Következtető-csomópont:

A csomópont következtető-csomópontja a zárt szemantikus-fának, ha minden közvetlen leszármazottja ellentmondás csomópont.

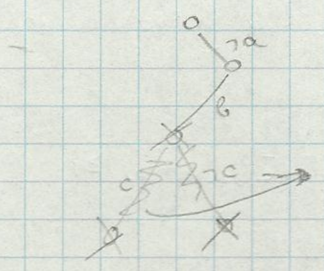
$(a \cup b) \wedge (\neg b \cup c) \wedge (a \cup \neg b \cup \neg c)$

ellentétes literálpár

rezolúció: az ellentéteseket elhagyjuk, a maradékot összevonjuk

$(a \cup \neg b)$

A rezolvens ellentmondás-csomópont lesz



ezeket le lehet törölni ⇒ csökken a fa mérete

Beszélhetünk üres szemantikus-fáról. Az üres clause-t tartalmazó clause set használható. (Bármit írnák, rögtön hamis lenne)

T: A clause set kielégíthetetlen ⟺, ha a rezolúciós metódus által levezethető az üres clause.

Clause set reprezentáció

pl.: $(a \vee \bar{b}) \rightarrow$ ez clause set

Ábrázolása: I: egydimenziós

1) logikai: A neki megfelelő KNF-ban leíró forma

(KNF: zárójelek között csak $\wedge$, zárójelben csak $\vee$ szerepelhet)

2) halmaz alapú: $\{\{a,b\}, \{\neg b, c\}, \{a, \neg b, \neg c\}, \{a, \neg b\}\}$

itt literálhalmazok vannak, a literálhalmaz a clause-nak felel meg.

3) algebrai: $(a+b)\cdot(\bar{b}+c)(a+\bar{b}+\bar{c})(a+\bar{b}) \geq 1$ (flip)

↳ error kiszűrhető

II. kétdimenziós:

1) Mátrix alapú:

sorok: clause, oszlopok: változók

1.	2.	3. változó
+	+	x
x	−	+
+	−	−
+	−	x
↑	↑	↑
a	b	c

PURE-LITERÁL: Ha 1 változó csak ⊕-ban v. csak ⊖-ban fordul elő. (Pl.: 'a' változó)

Ilyenkor felkiáltó, h. $a = true$.

UNIT CLAUSE: Ha 1 clause-ban csak 1 literál van.

pl.: $(a) \wedge (b \vee c)$

+	x	x
x	+	+

EMPTY CLAUSE : üres - clause

$\{\{\}; \{ \dots \}\}$

logikaiba nem írható fel

x x x → empty clause
+ + x

Műveletek:

1) UNIT PROPAGÁCIÓ:

Ha : ,a' igaz → előző példába behelyettesítve (pl.: I/3.)

$(a \vee b) \wedge (\neg b \vee T) \wedge (a \vee \neg b \vee F) \wedge (a \vee \neg b) \rightarrow (a \vee b) \wedge (a \vee \neg b) \wedge (a \vee \neg b)$

($\vee F$: elhagyható)

($\neg b \vee T$: mivel $\wedge$-el -nél ez nem számít, elhagyható)

Unit propagáció $(S, a) := \{ C \setminus \{\neg a\} \mid a \notin C \}$, $C \in S$

S: clause set; a: literál

$\setminus\{\neg a\}$: ki kell venni a literál negáltját a clause setből.

$a \notin C$: a literál, amelyre feltettük, h igaz, azt elhagyhatjuk

5. előadás X. 11.

Pl.: boltba (x) :- süt (nap), jóredvemvan (x)

$B(x) \Leftarrow S(n) \wedge J(x)$

$S(n) \wedge J(x) \Rightarrow B(x)$

$\neg S(n) \vee \neg J(x) \vee B(x)$

goal

Boltba (karcsi)

$B(k)$

$\neg B(k)$

$(\neg S(n) \vee J(x) \vee B(x)) \quad (\neg B(k))$

$x = k$

$(\neg S(n) \vee \neg J(k))$

$C_1 \wedge \dots \wedge C_n \Rightarrow G$

$C_1 \wedge \dots \wedge C_n \wedge \neg G$ → be kell látni, hogy ez ellentmondásos

Procedurális szemantika:

Van egy célista, ami kezdetben a goal részben lévő predikátumot tartalmaz.

Ha egy célra:

A célista egy predikátumhoz keresek egy olyan clause-t (felírásuk sorrendjében keresem), amelynek feje illeszkedik erre a célra, ha találok ilyet, akkor ezt a részcélt kitörlöm és helyébe beírom a megtalált clause-t.

Ha a célista üressé válik ⇒ megtaláltam a megoldást, a megoldást a mintaillesztés során kapott változó-helyettesítések képviselik.

Ha nincs első részcélra illeszkedő fej ⇒ vissza kell lépnem az előző állapotra és keresni egy új fejet illesztésre.

Példa:

Clause

apja (peter, kalman)

apja (kalman, zoltan)

nagyapja (x, y) :- apja (z, y), apja (x, z)

goal

nagyapja (peter, zoltan)

Célista:

Eredetben: (nagyapja (peter, zoltan))

találtunk egy olyan fejet, melyre az illeszkedik, kicseréljük a törzsre, lényegében

rezolválok:

(apja (z, zoltán), apja (peter, z))

az első felét nézem

(apja (peter, kalman))

()

az eredmény: Yes.

Példa:

clause

apja (peter, lajos)

apja (peter, kalman)

apja (kalman, zoltan)

nagyapja (x, y) :- apja (x, z), apja (z, y).

goal

nagyapja (peter, zoltan)

célista:

(nagyapja (peter, zoltan))

(apja (peter, z), apja (z, zoltan))

(apja (lajos, zoltan))

nilla ezt lépni, mert nincs jó fej

(apja (kalman, zoltan))

()

A prolog back-tracket használ.

Listák

domains

(a * jelenti a listát)

list = symbol *

szamlista = integer *

fej → [x ; x3] ← a lista farka, a farok mindig lista

A lista rekurzív adatszerkezet, rekurzívan kell feldolgozni.

Farokrekurzió: a rekurzív hívás az utolsó

[ ]

[1; [ ]] = [1]

[1; [2]] = [1,2]

[1; [2,3]] = [1,2,3]

domains

intlist = integer *

predicates

szumma (intlist, integer)

clauses

szumma ([ ], Y) :- Y = 0

← mintaillesztés

Ha nem üres a lista, szétszedjük fejre és farokra

szumma ([x | x1], Y) :- szumma (x1, Y2), Y = x + Y2

goal

szumma ([4,5], S).

(szumma ([4,5], S))

x = 4 , x1 = 5 , Y = S

(szumma ([5], Y2), S = 4 + Y2)

$x = 5$, $x_s = []$, $y = y_2$

(szumma $([], y_3)$, $y_2 = 5 + y_3$, $s = 4 + y_2$

$y = y_3$

($\underbrace{y_3 = 0}_{\text{igaz}}$, $y_2 = 5 + y_3$, $s = 4 + y_2$)

($y_2 = 5 + 0$, $s = 4 + y_2$)

($s = 4 + 5$)

($s = 9$)

()

SML-ben:

```
fun sum [] = 0 | sum (x::xs) = sum (xs) + x;
```

$f(x_1, x_2, \dots x_n)$

$pred(x_1, x_2, \dots, x_n, x_{n+1}) \equiv x_{n+1}$

Ø fgv-hez megtalálható a vele ekvivalens predikátum, úgy, hogy a predikátumnak eggyel több paramétere van, és ez egyenlő-e a fgv által visszaadott értékkel, ha a fgv-t az első n paramétere hívjuk meg.

Fgv-hívást mindig úgy szimulálok prologban, hogy az első n paraméter megegyezik a fgv-hívás paramétereivel, az n+1-dik paraméter pedig új változó, ebben kapom meg a fgv visszatérési értékét.