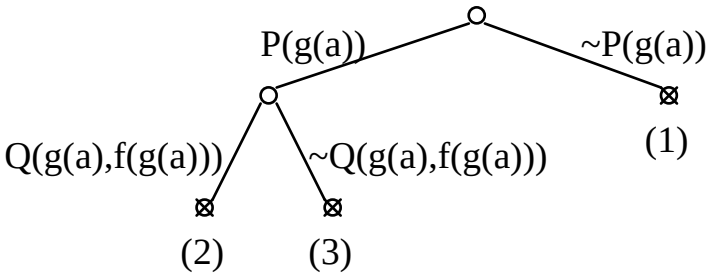
iii. pentru nodul 5, {P(a),~Q(a,a)} falsifica instanta de baza $\sim P(a) \vee Q(a,a)$ a clauzei $\sim P(x) \vee Q(x,a)$

5) O interpretare I care falsifica setul de clauze S este $I = \{\sim P(a), P(a) \vee \sim Q(a,a), Q(a,a)\}$.

5. O multime de instante de baza a lui S este $S' = \{P(f(a),a,g(f(a),b), \sim P(f(a),a,g(f(a),b)\}$, deoarece universul Herbrand este $H = \{a, b, f(a), f(b), g(a,b), g(b,b), f(f(a)),..., g(f(a),b),...\}$ De fapt sînt suficienti primii trei termeni, deoarece o instanta de baza a unei clauze C se obtine prin înlocuirea variabilelor din C cu membrii universului Herbrand al setului de clauze S ($C \in S$). În acest caz, S' a fost obtinut prin înlocuirile: $f(a)/x$, a/y si a/z.

6. O posibilitate de determinare a unei multimi de instante de baza nerealizabile este generarea unui arbore semantic închis T pentru setul de clauze S. Atunci multimea tuturor instantelor de baza ce sînt falsificate de nodurile de infirmare formeaza multimea dorita.

Pentru setul de clauze dat, multimile constante de nivel i sînt: $H_0 = \{a\}$, $H_1 = \{a, f(a), g(a)\}$, ..., universul Herbrand fiind $H = \{a, f(a), g(a), f(f(a)), f(g(a)),...\}$.



O multime de instante de baza nerealizabila a setului de clauze S este

$S' = \{P(g(a))\ (1), \sim Q(g(a), f(g(a)))\ (2), Q(g(a), f(g(a))) \vee \sim P(g(a))\}$

pentru fiecare instanta de baza specificîndu-se si nodul de infirmare care o falsifica.

7. O solutie posibila este urmatoarea:

Premise:

$(\forall x)(Intrat(x) \wedge \sim VIP(x) \rightarrow (\exists y)(Controlat(x,y) \wedge Vame^{o}(y)))$

$(\exists x)(Traficant(x) \wedge Intrat(x) \wedge (\forall y)(Controlat(x,y) \rightarrow Traficant(y)))$

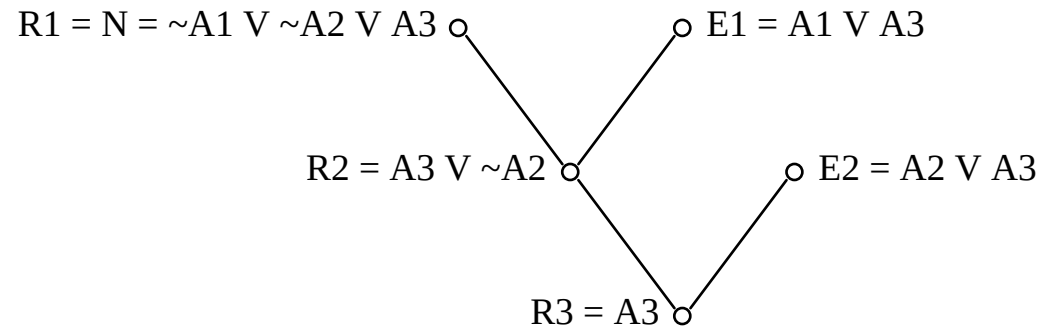
Concluzie:

$(\forall x)(\text{Traficant}(x) \rightarrow \sim \text{VIP}(x))$

9. 1) Setul de clauze $\{E_1, E_2, N\}$ este o multime de coincidenta semantica deoarece sînt îndeplinite conditiile definitiei:

   i. Electronii E_1 si E_2 sînt falsificati de interpretarea I.

   ii. Daca rezolventul $R_1 = N$, exista $R_2 = \text{rez}(R_1, E_1) = A_3 \vee \sim A_2 \vee A_3 = A_3 \vee \sim A_2$ si $R_3 = \text{rez}(R_2, E_2) = A_3$. În plus, literalii care rezolva sînt A_1 pentru $i = 1$ si A_2 pentru $i = 2$, cei mai mari din electronii care participa la rezolvare si PI-rezolventul R_3 este fals în interpretarea I.



   2) PI-rezolventul este deci $R_3 = A_3$.

   3) Multimile $\{E_1, N\}$ si $\{E_2, N\}$ nu sînt multimi de coincidenta semantica, deoarece $\text{rez}(E_1, N) = \sim A_2 \vee A_3$ este adevarat în interpretarea I si, analog, $\text{rez}(E_2, N) = \sim A_1 \vee A_3$ este adevarat în I. Daca ordonarea simbolurilor predicative se schimba în $A_3 > A_2 > A_1$, multimea $\{E_1, E_2, N\}$ nu mai este multime de coincidenta semantica, deoarece literalii care rezolva nu vor mai fi cei mai mari în electroni.

13. Transformînd formulele obtinute la problema 7 în forma clauzala, se obtine urmatorul set de clauze:

   (1) $\sim \text{Intrat}(x) \vee \text{VIP}(x) \vee \text{Controlat}(x, f(x))$

   (2) $\sim \text{Intrat}(x) \vee \text{VIP}(x) \vee \text{Vame}^o(f(x))$

   (3) Traficant(a)

   (4) Intrat(a)

   (5) $\sim \text{Controlat}(a, y) \vee \text{Traficant}(y)$

   (6) $\sim \text{Traficant}(x) \vee \sim \text{VIP}(x)$

   (7) $\sim \text{Traficant}(x) \vee \sim \text{Vame}^o(x)$

   1) Aplicînd algoritmul rezolutiei semantice pentru acest set de clauze, si ordonarea simbolurilor predicative $\text{Intrat} < \text{VIP} < \text{Controlat} < \text{Vame}^o < \text{Traficant}$, se obtin urmatoarele multimi initiale de clauze pozitive, respectiv nepozitive:

   $M = \{\text{Traficant}(a), \text{Intrat}(a)\}$ multimea clauzelor pozitive.

$N = \{VIP(x) \vee Controlat(x, f(x)) \vee \sim Intrat(x), VIP(x) \vee Vame^o(f(x)) \vee \sim Intrat(x),$
$Traficant(y) \vee \sim Controlat(a, y), \sim Traficant(x) \vee \sim VIP(x), \sim Traficant(x) \vee \sim Vame^o(x)\}$

$W_1 = \{VIP(a) \vee Controlat(a, f(a)), VIP(a) \vee Vame^o(f(a))\}$

$A_1 = W_1,\ B_1 = \phi$

$T = \{VIP(a) \vee Controlat(a, f(a)), VIP(a) \vee Vame^o(f(a))\}$

$M = T \cup M = \{Traficant(a), Intrat(a), VIP(a) \vee Controlat(a, f(a)), VIP(a) \vee Vame^o(f(a))\}$

Relaxînd conditia de ordonare a clauzelor se obtine multimea rezolventilor

$R = \{VIP(a) \vee Traficant(f(a)), VIP(a) \vee \sim Traficant(f(a))\}$

si respectiv multimile de clauze pozitive si nepozitive din R:

$A_0 = \{VIP(a) \vee Traficant(f(a))\},\ B_0 = \{VIP(a) \vee \sim Traficant(f(a))\}$

$W_1 = \phi,\ A_1 = \phi,\ B_1 = \phi$

$T = \{VIP(a) \vee Traficant(f(a))\}$

$M = \{Traficant(a), Intrat(a), VIP(a) \vee Controlat(a, f(a)), VIP(a) \vee Vame^o(f, a),$
$VIP(a) \vee Traficant(f(a))\}$

Relaxînd din nou conditia de ordonare a clauzelor se obtine multimea rezolventilor

$R = \{VIP(a) \vee \sim VIP(f(a)),\ VIP(a) \vee \sim Vame^o(f(a))\}$

si respectiv multimile de clauze pozitive si nepozitive din R:

$A_0 = \phi,\ B_0 = \{VIP(a) \vee \sim VIP(f(a)), VIP(a) \vee \sim Vame^o(f(a))\}$

$W_1 = \{VIP(a)\},\ A_1 = \{VIP(a)\},\ B_1 = \phi$

$T = A_0 \cup A_1 = \{VIP(a)\}$

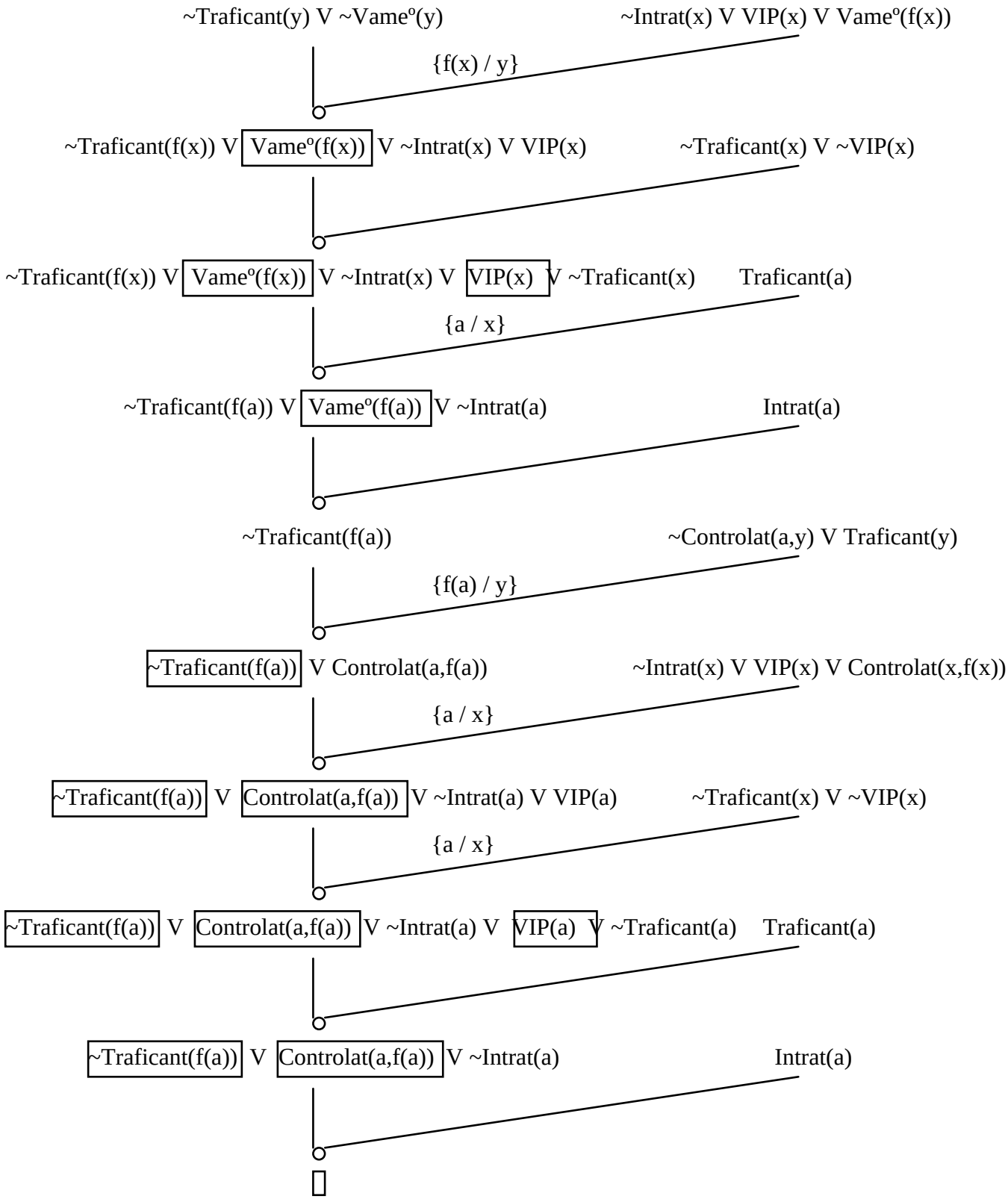
$M = \{Traficant(a), Intrat(a), VIP(a), VIP(a) \vee Controlat(a, f(a)), VIP(a) \vee Vame^o(f(a)),$
$VIP(a) \vee Traficant(f(a))\}$

$R = \{\sim Traficant(a)\}$

$A_0 = \phi,\ B_0 = \{\sim Traficant(a)\}$

$W_1 = \{\square\},\ A_1 = \{\square\},\ B_1 = \phi.$

2) Utilizînd rezolutia liniara cu clauze încadrate si informatia literalilor care rezolva, se obtine urmatoarea deductie a clauzei vide:


~Traficant(y) V ~Vameº(y)
~Intrat(x) V VIP(x) V Vameº(f(x))
{f(x) / y}
~Traficant(f(x)) V Vameº(f(x)) V ~Intrat(x) V VIP(x)
~Traficant(x) V ~VIP(x)
~Traficant(f(x)) V Vameº(f(x)) V ~Intrat(x) V VIP(x) V ~Traficant(x)
Traficant(a)
{a / x}
~Traficant(f(a)) V Vameº(f(a)) V ~Intrat(a)
Intrat(a)
~Traficant(f(a))
~Controlat(a,y) V Traficant(y)
{f(a) / y}
~Traficant(f(a)) V Controlat(a,f(a))
~Intrat(x) V VIP(x) V Controlat(x,f(x))
{a / x}
~Traficant(f(a)) V Controlat(a,f(a)) V ~Intrat(a) V VIP(a)
~Traficant(x) V ~VIP(x)
{a / x}
~Traficant(f(a)) V Controlat(a,f(a)) V ~Intrat(a) V VIP(a) V ~Traficant(a)
Traficant(a)
~Traficant(f(a)) V Controlat(a,f(a)) V ~Intrat(a)
Intrat(a)
□