Automatsko rezonovanje – beleške sa predavanja Iskazno rezonovanje

Filip Marić

*Matematički fakultet,
Univerzitet u Beogradu

Proletnji semestar 2018.

Pregled

Bulova algebra logike

- Iskazna logika je moderna verzija Bulove (George Bool 1815-1864) algebre logike.
- Pre Bula, formalistički pristup je najprisutniji u algebri — apstraktna algebra je uveliko razvijena.
- Bul uviđa da je moguće algebarskim (pa čak aritmetičkim) izrazima označavati i logičke iskaze i uspostaviti formalni račun (nalik na algebarske račune) koji daje pravila operisanja sa ovako zapisanim tvrđenjima.
- Osnovni objekti su iskazi koji predstavljaju tvrđenja (koja mogu biti tačna ili netačna).
- Iskazi su atomički što govori o tome da se njihova unutrašnja struktura za sada ne analizira. Analiza unutrašnje strukture iskaza dovodi do bogatijih logika (npr. logike prvog reda).

Pregled

Sintaksa iskaznih formula

- Izrazi iskazne logike nazivaju se iskazne formule.
- Na nivou apstraktne sintakse, grade se od
  - iskaza:
    - logičkih konstanti $\top$ i $\bot$ i
    - iskaza (atoma, iskaznih slova, iskaznih promenljivih)
  - primenom logičkih veznika (npr. *i*, *ili*, *ne*, *ako . . . onda . . .* , *ako i samo ako*, . . .).

Sintaksa iskaznih formula

U literaturi se koristi različita konkretna sintaksa za iskazne formule.

Srpski	Engleski	Simbolički	ASCII	Još simbolički
netačno	false	$\bot$	`false`	0, F
tačno	true	$\top$	`true`	1, T
ne p	not p	$\neg p$	`~p`	$\overline{p}$, $-p$, $\tilde{p}$
p i q	p and q	$p \wedge q$	`p/\q`	pq, $p \cdot q$, $p\&q$
p ili q	p or q	$p \vee q$	`p\/q`	$p + q$, $p \vert q$
ako p onda q	p implies q	$p \Rightarrow q$	`p==>q`	$p \to q$, $p \supset q$
p akko q	p iff q	$p \Leftrightarrow q$	`p<=>q`	$p \leftrightarrow q$, $p \equiv q$, $p \sim q$

Konkretna sintaksa iskaznih formula

Definicija

Neka je dat (najviše prebrojiv) skup atoma P. Skup iskaznih formula je najmanji skup reči nad azbukom $P \cup \{\top, \bot, \neg, \wedge, \vee, \Rightarrow, \Leftrightarrow, (,)\}$ koji zadovoljava:

- *iskazna slova i logičke konstante su iskazne formule,*
- *Ako je A iskazna formula, onda je i $\neg(A)$ iskazna formula.*
- *Ako su A i B iskazne formule, onda su i $(A) \wedge (B)$, $(A) \vee (B)$, $(A) \Rightarrow (B)$ i $(A) \Leftrightarrow (B)$ iskazne formule.*

Konkretna sintaksa iskaznih formula

Alternativno,

Definicija

Neka je dat (najviše prebrojiv) skup atoma P. Skup iskaznih formula jednak je jeziku generisnom sledećom kontekstno slobodnom gramatikom:

$$
\begin{array}{rcl}
\mathit{formula} & \longrightarrow & \top \\
& | & \bot \\
& | & p \quad (\text{za svako } p \in P) \\
& | & \neg(\mathit{formula}) \\
& | & (\mathit{formula}) \wedge (\mathit{formula}) \\
& | & (\mathit{formula}) \vee (\mathit{formula}) \\
& | & (\mathit{formula}) \Rightarrow (\mathit{formula}) \\
& | & (\mathit{formula}) \Leftrightarrow (\mathit{formula})
\end{array}
$$

Primeri iskaznih formula

Primer

$$((p) \wedge (q)) \Rightarrow ((r) \Leftrightarrow (\neg(p)))$$

$$(x_1) \wedge ((x_2) \vee ((x_1) \Rightarrow (\neg((x1) \Leftrightarrow (\neg(x_3))))))$$

Izostavljanje zagrada

Iako odstupa od date definicije, uobičajeno je da se koristi konvencija da se zagrade u zapisu formula mogu izostaviti i to u skladu sa sledećim prioritetom operatora, od višeg ka nižem ($\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$), sa levom asocijativnošću operatora $\wedge$ i $\vee$ i desnom asocijativnošću operatora $\Rightarrow$ i $\Leftrightarrow$. Zagrade oko pojedinačnih iskaznih slova se takođe izostavljaju.

Primer

$$p \vee p \wedge q \Rightarrow q \vee r \vee s$$

je skraćeni zapis za

$$((p) \vee ((p) \wedge (q))) \Rightarrow (((q) \vee (r)) \vee (s))$$

Literali

Često se, kao elementarne, razmatraju veoma jednostavne formule: atomi i njihove negacije.

Definicija

Literal je ili atom (pozitivan literal) ili negacija atoma (negativan literal).

Definicija

Suprotan literal atoma p je njegova negacija $\neg p$, dok je suprotan literal negaciji atoma $\neg p$ sam atom p. Suprotni literal literala l označavaćemo sa $\bar{l}$.

Reprezentacija u programskim jezicima

- Za internu reprezentaciju formula u programskim jezicima mnogo je pogodnija apstraktna sintaksa.
- Apstraktna sintaksa razrešava dileme o prioritetima operatora. Dodatno, nekada se i različite asocijativnosti apstrahuju uvođenjem n-arnih umesto binarnih veznika $\wedge$ i $\vee$.
- Programi se obično dopunjuju modulima za parsiranje koji imaju zadatak da formule na ulazu zadate u nekoj konkretnoj sintaksi prevedu u internu apstraktnu reprezentaciju i modulima za lepo štampanje koji imaju zadatak da internu reprezentaciju prikažu u nekoj konkretnoj sintaksi.
- U programskim jezicima se iskazne formule mogu predstaviti kao korisnički definisan tip podataka koji odgovara njihovoj apstraktnoj sintaksi.

Funkcionalni jezici

Funkcionalni jezici uglavnom imaju mogućnost direktnog definisanja algebarskih (najčešće rekurzivnih) tipova podataka. Npr. u jeziku Isabelle:

```
datatype formula =
      TRUE
    | FALSE
    | Var nat
    | Not  formula
    | And  formula formula (infixl "And" 100)
    | Or   formula formula (infixl "Or"  100)
    | Imp  formula formula (infixl "Imp" 100)
    | Iff  formula formula (infixl "Iff" 100)
```

C

```
enum formula_type {TRUE, FALSE, VAR, NOT, AND, OR, IMP, IFF};
typedef struct _formula {
   enum formula_type type;
   unsigned var_num;
   struct _formula* op1, op2;
} formula;
```

U svakom čvoru se ostavlja mogućnost smeštanja i relevantnih i nerelevantih podataka (npr. čvor `NOT` ne koristi `var_num` niti op2). Moguće su i ,,štedljivije" reprezentacije koje ne čuvaju u svakom čvoru sva moguća polja (koriste se obično unije).

C++

Svaki veznik se predstavlja zasebnom klasom, pri čemu sve klase nasleđuju apstraktnu baznu klasu `Formula`.

```
class Formula {...};

class False : public Formula {...};
class True : public Formula {...};

class Atom : public Formula {
...
private:
  unsigned var_num;
};
```

C++

```
class UnaryConnective : public Formula {
...
private:
  Formula *_op1;
};

class Not : public UnaryConnective {...};

class BinaryConnective : public Formula {
...
private:
  Formula *_op1, *_op2;
};

class And : public BinaryConnective {...};
class Or : public BinaryConnective {...};
class Imp : public BinaryConnective {...};
class Iff : public BinaryConnective {...};
```

Unutrašnja struktura iskaza

- Iako se iskazna logika ne interesuje za unutrašnju strukturu iskaza, to ne znači da iskazi ne smeju da imaju nikakvu unutrašnju strukturu.
- Kako bi se omogućilo da se ista implementacija koristi i za iskaze bez unutrašnje strukture (iskazna slova) i za iskaze sa strukturom (npr. atomičke formule logike prvog reda), atomima je moguće ostaviti mogućnost da imaju neki dalji sadržaj.

U funkcionalnom jeziku bi to izgledalo ovako:

```
datatype 'a formula =
     TRUE
   | FALSE
   | Atom 'a
   | Not  "'a formula"
   | And  "'a formula" "'a formula" (infixl "And" 100)
   | Or   "'a formula" "'a formula" (infixl "Or"  100)
   | Imp  "'a formula" "'a formula" (infixl "Imp" 100)
   | Iff  "'a formula" "'a formula" (infixl "Iff" 100)
```

U C-u bi se umesto `unsigned var_num` mogao koristiti `void* atom_content`, a u C++-u bi, dodatno, sve klase mogli parametrizovati sa `template <class AtomContent>`.

Semantika iskazne logike

- S obzirom da iskazne formule predstavljaju iskaze (tvrđenja), njihova vrednost je istinosna i one mogu biti tačne ili netačne.
- Slično kao što algebarski izrazi (npr. $x + y + 3$) imaju vrednosti samo ako su poznate vrednosti promenljivih (npr. x i y), tako i iskazne formule imaju istinitosnu vrednost samo pod uslovom da je poznata istinitosna vrednost svih atoma (koji u njoj učestvuju).

Valuacije

Valuacije određuju istinitosne vrednosti atoma. Ključno pitanje je da li su dopuštene parcijalne valuacije tj. da li valuacija obavezno definiše vrednost svakog atoma. Moguće su različite definicije.

Definicija

Valuacija je funkcija koja sve atome preslikava u neki dvočlan skup (npr. $\{$tačno, netačno$\}$, $\{T, F\}$, $\{0, 1\}$, ...). Skup $\{\top, \bot\}$, se često izbegava kako bi se napravila jasna razlika između sintaksnih simbola konstanti i semantičke vrednosti formule. Atom je tačan akko se preslikava u tačno (tj. T, 1, ...).

Primer

$$p \mapsto 1, q \mapsto 0, r \mapsto 1, \ldots$$

Atomi p i r su tačani u ovoj valuaciji, q je netačan, ...

Valuacije

Definicija

Valuacija je skup atoma. Atom je tačan akko pripada valuaciji.

Primer

$$\{p, r\}$$

Atomi p i r su tačani u ovoj valuaciji, q je netačan, . . .

Valuacije

Valuacije se ponekad definišu tako da direktno određuju vrednosti literala (ne samo atoma).

Definicija

(Parcijalna) valuacija je skup literala, koji ne sadrži dva suprotna literala. Literal je tačan akko pripada valuaciji, netačan ako njemu suprotan literal pripada valuaciji, a nedefinisan inače.

Primer

$$\{p, \neg q, r\}$$

Atomi p i r su tačni, q je netačan. Dalje, npr. literal $\neg p$ je netačan, dok su literali s i $\neg s$ nedefinisani.

Zadovoljenje.

- Činjenicu da je formula F tačna u valuaciji v označavaćemo sa $v \models F$. Kažemo još i da valuacija v zadovoljava formulu F, da je valuacija v model formule F itd.
- Uslovi pod kojima važi $v \models F$ definišu se rekurzivno po strukturi formule.
- Opet su moguće različite (međusobno ekvivalentne) definicije.

Definicija zadovoljenja

Definicija

- *Tačnost atoma je određena definicijom valuacije.*
- *Konstanta $\top$ je tačna u svakoj valuaciji ($v \models \top$). Konstanta $\bot$ je netačna u svakoj valuaciji ($v \not\models \bot$).*
- *Formula oblika $\neg F$ je tačna u valuaciji v akko je formula F netačna u valuaciji v (tj. $v \models \neg F$ akko $v \not\models F$).*
- *Formula oblika $F_1 \wedge F_2$ je tačna u valuaciji v akko su obe formule F_1 i F_2 tačne u valuaciji v (tj. $v \models F_1 \wedge F_2$ akko $v \models F_1$ i $v \models F_2$).*
- *Formula oblika $F_1 \vee F_2$ je tačna u valuaciji v akko je bar jedna od formula F_1 i F_2 tačna u valuaciji v (tj. $v \models F_1 \vee F_2$ akko $v \models F_1$ ili $v \models F_2$).*
- *Formula oblika $F_1 \Rightarrow F_2$ je tačna u valuaciji v akko je formula F_1 netačna ili je formula F_2 tačna u valuaciji v (tj. $v \models F_1 \Rightarrow F_2$ akko $v \not\models F_1$ ili $v \models F_2$).*
- *Formula oblika $F_1 \Leftrightarrow F_2$ je tačna u valuaciji v akko su formule F_1 i F_2 istovremeno tačne ili istovremeno netačne u valuaciji v (tj. $v \models F_1 \Leftrightarrow F_2$ akko $v \models F_1$ i $v \models F_2$ ili $v \not\models F_1$ i $v \not\models F_2$).*

Vrednost formule

Zadovoljenje (tj. tačnost) je moguće definisati i preko funkcije koja računa istinitosnu vrednost formule u datoj valuaciji. Funkciju I_v koja iskazne formule preslikava u skup istinitosnih vrednosti $\{0, 1\}$, definišemo rekurzivno na sledeći način.

Definicija

- $I_v(p) = 1$ *akko* $v \models p$.
- $I_v(\top) = 1$, $I_v(\bot) = 0$;
- $I_v(\neg F) = 1$ *akko je* $I_v(F) = 0$;
- $I_v(F_1 \wedge F_2) = 1$ *akko je* $I_v(F_1) = 1$ *i* $I_v(F_2) = 1$.
- $I_v(F_1 \vee F_2) = 1$ *akko je* $I_v(F_1) = 1$ *ili* $I_v(F_2) = 1$.
- $I_v(F_1 \Rightarrow F_2) = 1$ *akko je* $I_v(F_1) = 0$ *ili* $I_v(F_2) = 1$.
- $I_v(F_1 \Leftrightarrow F_2) = 1$ *akko je* $I_v(F_1) = I_v(F_2)$.

Važi $v \models F$ *akko je* $I_v(F) = 1$.

Odnos objektne i meta logike u prethodnim definicijama

- Primetimo da prethodne definicije definišu značenje npr. konjunkcije ($\wedge$), (opet?) korišćenjem konjunkcije (i).
- Ono što na prvi pogled deluje kao začarani krug, u suštini to nije.
- Naime, iskazna logika se ovde definiše u okviru šireg logičkog okvira — metalogike, koja je u ovom slučaju (neformalna) logika višeg reda izražena govornim jezikom.
- Veznik $\wedge$ pripada objektnoj, iskaznoj logici, dok je veznik i u ovom slučaju veznik koji pripada metalogici.

Istinitosna vrednost u funkcionalnom jeziku

Naredni kod implementira funkciju $I_v(F)$ (kroz funkciju `eval F v`).

```
primrec eval ::"'a formula => ('a => bool) => bool" where
  "eval FALSE v = False"
| "eval TRUE v = True"
| "eval (Atom x) v = v x"
| "eval (Not F) v = (~ eval F v)"
| "eval (F1 And F2) v = (eval F1 v /\ eval F2 v)"
| "eval (F1 Or F2) v = (eval F1 v \/ eval F2 v)"
| "eval (F1 Imp F2) v = (eval F1 v --> eval F2 v)"
| "eval (F1 Iff F2) v = (eval F1 v = eval F2 v)"
```

Istinitosna vrednost u C-u

```
int eval(formula* F, int (*v) (unsigned)) {
   switch(F->type) {
       case TRUE:  return 1;
       case FALSE: return 0;
       case VAR:  return (*v)(F->var_num);
       case NOT:   return !eval(F->op1, v);
       case AND:   return eval(F->op1, v) && eval(F->op2, v);
       case OR:    return eval(F->op1, v) || eval(F->op2, v);
       case IMP:   return !eval(F->op1, v) || eval(F->op2, v);
       case IFF:   return eval(F->op1, v) == eval(F->op2, v);
   }
}
```

Istinitosna vrednost u C++-u

```
class Formula {
public: ...
   virtual bool eval(bool (*v) (unsigned)) = 0;
};

bool  True::eval(bool (*v)(unsinged)) { return true; }
bool False::eval(bool (*v)(unsinged)) { return false; }
bool   Var::eval(bool (*v)(unsinged)) { return (*v)(_var_num); }
bool   Not::eval(bool (*v)(unsinged)) { return !_op1->eval(v); }

bool   And::eval(bool (*v)(unsinged)) {
   return _op1->eval(v) && _op2->eval(v);
}
bool    Or::eval(bool (*v)(unsinged)) {
   return _op1->eval(v) || _op2->eval(v);
}
bool   Imp::eval(bool (*v)(unsinged)) {
   return !_op1->eval(v) || _op2->eval(v);
}
bool   Iff::eval(bool (*v)(unsinged)) {
   return _op1->eval(v) == _op2->eval(v);
}
```

Predstavljanje valuacije rečnikom

- Prethodni kod koristi pokazivače na funkcije za predstavljanje valuacija
- Ako su valuacije predstavljene funkcijama, ne mogu se dinamički kreirati i menjati.
- Mnogo je pogodnije valuacije predstaviti kroz rečničke strukture podataka (npr. mape).
- Dodatno, ukoliko je domen konačan, a atomi kodirani samo rednim brojevima, moguće je čak koristiti obične nizove (ili vektore).

```
class Formula {
public: ...
   virtual bool eval(const std::map<unsigned, bool>& v) = 0;
};
...
bool   Var::eval(const std::map<unsigned, bool>& v) {
  return v[_var_num];
}
```

Predstavljanje valuacija

Naravno, najbolje je predstaviti valuacije zasebnom klasom sa operatorima za čitanje i postavljanje vrednosti atoma.

```
class Valuation {
public:
   bool operator[](unsinged p) const;
   bool& operator[](unsinged p);
private:
...
};
```

Ukoliko se želi proširiti unutrašnja reprezentacija atoma sa celobrojnog indeksa na nešto šire, ova klasa se može parametrizovati.

Primer implementacije valuacije

```
class Valuation {
public:
   bool operator[](unsinged p) const {
      std::map<unsigned, bool>::const_iterator it = _m.find(p);
      if (it == _m.end())
        throw "Valuation lookup error";
       return it->second;
   }

   bool& operator[](unsinged p) {
      _m[p];
   }
private:
  std::map<unsigned, bool> _m;
};
```

Zadovoljivost, nezadovoljivost, tautologičnost, porecivost

U algebri, neki izrazi su tačni bez obzira kako im se dodele vrednosti promenljivih (npr. $x^2 - y^2 = (x + y)(x - y)$), neki su tačni samo za neke vrednosti promenljivih (npr. $x^2 + 2x + 1 = 0$), a neki ni za koje vrednosti promenljivih (npr. $x^2 + 2x + 2 = 0$). Slična klasifikacija važi i za iskazne formule.

Definicija

- *Formula je zadovoljiva ako ima bar jedan model.*
- *Formula je nezadovoljiva (kontradikcija) ako nema nijedan model.*
- *Formula je tautologija (logički valjana) ako joj je svaka valuacija model.*
- *Formula je poreciva ako postoji valuacija koja joj nije model.*

Model skupa formula

U nekim slučajevima, potrebno je da je više formula istovremeno zadovoljeno.

Definicija

Valuacija v zadovoljava skup formula tj. predstavlja model skupa formula Γ (što označavamo sa $v \models \Gamma$) akko je v model svake formule iz Γ.
Skup formula je zadovoljiv akko ima model.

Obratiti pažnju da se traži da postoji jedna valuacija koja istovremeno zadovoljava sve formule.

Logičke posledice, ekvivalentne formule, ekvizadovoljive formule

Definicija

Formula F je logička posledica skupa formula Γ (što označavamo sa $\Gamma \models F$) akko je svaki model za skup Γ istovremeno i model za formulu F.
Formule F_1 i F_2 su logički ekvivalentne (što označavamo sa $F_1 \equiv F_2$) ako je svaki model formule F_1 ujedno model formule F_2 i obratno (tj. ako $A \models B$ i $B \models A$).
Formule F_1 i F_2 su ekvizadovoljive (slabo ekvivalentne) (što označavamo sa $F_1 \equiv_s F_2$) akko je F_1 zadovoljiva ako i samo ako je F_2 zadovoljiva.

Primeri

Primer

- *Formula p je logička posledica formule $p \wedge q$. Zaista za svaku valuaciju v za koju važi $v \vDash p \wedge q$, važi i $v \vDash p$.*
- *Formula $p \wedge q$ nije logička posledica formule $p \vee q$, iako valuacija $v = \{p, q\}$ zadovoljava obe. Npr. valuacija $v = \{p, \neg q\}$ zadovoljava $p \vee q$, ali ne i $p \wedge q$.*
- *Formule $p \wedge q$ i $q \wedge p$ su logički ekvivalentne.*
- *Formule $p \wedge q$ i $r \wedge (r \Leftrightarrow p \wedge q)$ su ekvizadovoljive (obe su zadovoljive), ali nisu ekvivalentne. Npr. valuacija $v = \{p, q, \neg r\}$ zadovoljava prvu, ali ne i drugu.*

Stav

- $F_1, \ldots, F_n \models F$ *akko je* $F_1 \wedge \ldots \wedge F_n \Rightarrow F$ *tautologija.*
- $\Gamma, F \models F'$ *akko* $\Gamma \models F \Rightarrow F'$.
- $F \equiv F'$ *akko je* $F \Leftrightarrow F'$ *tautologija.*
- *Ako je* $F_1 \equiv F_1'$ *i* $F_2 \equiv F_2'$ *tada je i*
  - $\neg F_1 \equiv \neg F_1'$,
  - $F_1 \wedge F_2 \equiv F_1' \wedge F_2'$,
  - $F_1 \vee F_2 \equiv F_1' \vee F_2'$,
  - $F_1 \Rightarrow F_2 \equiv F_1' \Rightarrow F_2'$, *i*
  - $F_1 \Leftrightarrow F_2 \equiv F_1' \Leftrightarrow F_2'$.

SAT problem

- Ispitivanje zadovoljivosti iskazne formule naziva se SAT problem.
- Centralni problem teorijskog računarstva.
- SAT je prvi problem za koji je dokazano da je NP kompletan.
- Ogromne praktične primene.

Problem ispitivanja tautologičnosti, lako se svodi na SAT.

Stav

- *Svaka tautologija je zadovoljiva.*
- *Svaka nezadovoljiva formula je poreciva.*
- *Formula je poreciva akko nije tautologija.*
- *Formula je nezadovoljiva akko nije zadovoljiva.*
- *Formula F je tautologija akko je $\neg F$ nezadovoljiva.*
- *Formula F je zadovoljiva akko je $\neg F$ poreciva.*

Pristupi rešavanja SAT problema

- Naivni metodi (metod istinitosnih tablica).
- Problem se tehnički jednostavnije rešava ako je formula u nekom od specijalnih oblika (DNF, KNF, BDD).
- Za DNF se zadovoljivost trivijalno ispituje, ali je postupak prevođenja u DNF netrivijalan. Slično je i sa BDD.
- Postoji efikasan postupak prevođenja formule u KNF (doduše koji ne čuva ekvivalentnost, ali čuva zadovoljivost).
- Zadovoljivost formula u KNF se dalje ispituje nekim od tzv. klauzalnih algoritama.
  - DP procedura (iskazna rezolucija)
  - DPLL procedura
  - CDCL SAT rešavači (unapređena DPLL procedura)
  - Stohastički SAT rešavači (zasnovani na randomizaciji — ne garantuju uvek definitivan odgovor).

O većini navedenih pristupa biće reči u okviru kursa.

Svojstva sistema (za ispitivanje (ne)zadovoljivosti)

Poželjna svojstva svakog sistema za ispitivanje (ne)zadovoljivosti:

Zaustavljanje – Za svaku ulaznu formulu, sistem se zaustavlja nakon primene konačno mnogo koraka.

Saglasnost – Ako sistem prijavi nezadovoljivost, polazna formula je zaista nezadovoljiva.

Potpunost – Ako je polazna formula nezadovoljiva, sistem će prijaviti nezadovoljivost.

Svojstva sistema (za dokazivanje)

O sistemima za dokazivanje će biti više reči u nastavku kursa, međutim, i oni imaju veoma slična poželjna svojstva:

Zaustavljanje – Za svaku ulaznu formulu, sistem se zaustavlja nakon primene konačno mnogo koraka.

Saglasnost – Ako sistem pronađe dokaz neke formule, polazna formula je zaista semantička posledica aksioma.

Potpunost – Ako je polazna formula semantička posledica aksioma, sistem će pronaći dokaz.

Pregled

Vrednost formule je određena vrednošću njenih atoma

Stav

- *Ako je skup atoma beskonačan (a najčešće uzimamo da je prebrojiv) postoji beskonačno (čak neprebrojivo) mnogo različitih valuacija.*
- *Skup atoma koji se javljaju u formuli je konačan.*
- *Ukoliko se dve valuacije poklapaju na skupu atoma koji se javlja u nekoj formuli, istinitosna vrednost formule u obe valuacije je jednaka.*

Za ispitivanje tautologičnosti (zadovoljivosti, ...) formule dovoljno je ispitati konačno mnogo (parcijalnih) valuacija (ako je n broj različitih atoma u formuli, onda je dovoljno ispitati 2^n različitih valuacija).

Istinitosna tablica – primer

Primer

$(p$	$\vee$	$(q$	$\wedge$	$r))$	$\wedge$	$(\neg$	p	$\vee$	$\neg$	$r)$
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	1	0	1	0	1	0	1
0	0	1	0	0	0	1	0	1	1	0
0	1	1	1	1	1	1	0	1	0	1
1	1	0	0	0	1	0	1	1	1	0
1	1	0	0	1	0	0	1	0	0	1
1	1	1	0	0	1	0	1	1	1	0
1	1	1	1	1	0	0	1	0	0	1

Skup atoma formule u C++-u

Naredni C++ kôd izračunava skup atoma (iskaznih promenljivih) koje se javljaju u formuli.

```
class Formula {
   virtual void atoms(set<unsigned>& ats) = 0;
};
void  True::atoms(set<unsigned>& ats) {}
void False::atoms(set<unsigned>>& ats) {}

void   Var::atoms(set<unsigned>& ats) {
   ats.insert(_var_num);
}

void UnaryConnective::atoms(set<unsigned>>& ats) {
   op1->atoms(ats);
}

void BinaryConnective::atoms(set<unsigned>& ats) {
   op1->atoms(ats); op2->atoms(ats);
}
```

Skup atoma formule u funkcionalnom jeziku

U funkcionalnom jeziku, pogodno je definisati i apstraktni funkcional koji obilazi formulu primenjujući datu funkciju na sve atome i akumulirajući rezultat.

```
fun overatoms where
 "overatoms f FALSE b = b"
| "overatoms f TRUE b = b"
| "overatoms f (Atom a) b = f a b "
| "overatoms f (Not F)  b = overatoms f F b"
| "overatoms f (F1 And F2) b = overatoms f F1 (overatoms f F2 b)"
| "overatoms f (F1 Or F2) b = overatoms f F1 (overatoms f F2 b)"
| "overatoms f (F1 Imp F2) b = overatoms f F1 (overatoms f F2 b)"
| "overatoms f (F1 Iff F2) b = overatoms f F1 (overatoms f F2 b)"

definition atoms where "atoms F = remdups (overatoms (op#) F [])"
```

Generisanje svih valuacija (varijacija)

Kako nabrojati sve valuacije za dati skup atoma?
Rekurzivno rešenje u funkcionalnom programskom jeziku.

```
primrec allvaluations where
  "allvaluations [] v l = v # l"
| "allvaluations (p#ps) v l =
              (allvaluations ps (v (p := False)) l) @
              (allvaluations ps (v (p := True)) l)"
```

Generisanje svih valuacija (varijacija)

Rekurzivno rešenje u C++-u.

```
void allvaluations(
    std::set<unsigned>::const_iterator atoms_begin,
    std::set<unsigned>::const_iterator atoms_end,
    Valuation& v, std::vector<Valuation>& res) {
  if (atoms_begin == atoms_end) res.push_back(v);
  else {
    unsinged p = *atoms_begin;
    std::set<unsigned>::const_iterator atoms_next =
        ++atoms_begin;
    v[p] = false;
    allvaluations(atoms_next, atoms_end, v, res);
    v[p] = true;
    allvaluations(atoms_next, atoms_end, v, res);
  }
}
```

Štampanje tablice istinitosti

```
void truth_table(Formula* f) {
  std::set<AtomContent> ats; f->atoms(ats);
  Valuation v; std::vector<Valuation> vals;
  allvaluations(ats.begin(), ats.end(), v, vals);

  std::vector<Valuation>::const_iterator it;
  for (it = vals.begin(); it != vals.end(); it++)
    std::cout << *it << "| " << f->eval(*it) << std::endl;
}
```

Generisanje leksikografski sledeće (valuacije) varijacije

000
001
010
011 Leksikografski redosled
100
101
110
111

Sledeća varijacija se može dobiti tako što se invertuje cifra po cifra od pozadi sve dok se ne invertuje prva nula ili se ne iscrpu sve cifre (u slučaju svih jedinica). Npr.
10110110101111
10110110110000

Generisanje leksikografski sledeće (valuacije) varijacije

```
class Valuation { ...
  void init(const std::set<unsigned>& ats) {
    std::set<unsigned>::const_iterator it;
    for (it = ats.begin(); it != ats.end(); it++)
      (*this)[*it] = false;
  }

  bool next() {
    std::map<unsigned, bool>::reverse_iterator it;
    for (it = _m.rbegin(); it != _m.rend(); it++) {
      it->second = !it->second;
      if (it->second == true)
         return true;
    }
    return false;
  }
};
```

Provera da li je formula tautologija

Ovim se dolazi do iterativne funkcije kojom se proverava da li je formula tautolgija.

```
bool tautology(Formula* f) {
  std::set<unsigned> ats; f->atoms(ats);
  Valuation v; v.init(ats);
  do {
    if (f->eval(v) == false)
      return false;
  } while (v.next());
  return true;
}
```

Provera da li je formula zadovoljiva

Slično, dolazi se i do iterativne funkcije kojom se proverava da li je formula zadovoljiva.

```
bool sat(Formula* f) {
  std::set<unsigned> ats; f->atoms(ats);
  Valuation v; v.init(ats);
  do {
    if (f->eval(v) == true)
      return true;
  } while (v.next());
  return false;
}
```

Pregled

Zamena

Definicija

Formula $G[F \to F']$ je zamena formule F formulom F' u formuli G i konstruiše se tako što se sva pojavljivanja potformule F u okviru neke formule G zamenjuju sa F'.

Primer

Ako u formuli

$$(p \Rightarrow q) \wedge r \vee (p \Rightarrow q),$$

zamenjujemo $p \Rightarrow q$ sa $\neg p \vee q$, dobija se formula:

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

Zamena u funkcionalnom jeziku

```
fun subst :: "'a formula => 'a formula => 'a formula => 'a formula"
where
"subst F Pat Subst =
  (if F = Pat then Subst
   else (case F of
       TRUE => TRUE
     | FALSE => FALSE
     | Atom p => Atom p
     | Not F' => Not (subst F' Pat Subst)
     | F1 And F2 => (subst F1 Pat Subst) And (subst F2 Pat Subst)
     | F1 Or F2 => (subst F1 Pat Subst) Or (subst F2 Pat Subst)
     | F1 Imp F2 => (subst F1 Pat Subst) Imp (subst F2 Pat Subst)
     | F1 Iff F2 => (subst F1 Pat Subst) Iff (subst F2 Pat Subst)
  ))"
```

Svojstva zamene

Stav

- *Ako je x atom, p i q formule, v valuacija, a v' valuacija koja se dobija postavljanjem vrednosti x na $I_v(q)$ u valuaciji v, onda je $I_v(p[x \to q]) = I_{v'}(p)$.*
- *Ako je p atom, F proizvoljna formula, a formula G je tautologija, onda je $G[p \to F]$ tautologija.*
- *Ako je $F \equiv F'$, onda je $G[F \to F'] \equiv G$.*

Svojstva zamene — primeri

Primer

- *Vrednost formule* $p \wedge (q \vee r)$ *u valuaciji* $v = \{\neg p, \neg q, r\}$ *jednaka je vrednosti formule* $p \wedge x$ *u valuaciji* $v' = \{\neg p, \neg q, r, x\}$. *Atom* x *je tačan u* v' *jer je* $q \vee r$ *tačno u* v.
- $p \wedge q \Leftrightarrow q \wedge p$ *je tautologija, pa je i* $(r \vee s) \wedge (s \wedge r) \Leftrightarrow (s \wedge r) \wedge (r \vee s)$ *tautologija.*
- $p \wedge q \equiv q \wedge p$ *pa je i* $r \vee (p \wedge q) \equiv r \vee (q \wedge p)$.

Pregled

Normalne forme

Određenim transformacijama algebarski izrazi se svode na oblike pogodnije za određene zadatke.

Primer

- *Npr.* $(x + y)(y - x) + y + x^2$ *se svodi na* $y^2 + y$.
- $x^3 + x^2y + xy + z$ *je ,,razvijeni oblik" polinoma – pogodno npr. za sabiranje dva polinoma*
- $(x + 1)(y + 2)(z + 3)$ *je ,,faktorisani oblik" polinoma – pogodno npr. za određivanje nula polinoma*

Slično se radi i u slučaju logičkih izraza.
Transformacije formula predstavljaju sintaksne operacije, pri čemu se obično zahteva njihova semantička opravdanost (tj. da se nakon primene transformacija dobijaju formule ekvivalentne polaznim).

Pristupi rešavanja SAT problema

- Naivni metodi (metod istinitosnih tablica).
- Problem se tehnički jednostavnije rešava ako je formula u nekom od specijalnih oblika (DNF, KNF, BDD).
- Za DNF se zadovoljivost trivijalno ispituje, ali je postupak prevođenja u DNF netrivijalan. Slično je i sa BDD.
- Postoji efikasan postupak prevođenja formule u KNF (doduše koji ne čuva ekvivalentnost, ali čuva zadovoljivost).
- Zadovoljivost formula u KNF se dalje ispituje nekim od tzv. klauzalnih algoritama.
  - DP procedura (iskazna rezolucija)
  - DPLL procedura
  - CDCL SAT rešavači (unapređena DPLL procedura)
  - Stohastički SAT rešavači (zasnovani na randomizaciji — ne garantuju uvek definitivan odgovor).

Prezapisivanje

Transformacije kojima se formule uprošćavaju i svode na normalne forme se često formulišu u terminima sistema za prezapisivanje (više o ovome u nastavku kursa).

Primer

- *Npr. ako se pravilo* $(X - Y)(X + Y) = X^2 - Y^2$ *primeni na izraz* $a + (b - c)(b + c) - a(d - b)$, *dobija se izraz* $a + b^2 - c^2 - a(d - b)$.
- *Ako se pravilo* $\neg(X \wedge Y) \equiv \neg X \vee \neg Y$ *primeni na izraz* $p \vee \neg(q \wedge r)$, *dobija se izraz* $p \vee (\neg q \vee \neg r)$.

Ukoliko pravilo podrazumeva zamenu neke podformule njoj ekvivalentnom formulom, semantička opravdanost njegove primene u principu sledi na osnovu teoreme o zameni.

Eliminisanje konstanti

Naredne logičke ekvivalencije mogu se upotrebiti za uprošćavanje formule eliminisanjem logičkih konstanti (preciznije, eliminišu se veznici koji se primenjuju na logičke konstante):

$$\neg\bot \equiv \top \qquad \neg\top \equiv \bot$$

$$P \wedge \bot \equiv \bot \qquad \bot \wedge P \equiv \bot$$

$$P \wedge \top \equiv P \qquad \top \wedge P \equiv P$$

$$P \vee \bot \equiv P \qquad \bot \vee P \equiv P$$

$$P \vee \top \equiv \top \qquad \top \vee P \equiv \top$$

$$P \Rightarrow \bot \equiv \neg P \qquad \bot \Rightarrow P \equiv \top$$

$$P \Rightarrow \top \equiv \top \qquad \top \Rightarrow P \equiv P$$

$$P \Leftrightarrow \bot \equiv \neg P \qquad \bot \Leftrightarrow P \equiv \neg P$$

$$P \Leftrightarrow \top \equiv P \qquad \top \Leftrightarrow P \equiv P$$

Implementacija u funkcionalnom jeziku

```
fun psimplify1 where
  "psimplify1 (Not FALSE) = TRUE"
| "psimplify1 (Not TRUE) = FALSE"
| "psimplify1 (F And TRUE) = F"
| "psimplify1 (F And FALSE) = FALSE"
| "psimplify1 (TRUE And F) = F"
| "psimplify1 (FALSE And F) = FALSE"
...
| "psimplify1 F = F"

primrec psimplify where
  "psimplify FALSE = FALSE"
| "psimplify TRUE = TRUE"
| "psimplify (Atom a) = Atom a"
| "psimplify (Not F) = psimplify1 (Not (psimplify F))"
| "psimplify (F1 And F2) = psimplify1 ((psimplify F1) And (psimplify F2))"
| "psimplify (F1 Or F2) = psimplify1 ((psimplify F1)  Or (psimplify F2))"
| "psimplify (F1 Imp F2) = psimplify1 ((psimplify F1) Imp (psimplify F2))"
| "psimplify (F1 Iff F2) = psimplify1 ((psimplify F1) Iff (psimplify F2))"
```

Eliminacija konstanti

Primer

Uprošćavanjem formule

$$(\top \Rightarrow (x \Leftrightarrow \bot)) \Rightarrow \neg(y \vee (\bot \wedge z))$$

dobija se

$$\neg x \Rightarrow \neg y$$

NNF

Definicija

NNF Formula je u negacionoj normalnoj formi (NNF) akko je sastavljena od literala korišćenjem isključivo veznika $\wedge$ i $\vee$ ili je logička konstanta ($\top$ ili $\bot$).

Primer

Formule $\top$, $\bot$, p, $p \wedge \neg q$ i $p \vee (q \wedge (\neg r) \vee s)$ su u NNF, dok $\neg\neg p$ i $p \wedge \neg(q \vee r)$ to nisu.

Uklanjanje implikacija i ekvivalencija

$$\begin{aligned} P \Rightarrow Q &\equiv \neg P \vee Q \\ \neg(P \Rightarrow Q) &\equiv P \wedge \neg Q \end{aligned}$$

$$\begin{aligned} P \Leftrightarrow Q &\equiv (P \wedge Q) \vee (\neg P \wedge \neg Q) \\ \neg(P \Leftrightarrow Q) &\equiv (P \wedge \neg Q) \vee (\neg P \wedge Q) \end{aligned}$$

ili

$$\begin{aligned} P \Leftrightarrow Q &\equiv (P \vee \neg Q) \wedge (\neg P \vee Q) \\ \neg(P \Leftrightarrow Q) &\equiv (P \vee Q) \wedge (\neg P \vee \neg Q) \end{aligned}$$

Spuštanje negacije

$$\begin{aligned} \neg(P \wedge Q) &\equiv \neg P \vee \neg Q \\ \neg(P \vee Q) &\equiv \neg P \wedge \neg Q \\ \neg\neg P &\equiv P \end{aligned}$$

Implementacija u funkcionalnom programskom jeziku

```
fun nnf' :: "'a formula => 'a formula" where
  "nnf' FALSE = FALSE"
| "nnf' TRUE = TRUE"
| "nnf' (Atom p) = Atom p"
| "nnf' (p And q) = (nnf' p) And (nnf' q)"
| "nnf' (p Or q) = (nnf' p) Or (nnf' q)"
| "nnf' (p Imp q) = (nnf' (Not p)) Or (nnf' q)"
| "nnf' (p Iff q) = ((nnf' p) And (nnf' q)) Or
                    ((nnf' (Not p)) And (nnf' (Not q)))"
| "nnf' (Not F) = (case F of
      Not p => nnf' p
    | p And q => (nnf' (Not p)) Or (nnf' (Not q))
    | p Or q => (nnf' (Not p)) And (nnf' (Not q))
    | p Imp q => (nnf' p) And (nnf' (Not q))
    | p Iff q => ((nnf' p) And (nnf' (Not q))) Or
                 ((nnf' (Not p)) And (nnf' q))
    | _ => (Not F)
  )"

definition nnf :: "'a formula => 'a formula" where
  "nnf f = nnf' (psimplify f)"
```

NNF

Primer

$$NNF[(p \Leftrightarrow q) \Leftrightarrow \neg(r \Rightarrow s)] \quad \equiv$$

$$NNF[p \Leftrightarrow q] \wedge NNF[\neg(r \Rightarrow s)] \vee$$
$$NNF[\neg(p \Leftrightarrow q)] \wedge NNF[\neg\neg(r \Rightarrow s)] \quad \equiv$$

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

Složenost NNF transformacije

- Zahvaljujući uvećanju formule prilikom eliminacije ekvivalencije, u najgorem slučaju, NNF formule sa n veznika može da sadrži više od 2^n veznika.
- Ukoliko je samo cilj da se negacija spusti do nivoa atoma, i ne insistira na potpunoj izgradnji NNF, tj. ako se dopusti zadržavanje ekvivalencija u formuli, moguće je izbeći eksponencijalno uvećanje. Negacija se može spustiti kroz ekvivalenciju na osnovu npr. $\neg(p \Leftrightarrow q) \equiv \neg p \Leftrightarrow q$.

DNF

Definicija

Formula je u disjunktivnoj normalnoj formi (DNF) ako je oblika $D_1 \vee \ldots \vee D_N$, *pri čemu je svaki disjunkt* D_i *oblika* $l_{i1} \wedge \ldots \wedge l_{im_i}$, *pri čemu su* l_{ij} *literali, tj. oblika je*

$$\bigvee_{i=1}^{N} \bigwedge_{j=1}^{m_i} l_{ij}$$

Ako je formula u *DNF* ona je i u *NNF*, uz dodatno ograničenje da je ,,disjunkcija konjunkcija".

KNF

Definicija

Formula je u konjunktivnoj normalnoj formi (KNF)[1] *ako je oblika* $K_1 \wedge \ldots \wedge K_N$, *pri čemu je svaki konjunkt* K_i *klauza, tj. oblika* $l_{i1} \vee \ldots \vee l_{im_i}$, *pri čemu su* l_{ij} *literali, tj. oblika je*

$$\bigwedge_{i=1}^{N} \bigvee_{j=1}^{m_i} l_{ij}$$

Ako je formula u *KNF* ona je i u *NNF*, uz dodatno ograničenje da je ,,konjunkcija disjunkcija".

[1] eng. CNF (Conjunctive Normal Form)

KNF i DNF date formule

Definicija

Za formulu F' kažemo da je DNF (KNF) formule F akko je F' u DNF (KNF) i važi $F \equiv F'$.

Stav

Svaka formula ima DNF (KNF).

- Naglasimo da postoji više različitih DNF (KNF) za istu polaznu formulu.
- Tretman konstanti $\top$ i $\bot$ je specifičan — neke definicije dopuštaju da su konstante u DNF (KNF). S druge strane, nekada se uzima da je $p \vee \neg p$ DNF za $\top$, a da je $p \wedge \neg p$ KNF za $\bot$.

Reprezentacija pomoću lista (skupova)

- Klasična definicija DNF i KNF nije precizna — veznici $\vee$ i $\wedge$ se obično definišu kao binarni, a onda najednom tretiraju kao n-arni (čak nekad i 0-arni).
- S obzirom na specijalnu strukturu, DNF i KNF formule je pogodno u posmatrati kao listu klauza, pri čemu se klauze opet predstavljaju kao liste literala. Npr. $(p \wedge \neg q) \vee (q \wedge \neg r \wedge s)$ se može predstaviti kao $[[p, \neg q], [q, \neg r, s]]$.
- S obzirom da se ponavljanja klauza i literala mogu ukloniti uz zadržavanje logičke ekvivalentnosti (na osnovu $P \wedge P \equiv P$ i $P \vee P \equiv P$), kao i da redosled klauza nije bitan (na osnovu $P \wedge Q \equiv Q \wedge P$ i $P \vee Q \equiv Q \vee P$), umesto lista se mogu koristiti skupovi literala, odnosno skupovi skupova literala.
  Npr. $\{\{p, \neg q\}, \{q, \neg r, s\}\}$.

Konverzija lista (skupova) u formule

- Prilikom prevođenja u konjunkciju, elementi liste se povežu veznikom $\wedge$, osim u specijalnom slučaju prazne liste koja se prevodi u $\top$.
- Prilikom prevođenja u disjunkciju, elementi liste se povežu veznikom $\vee$, osim u specijalnom slučaju prazne liste koja se prevodi u $\bot$.

```
primrec fold1 where
  "fold1 f (h # t) = (if t = [] then h else f h (fold1 f t))"
definition list_disj :: "'a formula list => 'a formula" where
  "list_disj l = (if l = [] then FALSE else fold1 mk_or l)"
definition list_conj :: "'a formula list => 'a formula" where
  "list_conj l = (if l = [] then TRUE else fold1 mk_and l)"
definition dnf2form :: "'a formula list list => 'a formula" where
  "dnf2form l = list_disj (map list_conj l)"
definition cnf2form :: "'a formula list list => 'a formula" where
  "cnf2form l = list_conj (map list_disj l)"
```

DIMACS CNF

- DIMACS CNF — Standardni format zapisa KNF formula u tekstualne datoteke.
- Koristi se kao ulaz SAT rešavača.

Primer

$$(x_1 \vee \neg x_2) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge x_3$$

```
p cnf 3 4
1 -2 0
-2 3 0
-1 2 -3 0
3 0
```

Prevođenje prezapisivanjem distributivnosti

NNF formula se može prevesti u DNF primenom logičkih ekvivalencija

$$\begin{aligned} P \wedge (Q \vee R) &\equiv (P \wedge Q) \vee (P \wedge R) \\ (Q \vee R) \wedge P &\equiv (Q \wedge P) \vee (R \wedge P) \end{aligned}$$

NNF formula se može prevesti u KNF primenom logičkih ekvivalencija

$$\begin{aligned} P \vee (Q \wedge R) &\equiv (P \vee Q) \wedge (P \vee R) \\ (Q \wedge R) \vee P &\equiv (Q \vee P) \wedge (R \vee P) \end{aligned}$$

Korektnost ove procedure se zasniva na teoremi o zameni.

Implementacija u funkcionalnom jeziku

```
fun distribdnf :: "'a formula => formula" where
  "distribdnf (p And (q Or r)) =
      (distribdnf (p And q)) Or (distribdnf (p And r))"
| "distribdnf ((p Or q) And r) =
      (distribdnf (p And r)) Or (distribdnf (q And r))"
| "distribdnf F = F"

fun rawdnf :: "'a formula => 'a formula" where
  "rawdnf (p And q) = distribdnf ((rawdnf p) And (rawdnf q))"
| "rawdnf (p Or q) = (rawdnf p) Or (rawdnf q)"
| "rawdnf F = F"
```

Primer

Primer

$$
\begin{aligned}
& \mathit{rawdnf}\,[(p \vee (q \wedge r)) \wedge (\neg p \vee \neg r)] && \equiv \\
& \mathit{distribdnf}\,[\mathit{rawdnf}\,[p \vee (q \wedge r)] \wedge \mathit{rawdnf}\,[\neg p \vee \neg r]] && \equiv \\
& \mathit{distribdnf}\,[(\mathit{rawdnf}\,[p] \vee \mathit{rawdnf}\,[q \wedge r]) \wedge (\mathit{rawdnf}\,[\neg p] \vee \mathit{rawdnf}\,[\neg r])] && \equiv \\
& \mathit{distribdnf}\,[(p \vee \mathit{distribdnf}\,[q \wedge r]) \wedge (\neg p \vee \neg r)] && \equiv \\
& \mathit{distribdnf}\,[(p \vee (q \wedge r)) \wedge (\neg p \vee \neg r)] && \equiv \\
& (p \wedge \neg p) \vee (q \wedge r \wedge \neg p) \vee (p \wedge \neg r) \vee (q \wedge r \wedge \neg r)
\end{aligned}
$$

Primetimo da se dobijena DNF može dalje uprostiti (npr. $p \wedge \neg p$, kao i $q \wedge r \wedge \neg r$ su logički ekvivalentne sa $\bot$ i mogu se ukloniti).

Implementacija DNF za reprezentaciju u obliku lista

```
definition distrib where
  "distrib l1 l2 = map (% (x, y). x @ y) (allpairs l1 l2)"

fun purednf :: "'a formula => 'a formula list list" where
  "purednf (p And q) = distrib (purednf p) (purednf q)"
| "purednf (p Or q) = (purednf p) @ (purednf q)"
| "purednf F = [[F]]"

definition trivial where
"trivial l ==
  let pos = filter positive l; neg = filter negative l in
  intersect pos (map negate neg) ~= []"

fun simpdnf :: "'a formula => 'a formula list list" where
  "simpdnf FALSE = []"
| "simpdnf TRUE = [[]]"
| "simpdnf F = filter (% c. !trivial c) (purednf (nnf F))"

definition "dnf F = dnf2form (simpdnf F)"
```

Implementacija KNF za reprezentaciju u obliku lista

Za izgradnju KNF procedure može (naravno, ne mora) se iskoristiti dualnost.

$$\neg p \equiv \bigvee_{i=1}^{m} \bigwedge_{j=1}^{n} p_{ij}$$

akko

$$p \equiv \bigwedge_{i=1}^{m} \bigvee_{j=1}^{n} \neg p_{ij}$$

```
definition "purecnf F = map (map negate) (purednf (nnf (Not F)))"

fun simpcnf :: "'a formula => 'a formula list list" where
  "simpcnf FALSE = [[]]"
| "simpcnf TRUE = []"
| "simpdnf F = filter (% c. !trivial c) (purecnf F)"

definition "dnf F = cnf2form (simpdnf F)"
```

Veza između DNF i istinitosnih tablica

Definicija

Ako je data formula F koja nije kontradikcija, i koja sadrži atome $p_1, \ldots, p_n$, njena kanonska DNF je

$$\bigvee_{v \models F} \bigwedge_{i=1}^{n} p_i^v,$$

pri čemu je

$$p_i^v = \begin{cases} p_i & ako\ v \models p_i \\ \neg p_i & inače \end{cases}$$

Veza između KNF i istinitosnih tablica

Definicija

Ako je data formula F koja nije tautologija, i koja sadrži atome p_1, ..., p_n, njena kanonska KNF je

$$\bigwedge_{v \not\models F} \bigvee_{i=1}^{n} p_i^{\hat{v}},$$

pri čemu je

$$p_i^{\hat{v}} = \begin{cases} \neg p_i & ako\ v \models p_i \\ p_i & inače \end{cases}$$

Kanonska DNF/KNF - primer

p	q	r	$(p \vee (q \wedge r)) \wedge (\neg p \vee \neg r)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Kanonska DNF:

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

Kanonska KNF:

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

Minimalizacija DNF (KNF)

Postoje tehnike koje minimalizuju DNF (ili KNF).

- Algebarske transformacije
- Karnoove (Karnaugh) mape
- Kvin-MekKlaski (QuineMcCluskey) algoritam (prosti implikanti)

Složenost DNF i KNF procedura

- S obzirom da se baziraju na NNF, jasno je da distributivne DNF i KNF mogu eksponencijalno da uvećaju formulu.
- Takođe, i pravila distributivnosti sama za sebe doprinose eksponencijalnom uvećavanju. Npr. prevođenje $(p_1 \vee q_1) \wedge \ldots \wedge (p_n \vee q_n)$ u KNF ima 2^n klauza.
- Slično, i kanonske DNF i KNF mogu da budu eksponencijalno velike u odnosu na polaznu formulu.
- Ovakvo uvećanje je neizbežno ako se insistira da dobijena formula bude ekvivalentna polaznoj.
- Navedene činjenice čine sve prethodno navedene tehnike neupotrebljive u većini praktičnih zadataka velike dimenzije.

Pristupi rešavanja SAT problema

- Naivni metodi (metod istinitosnih tablica).
- Problem se tehnički jednostavnije rešava ako je formula u nekom od specijalnih oblika (DNF, KNF, BDD).
- Za DNF se zadovoljivost trivijalno ispituje, ali je postupak prevođenja u DNF netrivijalan. Slično je i sa BDD.
- Postoji efikasan postupak prevođenja formule u KNF (doduše koji ne čuva ekvivalentnost, ali čuva zadovoljivost).
- Zadovoljivost formula u KNF se dalje ispituje nekim od tzv. klauzalnih algoritama.
  - DP procedura (iskazna rezolucija)
  - DPLL procedura
  - CDCL SAT rešavači (unapređena DPLL procedura)
  - Stohastički SAT rešavači (zasnovani na randomizaciji — ne garantuju uvek definitivan odgovor).

Definiciona (Cajtinova) KNF

- U većini primena, nije neophodno da KNF bude logički ekvivalentna polaznoj — dovoljno je da bude ekvizadovoljiva.
- Ekvizadovoljiva KNF se može izgraditi tako da dobijena formula bude samo za konstantni faktor veća od polazne.
- Dualno, postoji definiciona DNF koja je ekvivalidna polaznoj.

Definiciona (Cajtinova) KNF

- Ideja potiče od Cajtina (Tseitin) — za potformule uvode se novi atomi (iskazna slova).

Primer

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

$$(p \vee s_1) \wedge (\neg p \vee \neg r) \wedge (s_1 \Leftrightarrow q \wedge r)$$

$$s_2 \wedge (\neg p \vee \neg r) \wedge (s_1 \Leftrightarrow q \wedge r) \wedge (s_2 \Leftrightarrow p \vee s_1)$$

$$s_2 \wedge s_3 \wedge (s_1 \Leftrightarrow q \wedge r) \wedge (s_2 \Leftrightarrow p \vee s_1) \wedge (s_3 \Leftrightarrow \neg p \vee \neg r)$$

$$s_4 \wedge (s_1 \Leftrightarrow q \wedge r) \wedge (s_2 \Leftrightarrow p \vee s_1) \wedge (s_3 \Leftrightarrow \neg p \vee \neg r) \wedge (s_4 \Leftrightarrow s_2 \wedge s_3)$$

Definiciona (Cajtinova) KNF

- Definicione ekvivalencije se klasično prevode u KNF

Primer

$$s_4 \wedge (s_1 \Leftrightarrow q \wedge r) \wedge (s_2 \Leftrightarrow p \vee s_1) \wedge (s_3 \Leftrightarrow \neg p \vee \neg r) \wedge (s_4 \Leftrightarrow s_2 \wedge s_3)$$

$$\begin{aligned}
&s_4 \\
&(\neg s_1 \vee q) \wedge (\neg s_1 \vee r) \wedge (\neg q \vee \neg r \vee s_1) \wedge \\
&(\neg s_2 \vee p \vee s_1) \wedge (\neg p \vee s_2) \wedge (\neg s_1 \vee s_2) \wedge \\
&(\neg s_3 \vee \neg p \vee \neg r) \wedge (p \vee s_3) \wedge (r \vee s_3) \wedge \\
&(\neg s_4 \vee s_2) \wedge (\neg s_4 \vee s_3) \wedge (\neg s_2 \vee \neg s_3 \vee s_4)
\end{aligned}$$

Definiciona KNF – implementacija

```
function maincnf and defstep where
"maincnf F defs n =
  (case F of
       F1 And F2 => defstep mk_and F1 F2 defs n
     | F1 Or F2  => defstep mk_or  F1 F2 defs n
     | F1 Imp F2 => defstep mk_imp F1 F2 defs n
     | _         => (F, defs, n))"
| "defstep ope p q defs n =
   (let (a1, defs1, n1) = maincnf p defs  n;
        (a2, defs2, n2) = maincnf q defs1 n1;
        F = ope a1 a2;
        (a, n3) = mk_prop n2 in
          (a, defs2 @ [(a, F)], n3)
    )"

definition defcnf where
"defcnf F ==
   let (F', defs, _) = maincnf (F, [], 1);
       l = concat (map simpcnf (F' # (map (% (a, F). (a Iff F)) defs))) in
     cnf2form l"
```

Ekvizadovoljivost definicione KNF

Teorema

Ako se s_i ne javlja u q tada su $p[s_i \to q]$ i $p \wedge (s_i \Leftrightarrow q)$ ekvizadovoljive. Preciznije, svaki model za $p[s_i \to q]$ se može proširiti do modela za $p \wedge (s_i \Leftrightarrow q)$ dok je svaki model za $p \wedge (s_i \Leftrightarrow q)$ ujedno i model za $p[s_i \to q]$.

Dokaz

Neka $v \vDash p[s_i \to q]$. Posmatrajmo valuaciju v' koja menja v samo postavljajući promenljivoj s_i vrednost na $I_v(q)$. Važi $v' \vDash p$ (na osnovu svojstava zamene), i važi i $v' \vDash s_i \Leftrightarrow q$ jer je $I_{v'}(q) = I_v(q) = I_v(s_i)$ (pošto se s_i ne javlja u q), te važi $v \vDash p \wedge (s_i \Leftrightarrow q)$.
Obratno, ako $v \vDash p \wedge (s_i \Leftrightarrow q)$, tada $v \vDash p$ i $v \vDash s_i \Leftrightarrow q$ pa je $I_v(s_i) = I_v(q)$. Zato $v \vDash p[s_i \to q]$ (na osnovu svojstava zamene).

Pregled

Pristupi rešavanja SAT problema

- Naivni metodi (metod istinitosnih tablica).
- Problem se tehnički jednostavnije rešava ako je formula u nekom od specijalnih oblika (DNF, KNF, BDD).
- Za DNF se zadovoljivost trivijalno ispituje, ali je postupak prevođenja u DNF netrivijalan. Slično je i sa BDD.
- Postoji efikasan postupak prevođenja formule u KNF (doduše koji ne čuva ekvivalentnost, ali čuva zadovoljivost).
- Zadovoljivost formula u KNF se dalje ispituje nekim od tzv. klauzalnih algoritama.
  - DP procedura (iskazna rezolucija)
  - DPLL procedura
  - CDCL SAT rešavači (unapređena DPLL procedura)
  - Stohastički SAT rešavači (zasnovani na randomizaciji — ne garantuju uvek definitivan odgovor).

DPLL procedura — pregled

- Ispituje zadovoljivost iskaznih formula (u KNF).
- DPLL (Davis-Putnam-Logemann-Loveland, 1962).
- Izmena DP procedure (Davis-Putnam, 1961) u cilju smanjivanja utroška memorije.
- Deo procedure pobijanja za logiku prvog reda.
- Osnova savremenih CDCL SAT rešavača.
- Razlikovaćemo DPLL pretragu i DPLL proceduru (koja uz pretragu uključuje i elemente zaključivanja).

DPLL procedura — ideja koraka pretrage

- Pretraga za zadovoljavajućom valuacijom u okviru DPLL procedure, zasniva se na ispitivanju obe moguće istinitosne vrednosti promenljivih koje se javljaju u formuli.
- Postavljanje vrednosti nekoj promenljivoj indukuje vrednost oba njoj odgovarajuća literala. Iz tehničkih razloga, elementarnim korakom ćemo smatrati ,,postavljanje literala na tačno".
- Vrednost nekog literala l se ,,postavi na tačno" i ispita se zadovoljivost. Ako se ne dobije zadovoljivost, onda se vrednost njemu suprotnog literala $\overline{l}$,,postavi na tačno" i ispita se zadovoljivost.
- ,,Postavljanje literala na tačno" dovodi do uprošćavanja formule (dok se ne stigne do formule čija se zadovoljivost trivijalno očitava — npr. logičke konstante).
- Razne varijante koraka ,,literal se postavi na tačno".

DPLL procedura — $F[[l \to \top]]$

- Literal se može ,,postaviti na tačno" tako što se sva njegova pojavljivanja zamene sa logičkom konstantom $\top$, a pojavljivanja njemu suprotnog literala zamene sa logičkom konstantom $\bot$.
- Neka $F[[l \to \top]]$ označava formulu $F[l \to \top][\bar{l} \to \bot]$.
- Neka $F[[l \to \bot]]$ označava formulu $F[\bar{l} \to \top][l \to \bot]$ (tj. $F[[\bar{l} \to \top]]$).

Korak split

Pretraga u okviru DPLL procedure je zasnovana na koraku split koji ispitivanje zadovoljivosti formule F svodi na ispitivanje zadovoljivosti formula $F[[l \to \top]]$ i $F[[l \to \bot]]$.

Stav (Split)

Neka je F formula, a l literal. Formula F je zadovoljiva, ako i samo ako je zadovoljiva formula $F[[l \to \top]]$ ili je zadovoljiva formula $F[[l \to \bot]]$.

DPLL procedura — KNF formule

- Formule oblika $F[[l \to \top]]$ se mogu uprostiti (eliminacijom konstanti).
- Ovo je prilično jednostavno, ukoliko je formula F u KNF, tj. ako je predstavljena skupom klauza $\mathcal{F}$.
  - Iz svih klauza se ukloni literal $\bar{l}$ (tj. $\bot$ nakon zamene).
  - Uklone se sve klauze koje sadrže literal l (tj. $\top$ nakon zamene).
- Sa logičkog stanovišta najbolje je formulu smatrati skupom klauza, a klauze skupovima literala. Sa stanovišta implementacije, često se umesto skupova koriste liste (nizovi, vektori, . . .).

DPLL i tautologične klauze

- Klauze koje sadrže međusobno suprotne literale ne utiču na zadovoljivost formule.
- Ove klauze je moguće ukloniti pre primene procedure (često već u fazi prevođenja u KNF).
- Ovaj korak ne utiče suštinski na dalje izlaganje.

DPLL pretraga — trivijalni slučajevi

- DPLL pretraga prijavljuje zadovoljivost ukoliko je skup klauza prazan.
- DPLL pretraga prijavljuje nezadovoljivost ukoliko skup klauza sadrži praznu klauzu.

DPLL pretraga — pseudokod

```
function dpll (F : CNF Formula) : (SAT, UNSAT)
begin
   if C is empty then return SAT
   else if there is an empty clause in F then return UNSAT
   else begin
      select a literal l occurring in F
      if dpll(F[l → ⊤]) = SAT then return SAT
      else return dpll(F[l → ⊥])
   end
end
```

Implementacija DPLL pretrage — zamena u KNF

```
definition
setLiteralTrue :: "Literal => Literal set set => Literal set set"
where
  "setLiteralTrue l F ==
     let F' = filter (% c. l ~: c) F;
         F'' = map (% c. remove (negate l) c) F' in
         F''"
```

Implementacija DPLL pretrage — odabir literala za split

- Veoma važna odluka za efikasnost procedure.
- Literal bi trebalo da bude neki od literala koji se javlja u tekućem skupu klauza.

Trivijalna (neefikasna) implementacija bira bilo koji literal iz formule:

```
definition
selectLiteral :: "Literal set set => Literal" where
  "selectLiteral F = some_elem (Union F)"
```

Implementacija DPLL pretrage

```
function dpll :: "Literal set set => bool" where
"dpll F ==
   if F = {} then True
   else if {} : F then False
   else
     let l = selectLiteral F in
     if dpll (setLiteralTrue l F) then True
     else dpll (setLiteralTrue (negate l) F)"
```

Korektnost DPLL pretrage — osnovne leme

Stav

1. *Ako* $v \models l$ *i* $v \models c$, *onda i* $v \models c \setminus \{\bar{l}\}$.
2. *Ako* $v \models \mathcal{F}$ *i* $v \models l$ *onda i* $v \models \mathcal{F}[[l \to \top]]$.
3. *Ako* $v \models \mathcal{F}[[l \to \top]]$ *onda postoji* v' *tako da* $v' \models l$ *i* $v' \models \mathcal{F}$.

Korektnost DPLL pretrage — osnovne leme — dokaz

Dokaz

1. *Pošto* $v \models c$, *i* $v \not\models \bar{l}$ *u* c *postoji literal* l' *različit od* $\bar{l}$ *tako da* $v \models l'$. *Važi da je* $l' \in c \setminus \{\bar{l}\}$, *pa zato i* $v \models c \setminus \{\bar{l}\}$.
2. *Neka je* v *model za* $\mathcal{F}$. *Klauze iz* $\mathcal{F}[[l \to \top]]$ *koje su i u* $\mathcal{F}$ *su trivijalno zadovoljene, dok za svaku drugu klauzu* c *iz* $\mathcal{F}[[l \to \top]]$ *postoji klauza* c' *iz* $\mathcal{F}$ *tako da je* $c' = c \cup \{\bar{l}\}$. *Pošto* $v \models c'$ *i* $v \not\models \bar{l}$, *onda* $v \models c$.
3. *Neka je* v *model za* $\mathcal{F}[[l \to \top]]$, *i neka je* v' *valuacija koja se dobija od* v *postavljanjem vrednosti literala* l *na tačno. Klauze iz* $\mathcal{F}$ *koje ne sadrže ni* l *ni* $\bar{l}$ *se nalaze i u* $\mathcal{F}[[l \to \top]]$ *tako da su zadovoljene u* v, *pa i u* v'. *Klauze iz* $\mathcal{F}$ *koje sadrže* l *su zadovoljene samom konstrukcijom* v'. *Na kraju, za svaku klauzu* $c \in \mathcal{F}$ *koja sadrži* $\bar{l}$ *postoji klauza* $c' \in \mathcal{F}[[l \to \top]]$, *koja ne sadrži* $\bar{l}$, *takva da je* $c = c' \cup \{\bar{l}\}$. *Pošto* $v \models c'$, *to u njoj postoji literal (različit od* $\bar{l}$*) tačan u* v *i on je tačan i u* v', *te* $v' \models c$.

Korektnost koraka split

Stav (Split)

Skup klauza $\mathcal{F}$ je zadovoljiv, ako i samo ako je zadovoljiv $\mathcal{F}[[l \to \top]]$ ili je zadovoljiv $\mathcal{F}[[l \to \bot]]$, za neki literal l.

Dokaz

Neka je v model za $\mathcal{F}$. Ako je $v \vDash l$, onda je v model i za $\mathcal{F}[[l \to \top]]$, a ako $v \nvDash l$, pošto $v \vDash \bar{l}$, onda je v model za $\mathcal{F}[[l \to \bot]]$.
Ako je v model za $\mathcal{F}[[l \to \top]]$, onda postoji v' (dobijena od v postavljanjem vrednosti l na tačno) tako da je v' model za $\mathcal{F}$.
Ako je v model za $\mathcal{F}[[l \to \bot]]$, onda postoji v' (dobijena od v postavljanjem vrednosti l na netačno) tako da je v' model za $\mathcal{F}$.

Korektnost trivijalnih slučajeva

Stav

- *Prazan skup klauza je zadovoljen u svakoj valuaciji.*
- *Prazna klauza nema model.*
- *Ako skup formula sadrži praznu klauzu on nema model (tj. nije zadovoljiv).*

Korektnost DPLL pretrage

Teorema (Zaustavljanje)

Ukoliko se korak split uvek primenjuje na literal koji je deo tekućeg skupa klauza, DPLL pretraga se zaustavlja.

Dokaz

Broj različitih promenljivih u tekućem skupu klauza se smanjuje pri svakom rekurzivnom pozivu, pa se procedura mora zaustaviti.

Teorema (Saglasnost i potpunost)

Skup klauza $\mathcal{F}$ je zadovoljiv ako i samo je `dpll` $\mathcal{F} = True$.

Dokaz

Indukcijom, na osnovu definicije funkcije `dppl`*, uz korišćenje lema o korektnosti pojedinačnih koraka.*

Elementi zaključivanja

- Prethodna procedura ne uključuje nikakve elemente zaključivanja.
- U nekim slučajevima može se izbeći analiziranje obe grane u split pravilu.
- DPLL procedura uvodi dva ovakva koraka koji značajno mogu da smanje prostor pretrage.
  - propagacija jediničnih klauza (eng. unit propagation)
  - eliminacija ,,čistih" literala (eng. pure literal)

Primeri

Primer

$$\mathcal{F}_0 = \{\{\neg x_1, \neg x_2\}, \{x_2\}, \{\neg x_2, x_3, \neg x_4\}, \{\neg x_3, x_4\}, \{x_4, x_5\}, \{\neg x_3, \neg x_4, x_1\}\}$$

Unit propagate: *valuacija koja x_2 postavlja na netačno ne može biti model.*

$$\mathcal{F}_1 = \mathcal{F}_0[[x_2 \to \top]] = \{\{\neg x_1\}, \{x_3, \neg x_4\}, \{\neg x_3, x_4\}, \{x_4, x_5\}, \{\neg x_3, \neg x_4, x_1\}\}$$

Unit propagate: *valuacija koja x_1 postavlja na tačno ne može biti model.*

$$\mathcal{F}_2 = \mathcal{F}_1[[x_1 \to \bot]] = \{\{x_3, \neg x_4\}, \{\neg x_3, x_4\}, \{x_4, x_5\}, \{\neg x_3, \neg x_4\}\}$$

Pure literal: *Ne smeta da se x_5 postavi na tačno.*

$$\mathcal{F}_3 = \mathcal{F}_2[[x_5 \to \top]] = \{\{x_3, \neg x_4\}, \{\neg x_3, x_4\}, \{\neg x_3, \neg x_4\}\}$$

Tek sada kreće pretraga.

Elementi zaključivanja

Unit propagation — Ako formula sadrži klauzu sa tačno jednim literalom $\{l\}$, onda taj literal mora biti tačan u eventualnoj zadovoljavajućoj valuaciji.

Pure literal — Ako se u formuli javlja neki literal l, a ne javlja se njemu suprotan literal $\bar{l}$, onda taj literal može biti postavljen na tačno u eventualnoj zadovoljavajućoj valuaciji.

DPLL procedura — pseudokod

```
function dpll (F : CNF Formula) : bool
begin
   if F is empty then return True
   else if there is an empty clause in F then return False
   else if there is a pure literal l in F then return dpll(F[l → ⊤])
   else there is a unit clause [l] in F then return dpll(C[l → ⊤])
   else begin
      select a literal l occurring in F
      return dpll(F[l → ⊤]) or dpll(F[l → ⊥])
   end
end
```

Implementacija koraka propagacije jediničnih klauza

```
definition
unit_propagate :: "Literal set set => Literal set set option"
where
"unit_propagate F ==
  let units = Union (filter (% c. card c = 1) F) in
  (if units = {} then
     None
   else
     let l = some_elem units in
     Some (setLiteralTrue l F))"
```

Implementacija koraka propagacije jediničnih klauza

Moguće je i istovremeno propagirati sve jedinične klauze.

```
definition
unit_propagate :: "Literal set set => Literal set set option"
where
"unit_propagate F ==
  let units = Union (filter (% c. card c = 1) F) in
  (if units = {} then
     None
   else
     let F' = filter (% c. intersect c units = {}) F;
         F'' = map (% c. c - (map negate units)) F' in
         F'')"
```

Implementacija koraka eliminacije čistih literala

```
definition
pure_literal :: "Literal set set => Literal set set option"
where
"pure_literal F ==
   let lits = Union F;
       pos = filter positive lits;
       neg = map negate (filter negative lits);
       pos_only = pos - neg;   neg_only = neg - pos;
       pure = union pos_only (map negate neg_only) in
     (if pure = {} then
         None
      else
         let l = some_elem pure in
            Some (setLiteralTrue l F))"
```

Implementacija koraka eliminacije čistih literala

Moguće je istovremeno ukloniti sve čiste literale.

```
definition
pure_literal :: "Literal set set => Literal set set option"
where
"pure_literal F ==
   let lits = Union F;
       pos = filter positive lits;
       neg = map negate (filter negative lits);
       pos_only = pos - neg;   neg_only = neg - pos;
       pure = union pos_only (map negate neg_only) in
     (if pure = {} then
         None
      else
         Some (filter (% c. intersect c pure = {}) F))"
```

Split kao dodavanje jediničnih klauza

- Literal se može ,,postaviti na tačno" tako što se konjunkcijom pridruži formuli, tj. doda kao jedinična klauza.

Stav

Neka je F formula, a l literal. Formula F je zadovoljiva, ako i samo ako je zadovoljiva formula $F \wedge l$ ili je zadovoljiva formula $F \wedge \bar{l}$.

Dokaz

Neka je v model formule F. Ako je u ovom modelu literal l tačan, tada je v model i za $F \wedge l$, a ako je u ovom modelu literal l netačan, tada je v model za $F \wedge \bar{l}$.
Obratno, svaki model za $F \wedge l$ je trivijalno model i za F, i svaki model za $F \wedge \bar{l}$ je trivijalno model i za F.

Implementacija DPLL procedure

```
function dpll :: "Literal set set => bool" where
"dpll F ==
   if F = {} then
      True
   else if {} : F then
      False
   else (case pure_literal clauses of
      Some clauses' => dpll clauses'
      | None => (case unit_propagate clauses of
        Some clauses' => dpll clauses'
      | None =>
        (let l = selectLiteral F in
          if dpll (union F {l}) then
             True
          else
             dpll (union F (negate l)))))"
```

Korektnost koraka propagacije jediničnih klauza

Stav (Unit propagate)

Ukoliko je $\{l\} \in \mathcal{F}$*, tada je*

$$\mathcal{F} \equiv_s \mathcal{F}[[l \to \top]].$$

Dokaz

Neka je v *model za* $\mathcal{F}$*. Pošto je* $\{l\} \in \mathcal{F}$*, važi* $v \vDash l$*, te* $v \vDash \mathcal{F}[[l \to \top]]$*.*
Ako je v *model za* $\mathcal{F}[[l \to \top]]$*, onda postoji* v' *(dobijena od* v *postavljanjem vrednosti* l *na tačno) tako da je* v' *model za* $\mathcal{F}$*.*

Korektnost koraka propagacije jediničnih klauza

Stav (Unit propagate)

Ukoliko je $\{l\} \in \mathcal{F}$*, tada je*

$$\mathcal{F} \equiv_s \mathcal{F}[[l \to \top]].$$

Dokaz (Alternativni dokaz)

Ako je $\{l\} \in \mathcal{F}$ *onda* $\mathcal{F}[[l \to \bot]]$ *sadrži praznu klauzu pa nije zadovoljiva. Tvrđenje tada sledi na osnovu stava o koraku split.*

Korektnost koraka eliminacije čistih literala

Stav (Pure literal)

Ako se u skupu klauza $\mathcal{F}$ javlja literal l, a ne javlja literal $\bar{l}$, tada je

$$\mathcal{F} \equiv_s \mathcal{F}[[l \to \top]].$$

Dokaz

Ako je $\mathcal{F}$ zadovoljiv, zadovoljiv je i $\mathcal{F}[[l \to \top]]$ (kao njegov podskup). Ako je v model za $\mathcal{F}[[l \to \top]]$, onda postoji v' (dobijena od v postavljanjem vrednosti l na tačno) tako da je v' model za $\mathcal{F}$.

Korektnost koraka eliminacije čistih literala

Stav (Pure literal)

Ako se u skupu klauza $\mathcal{F}$ javlja literal l, a ne javlja literal $\bar{l}$, tada je

$$\mathcal{F} \equiv_s \mathcal{F}[[l \to \top]].$$

Dokaz (Alternativni dokaz)

Pošto je $l \in \mathcal{F}$, a $\bar{l} \notin \mathcal{F}$, važi da je $\mathcal{F}[[l \to \top]] \subseteq \mathcal{F}[[l \to \bot]]$. Tada, ako je zadovoljiv $\mathcal{F}[[l \to \bot]]$, zadovoljiv je i $\mathcal{F}[[l \to \top]]$, pa tvrđenje sledi na osnovu stava o koraku split.

Efikasnost DPLL procedure

- U najgorem slučaju, DPLL procedura zahteva eksponencijalni broj koraka u odnosu na veličinu ulazne formule (korak split generiše eksponencijalnost). Ovaj problem nije moguće razrešiti.
- Osnovna varijanta procedure je izrazito neefikasna:
  - Pri svakom koraku vrši se modifikacija formule (skupa klauza).
  - Rekurzivna implementacija značajno kvari efikasnost (formule koje mogu da budu veoma velike se prenose kao parametar funkcije i slažu na programski stek).

Prosleđivanje parcijalnih valuacija

- Umesto zamene sa $\top$ u formuli, literal se može ,,postaviti na tačno" tako što se formula ne menja, dok se zasebno održava parcijalna valuacija u kojoj se ,,čuva" vrednost svih promenljivih i literala.
- Parcijalna valuacija može da se shvati kao skup literala koji je konzistentan tj. ne sadrži istovremeno dva suprotna literala.
- Pošto je valuacija parcijalna, promenljiva, odnosno literal, mogu u njoj da budu tačni, netačni ili nedefinisani.

DPLL procedura — prosleđivanje valuacija — pseudokod

```
function dpll (M : Valuation) : (SAT, UNSAT)
begin
   if M ⊨¬ F (i.e., there is c ∈ F s.t. M ⊨¬ c) then return UNSAT
   else if M ⊨ F then return SAT
   else there is a unit clause (i.e., there is a clause
      l ∨ l_1 ∨ ... ∨ l_k in F s.t. l, l̄ ∉ M, l̄_1, ..., l̄_k ∈ M) then
      return dpll(M ∪ {l})
   else begin
      select a literal l s.t. l ∈ F, l, l̄ ∉ M
      if dpll(M ∪ {l}) = SAT then return SAT
      else return dpll(M ∪ {l̄})
   end
end
```

Provera tačnosti u parcijalnoj valuaciji?

- Pokazuje se da postoje efikasni postupci za proveru da li u formuli postoji netačna ili jedinična klauza, ali ne postoji efikasan postupak kojim bi se utvrdilo da li je formula tačna u nekoj parcijalnoj valuaciji.
- Može li se taj test izbeći?

Stav

Ukoliko su sve promenljive iz $\mathcal{F}$ definisane u parcijalnoj valuaciji M, tada je $\mathcal{F}$ ili tačna ili netačna u M, tj. $M \models \mathcal{F}$ ili $M \models \neg \mathcal{F}$.

DPLL procedura — prosleđivanje valuacija — pseudokod

```
function dpll (M : Valuation) : (SAT, UNSAT)
begin
   if M ⊨¬ F (i.e., there is c ∈ F s.t. M ⊨¬ c) then return UNSAT
   else if M is total wrt. the variables of F then return SAT
   else there is a unit clause (i.e., there is a clause
      l ∨ l_1 ∨ ... ∨ l_k in F s.t. l, l̄ ∉ M, l̄_1, ..., l̄_k ∈ M) then
      return dpll(M ∪ {l})
   else begin
      select a literal l s.t. l ∈ F, l, l̄ ∉ M
      if dpll(M ∪ {l}) = SAT then return SAT
      else return dpll(M ∪ {l̄})
   end
end
```

Veza sa istinitosnim tablicama

- DPLL pretraga uz prenos parcijalnih valuacija prilično podseća na metod ispitivanja zadovoljivosti korišćenjem istinitosnih tablica.
- Osnovna razlika je što se provera vrednosti vrši rano — prilikom svakog proširivanja valuacije atomom, a ne tek kada se valuacija popuni svim atomima iz formule.

Pojam jedinične klauze

- Ukoliko se vrši uprošćavanje formule, iz klauza se uklanjaju literali i jedinične klauze su one koje imaju tačno jedan literal.
- U slučaju da se ne vrši uprošćavanje formule već izgradnja parcijalne valuacije, pojam jediničnih klauza se menja.

Definicija

Klauza c je jedinična u odnosu na parcijalnu valuaciju v akko sadrži literal l nedefinisan u v, dok su joj svi literali različiti od l netačni u v.

Slično se može proširiti i pojam čistih literala.

Primer

Primer

$$\mathcal{F} = \{\{\neg x_1, \neg x_2\}, \{x_2\}, \{\neg x_2, x_3, \neg x_4\}, \{\neg x_3, x_4\}, \{x_4, x_5\}\}$$

$$v = \{\}$$

Unit propagate: *Klauza* $\{x_2\}$ *je jedinična.*

$$v = \{x_2\}$$

Unit propagate: *Klauza* $\{\neg x_1, \neg x_2\}$ *je jedinična.*

$$v = \{x_2, \neg x_1\}$$

Pure literal: *Literal* x_5 *je čist.*

$$v = \{x_2, \neg x_1, x_5\}$$

Tek sada kreće pretraga.

Primer

$$\mathcal{F} = \{\{\neg x_1, \neg x_2\}, \{x_2\}, \{\neg x_2, x_3, \neg x_4\}, \{\neg x_3, x_4\}, \{x_4, x_5\}, \{\neg x_3, \neg x_4, x_1\}\}$$

dpll($\{\}$)
dpll($\{x_2\}$) — *unitPropagate* ($c = \{x_2\}$)
dpll($\{x_2, \neg x_1\}$) — *unitPropagate* ($c = \{\neg x_1, \neg x_2\}$)
dpll($\{x_2, \neg x_1, x_5\}$) – *pureLiteral* (x_5)
dpll($\{x_2, \neg x_1, x_5, x_3\}$) — *split* ($x_3$)
dpll($\{x_2, \neg x_1, x_5, x_3, x_4\}$) — *unitPropagate* ($c = \{\neg x_3, x_4\}$)
return False — $c = \{\neg x_3, \neg x_4, x_1\}$

dpll($\{x_2, \neg x_1, x_5, \neg x_3\}$) — *split* ($x_3$) – *druga grana*
dpll($\{x_2, \neg x_1, x_5, \neg x_3, \neg x_4\}$) — *unitPropagate* ($c = \{\neg x_2, x_3, \neg x_4\}$)
return True

Primer — korak ka iterativnoj implementaciji?

Primer

$$\mathcal{F} = \{\{\neg x_1, \neg x_2\}, \{x_2\}, \{\neg x_2, x_3, \neg x_4\}, \{\neg x_3, x_4\}, \{x_4, x_5\}, \{\neg x_3, \neg x_4, x_1\}\}$$

pravilo	M
unitPropagate$(c = \{x_2\})$	$\{x_2\}$
unitPropagate$(c = \{\neg x_1, \neg x_2\})$	$\{x_2, \neg x_1\}$
pureLiteral (x_5)	$\{x_2, \neg x_1, x_5\}$
decide (x_3)	$\{x_2, \neg x_1, x_5, x_3\}$
unitPropagate$(c = \{\neg x_3, x_4\}\})$	$\{x_2, \neg x_1, x_5, x_3, x_4\}$
backtrack $(M \models \neg \{\neg x_3, \neg x_4, x_1\})$	$\{x_2, \neg x_1, x_5, \neg x_3\}$
unitPropagate $(c = \{\neg x_2, x_3, \neg x_4\}\})$	$\{x_2, \neg x_1, x_5, \neg x_3, \neg x_4\}$

Kako znati do kog literala se vraćamo?

Primer — korak ka iterativnoj implementaciji?

Primer

$$\mathcal{F} = \{\{\neg x_1, \neg x_2\}, \{x_2\}, \{\neg x_2, x_3, \neg x_4\}, \{\neg x_3, x_4\}, \{x_4, x_5\}, \{\neg x_3, \neg x_4, x_1\}\}$$

pravilo	M
unitPropagate$(c = \{x_2\})$	$[x_2]$
unitPropagate$(c = \{\neg x_1, \neg x_2\})$	$[x_2, \neg x_1]$
pureLiteral (x_5)	$[x_2, \neg x_1, x_5]$
decide (x_3)	$[x_2, \neg x_1, x_5, \vert x_3]$
unitPropagate$(c = \{\neg x_3, x_4\}\})$	$[x_2, \neg x_1, x_5, \vert x_3, x_4]$
backtrack $(M \models \neg \{\neg x_3, \neg x_4, x_1\})$	$[x_2, \neg x_1, x_5, \neg x_3]$
unitPropagate $(c = \{\neg x_2, x_3, \neg x_4\}\})$	$[x_2, \neg x_1, x_5, \neg x_3, \neg x_4]$

Kako znati do kog literala se vraćamo?
Valuacije moraju biti liste označenih literala.

Apstraktni opisi procedure

- Razmatranje procedure opisane na nivou koda je obično suviše komplikovano jer sadrži dosta implementacionih detalja.
- Umesto koda, moguće je opisati tekuće stanje procedure i dati pravila koja opisuju kako se može preći iz stanja u stanje.

Iterativna implementacija — sistem stanja

DPLL pretraga

Stanje rešavača

- M - označena valuacija

Decide:

$$\frac{l \in F \quad l, \bar{l} \notin M}{M := M \,|\, l}$$

UnitPropagate:

$$\frac{l \vee l_1 \vee \ldots \vee l_k \in F \qquad \bar{l}_1, \ldots, \bar{l}_k \in M \qquad l, \bar{l} \notin M}{M := M \; l}$$

Backtrack:

$$\frac{M \models \neg F \quad M = M' \,|\, l \, M'' \quad \text{decisions } M'' = [\,]}{M := M' \; \bar{l}}$$

Primer

$F = [[-1, +2], [-1, -3], [-2, +4, +5], [+3, -4, -5], [-4, +5]]$

Primenjeno pravilo	*sat?*	M
	UNDEF	$[\,]$
Decide $(l = +1)$	*UNDEF*	$[\mid +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +1,\ +2]$
UnitProp $(c = [-1, -3],\ l = -3)$	*UNDEF*	$[\mid +1, +2,\ -3]$
Decide $(l = +4)$	*UNDEF*	$[\mid +1, +2, -3, \mid +4]$
UnitProp $(c = [-4, +5],\ l = +5)$	*UNDEF*	$[\mid +1, +2, -3, \mid +4, +5]$
Backtrack $(M \models \neg\, [+3, -4, -5])$	*UNDEF*	$[\mid +1, +2, -3, -4]$
UnitProp $(c = [-2, +4, +5],\ l = +5)$	*UNDEF*	$[\mid +1, +2, -3, -4,\ +5]$
$M \not\models \neg\, F$, $(\text{vars}\ M) = (\text{vars}\ F)$	*SAT*	$[\mid +1, +2, -3, -4, +5]$

Primer

$F = [[-1, +2], [-1, -3], [-2, +4, +5], [+3, -4, -5], [-4, +5]]$

Primenjeno pravilo	*sat?*	M
	UNDEF	$[\,]$
Decide $(l = +1)$	*UNDEF*	$[\mid +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +1,\ +2]$
UnitProp $(c = [-1, -3],\ l = -3)$	*UNDEF*	$[\mid +1, +2,\ -3]$
Decide $(l = +4)$	*UNDEF*	$[\mid +1, +2, -3, \mid +4]$
UnitProp $(c = [-4, +5],\ l = +5)$	*UNDEF*	$[\mid +1, +2, -3, \mid +4, +5]$
Backtrack $(M \models \neg\, [+3, -4, -5])$	*UNDEF*	$[\mid +1, +2, -3, -4]$
UnitProp $(c = [-2, +4, +5],\ l = +5)$	*UNDEF*	$[\mid +1, +2, -3, -4,\ +5]$
$M \not\models \neg\, F$, $(\text{vars}\ M) = (\text{vars}\ F)$	*SAT*	$[\mid +1, +2, -3, -4, +5]$

Primer

$F = [[-1, +2], [-1, -3], [-2, +4, +5], [+3, -4, -5], [-4, +5]]$

Primenjeno pravilo	*sat?*	*M*
	UNDEF	$[\,]$
Decide $(l = +1)$	*UNDEF*	$[\mid +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +1,\ +2]$
UnitProp $(c = [-1, -3],\ l = -3)$	*UNDEF*	$[\mid +1, +2,\ -3]$
Decide $(l = +4)$	*UNDEF*	$[\mid +1, +2, -3, \mid +4]$
UnitProp $(c = [-4, +5],\ l = +5)$	*UNDEF*	$[\mid +1, +2, -3, \mid +4, +5]$
Backtrack $(M \models \neg\, [+3, -4, -5])$	*UNDEF*	$[\mid +1, +2, -3, -4]$
UnitProp $(c = [-2, +4, +5],\ l = +5)$	*UNDEF*	$[\mid +1, +2, -3, -4,\ +5]$
$M \not\models \neg\, F$, $(\text{vars}\ M) = (\text{vars}\ F)$	*SAT*	$[\mid +1, +2, -3, -4, +5]$

Primer

$F = [[-1, +2], [-1, -3], [-2, +4, +5], [+3, -4, -5], [-4, +5]]$

Primenjeno pravilo	*sat?*	M
	UNDEF	$[\,]$
Decide $(l = +1)$	*UNDEF*	$[\mid +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +1,\ +2]$
UnitProp $(c = [-1, -3],\ l = -3)$	*UNDEF*	$[\mid +1, +2,\ -3]$
Decide $(l = +4)$	*UNDEF*	$[\mid +1, +2, -3, \mid +4]$
UnitProp $(c = [-4, +5],\ l = +5)$	*UNDEF*	$[\mid +1, +2, -3, \mid +4, +5]$
Backtrack $(M \models \neg\, [+3, -4, -5])$	*UNDEF*	$[\mid +1, +2, -3, -4]$
UnitProp $(c = [-2, +4, +5],\ l = +5)$	*UNDEF*	$[\mid +1, +2, -3, -4,\ +5]$
$M \not\models \neg\, F$, $(\text{vars}\ M) = (\text{vars}\ F)$	*SAT*	$[\mid +1, +2, -3, -4, +5]$

Primer

$F = [[-1, +2], [-1, -3], [-2, +4, +5], [+3, -4, -5], [-4, +5]]$

Primenjeno pravilo	*sat?*	M
	UNDEF	$[\,]$
Decide $(l = +1)$	*UNDEF*	$[\mid +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +1,\ +2]$
UnitProp $(c = [-1, -3],\ l = -3)$	*UNDEF*	$[\mid +1, +2,\ -3]$
Decide $(l = +4)$	*UNDEF*	$[\mid +1, +2, -3, \mid +4]$
UnitProp $(c = [-4, +5],\ l = +5)$	*UNDEF*	$[\mid +1, +2, -3, \mid +4, +5]$
Backtrack $(M \models \neg\, [+3, -4, -5])$	*UNDEF*	$[\mid +1, +2, -3, -4]$
UnitProp $(c = [-2, +4, +5],\ l = +5)$	*UNDEF*	$[\mid +1, +2, -3, -4,\ +5]$
$M \not\models \neg\, F$, $(\text{vars}\ M) = (\text{vars}\ F)$	*SAT*	$[\mid +1, +2, -3, -4, +5]$

Primer

$F = [[-1, +2], [-1, -3], [-2, +4, +5], [+3, -4, -5], [-4, +5]]$

Primenjeno pravilo	*sat?*	M
	UNDEF	$[\,]$
Decide $(l = +1)$	*UNDEF*	$[\mid +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +1,\ +2]$
UnitProp $(c = [-1, -3],\ l = -3)$	*UNDEF*	$[\mid +1, +2,\ -3]$
Decide $(l = +4)$	*UNDEF*	$[\mid +1, +2, -3, \mid +4]$
UnitProp $(c = [-4, +5],\ l = +5)$	*UNDEF*	$[\mid +1, +2, -3, \mid +4, +5]$
Backtrack $(M \models \neg\, [+3, -4, -5])$	*UNDEF*	$[\mid +1, +2, -3, -4]$
UnitProp $(c = [-2, +4, +5],\ l = +5)$	*UNDEF*	$[\mid +1, +2, -3, -4,\ +5]$
$M \not\models \neg\, F$, $(\text{vars}\ M) = (\text{vars}\ F)$	*SAT*	$[\mid +1, +2, -3, -4, +5]$

Primer

$F = [[-1, +2], [-1, -3], [-2, +4, +5], [+3, -4, -5], [-4, +5]]$

Primenjeno pravilo	*sat?*	M
	UNDEF	$[\,]$
Decide $(l = +1)$	*UNDEF*	$[\mid +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +1,\ +2]$
UnitProp $(c = [-1, -3],\ l = -3)$	*UNDEF*	$[\mid +1, +2,\ -3]$
Decide $(l = +4)$	*UNDEF*	$[\mid +1, +2, -3, \mid +4]$
UnitProp $(c = [-4, +5],\ l = +5)$	*UNDEF*	$[\mid +1, +2, -3, \mid +4, +5]$
Backtrack $(M \models \neg\, [+3, -4, -5])$	*UNDEF*	$[\mid +1, +2, -3, -4]$
UnitProp $(c = [-2, +4, +5],\ l = +5)$	*UNDEF*	$[\mid +1, +2, -3, -4,\ +5]$
$M \not\models \neg\, F$, $(\text{vars}\ M) = (\text{vars}\ F)$	*SAT*	$[\mid +1, +2, -3, -4, +5]$

Primer

$F = [[-1, +2], [-1, -3], [-2, +4, +5], [+3, -4, -5], [-4, +5]]$

Primenjeno pravilo	*sat?*	M
	UNDEF	$[\,]$
Decide $(l = +1)$	*UNDEF*	$[\mid +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +1,\ +2]$
UnitProp $(c = [-1, -3],\ l = -3)$	*UNDEF*	$[\mid +1, +2,\ -3]$
Decide $(l = +4)$	*UNDEF*	$[\mid +1, +2, -3, \mid +4]$
UnitProp $(c = [-4, +5],\ l = +5)$	*UNDEF*	$[\mid +1, +2, -3, \mid +4, +5]$
Backtrack $(M \models \neg [+3, -4, -5])$	*UNDEF*	$[\mid +1, +2, -3, -4]$
UnitProp $(c = [-2, +4, +5],\ l = +5)$	*UNDEF*	$[\mid +1, +2, -3, -4,\ +5]$
$M \not\models \neg F$, $(\text{vars } M) = (\text{vars } F)$	*SAT*	$[\mid +1, +2, -3, -4, +5]$

Primer

$F = [[-1, +2], [-1, -3], [-2, +4, +5], [+3, -4, -5], [-4, +5]]$

Primenjeno pravilo	*sat?*	M
	UNDEF	$[\,]$
Decide $(l = +1)$	*UNDEF*	$[\mid +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +1,\ +2]$
UnitProp $(c = [-1, -3],\ l = -3)$	*UNDEF*	$[\mid +1, +2,\ -3]$
Decide $(l = +4)$	*UNDEF*	$[\mid +1, +2, -3, \mid +4]$
UnitProp $(c = [-4, +5],\ l = +5)$	*UNDEF*	$[\mid +1, +2, -3, \mid +4, +5]$
Backtrack $(M \models \neg\, [+3, -4, -5])$	*UNDEF*	$[\mid +1, +2, -3, -4]$
UnitProp $(c = [-2, +4, +5],\ l = +5)$	*UNDEF*	$[\mid +1, +2, -3, -4,\ +5]$
$M \not\models \neg\, F$, $(\text{vars}\ M) = (\text{vars}\ F)$	*SAT*	$[\mid +1, +2, -3, -4, +5]$

Primer

Neka je $\mathcal{F}_0 = [\ [-1,+2],\ [-1,-3,+5,+7],\ [-1,-2,+5,-7],\ [-2,+3],\ [+2,+4],\ [-2,-5,+7],\ [-3,-6,-7],\ [-5,+6]\ ]$.

pravilo	M
	$[\,]$
decide $(l = +1)$,	$[\vert +1]$
unitPropagate $(c = [-1,+2],\ l = +2)$	$[\vert +1,+2]$
unitPropagate $(c = [-2,+3],\ l = +3)$	$[\vert +1,+2,+3]$
decide $(l = +4)$	$[\vert +1,+2,+3,\vert +4]$
decide $(l = +5)$	$[\vert +1,+2,+3,\vert +4,\vert +5]$
unitPropagate $(c = [-5,+6],\ l = +6)$	$[\vert +1,+2,+3,\vert +4,\vert +5,+6]$
unitPropagate $(c = [-2,-5,+7],\ l = +7)$	$[\vert +1,+2,+3,\vert +4,\vert +5,+6,+7]$
backtrack $(M \models \neg\,[-3,-6,-7])$	$[\vert +1,+2,+3,\vert +4,-5]$
unitPropagate $(c = [-1,-3,+5,+7],\ l = +7)$	$[\vert +1,+2,+3,\vert +4,-5,+7]$
backtrack $(M \models \neg\,[-1,-2,+5,-7])$	$[\vert +1,+2,+3,-4]$
decide $(l = +5)$	$[\vert +1,+2,+3,-4,\vert +5]$
unitPropagate $(c = [-5,+6],\ l = +6)$	$[\vert +1,+2,+3,-4,\vert +5,+6]$
unitPropagate $(c = [-2,-5,+7],\ l = +7)$	$[\vert +1,+2,+3,-4,\vert +5,+6,+7]$

Primer

Neka je $\mathcal{F}_0 = [\ [-1,+2],\ [-1,-3,+5,+7],\ [-1,-2,+5,-7],\ [-2,+3],\ [+2,+4],\ [-2,-5,+7],\ [-3,-6,-7],\ [-5,+6]\ ]$.

pravilo	M
backtrack $(M \models \neg [-3,-6,-7])$	$[\mid +1,+2,+3,-4,-5]$
unitPropagate $(c = [-1,-3,+5,+7],\ l = +7)$	$[\mid +1,+2,+3,-4,-5,+7]$
backtrack $(M \models \neg [-1,-2,+5,-7]$	$[-1]$
decide $(l = +2)$	$[-1,\mid +2]$
unitPropagate $(c = [-2,+3],\ l = +3)$	$[-1,\mid +2,+3]$
decide $(l = +4)$	$[-1,\mid +2,+3,\mid +4]$
decide $(l = +5)$	$[-1,\mid +2,+3,\mid +4,\mid +5]$
unitPropagate $(c = [-5,+6],\ l = +6)$	$[-1,\mid +2,+3,\mid +4,\mid +5,+6]$
unitPropagate $(c = [-2,-5,+7],\ l = +7)$	$[-1,\mid +2,+3,\mid +4,\mid +5,+6,+7]$
backtrack $M \models \neg [-3,-6,-7]$	$[-1,\mid +2,+3,\mid +4,-5]$
decide $(l = +6)$	$[-1,\mid +2,+3,\mid +4,-5,\mid +6]$
unitPropagate $(c = [-3,-6,-7],\ l = -7)$	$[-1,\mid +2,+3,\mid +4,-5,\mid +6,-7]$

Ključna pitanja za implementaciju

- Prilikom izmene valuacije M, kako efikasno otkriti klauze iz F koje su postale netačne i/ili jedinične? Neophodno je korišćenje specijalizovanih kompleksnih struktura podataka (npr. shema dva posmatrana literala).
- Kako odabrati literal za pravilo Decide? Neophodno je koristiti pametne heuristike koje ovo kontrolišu.

Ključna pitanja za implementaciju

- Prilikom izmene valuacije M, kako efikasno otkriti klauze iz F koje su postale netačne i/ili jedinične? Neophodno je korišćenje specijalizovanih kompleksnih struktura podataka (npr. shema dva posmatrana literala).
- Kako odabrati literal za pravilo Decide? Neophodno je koristiti pametne heuristike koje ovo kontrolišu.

Ključna pitanja za implementaciju

- Prilikom izmene valuacije M, kako efikasno otkriti klauze iz F koje su postale netačne i/ili jedinične? Neophodno je korišćenje specijalizovanih kompleksnih struktura podataka (npr. shema dva posmatrana literala).
- Kako odabrati literal za pravilo Decide? Neophodno je koristiti pametne heuristike koje ovo kontrolišu.

Primer

$F_0 = [[-1, +2], [-3, +4], [-1, -3, +5], [-2, -4, -5], [-2, +3, +5, -6], [-1, +3, -5, -6], [+1, -6], [+1, +7]]$.

Primenjeno pravilo	*satFlag*	*M*
	UNDEF	$[]$
Decide $(l = +6)$	*UNDEF*	$[\mid +6]$
UnitProp $(c = [+1, -6],\ l = +1)$	*UNDEF*	$[\mid +6, +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +6, +1, +2]$
Decide $(l = +7)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7]$
Decide $(l = +3)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3]$
UnitProp $(c = [-3, +4],\ l = +4)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4]$
UnitProp $(c = [-1, -3, +5],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$
Backtrack $(M \models \neg [-2, -4, -5])$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, -3]$
UnitProp $(c = [-2, +3, +5, -6],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, -3, +5]$
Backtrack $(M \models \neg [-1, +3, -5, -6])$	*UNDEF*	$[\mid +6, +1, +2, -7]$
Decide $(l = +3)$	*UNDEF*	$[\mid +6, +1, +2, -7, \mid +3]$
UnitProp $(c = [-3, +4],\ l = +4)$	*UNDEF*	$[\mid +6, +1, +2, -7, \mid +3, +4]$
UnitProp $(c = [-1, -3, +5],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, -7, \mid +3, +4, +5]$
Backtrack $(M \models \neg [-2, -4, -5])$	*UNDEF*	$[\mid +6, +1, +2, -7, -3]$
UnitProp $(c = [-2, +3, +5, -6],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, -7, -3, +5]$
Backtrack $(M \models \neg [-1, +3, -5, -6])$	*UNDEF*	$[-6]$

Primer

$F_0 = [[-1, +2], [-3, +4], [-1, -3, +5], [-2, -4, -5], [-2, +3, +5, -6], [-1, +3, -5, -6], [+1, -6], [+1, +7]]$.

Primenjeno pravilo	*satFlag*	*M*
	UNDEF	$[]$
Decide $(l = +6)$	*UNDEF*	$[\mid +6]$
UnitProp $(c = [+1, -6],\ l = +1)$	*UNDEF*	$[\mid +6, +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +6, +1, +2]$
Decide $(l = +7)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7]$
Decide $(l = +3)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3]$
UnitProp $(c = [-3, +4],\ l = +4)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4]$
UnitProp $(c = [-1, -3, +5],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$
Backtrack $(M \models \neg [-2, -4, -5])$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, -3]$
UnitProp $(c = [-2, +3, +5, -6],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, -3, +5]$
Backtrack $(M \models \neg [-1, +3, -5, -6])$	*UNDEF*	$[\mid +6, +1, +2, -7]$
Decide $(l = +3)$	*UNDEF*	$[\mid +6, +1, +2, -7, \mid +3]$
UnitProp $(c = [-3, +4],\ l = +4)$	*UNDEF*	$[\mid +6, +1, +2, -7, \mid +3, +4]$
UnitProp $(c = [-1, -3, +5],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, -7, \mid +3, +4, +5]$
Backtrack $(M \models \neg [-2, -4, -5])$	*UNDEF*	$[\mid +6, +1, +2, -7, -3]$
UnitProp $(c = [-2, +3, +5, -6],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, -7, -3, +5]$
Backtrack $(M \models \neg [-1, +3, -5, -6])$	*UNDEF*	$[-6]$

Problemi

- Niz koraka nakon pretpostavke $+7$ je pokazao da ni $+3$ ni -3 nisu kompatibilni sa prethodnim pretpostavkama.
- pravilo Backtrack menja uvek samo poslednju pretpostavku.
- Isti niz koraka pokazuje da ni ovaj put ni $+3$ ni -3 nisu kompatibilni sa prethodnim pretpostavkama.
- Pretpostavka $+7$ je potpuno irelevantna za konflikt koji se dogodio i ponavljanje postupka sa -7 je gubitak vremena.

Rešenje: analiza konflikata (eng. conflict analysis) i uvođenje povratnih skokova (eng. backjumping).

Problemi

- Niz koraka nakon pretpostavke $+7$ je pokazao da ni $+3$ ni -3 nisu kompatibilni sa prethodnim pretpostavkama.
- pravilo Backtrack menja uvek samo poslednju pretpostavku.
- Isti niz koraka pokazuje da ni ovaj put ni $+3$ ni -3 nisu kompatibilni sa prethodnim pretpostavkama.
- Pretpostavka $+7$ je potpuno irelevantna za konflikt koji se dogodio i ponavljanje postupka sa -7 je gubitak vremena.

Rešenje: analiza konflikata (eng. conflict analysis) i uvođenje povratnih skokova (eng. backjumping).

Primer

Primenjeno pravilo	*satFlag*	*M*
...		
Backtrack $(M \models \neg [-1, 3, -5, -6])$	*UNDEF*	$[-6]$
Decide $(l = 1)$	*UNDEF*	$[-6, \|1]$
UnitProp $(c = [-1, 2],\ l = 2)$	*UNDEF*	$[-6, \|1, 2]$
Decide $(l = 7)$	*UNDEF*	$[-6, \|1, 2, \|7]$
Decide $(l = 3)$	*UNDEF*	$[-6, \|1, 2, \|7, \|3]$
UnitProp $(c = [-3, 4],\ l = 4)$	*UNDEF*	$[-6, \|1, 2, \|7, \|3, 4]$
UnitProp $(c = [-1, -3, 5],\ l = 5)$	*UNDEF*	$[-6, \|1, 2, \|7, \|3, 4, 5]$
Backtrack $(M \models \neg [-2, -4, -5])$	*UNDEF*	$[-6, \|1, 2, \|7, -3]$
Decide $(l = 4)$	*UNDEF*	$[-6, \|1, 2, \|7, -3, \|4]$
UnitProp $(c = [-2, -4, -5],\ l = -5)$	*UNDEF*	$[-6, \|1, 2, \|7, -3, \|4, -5]$
$M \not\models \neg F_0$, $(\text{vars } M) = (\text{vars } F_0)$	*SAT*	$[-6, \|1, 2, \|7, -3, \|4, -5]$

Primer

Primenjeno pravilo	*satFlag*	*M*
...		
Backtrack $(M \models \neg\, [-1, 3, -5, -6])$	*UNDEF*	$[-6]$
Decide $(l = 1)$	*UNDEF*	$[-6, \vert 1]$
UnitProp $(c = [-1, 2],\ l = 2)$	*UNDEF*	$[-6, \vert 1, 2]$
Decide $(l = 7)$	*UNDEF*	$[-6, \vert 1, 2, \vert 7]$
Decide $(l = 3)$	*UNDEF*	$[-6, \vert 1, 2, \vert 7, \vert 3]$
UnitProp $(c = [-3, 4],\ l = 4)$	*UNDEF*	$[-6, \vert 1, 2, \vert 7, \vert 3, 4]$
UnitProp $(c = [-1, -3, 5],\ l = 5)$	*UNDEF*	$[-6, \vert 1, 2, \vert 7, \vert 3, 4, 5]$
Backtrack $(M \models \neg\, [-2, -4, -5])$	*UNDEF*	$[-6, \vert 1, 2, \vert 7, -3]$
Decide $(l = 4)$	*UNDEF*	$[-6, \vert 1, 2, \vert 7, -3, \vert 4]$
UnitProp $(c = [-2, -4, -5],\ l = -5)$	*UNDEF*	$[-6, \vert 1, 2, \vert 7, -3, \vert 4, -5]$
$M \not\models \neg\, F_0$, (vars M) = (vars F_0)	*SAT*	$[-6, \vert 1, 2, \vert 7, -3, \vert 4, -5]$

Problemi - nastavak

- Ista vrsta redundantnosti može da se javi i u nekom kontekstu (npr. sa literalom -6 umesto literala 6).

Rešenje: učenje iz ranijih konflikata (eng. learning).

Problemi - nastavak

- Ista vrsta redundantnosti može da se javi i u nekom kontekstu (npr. sa literalom -6 umesto literala 6).

Rešenje: učenje iz ranijih konflikata (eng. learning).

Analiza konflikata

Primer

Primenjeno pravilo	*satFlag*	*M*
	UNDEF	$[\,]$
Decide $(l = +6)$	*UNDEF*	$[\mid +6]$
UnitProp $(c = [+1, -6],\ l = +1)$	*UNDEF*	$[\mid +6, +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +6, +1, +2]$
Decide $(l = +7)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7]$
Decide $(l = +3)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3]$
UnitProp $(c = [-3, +4],\ l = +4)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4]$
UnitProp $(c = [-1, -3, +5],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$
Conflict $(M \models \neg [-2, -4, -5])$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$

Zašto je došlo do konflikta? Zbog literala $+2$, $+4$ i $+5$.

A zbog čega je prisutan literal $+5$? Zbog literala $+1$ i $+3$.

A zbog čega je prisutan literal $+4$? Zbog literala $+3$.

Dakle, uz literale $+1$ i $+2$, ne ide literal $+3$.

Analiza konflikata

Primer

Primenjeno pravilo	*satFlag*	*M*
	UNDEF	$[\,]$
Decide $(l = +6)$	*UNDEF*	$[\mid +6]$
UnitProp $(c = [+1, -6],\ l = +1)$	*UNDEF*	$[\mid +6, +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +6, +1, +2]$
Decide $(l = +7)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7]$
Decide $(l = +3)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3]$
UnitProp $(c = [-3, +4],\ l = +4)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4]$
UnitProp $(c = [-1, -3, +5],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$
Conflict $(M \models \neg [-2, -4, -5])$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$

Zašto je došlo do konflikta? Zbog literala +2, +4 i +5.

A zbog čega je prisutan literal +5? Zbog literala +1 i +3.

A zbog čega je prisutan literal +4? Zbog literala +3.

Dakle, uz literale +1 i +2, ne ide literal +3.

Analiza konflikata

Primer

Primenjeno pravilo	*satFlag*	*M*
	UNDEF	$[\,]$
Decide $(l = +6)$	*UNDEF*	$[\mid +6]$
UnitProp $(c = [+1, -6],\ l = +1)$	*UNDEF*	$[\mid +6, +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +6, +1, +2]$
Decide $(l = +7)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7]$
Decide $(l = +3)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3]$
UnitProp $(c = [-3, +4],\ l = +4)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4]$
UnitProp $(c = [-1, -3, +5],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$
Conflict $(M \models \neg [-2, -4, -5])$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$

$$\{+2, +4, +5\}$$

Zašto je došlo do konflikta? Zbog literala +2, +4 i +5.

A zbog čega je prisutan literal +5? Zbog literala +1 i +3.

A zbog čega je prisutan literal +4? Zbog literala +3.

Dakle, uz literale +1 i +2, ne ide literal +3.

Analiza konflikata

Primer

Primenjeno pravilo	*satFlag*	*M*
	UNDEF	$[\,]$
Decide $(l = +6)$	*UNDEF*	$[\mid +6]$
UnitProp $(c = [+1, -6],\ l = +1)$	*UNDEF*	$[\mid +6, +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +6, +1, +2]$
Decide $(l = +7)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7]$
Decide $(l = +3)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3]$
UnitProp $(c = [-3, +4],\ l = +4)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4]$
UnitProp $(c = [-1, -3, +5],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$
Conflict $(M \models \neg [-2, -4, -5])$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$

$$\{+2, +4, +5\}$$

Zašto je došlo do konflikta? Zbog literala +2, +4 i +5.

A zbog čega je prisutan literal +5? Zbog literala +1 i +3.

A zbog čega je prisutan literal +4? Zbog literala +3.

Dakle, uz literale +1 i +2, ne ide literal +3.

Analiza konflikata

Primer

Primenjeno pravilo	*satFlag*	*M*
	UNDEF	$[\,]$
Decide $(l = +6)$	*UNDEF*	$[\mid +6]$
UnitProp $(c = [+1, -6],\ l = +1)$	*UNDEF*	$[\mid +6, +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +6, +1, +2]$
Decide $(l = +7)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7]$
Decide $(l = +3)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3]$
UnitProp $(c = [-3, +4],\ l = +4)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4]$
UnitProp $(c = [-1, -3, +5],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$
Conflict $(M \models \neg [-2, -4, -5])$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$

$$\{+2, +4, +5\}$$

Zašto je došlo do konflikta? Zbog literala +2, +4 i +5.

A zbog čega je prisutan literal +5? Zbog literala +1 i +3.

A zbog čega je prisutan literal +4? Zbog literala +3.

Dakle, uz literale +1 i +2, ne ide literal +3.

Analiza konflikata

Primer

Primenjeno pravilo	*satFlag*	*M*
	UNDEF	$[\,]$
Decide $(l = +6)$	*UNDEF*	$[\mid +6]$
UnitProp $(c = [+1, -6],\ l = +1)$	*UNDEF*	$[\mid +6, +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +6, +1, +2]$
Decide $(l = +7)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7]$
Decide $(l = +3)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3]$
UnitProp $(c = [-3, +4],\ l = +4)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4]$
UnitProp $(c = [-1, -3, +5],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$
Conflict $(M \models \neg [-2, -4, -5])$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$

$$\{+1, +2, +3, +4\}$$

Zašto je došlo do konflikta? Zbog literala +2, +4 i +5.

A zbog čega je prisutan literal +5? Zbog literala +1 i +3.

A zbog čega je prisutan literal +4? Zbog literala +3.

Dakle, uz literale +1 i +2, ne ide literal +3.

Analiza konflikata

Primer

Primenjeno pravilo	*satFlag*	*M*
	UNDEF	$[\,]$
Decide $(l = +6)$	*UNDEF*	$[\mid +6]$
UnitProp $(c = [+1, -6],\ l = +1)$	*UNDEF*	$[\mid +6, +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +6, +1, +2]$
Decide $(l = +7)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7]$
Decide $(l = +3)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3]$
UnitProp $(c = [-3, +4],\ l = +4)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4]$
UnitProp $(c = [-1, -3, +5],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$
Conflict $(M \models \neg [-2, -4, -5])$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$

$$\{+1, +2, +3, +4\}$$

Zašto je došlo do konflikta? Zbog literala +2, +4 i +5.

A zbog čega je prisutan literal +5? Zbog literala +1 i +3.

A zbog čega je prisutan literal +4? Zbog literala +3.

Dakle, uz literale +1 i +2, ne ide literal +3.

Analiza konflikata

Primer

Primenjeno pravilo	*satFlag*	*M*
	UNDEF	$[\,]$
Decide $(l = +6)$	*UNDEF*	$[\mid +6]$
UnitProp $(c = [+1, -6],\ l = +1)$	*UNDEF*	$[\mid +6, +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +6, +1, +2]$
Decide $(l = +7)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7]$
Decide $(l = +3)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3]$
UnitProp $(c = [-3, +4],\ l = +4)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4]$
UnitProp $(c = [-1, -3, +5],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$
Conflict $(M \models \neg [-2, -4, -5])$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$

$$\{+1, +2, +3, +4\}$$

Zašto je došlo do konflikta? Zbog literala +2, +4 i +5.

A zbog čega je prisutan literal +5? Zbog literala +1 i +3.

A zbog čega je prisutan literal +4? Zbog literala +3.

Dakle, uz literale +1 i +2, ne ide literal +3.

Analiza konflikata

Primer

Primenjeno pravilo	*satFlag*	*M*
	UNDEF	$[\,]$
Decide $(l = +6)$	*UNDEF*	$[\mid +6]$
UnitProp $(c = [+1, -6],\ l = +1)$	*UNDEF*	$[\mid +6, +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +6, +1, +2]$
Decide $(l = +7)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7]$
Decide $(l = +3)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3]$
UnitProp $(c = [-3, +4],\ l = +4)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4]$
UnitProp $(c = [-1, -3, +5],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$
Conflict $(M \models \neg [-2, -4, -5])$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$

$$\{+1, +2, +3\}$$

Zašto je došlo do konflikta? Zbog literala +2, +4 i +5.

A zbog čega je prisutan literal +5? Zbog literala +1 i +3.

A zbog čega je prisutan literal +4? Zbog literala +3.

Dakle, uz literale +1 i +2, ne ide literal +3.

Analiza konflikata

Primer

Primenjeno pravilo	*satFlag*	*M*
	UNDEF	$[\,]$
Decide $(l = +6)$	*UNDEF*	$[\mid +6]$
UnitProp $(c = [+1, -6],\ l = +1)$	*UNDEF*	$[\mid +6, +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +6, +1, +2]$
Decide $(l = +7)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7]$
Decide $(l = +3)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3]$
UnitProp $(c = [-3, +4],\ l = +4)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4]$
UnitProp $(c = [-1, -3, +5],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$
Conflict $(M \models \neg [-2, -4, -5])$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$

$$\{+1, +2, +3\}$$

Zašto je došlo do konflikta? Zbog literala +2, +4 i +5.

A zbog čega je prisutan literal +5? Zbog literala +1 i +3.

A zbog čega je prisutan literal +4? Zbog literala +3.

Dakle, uz literale +1 i +2, ne ide literal +3.

Analiza konflikata

Primer

Primenjeno pravilo	*satFlag*	*M*
	UNDEF	$[\,]$
Decide $(l = +6)$	*UNDEF*	$[\mid +6]$
UnitProp $(c = [+1, -6],\ l = +1)$	*UNDEF*	$[\mid +6, +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +6, +1, +2]$
Decide $(l = +7)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7]$
Decide $(l = +3)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3]$
UnitProp $(c = [-3, +4],\ l = +4)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4]$
UnitProp $(c = [-1, -3, +5],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$
Conflict $(M \models \neg [-2, -4, -5])$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$
Backjump $(C = [-1, -2, -3])$	*UNDEF*	$[\mid +6, +1, +2, -3]$

$$\{+1, +2, +3\}$$

Zašto je došlo do konflikta? Zbog literala +2, +4 i +5.

A zbog čega je prisutan literal +5? Zbog literala +1 i +3.

A zbog čega je prisutan literal +4? Zbog literala +3.

Dakle, uz literale +1 i +2, ne ide literal +3.

Analiza konflikata kao rezolucija

Stablo rezolucije

$$\dfrac{\dfrac{[-2,-4,-5] \qquad [-1,-3,5]}{[-1,-2,-3,-4]} \qquad [-3,4]}{[-1,-2,-3]}$$

Dobijena klauza povratnog skoka je semantička posledica polaznog skupa klauza.

Analiza konflikata kao rezolucija

Stablo rezolucije

$$\dfrac{\dfrac{[-2,-4,-5] \qquad [-1,-3,5]}{[-1,-2,-3,-4]} \qquad [-3,4]}{[-1,-2,-3]}$$

Dobijena klauza povratnog skoka je semantička posledica polaznog skupa klauza.

Analiza konflikata kao rezolucija

Stablo rezolucije

$$\dfrac{\dfrac{[-2,-4,-5] \qquad [-1,-3,5]}{[-1,-2,-3,-4]} \qquad [-3,4]}{[-1,-2,-3]}$$

Dobijena klauza povratnog skoka je semantička posledica polaznog skupa klauza.

Analiza konflikata kao rezolucija

Stablo rezolucije

$$\dfrac{\dfrac{[-2,-4,-5] \qquad [-1,-3,5]}{[-1,-2,-3,-4]} \qquad [-3,4]}{[-1,-2,-3]}$$

Dobijena klauza povratnog skoka je semantička posledica polaznog skupa klauza.

Analiza konflikata kao rezolucija

Stablo rezolucije

$$\dfrac{\dfrac{[-2,-4,-5] \qquad [-1,-3,5]}{[-1,-2,-3,-4]} \qquad [-3,4]}{[-1,-2,-3]}$$

Dobijena klauza povratnog skoka je semantička posledica polaznog skupa klauza.

Sistem sa analizom konflikata

Decide:

$$\frac{l \in F \quad l, \bar{l} \notin M}{M := M \,|\, l}$$

UnitPropagate:

$$\frac{l \vee l_1 \vee \ldots \vee l_k \in F \qquad \bar{l}_1, \ldots, \bar{l}_k \in M \qquad l, \bar{l} \notin M}{M := M \; l}$$

Conflict:

$$\frac{l_1 \vee \ldots l_k \in F \qquad \bar{l}_1, \ldots, \bar{l}_k \in M}{C := l_1 \vee \ldots l_k}$$

Explain:

$$\frac{\bar{l} \in C \qquad l \vee l_1 \vee \ldots \vee l_k \in F \qquad \bar{l}_1, \ldots, \bar{l}_k \prec^M l}{C := (C \setminus \bar{l}) \vee l_1 \vee \ldots \vee l_k}$$

Backjump:

$$\frac{C = l \vee l_1 \vee \ldots \vee l_k \qquad C \in F \qquad \text{level } \bar{l} > m \geq \text{level } \bar{l}_i}{M := (\text{prefixToLevel } m \; M) \; l}$$

Učenje

Sistem sa učenjem

Stanje rešavača:

- ...
- F - formula koja se vremenom proširuje

...

Learn:

$$\frac{F \models c}{F := F \wedge c}$$

Obično se uče isključivo klauze povratnog skoka.

Primer

$F_0 = [[-1, +2], [-3, +4], [-1, -3, +5], [-2, -4, -5], [-2, +3, +5, -6], [-1, +3, -5, -6], [+1, -6], [+1, +7]\ ]$.

Primenjeno pravilo	*satFlag*	M/C
	UNDEF	$[\,]$
Decide $(l = +6)$	*UNDEF*	$[\vert +6]$
UnitProp $(c = [+1, -6],\ l = +1)$	*UNDEF*	$[\vert +6, +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\vert +6, +1, +2]$
Decide $(l = +7)$	*UNDEF*	$[\vert +6, +1, +2, \vert +7]$
Decide $(l = +3)$	*UNDEF*	$[\vert +6, +1, +2, \vert +7, \vert +3]$
UnitProp $(c = [-3, +4],\ l = +4)$	*UNDEF*	$[\vert +6, +1, +2, \vert +7, \vert +3, +4]$
UnitProp $(c = [-1, -3, +5],\ l = +5)$	*UNDEF*	$[\vert +6, +1, +2, \vert +7, \vert +3, +4, +5]$
Conflict $(M \vDash \neg [-2, -4, -5])$	*UNDEF*	$[-2, -4, -5]$
Explain $(l = +5,\ c = [-1, -3, +5])$	*UNDEF*	$[-1, -2, -3, -4]$
Explain $(l = 4,\ c = [-3, +4])$	*UNDEF*	$[-1, -2, -3]$
Learn *(C = [-1, -2, -3])*	*UNDEF*	$[-1, -2, -3]$
Backjump $(C = [-1, -2, -3])$	*UNDEF*	$[\vert +6, +1, +2, -3]$
UnitProp $(c = [-2, +3, +5, -6],\ l = +5)$	*UNDEF*	$[\vert +6, +1, +2, -3, +5]$
Conflict $(M \vDash \neg [-1, +3, -5, -6])$	*UNDEF*	$[-1, +3, -5, -6]$
Explain $(l = +5,\ c = [-2, +3, +5, -6])$	*UNDEF*	$[-1, -2, +3, -6]$
Explain $(l = -3,\ c = [-1, -2, -3])$	*UNDEF*	$[-1, -2, -6]$
Explain $(l = +2,\ c = [-1, +2])$	*UNDEF*	$[-1, -6]$
Explain $(l = +1,\ c = [+1, -6])$	*UNDEF*	$[-6]$
Learn *(C = [-6])*	*UNDEF*	$[-6]$
Backjump $(C = [-6])$	*UNDEF*	$[-6]$

Primer

$F_0 = [[-1, +2], [-3, +4], [-1, -3, +5], [-2, -4, -5], [-2, +3, +5, -6], [-1, +3, -5, -6], [+1, -6], [+1, +7]]$.

Primenjeno pravilo	*satFlag*	M/C
	UNDEF	$[]$
Decide $(l = +6)$	*UNDEF*	$[\|+6]$
UnitProp $(c = [+1, -6], l = +1)$	*UNDEF*	$[\|+6, +1]$
UnitProp $(c = [-1, +2], l = +2)$	*UNDEF*	$[\|+6, +1, +2]$
Decide $(l = +7)$	*UNDEF*	$[\|+6, +1, +2, \|+7]$
Decide $(l = +3)$	*UNDEF*	$[\|+6, +1, +2, \|+7, \|+3]$
UnitProp $(c = [-3, +4], l = +4)$	*UNDEF*	$[\|+6, +1, +2, \|+7, \|+3, +4]$
UnitProp $(c = [-1, -3, +5], l = +5)$	*UNDEF*	$[\|+6, +1, +2, \|+7, \|+3, +4, +5]$
Conflict $(M \models \neg[-2, -4, -5])$	*UNDEF*	$[-2, -4, -5]$
Explain $(l = +5, c = [-1, -3, +5])$	*UNDEF*	$[-1, -2, -3, -4]$
Explain $(l = 4, c = [-3, +4])$	*UNDEF*	$[-1, -2, -3]$
Learn *(C = [-1, -2, -3])*	*UNDEF*	$[-1, -2, -3]$
Backjump $(C = [-1, -2, -3])$	*UNDEF*	$[\|+6, +1, +2, -3]$
UnitProp $(c = [-2, +3, +5, -6], l = +5)$	*UNDEF*	$[\|+6, +1, +2, -3, +5]$
Conflict $(M \models \neg[-1, +3, -5, -6])$	*UNDEF*	$[-1, +3, -5, -6]$
Explain $(l = +5, c = [-2, +3, +5, -6])$	*UNDEF*	$[-1, -2, +3, -6]$
Explain $(l = -3, c = [-1, -2, -3])$	*UNDEF*	$[-1, -2, -6]$
Explain $(l = +2, c = [-1, +2])$	*UNDEF*	$[-1, -6]$
Explain $(l = +1, c = [+1, -6])$	*UNDEF*	$[-6]$
Learn *(C = [-6])*	*UNDEF*	$[-6]$
Backjump $(C = [-6])$	*UNDEF*	$[-6]$

Primer

$F_0 = [[-1, +2], [-3, +4], [-1, -3, +5], [-2, -4, -5], [-2, +3, +5, -6], [-1, +3, -5, -6], [+1, -6], [+1, +7]]$.

Primenjeno pravilo	*satFlag*	M/C
	UNDEF	$[]$
Decide $(l = +6)$	*UNDEF*	$[\|+6]$
UnitProp $(c = [+1, -6], l = +1)$	*UNDEF*	$[\|+6, +1]$
UnitProp $(c = [-1, +2], l = +2)$	*UNDEF*	$[\|+6, +1, +2]$
Decide $(l = +7)$	*UNDEF*	$[\|+6, +1, +2, \|+7]$
Decide $(l = +3)$	*UNDEF*	$[\|+6, +1, +2, \|+7, \|+3]$
UnitProp $(c = [-3, +4], l = +4)$	*UNDEF*	$[\|+6, +1, +2, \|+7, \|+3, +4]$
UnitProp $(c = [-1, -3, +5], l = +5)$	*UNDEF*	$[\|+6, +1, +2, \|+7, \|+3, +4, +5]$
Conflict $(M \models \neg[-2, -4, -5])$	*UNDEF*	$[-2, -4, -5]$
Explain $(l = +5, c = [-1, -3, +5])$	*UNDEF*	$[-1, -2, -3, -4]$
Explain $(l = 4, c = [-3, +4])$	*UNDEF*	$[-1, -2, -3]$
Learn *(C = [-1, -2, -3])*	*UNDEF*	$[-1, -2, -3]$
Backjump $(C = [-1, -2, -3])$	*UNDEF*	$[\|+6, +1, +2, -3]$
UnitProp $(c = [-2, +3, +5, -6], l = +5)$	*UNDEF*	$[\|+6, +1, +2, -3, +5]$
Conflict $(M \models \neg[-1, +3, -5, -6])$	*UNDEF*	$[-1, +3, -5, -6]$
Explain $(l = +5, c = [-2, +3, +5, -6])$	*UNDEF*	$[-1, -2, +3, -6]$
Explain $(l = -3, c = [-1, -2, -3])$	*UNDEF*	$[-1, -2, -6]$
Explain $(l = +2, c = [-1, +2])$	*UNDEF*	$[-1, -6]$
Explain $(l = +1, c = [+1, -6])$	*UNDEF*	$[-6]$
Learn *(C = [-6])*	*UNDEF*	$[-6]$
Backjump $(C = [-6])$	*UNDEF*	$[-6]$

Primer

$F_0 = [[-1, +2], [-3, +4], [-1, -3, +5], [-2, -4, -5], [-2, +3, +5, -6], [-1, +3, -5, -6], [+1, -6], [+1, +7]\ ]$.

Primenjeno pravilo	*satFlag*	M/C
	UNDEF	$[\,]$
Decide $(l = +6)$	*UNDEF*	$[\mid +6]$
UnitProp $(c = [+1, -6],\ l = +1)$	*UNDEF*	$[\mid +6, +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +6, +1, +2]$
Decide $(l = +7)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7]$
Decide $(l = +3)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3]$
UnitProp $(c = [-3, +4],\ l = +4)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4]$
UnitProp $(c = [-1, -3, +5],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$
Conflict $(M \models \neg [-2, -4, -5])$	*UNDEF*	$[-2, -4, -5]$
Explain $(l = +5,\ c = [-1, -3, +5])$	*UNDEF*	$[-1, -2, -3, -4]$
Explain $(l = 4,\ c = [-3, +4])$	*UNDEF*	$[-1, -2, -3]$
Learn *(C = [-1, -2, -3])*	*UNDEF*	$[-1, -2, -3]$
Backjump $(C = [-1, -2, -3])$	*UNDEF*	$[\mid +6, +1, +2, -3]$
UnitProp $(c = [-2, +3, +5, -6],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, -3, +5]$
Conflict $(M \models \neg [-1, +3, -5, -6])$	*UNDEF*	$[-1, +3, -5, -6]$
Explain $(l = +5,\ c = [-2, +3, +5, -6])$	*UNDEF*	$[-1, -2, +3, -6]$
Explain $(l = -3,\ c = [-1, -2, -3])$	*UNDEF*	$[-1, -2, -6]$
Explain $(l = +2,\ c = [-1, +2])$	*UNDEF*	$[-1, -6]$
Explain $(l = +1,\ c = [+1, -6])$	*UNDEF*	$[-6]$
Learn *(C = [-6])*	*UNDEF*	$[-6]$
Backjump $(C = [-6])$	*UNDEF*	$[-6]$

Primer

$F_0 = [[-1, +2], [-3, +4], [-1, -3, +5], [-2, -4, -5], [-2, +3, +5, -6], [-1, +3, -5, -6], [+1, -6], [+1, +7], [-1, -2, -3]\ ]$.

Primenjeno pravilo	*satFlag*	*M/C*
	UNDEF	$[\,]$
Decide $(l = +6)$	*UNDEF*	$[\mid +6]$
UnitProp $(c = [+1, -6],\ l = +1)$	*UNDEF*	$[\mid +6, +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +6, +1, +2]$
Decide $(l = +7)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7]$
Decide $(l = +3)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3]$
UnitProp $(c = [-3, +4],\ l = +4)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4]$
UnitProp $(c = [-1, -3, +5],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$
Conflict $(M \models \neg [-2, -4, -5])$	*UNDEF*	$[-2, -4, -5]$
Explain $(l = +5,\ c = [-1, -3, +5])$	*UNDEF*	$[-1, -2, -3, -4]$
Explain $(l = 4,\ c = [-3, +4])$	*UNDEF*	$[-1, -2, -3]$
Learn *(C = [-1, -2, -3])*	*UNDEF*	$[-1, -2, -3]$
Backjump $(C = [-1, -2, -3])$	*UNDEF*	$[\mid +6, +1, +2, -3]$
UnitProp $(c = [-2, +3, +5, -6],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, -3, +5]$
Conflict $(M \models \neg [-1, +3, -5, -6])$	*UNDEF*	$[-1, +3, -5, -6]$
Explain $(l = +5,\ c = [-2, +3, +5, -6])$	*UNDEF*	$[-1, -2, +3, -6]$
Explain $(l = -3,\ c = [-1, -2, -3])$	*UNDEF*	$[-1, -2, -6]$
Explain $(l = +2,\ c = [-1, +2])$	*UNDEF*	$[-1, -6]$
Explain $(l = +1,\ c = [+1, -6])$	*UNDEF*	$[-6]$
Learn *(C = [-6])*	*UNDEF*	$[-6]$
Backjump $(C = [-6])$	*UNDEF*	$[-6]$

Primer

$F_0 = [[-1, +2], [-3, +4], [-1, -3, +5], [-2, -4, -5], [-2, +3, +5, -6], [-1, +3, -5, -6], [+1, -6], [+1, +7], [-1, -2, -3]\ ]$.

Primenjeno pravilo	*satFlag*	*M/C*
	UNDEF	$[\,]$
Decide $(l = +6)$	*UNDEF*	$[\mid +6]$
UnitProp $(c = [+1, -6],\ l = +1)$	*UNDEF*	$[\mid +6, +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +6, +1, +2]$
Decide $(l = +7)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7]$
Decide $(l = +3)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3]$
UnitProp $(c = [-3, +4],\ l = +4)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4]$
UnitProp $(c = [-1, -3, +5],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$
Conflict $(M \models \neg [-2, -4, -5])$	*UNDEF*	$[-2, -4, -5]$
Explain $(l = +5,\ c = [-1, -3, +5])$	*UNDEF*	$[-1, -2, -3, -4]$
Explain $(l = 4,\ c = [-3, +4])$	*UNDEF*	$[-1, -2, -3]$
Learn *(C = [-1, -2, -3])*	*UNDEF*	$[-1, -2, -3]$
Backjump $(C = [-1, -2, -3])$	*UNDEF*	$[\mid +6, +1, +2, -3]$
UnitProp $(c = [-2, +3, +5, -6],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, -3, +5]$
Conflict $(M \models \neg [-1, +3, -5, -6])$	*UNDEF*	$[-1, +3, -5, -6]$
Explain $(l = +5,\ c = [-2, +3, +5, -6])$	*UNDEF*	$[-1, -2, +3, -6]$
Explain $(l = -3,\ c = [-1, -2, -3])$	*UNDEF*	$[-1, -2, -6]$
Explain $(l = +2,\ c = [-1, +2])$	*UNDEF*	$[-1, -6]$
Explain $(l = +1,\ c = [+1, -6])$	*UNDEF*	$[-6]$
Learn *(C = [-6])*	*UNDEF*	$[-6]$
Backjump $(C = [-6])$	*UNDEF*	$[-6]$

Primer

$F_0 = [[-1, +2], [-3, +4], [-1, -3, +5], [-2, -4, -5], [-2, +3, +5, -6], [-1, +3, -5, -6], [+1, -6], [+1, +7], [-1, -2, -3]\ ]$.

Primenjeno pravilo	*satFlag*	*M/C*
	UNDEF	$[\,]$
Decide $(l = +6)$	*UNDEF*	$[\mid +6]$
UnitProp $(c = [+1, -6],\ l = +1)$	*UNDEF*	$[\mid +6, +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +6, +1, +2]$
Decide $(l = +7)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7]$
Decide $(l = +3)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3]$
UnitProp $(c = [-3, +4],\ l = +4)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4]$
UnitProp $(c = [-1, -3, +5],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$
Conflict $(M \models \neg [-2, -4, -5])$	*UNDEF*	$[-2, -4, -5]$
Explain $(l = +5,\ c = [-1, -3, +5])$	*UNDEF*	$[-1, -2, -3, -4]$
Explain $(l = 4,\ c = [-3, +4])$	*UNDEF*	$[-1, -2, -3]$
Learn *(C = [-1, -2, -3])*	*UNDEF*	$[-1, -2, -3]$
Backjump $(C = [-1, -2, -3])$	*UNDEF*	$[\mid +6, +1, +2, -3]$
UnitProp $(c = [-2, +3, +5, -6],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, -3, +5]$
Conflict $(M \models \neg [-1, +3, -5, -6])$	*UNDEF*	$[-1, +3, -5, -6]$
Explain $(l = +5,\ c = [-2, +3, +5, -6])$	*UNDEF*	$[-1, -2, +3, -6]$
Explain $(l = -3,\ c = [-1, -2, -3])$	*UNDEF*	$[-1, -2, -6]$
Explain $(l = +2,\ c = [-1, +2])$	*UNDEF*	$[-1, -6]$
Explain $(l = +1,\ c = [+1, -6])$	*UNDEF*	$[-6]$
Learn *(C = [-6])*	*UNDEF*	$[-6]$
Backjump $(C = [-6])$	*UNDEF*	$[-6]$

Primer

$F_0 = [[-1, +2], [-3, +4], [-1, -3, +5], [-2, -4, -5], [-2, +3, +5, -6], [-1, +3, -5, -6], [+1, -6], [+1, +7], [-1, -2, -3]\ ]$.

Primenjeno pravilo	*satFlag*	*M/C*
	UNDEF	$[\,]$
Decide *($l = +6$)*	*UNDEF*	$[\vert +6]$
UnitProp *($c = [+1, -6]$, $l = +1$)*	*UNDEF*	$[\vert +6, +1]$
UnitProp *($c = [-1, +2]$, $l = +2$)*	*UNDEF*	$[\vert +6, +1, +2]$
Decide *($l = +7$)*	*UNDEF*	$[\vert +6, +1, +2, \vert +7]$
Decide *($l = +3$)*	*UNDEF*	$[\vert +6, +1, +2, \vert +7, \vert +3]$
UnitProp *($c = [-3, +4]$, $l = +4$)*	*UNDEF*	$[\vert +6, +1, +2, \vert +7, \vert +3, +4]$
UnitProp *($c = [-1, -3, +5]$, $l = +5$)*	*UNDEF*	$[\vert +6, +1, +2, \vert +7, \vert +3, +4, +5]$
Conflict *($M \models \neg [-2, -4, -5]$)*	*UNDEF*	$[-2, -4, -5]$
Explain *($l = +5$, $c = [-1, -3, +5]$)*	*UNDEF*	$[-1, -2, -3, -4]$
Explain *($l = 4$, $c = [-3, +4]$)*	*UNDEF*	$[-1, -2, -3]$
Learn *(C = [-1, -2, -3])*	*UNDEF*	$[-1, -2, -3]$
Backjump *($C = [-1, -2, -3]$)*	*UNDEF*	$[\vert +6, +1, +2, -3]$
UnitProp *($c = [-2, +3, +5, -6]$, $l = +5$)*	*UNDEF*	$[\vert +6, +1, +2, -3, +5]$
Conflict *($M \models \neg [-1, +3, -5, -6]$)*	*UNDEF*	$[-1, +3, -5, -6]$
Explain *($l = +5$, $c = [-2, +3, +5, -6]$)*	*UNDEF*	$[-1, -2, +3, -6]$
Explain *($l = -3$, $c = [-1, -2, -3]$)*	*UNDEF*	$[-1, -2, -6]$
Explain *($l = +2$, $c = [-1, +2]$)*	*UNDEF*	$[-1, -6]$
Explain *($l = +1$, $c = [+1, -6]$)*	*UNDEF*	$[-6]$
Learn *(C = [-6])*	*UNDEF*	$[-6]$
Backjump *($C = [-6]$)*	*UNDEF*	$[-6]$

Primer

$F_0 = [[-1, +2], [-3, +4], [-1, -3, +5], [-2, -4, -5], [-2, +3, +5, -6], [-1, +3, -5, -6], [+1, -6], [+1, +7], [-1, -2, -3]\ ]$.

Primenjeno pravilo	*satFlag*	*M/C*
	UNDEF	$[\,]$
Decide $(l = +6)$	*UNDEF*	$[\|+6]$
UnitProp $(c = [+1, -6],\ l = +1)$	*UNDEF*	$[\|+6, +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\|+6, +1, +2]$
Decide $(l = +7)$	*UNDEF*	$[\|+6, +1, +2, \|+7]$
Decide $(l = +3)$	*UNDEF*	$[\|+6, +1, +2, \|+7, \|+3]$
UnitProp $(c = [-3, +4],\ l = +4)$	*UNDEF*	$[\|+6, +1, +2, \|+7, \|+3, +4]$
UnitProp $(c = [-1, -3, +5],\ l = +5)$	*UNDEF*	$[\|+6, +1, +2, \|+7, \|+3, +4, +5]$
Conflict $(M \models \neg [-2, -4, -5])$	*UNDEF*	$[-2, -4, -5]$
Explain $(l = +5,\ c = [-1, -3, +5])$	*UNDEF*	$[-1, -2, -3, -4]$
Explain $(l = 4,\ c = [-3, +4])$	*UNDEF*	$[-1, -2, -3]$
Learn *(C = [-1, -2, -3])*	*UNDEF*	$[-1, -2, -3]$
Backjump $(C = [-1, -2, -3])$	*UNDEF*	$[\|+6, +1, +2, -3]$
UnitProp $(c = [-2, +3, +5, -6],\ l = +5)$	*UNDEF*	$[\|+6, +1, +2, -3, +5]$
Conflict $(M \models \neg [-1, +3, -5, -6])$	*UNDEF*	$[-1, +3, -5, -6]$
Explain $(l = +5,\ c = [-2, +3, +5, -6])$	*UNDEF*	$[-1, -2, +3, -6]$
Explain $(l = -3,\ c = [-1, -2, -3])$	*UNDEF*	$[-1, -2, -6]$
Explain $(l = +2,\ c = [-1, +2])$	*UNDEF*	$[-1, -6]$
Explain $(l = +1,\ c = [+1, -6])$	*UNDEF*	$[-6]$
Learn *(C = [-6])*	*UNDEF*	$[-6]$
Backjump $(C = [-6])$	*UNDEF*	$[-6]$

Primer

$F_0 = [[-1, +2], [-3, +4], [-1, -3, +5], [-2, -4, -5], [-2, +3, +5, -6], [-1, +3, -5, -6], [+1, -6], [+1, +7], [-1, -2, -3]\ ]$.

Primenjeno pravilo	*satFlag*	*M/C*
	UNDEF	$[\,]$
Decide $(l = +6)$	*UNDEF*	$[\|+6]$
UnitProp $(c = [+1, -6],\ l = +1)$	*UNDEF*	$[\|+6, +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\|+6, +1, +2]$
Decide $(l = +7)$	*UNDEF*	$[\|+6, +1, +2, \|+7]$
Decide $(l = +3)$	*UNDEF*	$[\|+6, +1, +2, \|+7, \|+3]$
UnitProp $(c = [-3, +4],\ l = +4)$	*UNDEF*	$[\|+6, +1, +2, \|+7, \|+3, +4]$
UnitProp $(c = [-1, -3, +5],\ l = +5)$	*UNDEF*	$[\|+6, +1, +2, \|+7, \|+3, +4, +5]$
Conflict $(M \models \neg [-2, -4, -5])$	*UNDEF*	$[-2, -4, -5]$
Explain $(l = +5,\ c = [-1, -3, +5])$	*UNDEF*	$[-1, -2, -3, -4]$
Explain $(l = 4,\ c = [-3, +4])$	*UNDEF*	$[-1, -2, -3]$
Learn *(C = [-1, -2, -3])*	*UNDEF*	$[-1, -2, -3]$
Backjump $(C = [-1, -2, -3])$	*UNDEF*	$[\|+6, +1, +2, -3]$
UnitProp $(c = [-2, +3, +5, -6],\ l = +5)$	*UNDEF*	$[\|+6, +1, +2, -3, +5]$
Conflict $(M \models \neg [-1, +3, -5, -6])$	*UNDEF*	$[-1, +3, -5, -6]$
Explain $(l = +5,\ c = [-2, +3, +5, -6])$	*UNDEF*	$[-1, -2, +3, -6]$
Explain $(l = -3,\ c = [-1, -2, -3])$	*UNDEF*	$[-1, -2, -6]$
Explain $(l = +2,\ c = [-1, +2])$	*UNDEF*	$[-1, -6]$
Explain $(l = +1,\ c = [+1, -6])$	*UNDEF*	$[-6]$
Learn *(C = [-6])*	*UNDEF*	$[-6]$
Backjump $(C = [-6])$	*UNDEF*	$[-6]$

Primer

$F_0 = [[-1, +2], [-3, +4], [-1, -3, +5], [-2, -4, -5], [-2, +3, +5, -6], [-1, +3, -5, -6], [+1, -6], [+1, +7], [-1, -2, -3]\ ]$.

Primenjeno pravilo	*satFlag*	M/C
	UNDEF	$[\,]$
Decide $(l = +6)$	*UNDEF*	$[\mid +6]$
UnitProp $(c = [+1, -6],\ l = +1)$	*UNDEF*	$[\mid +6, +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\mid +6, +1, +2]$
Decide $(l = +7)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7]$
Decide $(l = +3)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3]$
UnitProp $(c = [-3, +4],\ l = +4)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4]$
UnitProp $(c = [-1, -3, +5],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, \mid +7, \mid +3, +4, +5]$
Conflict $(M \models \neg [-2, -4, -5])$	*UNDEF*	$[-2, -4, -5]$
Explain $(l = +5,\ c = [-1, -3, +5])$	*UNDEF*	$[-1, -2, -3, -4]$
Explain $(l = 4,\ c = [-3, +4])$	*UNDEF*	$[-1, -2, -3]$
Learn *(C = [-1, -2, -3])*	*UNDEF*	$[-1, -2, -3]$
Backjump $(C = [-1, -2, -3])$	*UNDEF*	$[\mid +6, +1, +2, -3]$
UnitProp $(c = [-2, +3, +5, -6],\ l = +5)$	*UNDEF*	$[\mid +6, +1, +2, -3, +5]$
Conflict $(M \models \neg [-1, +3, -5, -6])$	*UNDEF*	$[-1, +3, -5, -6]$
Explain $(l = +5,\ c = [-2, +3, +5, -6])$	*UNDEF*	$[-1, -2, +3, -6]$
Explain $(l = -3,\ c = [-1, -2, -3])$	*UNDEF*	$[-1, -2, -6]$
Explain $(l = +2,\ c = [-1, +2])$	*UNDEF*	$[-1, -6]$
Explain $(l = +1,\ c = [+1, -6])$	*UNDEF*	$[-6]$
Learn *(C = [-6])*	*UNDEF*	$[-6]$
Backjump $(C = [-6])$	*UNDEF*	$[-6]$

Primer

$F_0 = [[-1, +2], [-3, +4], [-1, -3, +5], [-2, -4, -5], [-2, +3, +5, -6], [-1, +3, -5, -6], [+1, -6], [+1, +7], [-1, -2, -3]\ ]$.

Primenjeno pravilo	*satFlag*	M/C
	UNDEF	$[\,]$
Decide $(l = +6)$	*UNDEF*	$[\|+6]$
UnitProp $(c = [+1, -6],\ l = +1)$	*UNDEF*	$[\|+6, +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\|+6, +1, +2]$
Decide $(l = +7)$	*UNDEF*	$[\|+6, +1, +2, \|+7]$
Decide $(l = +3)$	*UNDEF*	$[\|+6, +1, +2, \|+7, \|+3]$
UnitProp $(c = [-3, +4],\ l = +4)$	*UNDEF*	$[\|+6, +1, +2, \|+7, \|+3, +4]$
UnitProp $(c = [-1, -3, +5],\ l = +5)$	*UNDEF*	$[\|+6, +1, +2, \|+7, \|+3, +4, +5]$
Conflict $(M \models \neg [-2, -4, -5])$	*UNDEF*	$[-2, -4, -5]$
Explain $(l = +5,\ c = [-1, -3, +5])$	*UNDEF*	$[-1, -2, -3, -4]$
Explain $(l = 4,\ c = [-3, +4])$	*UNDEF*	$[-1, -2, -3]$
Learn *(C = [-1, -2, -3])*	*UNDEF*	$[-1, -2, -3]$
Backjump $(C = [-1, -2, -3])$	*UNDEF*	$[\|+6, +1, +2, -3]$
UnitProp $(c = [-2, +3, +5, -6],\ l = +5)$	*UNDEF*	$[\|+6, +1, +2, -3, +5]$
Conflict $(M \models \neg [-1, +3, -5, -6])$	*UNDEF*	$[-1, +3, -5, -6]$
Explain $(l = +5,\ c = [-2, +3, +5, -6])$	*UNDEF*	$[-1, -2, +3, -6]$
Explain $(l = -3,\ c = [-1, -2, -3])$	*UNDEF*	$[-1, -2, -6]$
Explain $(l = +2,\ c = [-1, +2])$	*UNDEF*	$[-1, -6]$
Explain $(l = +1,\ c = [+1, -6])$	*UNDEF*	$[-6]$
Learn $(C = [-6])$	*UNDEF*	$[-6]$
Backjump $(C = [-6])$	*UNDEF*	$[-6]$

Primer

$F_0 = [[-1, +2], [-3, +4], [-1, -3, +5], [-2, -4, -5], [-2, +3, +5, -6], [-1, +3, -5, -6], [+1, -6], [+1, +7], [-1, -2, -3]$, $[-6]$].

Primenjeno pravilo	*satFlag*	*M/C*
	UNDEF	[ ]
Decide *($l = +6$)*	*UNDEF*	$[\vert +6]$
UnitProp *($c = [+1, -6]$, $l = +1$)*	*UNDEF*	$[\vert +6, +1]$
UnitProp *($c = [-1, +2]$, $l = +2$)*	*UNDEF*	$[\vert +6, +1, +2]$
Decide *($l = +7$)*	*UNDEF*	$[\vert +6, +1, +2, \vert +7]$
Decide *($l = +3$)*	*UNDEF*	$[\vert +6, +1, +2, \vert +7, \vert +3]$
UnitProp *($c = [-3, +4]$, $l = +4$)*	*UNDEF*	$[\vert +6, +1, +2, \vert +7, \vert +3, +4]$
UnitProp *($c = [-1, -3, +5]$, $l = +5$)*	*UNDEF*	$[\vert +6, +1, +2, \vert +7, \vert +3, +4, +5]$
Conflict *($M \models \neg [-2, -4, -5]$)*	*UNDEF*	$[-2, -4, -5]$
Explain *($l = +5$, $c = [-1, -3, +5]$)*	*UNDEF*	$[-1, -2, -3, -4]$
Explain *($l = 4$, $c = [-3, +4]$)*	*UNDEF*	$[-1, -2, -3]$
Learn *(C = [-1, -2, -3])*	*UNDEF*	$[-1, -2, -3]$
Backjump *($C = [-1, -2, -3]$)*	*UNDEF*	$[\vert +6, +1, +2, -3]$
UnitProp *($c = [-2, +3, +5, -6]$, $l = +5$)*	*UNDEF*	$[\vert +6, +1, +2, -3, +5]$
Conflict *($M \models \neg [-1, +3, -5, -6]$)*	*UNDEF*	$[-1, +3, -5, -6]$
Explain *($l = +5$, $c = [-2, +3, +5, -6]$)*	*UNDEF*	$[-1, -2, +3, -6]$
Explain *($l = -3$, $c = [-1, -2, -3]$)*	*UNDEF*	$[-1, -2, -6]$
Explain *($l = +2$, $c = [-1, +2]$)*	*UNDEF*	$[-1, -6]$
Explain *($l = +1$, $c = [+1, -6]$)*	*UNDEF*	$[-6]$
Learn *(C = [-6])*	*UNDEF*	$[-6]$
Backjump *($C = [-6]$)*	*UNDEF*	$[-6]$

Primer

$F_0 = [[-1, +2], [-3, +4], [-1, -3, +5], [-2, -4, -5], [-2, +3, +5, -6], [-1, +3, -5, -6], [+1, -6], [+1, +7], [-1, -2, -3]\ , [-6]\ ]$.

Primenjeno pravilo	*satFlag*	M/C
	UNDEF	$[\,]$
Decide $(l = +6)$	*UNDEF*	$[\|+6]$
UnitProp $(c = [+1, -6],\ l = +1)$	*UNDEF*	$[\|+6, +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[\|+6, +1, +2]$
Decide $(l = +7)$	*UNDEF*	$[\|+6, +1, +2, \|+7]$
Decide $(l = +3)$	*UNDEF*	$[\|+6, +1, +2, \|+7, \|+3]$
UnitProp $(c = [-3, +4],\ l = +4)$	*UNDEF*	$[\|+6, +1, +2, \|+7, \|+3, +4]$
UnitProp $(c = [-1, -3, +5],\ l = +5)$	*UNDEF*	$[\|+6, +1, +2, \|+7, \|+3, +4, +5]$
Conflict $(M \models \neg [-2, -4, -5])$	*UNDEF*	$[-2, -4, -5]$
Explain $(l = +5,\ c = [-1, -3, +5])$	*UNDEF*	$[-1, -2, -3, -4]$
Explain $(l = 4,\ c = [-3, +4])$	*UNDEF*	$[-1, -2, -3]$
Learn *(C = [-1, -2, -3])*	*UNDEF*	$[-1, -2, -3]$
Backjump $(C = [-1, -2, -3])$	*UNDEF*	$[\|+6, +1, +2, -3]$
UnitProp $(c = [-2, +3, +5, -6],\ l = +5)$	*UNDEF*	$[\|+6, +1, +2, -3, +5]$
Conflict $(M \models \neg [-1, +3, -5, -6])$	*UNDEF*	$[-1, +3, -5, -6]$
Explain $(l = +5,\ c = [-2, +3, +5, -6])$	*UNDEF*	$[-1, -2, +3, -6]$
Explain $(l = -3,\ c = [-1, -2, -3])$	*UNDEF*	$[-1, -2, -6]$
Explain $(l = +2,\ c = [-1, +2])$	*UNDEF*	$[-1, -6]$
Explain $(l = +1,\ c = [+1, -6])$	*UNDEF*	$[-6]$
Learn *(C = [-6])*	*UNDEF*	$[-6]$
Backjump $(C = [-6])$	*UNDEF*	$[-6]$

Primer

$F_0 = [[-1, +2], [-3, +4], [-1, -3, +5], [-2, -4, -5], [-2, +3, +5, -6], [-1, +3, -5, -6], [+1, -6], [+1, +7]\ , [-1, -2, -3]\ , [-6]\ ]$.

Primenjeno pravilo	*satFlag*	M/C
Backjump $(C = [-6])$	*UNDEF*	$[-6]$
Decide $(l = +1)$	*UNDEF*	$[-6, \vert +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[-6, \vert +1, +2]$
UnitProp $(c = [-1, -2, -3],\ l = -3)$	*UNDEF*	$[-6, \vert +1, +2, -3]$
Decide $(l = +4)$	*UNDEF*	$[-6, \vert +1, +2, -3, \vert 4]$
UnitProp $(c = [-2, -4, -5],\ l = -5)$	*UNDEF*	$[-6, \vert +1, +2, -3, \vert 4, -5]$
Decide $(l = +7)$	*UNDEF*	$[-6, \vert +1, +2, -3, \vert 4, -5, \vert +7]$
$M \not\models \neg F$, $(\text{vars } M) = (\text{vars } F)$	*SAT*	$[-6, \vert +1, +2, -3, \vert 4, -5, \vert +7]$

Primer

$F_0 = [[-1, +2], [-3, +4], [-1, -3, +5], [-2, -4, -5], [-2, +3, +5, -6], [-1, +3, -5, -6], [+1, -6], [+1, +7]\ , [-1, -2, -3]\ , [-6]\ ]$.

Primenjeno pravilo	*satFlag*	M/C
Backjump $(C = [-6])$	*UNDEF*	$[-6]$
Decide $(l = +1)$	*UNDEF*	$[-6, \mid +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[-6, \mid +1, +2]$
UnitProp $(c = [-1, -2, -3],\ l = -3)$	*UNDEF*	$[-6, \mid +1, +2, -3]$
Decide $(l = +4)$	*UNDEF*	$[-6, \mid +1, +2, -3, \mid 4]$
UnitProp $(c = [-2, -4, -5],\ l = -5)$	*UNDEF*	$[-6, \mid +1, +2, -3, \mid 4, -5]$
Decide $(l = +7)$	*UNDEF*	$[-6, \mid +1, +2, -3, \mid 4, -5, \mid +7]$
$M \not\models \neg F$, $(\text{vars } M) = (\text{vars } F)$	*SAT*	$[-6, \mid +1, +2, -3, \mid 4, -5, \mid +7]$

Primer

$F_0 = [[-1, +2], [-3, +4], [-1, -3, +5], [-2, -4, -5], [-2, +3, +5, -6], [-1, +3, -5, -6], [+1, -6], [+1, +7]\ , [-1, -2, -3]\ , [-6]\ ]$.

Primenjeno pravilo	*satFlag*	M/C
Backjump $(C = [-6])$	*UNDEF*	$[-6]$
Decide $(l = +1)$	*UNDEF*	$[-6, \vert +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[-6, \vert +1, +2]$
UnitProp $(c = [-1, -2, -3],\ l = -3)$	*UNDEF*	$[-6, \vert +1, +2, -3]$
Decide $(l = +4)$	*UNDEF*	$[-6, \vert +1, +2, -3, \vert 4]$
UnitProp $(c = [-2, -4, -5],\ l = -5)$	*UNDEF*	$[-6, \vert +1, +2, -3, \vert 4, -5]$
Decide $(l = +7)$	*UNDEF*	$[-6, \vert +1, +2, -3, \vert 4, -5, \vert +7]$
$M \not\models \neg F$, $(\text{vars } M) = (\text{vars } F)$	*SAT*	$[-6, \vert +1, +2, -3, \vert 4, -5, \vert +7]$

Primer

$F_0 = [[-1, +2], [-3, +4], [-1, -3, +5], [-2, -4, -5], [-2, +3, +5, -6], [-1, +3, -5, -6], [+1, -6], [+1, +7]\ , [-1, -2, -3]\ , [-6]\ ]$.

Primenjeno pravilo	*satFlag*	M/C
Backjump $(C = [-6])$	*UNDEF*	$[-6]$
Decide $(l = +1)$	*UNDEF*	$[-6, \mid +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[-6, \mid +1, +2]$
UnitProp $(c = [-1, -2, -3],\ l = -3)$	*UNDEF*	$[-6, \mid +1, +2, -3]$
Decide $(l = +4)$	*UNDEF*	$[-6, \mid +1, +2, -3, \mid 4]$
UnitProp $(c = [-2, -4, -5],\ l = -5)$	*UNDEF*	$[-6, \mid +1, +2, -3, \mid 4, -5]$
Decide $(l = +7)$	*UNDEF*	$[-6, \mid +1, +2, -3, \mid 4, -5, \mid +7]$
$M \not\models \neg F$, $(\text{vars } M) = (\text{vars } F)$	*SAT*	$[-6, \mid +1, +2, -3, \mid 4, -5, \mid +7]$

Primer

$F_0 = [[-1, +2], [-3, +4], [-1, -3, +5], [-2, -4, -5], [-2, +3, +5, -6], [-1, +3, -5, -6], [+1, -6], [+1, +7]\ , [-1, -2, -3]\ , [-6]\ ]$.

Primenjeno pravilo	*satFlag*	M/C
Backjump $(C = [-6])$	*UNDEF*	$[-6]$
Decide $(l = +1)$	*UNDEF*	$[-6, \vert +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[-6, \vert +1, +2]$
UnitProp $(c = [-1, -2, -3],\ l = -3)$	*UNDEF*	$[-6, \vert +1, +2, -3]$
Decide $(l = +4)$	*UNDEF*	$[-6, \vert +1, +2, -3, \vert 4]$
UnitProp $(c = [-2, -4, -5],\ l = -5)$	*UNDEF*	$[-6, \vert +1, +2, -3, \vert 4, -5]$
Decide $(l = +7)$	*UNDEF*	$[-6, \vert +1, +2, -3, \vert 4, -5, \vert +7]$
$M \not\models \neg F$, $(\text{vars } M) = (\text{vars } F)$	*SAT*	$[-6, \vert +1, +2, -3, \vert 4, -5, \vert +7]$

Primer

$F_0 = [[-1, +2], [-3, +4], [-1, -3, +5], [-2, -4, -5], [-2, +3, +5, -6], [-1, +3, -5, -6], [+1, -6], [+1, +7]\ , [-1, -2, -3]\ , [-6]\ ]$.

Primenjeno pravilo	*satFlag*	M/C
Backjump $(C = [-6])$	*UNDEF*	$[-6]$
Decide $(l = +1)$	*UNDEF*	$[-6, \vert +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[-6, \vert +1, +2]$
UnitProp $(c = [-1, -2, -3],\ l = -3)$	*UNDEF*	$[-6, \vert +1, +2, -3]$
Decide $(l = +4)$	*UNDEF*	$[-6, \vert +1, +2, -3, \vert 4]$
UnitProp $(c = [-2, -4, -5],\ l = -5)$	*UNDEF*	$[-6, \vert +1, +2, -3, \vert 4, -5]$
Decide $(l = +7)$	*UNDEF*	$[-6, \vert +1, +2, -3, \vert 4, -5, \vert +7]$
$M \not\models \neg F$, $(\text{vars } M) = (\text{vars } F)$	*SAT*	$[-6, \vert +1, +2, -3, \vert 4, -5, \vert +7]$

Primer

$F_0 = [[-1, +2], [-3, +4], [-1, -3, +5], [-2, -4, -5], [-2, +3, +5, -6], [-1, +3, -5, -6], [+1, -6], [+1, +7]\ , [-1, -2, -3]\ , [-6]\ ]$.

Primenjeno pravilo	*satFlag*	M/C
Backjump $(C = [-6])$	*UNDEF*	$[-6]$
Decide $(l = +1)$	*UNDEF*	$[-6, \mid +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[-6, \mid +1, +2]$
UnitProp $(c = [-1, -2, -3],\ l = -3)$	*UNDEF*	$[-6, \mid +1, +2, -3]$
Decide $(l = +4)$	*UNDEF*	$[-6, \mid +1, +2, -3, \mid 4]$
UnitProp $(c = [-2, -4, -5],\ l = -5)$	*UNDEF*	$[-6, \mid +1, +2, -3, \mid 4, -5]$
Decide $(l = +7)$	*UNDEF*	$[-6, \mid +1, +2, -3, \mid 4, -5, \mid +7]$
$M \not\models \neg F$, $(\text{vars } M) = (\text{vars } F)$	*SAT*	$[-6, \mid +1, +2, -3, \mid 4, -5, \mid +7]$

Primer

$F_0 = [[-1, +2], [-3, +4], [-1, -3, +5], [-2, -4, -5], [-2, +3, +5, -6], [-1, +3, -5, -6], [+1, -6], [+1, +7]\ , [-1, -2, -3]\ , [-6]\ ]$.

Primenjeno pravilo	*satFlag*	M/C
Backjump $(C = [-6])$	*UNDEF*	$[-6]$
Decide $(l = +1)$	*UNDEF*	$[-6, \vert +1]$
UnitProp $(c = [-1, +2],\ l = +2)$	*UNDEF*	$[-6, \vert +1, +2]$
UnitProp $(c = [-1, -2, -3],\ l = -3)$	*UNDEF*	$[-6, \vert +1, +2, -3]$
Decide $(l = +4)$	*UNDEF*	$[-6, \vert +1, +2, -3, \vert 4]$
UnitProp $(c = [-2, -4, -5],\ l = -5)$	*UNDEF*	$[-6, \vert +1, +2, -3, \vert 4, -5]$
Decide $(l = +7)$	*UNDEF*	$[-6, \vert +1, +2, -3, \vert 4, -5, \vert +7]$
$M \not\models_{\neg} F$, (vars M) = (vars F)	*SAT*	$[-6, \vert +1, +2, -3, \vert 4, -5, \vert +7]$

Zaboravljanje

- Ukoliko broj naučenih klauza postane preveliki, pronalaženje jediničnih i konfliktnih klauza se usporava.
- Neke naučene klauze se posle određenog vremena uklanjaju.
- Zaboravljanje je kontrolisano heuristikama.

Otpočinjanje iznova

- U nekim trenucima je korisno pretragu prekinuti i započeti iznova.
- Postoji nada da će nas klauze koje su u međuvremenu naučene odvesti u neku drugu (lakšu) granu stabla pretrage.
- Otpočinjanje iznova pokazuje dobra svojstva kada se koristi uz određenih procenat slučajnih odluka u toku pretrage.
- Otpočinjanje iznova je kontrolisano heuristikama.

Pregled

Pristupi rešavanja SAT problema

- Naivni metodi (metod istinitosnih tablica).
- Problem se tehnički jednostavnije rešava ako je formula u nekom od specijalnih oblika (DNF, KNF, BDD).
- Za DNF se zadovoljivost trivijalno ispituje, ali je postupak prevođenja u DNF netrivijalan. Slično je i sa BDD.
- Postoji efikasan postupak prevođenja formule u KNF (doduše koji ne čuva ekvivalentnost, ali čuva zadovoljivost).
- Zadovoljivost formula u KNF se dalje ispituje nekim od tzv. klauzalnih algoritama.
  - DP procedura (iskazna rezolucija)
  - DPLL procedura
  - CDCL SAT rešavači (unapređena DPLL procedura)
  - Stohastički SAT rešavači (zasnovani na randomizaciji — ne garantuju uvek definitivan odgovor).

Metod rezolucije

- Metod otkriven još 1960.-tih (Davis, Putnam, Robinson, . . .).
- Metod dokazivanja nezadovoljivosti (pobijanja) skupa formula:
  - iskazne logike,
  - logike prvog reda.

Reprezentacija klauza?

- Važno pitanje pre primene metoda rezolucije je reprezentacija klauza.
  - Ukoliko su klauze skupovi literala (tj. tako da ponavljanje literala nije dopušteno), u iskaznom slučaju, dovoljno je samo pravilo binarne rezolucije.
  - Ukoliko su klauze liste literala (tj. ako je ponavljanje literala dopušteno) uz pravilo rezolucije potrebno je pravilo pune rezolucije, ili kombinacija pravila binarne rezolucije i pravila faktorisanja. U slučaju logike prvog reda situacija je uvek ovakva.

U nastavku će biti uvek podrazumevana reprezentacija klauza u obliku skupa literala.

Pravilo binarne rezolucije

$$\frac{\mathcal{C}' \cup \{l\} \quad \mathcal{C}'' \cup \{\bar{l}\}}{\mathcal{C}' \cup \mathcal{C}''}$$

Suštinski, opravdanje ovog pravila je u tranzitivnosti implikacije.

$$\frac{\overline{\mathcal{C}'} \Rightarrow l \quad l \Rightarrow \mathcal{C}''}{\overline{\mathcal{C}'} \Rightarrow \mathcal{C}''}$$

Dobijena klauza $\mathcal{C}' \cup \mathcal{C}''$ se naziva rezolventa, dok se polazne klauze $\mathcal{C}' \cup \{l\}$ i $\mathcal{C}'' \cup \{\bar{l}\}$ nazivaju njenim roditeljima.
Rezolucija klauza $\mathcal{C}_1$ i $\mathcal{C}_2$ preko literala l će biti označavana i sa $\mathcal{C}_1 \oplus_l \mathcal{C}_2$.

Metod rezolucije

- Metod rezolucije se zasniva na iterativnoj primeni pravila (binarne) rezolucije na skup klauza, dodavanjem rezolventi u polazni skup, bez brisanja roditelja, sve dok postoje nove rezolvente ili u polazni skup nije dodana prazna klauza.
- Ako je u skup klauza dodana prazna klauza metod rezolucije prijavljuje nezadovoljivost.
- Ako nije moguće generisati nove rezolvente, metod rezolucije prijavljuje zadovoljivost.

Metod rezolucije — primeri

Primer

$$\{\{\neg p, \neg q, r\}, \{\neg p, q\}, \{p\}, \{\neg r\}\}$$

$$\begin{array}{lcll} \mathcal{C}_1 & : & \{\neg p, \neg q, r\} & \\ \mathcal{C}_2 & : & \{\neg p, q\} & \\ \mathcal{C}_3 & : & \{p\} & \\ \mathcal{C}_4 & : & \{\neg r\} & \\ \hline \mathcal{C}_5 & : & \{\neg p, r\} & \mathcal{C}_1 \oplus_q \mathcal{C}_2 \\ \mathcal{C}_6 & : & \{\neg p\} & \mathcal{C}_4 \oplus_r \mathcal{C}_5 \\ \mathcal{C}_7 & : & \{\} & \mathcal{C}_3 \oplus_p \mathcal{C}_6 \end{array}$$

Zaustavljanje rezolucije

Teorema

Metod rezolucije se zaustavlja.

Dokaz

Pošto je početni skup atoma konačan, postoji konačno mnogo različitih klauza sagrađenih od njegovih elemenata (za n atoma, postoji 2^{2n} različitih klauza). U svakom koraku u skup klauza se dodaje nova klauza koja sadrži samo literale iz polaznog skupa, te može da bude samo konačno mnogo koraka.

Saglasnost rezolucije

Lema (Saglasnost pravila rezolucije)

Ako je $\mathcal{C}_1 \in \Gamma$ *i* $\mathcal{C}_2 \in \Gamma$, *skupovi klauza* Γ *i* $\Gamma \cup \{\mathcal{C}_1 \oplus_l \mathcal{C}_2\}$ *su logički ekvivalentni.*

Dokaz

Ako je v *model za* $\Gamma \cup \{\mathcal{C}_1 \oplus_l \mathcal{C}_2\}$, *tada je, trivijalno,* v *model i za* Γ. *Obratno, neka je* v *model za* Γ. *Razmotrimo sledeće slučajeve:*

- $v \not\models l$. *Tada u* $\mathcal{C}_1$ *postoji literal* l' *različit od* l *takav da* $v \models l'$. *No,* $l' \in \mathcal{C}_1 \oplus_l \mathcal{C}_2$ *te* $v \models \mathcal{C}_1 \oplus_l \mathcal{C}_2$.
- $v \models l$. *Tada* $v \not\models \bar{l}$ *pa u* $\mathcal{C}_2$ *postoji literal* l' *različit od* $\bar{l}$ *takav da* $v \models l'$. *No,* $l' \in \mathcal{C}_1 \oplus_l \mathcal{C}_2$ *te* $v \models \mathcal{C}_1 \oplus_l \mathcal{C}_2$.

Saglasnost metoda rezolucije

Teorema

Ako metod rezolucije prijavi nezadovoljivost (tj. ako izvede praznu klauzu), onda je polazni skup klauza nezadovoljiv.

Dokaz

U svakom koraku metoda rezolucije u skup klauza se dodaje rezolventa neka dva njegova člana i novodobijeni skup je ekvivalentan prethodnom, pa samim tim, na osnovu tranzitivnosti logičke ekvivalentnosti, i polaznom. Kako poslednji skup sadrži praznu klauzu, on nije zadovoljiv pa ne može biti ni njemu ekvivalentan polazni skup klauza.

Potpunost rezolucije

Teorema

Ako je polazni skup klauza nezadovoljiv, onda metod rezolucije prijavljuje nezadovoljivost (tj. izvodi praznu klauzu).

Davis-Putnam procedura

- DP procedura (Davis-Putnam, 1961).
- Ne mešati sa DPLL procedurom (Davis-Putnam-Logemann-Loveland, 1962).
- Suštinski zasnovana na rezoluciji.
- Deo veće procedure za logiku prvog reda.

- 3 koraka
  - propagacija jediničnih klauza (unit propagation)
  - eliminacija ,,čistih" literala (pure literal)
  - eliminacija promenljive (variable elimination)
- Suštinski korak je eliminacija i on je zasnovan na rezoluciji.
- Rezolucija se primenjuje sistematski i iscrpno promenljivu po promenljivu, pri čemu se originalne klauze u svakom koraku uklanjaju.

Korak eliminacije promenljive

Promenljiva p se eliminiše tako što se sve klauze polaznog skupa koje sadrže p i sve klauze koje sadrže $\neg p$ rezolivraju i zamene dobijenim rezolventama.
Dakle, umesto skupa $\mathcal{F}$ posmatra se skup

$$\{\mathcal{C}' \oplus_p \mathcal{C}'' \,|\, \mathcal{C}' \in \mathcal{F}, p \in \mathcal{C}', \mathcal{C}'' \in \mathcal{F}, \neg p \in \mathcal{C}''\} \cup$$
$$\{\mathcal{C} \,|\, \mathcal{C} \in \mathcal{F}, p \notin \mathcal{C}, \neg p \notin \mathcal{C}\}$$

Tautologične klauze i eliminacija promenljive

- Detalj: ukoliko klauza sadrži i p i $\neg p$, nakon njenog rezolviranja, ne uklanja se promenljiva p.
- Tautologične klauze moguće je ukloniti iz skupa pre početka procedure i eliminisati ih prilikom svakog koraka eliminacije.

Implementacija koraka eliminacije

```
definition
resolve_on :: "Literal => Literal set set => Literal set set" where
"resolve_on p clauses ==
   (pos, notpos) = partition (% c. p : c) clauses;
   (neg, other) =  partition (% c. (negate p) : c) notpos;
   pos' = map (% c. remove p c) pos;
   neg' = map (% c. remove (negate p) c) neg;
   res = map (% (a, b). sup a b) (allpairs pos' neg') in
   sup (filter (% c. ~trivial c) res) other"

definition
eliminate_var :: "Literal set set => Literal set set" where
"eliminate_var clauses ==
   let pos = filter positive (Supremum clauses);
       p = some_elem pos in
       resolve_on p clauses"
```

Implementacija glavne DP procedure

```
function dp :: "Literal set set => bool" where
"dp clauses =
  (if clauses = {} then
      True
   else (if {} : clauses then
      False
   else (case pure_literal clauses of
      Some clauses' => dp clauses'
      | None => (case unit_propagate clauses of
        Some clauses' => dp clauses'
      | None => dp (eliminate_var clauses)))))"
```

Korektnost koraka eliminacije

Teorema

Neka je

$$\mathcal{F} = \{\mathcal{C}'_i \cup \{p\} \mid 1 \leq i \leq m\} \cup \{\mathcal{C}''_j \cup \{\neg p\} \mid 1 \leq j \leq n\} \cup \mathcal{F}_0,$$

pri čemu $\mathcal{C}'_i$ ne sadrže $\neg p$, $\mathcal{C}''_j$ ne sadrže p, dok klauze iz $\mathcal{F}_0$ ako sadrže p ili $\neg p$, onda sadrže i drugi (tj. tautologične su).
Neka je

$$\mathcal{F}' = \{\mathcal{C}'_i \cup \mathcal{C}''_j \mid 1 \leq i \leq m, 1 \leq j \leq n\} \cup \mathcal{F}_0.$$

Tada je

$$\mathcal{F} \equiv_s \mathcal{F}'.$$

Korektnost koraka eliminacije

Dokaz

Ako je $\mathcal{F}$ zadovoljiv, na osnovu leme o pravilu rezolucije, zadovoljiv je i $\mathcal{F}'$.
Neka je v valuacija koja zadovoljava $\mathcal{F}'$. Valuacija v istovremeno zadovoljava sve $\mathcal{C}_i'$ ili istovremeno zadovoljava sve $\mathcal{C}_j''$. Zaista, pošto v zadovoljava sve $\mathcal{C}_i' \cup \mathcal{C}_j''$, ako v ne zadovoljava neko $\mathcal{C}_i'$, onda zadovoljava sve $\mathcal{C}_j''$, i obratno.
Ako v zadovoljava sve $\mathcal{C}_i'$, posmatrajmo valuaciju v' dobijenu od v' postavljanjem p na netačno. Sve klauze $\mathcal{C}_i' \cup \{p\}$ su zadovoljene u v' jer su i sve $\mathcal{C}_i'$, sve klauze $\mathcal{C}_j'' \cup \{\neg p\}$ jer je p netačno, dok su klauze iz $\mathcal{F}_0$ zadovoljene jer ili ne sadrže p ili su tautologične.
Ako v zadovoljava sve $\mathcal{C}_j''$, dokaz je analogan.

Pregled

Metod tabloa

- Metod analitičkih tabloa
- Bet (Everth Beth), Hintika (Hintikka) 1955., Smaljan (Raymond Smullyan) 1971.
- Koristi se za pokazivanje nezadovoljivosti formula (može se koristiti za proveru tautologičnosti)
- Metod tabloa donekle odgovara (lenjom) prevođenju formule u DNF.
- Tabloi se obično prikazuju u obliku stabla.

Iskazni tablo – pravila

$$\frac{A \wedge B}{\begin{array}{c} A \\ B \end{array}}$$

$$\frac{\neg(A \vee B)}{\begin{array}{c} \neg A \\ \neg B \end{array}}$$

$$\frac{\neg(A \Rightarrow B)}{\begin{array}{c} A \\ \neg B \end{array}}$$

$$\frac{\neg(A \wedge B)}{\neg A \mid \neg B}$$

$$\frac{A \vee B}{A \mid B}$$

$$\frac{A \Rightarrow B}{\neg A \mid B}$$

Slična pravila se mogu definisati i za druge logičke veznike.

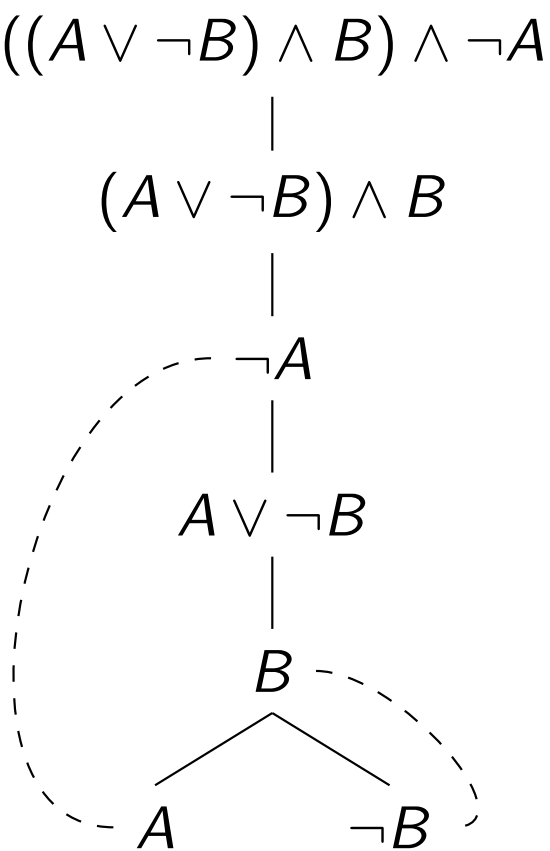
1. Konstrukcija tabloa kreće od stabla koje u svom jedinom svom čvoru (korenu) sadrži formulu čija se zadovoljivost ispituje.
2. U svakom koraku, moguće je stablo proširiti naniže primenom nekog od datih pravila na neku formulu koja se nalaze u putanji od korena do lista koji se proširuje, a na koju to pravilo ranije nije bilo primenjeno.
3. Tablo koji se ne može proširiti, naziva se zasićenim.
4. Putanja je zatvorena ukoliko sadrži formulu i njenu negaciju (obično su u pitanju kontradiktorni literali). Tablo je zatvoren ako mu je svaka putanja zatvorena.

Teorema (Korektnost metoda tabloa)

Metod tabloa se zaustavlja za svaku iskaznu formulu i dobijeni tablo je zatvoren ako i samo ako je polazna iskazna formula nezadovoljiva.

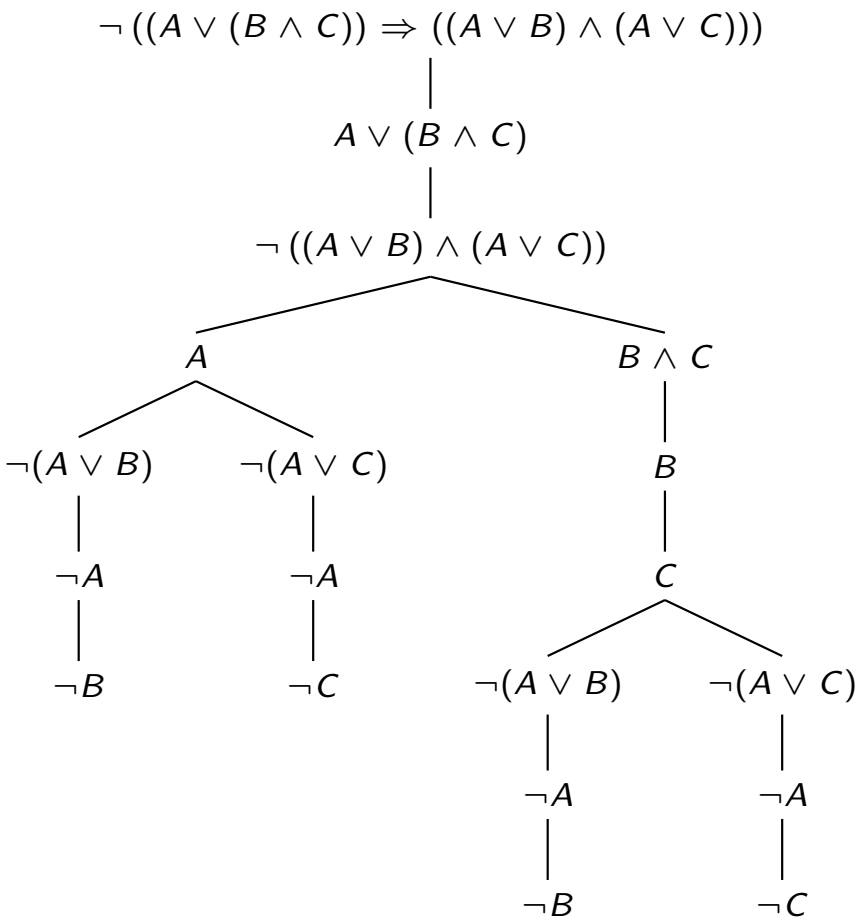
Primer

Pokažimo da je $((A \vee \neg B) \wedge B) \wedge \neg A$ nezadovoljiva formula.



Primer

Pokažimo da je $(A \vee (B \wedge C)) \Rightarrow ((A \vee B) \wedge (A \vee C)$ tautologija.



Redosled primene pravila

- Redosled primene pravila nije bitan za korektnost procedure.
- Bolja efikasnost se dobija ukoliko se negranajućim pravilima da prioritet.

Pregled

Semantički ili deduktivni pristup?

- Većina prikazanih metoda za ispitivanje zadovoljivosti (tj. dualno, pokazivanja tautologičnosti) su bili semantičke prirode.
- Uspešnost semantičkih metoda u iskaznoj logici se pre svega zasniva na činjenici da je dovoljno ispitati konačno mnogo valuacija.
- Kod bogatijih logika, skup relevantnih valuacija (i interpretacija) formule će biti obično beskonačan tako da semantički metodi obično nisu jednostavno primenljivi.

Šta su deduktivni sistemi?

- Deduktivni (ili formalni) sistemi se sastoje od aksioma i pravila izvođenja.
- Formule koje se iz aksioma mogu izvesti konačnom primenom pravila izvđenja, nazivaju se teoreme datog sistema.
- Činjenica da formula F teorema, obeležava se najčešće sa

$$\vdash F.$$

Činjenica da se formula F može dokazati, uz dodatno korišćenje pretpostavki iz nekog skupa Γ, obeležava se najčešće sa

$$\Gamma \vdash F.$$

Semantička relacija $\models$ i deduktivna relacija $\vdash$ moraju biti tesno povezane.

Definicija

Deduktivni sistem za iskaznu logiku je saglasan ako može da dokaže samo tautologije, tj. ako važi $\vdash F$, onda važi i $\models F$. Takođe, ako se formula može dokazati iz nekog skupa pretpostavki, onda je ona logička posledica tog skupa, tj. ako važi $\Gamma \vdash F$, onda važi i $\Gamma \models F$.

Definicija

Deduktivni sistem za neku logiku je potpun ako može da dokaže svaku tautologiju, tj. ako važi $\models F$, onda važi i $\vdash F$. Takođe, ako je formula F logička posledica skupa Γ, onda ona može biti dokazana, tj. ako važi $\Gamma \models F$, onda važi i $\Gamma \vdash F$.

Najčešće korišćeni deduktivni sistemi za logiku

1. Hilbertov sistem
2. Prirodna dedukcija
3. Račun sekvenata

Hilbertov sistem

- Više varijanti — teorija H.
- Minimalistički sistem — samo tri sheme aksioma i jedno pravilo izvođenja. Nije pogodan za praktičnu upotrebu.
- Jednostavnosti radi, razmatraju se samo formule koje sadrže iskazne atome i veznike $\neg$ i $\Rightarrow$, dok se ostali veznici i konstante smatraju skraćenicama (npr. $A \wedge B$ se svodi na $\neg(A \Rightarrow \neg B)$, $A \vee B$ se svodi na $\neg A \Rightarrow B$, $\top$ se svodi na $A \Rightarrow A$, $\bot$ se svodi na $\neg(A \Rightarrow A)$).

Definicija

Sheme aksioma:

(A1): $A \Rightarrow (B \Rightarrow A)$

(A2): $(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$

(A3): $(\neg B \Rightarrow \neg A) \Rightarrow (A \Rightarrow B)$

Pravilo izvođenja je modus ponens.

$$\frac{A \quad A \Rightarrow B}{B}$$

Dokaz u Hilbertovom sistemu

Definicija

Dokaz formule F u Hilbertovom sistemu je lista formula koja sadrži F i koja zadovoljava da je svaka formula u listi ili instanca neke sheme aksioma ili se dobija primenom modus ponensa na neke prethodne elemente u listi. Ako formula F ima dokaz, pišemo

$$\vdash_H F.$$

Dokaz formule F iz pretpostavki iz skupa Γ dopušta da se u listi ravnopravno sa aksiomama koriste i formule skupa Γ. Ako formula F ima dokaz iz pretpostavki iz skupa Γ, pišemo

$$\Gamma \vdash_H F.$$

Hilbertov sistem — primer dokaza

Sheme aksioma:

(A1): $A \Rightarrow (B \Rightarrow A)$

(A2): $(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$

(A3): $(\neg B \Rightarrow \neg A) \Rightarrow (A \Rightarrow B)$

Pravilo izvođenja je modus ponens.

$$\frac{A \quad A \Rightarrow B}{B}$$

1.	$(p \Rightarrow ((p \Rightarrow p) \Rightarrow p)) \Rightarrow ((p \Rightarrow (p \Rightarrow p)) \Rightarrow (p \Rightarrow p))$	A1
2.	$p \Rightarrow ((p \Rightarrow p) \Rightarrow p)$	A2
3.	$(p \Rightarrow (p \Rightarrow p)) \Rightarrow (p \Rightarrow p)$	MP(1, 2)
4.	$p \Rightarrow (p \Rightarrow p)$	A1
5.	$p \Rightarrow p$	MP(3, 4)

$$\dfrac{\dfrac{\text{A1: } (p \Rightarrow ((p \Rightarrow p) \Rightarrow p)) \Rightarrow ((p \Rightarrow (p \Rightarrow p)) \Rightarrow (p \Rightarrow p)) \qquad \text{A2: } p \Rightarrow ((p \Rightarrow p) \Rightarrow p)}{(p \Rightarrow (p \Rightarrow p)) \Rightarrow (p \Rightarrow p)} \qquad \text{A1:} p \Rightarrow (p \Rightarrow p)}{p \Rightarrow p}$$

Svojstva Hilbertovog sistema

Teorema (Teorema dedukcije)

Ako važi $\Gamma, A \vdash_H B$ *onda važi i* $\Gamma \vdash_H A \Rightarrow B$.

Teorema (Saglasnost i potpunost Hilbertovog sistema)

Formula F *je tautologija (tj.* $\models F$*) ako i samo ako je dokaziva u sistemu* H *(tj.* $\vdash_H F$*).*

Prirodna dedukcija

- Gencen (Gerhard Gentzen) 1935.
- Prati uobičajene postupke koje matematičari koriste prilikom dokazivanja teorema.
- Verzije za klasičnu (NK) i intuicionističku logiku (NJ).
- Različite forme: oslobađanje pretpostavki ili eksplicitni konteksti.
- Dve grupe pravila: eliminacija (E) i uvođenje (I).
- Dokazi u obliku stabla u čijem dnu je formula koja je dokazana.

Pravila

Negacija

$$\dfrac{\begin{array}{c}[A]^1\\ \vdots\\ \bot\end{array}}{\neg A}\ \neg I^1 \qquad \dfrac{A \qquad \neg A}{\bot}\ \neg E$$

Konjunkcija

$$\dfrac{A \qquad B}{A \wedge B}\ \wedge I \qquad \dfrac{A \wedge B}{A}\ \wedge E1 \qquad \dfrac{A \wedge B}{B}\ \wedge E2$$

Disjunkcija

$$\dfrac{A}{A \vee B}\ \vee I1 \qquad \dfrac{B}{A \vee B}\ \vee I2$$

$$\dfrac{A \vee B \qquad \begin{array}{c}[A]^1\\ \vdots\\ C\end{array} \qquad \begin{array}{c}[B]^2\\ \vdots\\ C\end{array}}{C}\ \vee E^{1,2}$$

Pravila

Implikacija

$$\frac{\begin{array}{c}[A]^1\\ \vdots\\ B\end{array}}{A \Rightarrow B} \Rightarrow I^1 \qquad \frac{A \qquad A \Rightarrow B}{B} \Rightarrow E$$

Logičke konstante

$$\frac{\bot}{A} \bot E \qquad \frac{}{\top} \top I$$

Klasična naspram intuicionističke logike

- Intuicionistička prirodna dedukcija dopušta isključivo prethodno navedena pravila.
- Klasičnu prirodnu dedukciju karakteriše bilo koje od sledećih svojstava:

Klasična pravila

$$\frac{}{A \vee \neg A}\ \text{ExcludedMiddle} \qquad \frac{\neg\neg A}{A}\ \text{DoubleNegation} \qquad \frac{\begin{array}{c}[\neg A]^1 \\ \vdots \\ \bot\end{array}}{A}\ \textit{Contradiction}^1$$

Prirodna dedukcija – primer

Primer

$$\dfrac{\dfrac{\dfrac{[\neg(A \vee B)]^1 \qquad \dfrac{[A]^2}{A \vee B}\vee I1}{\bot}\neg E}{\neg A}\neg I^2 \qquad \dfrac{\dfrac{\dfrac{[\neg(A \vee B)]^1 \qquad \dfrac{[B]^3}{A \vee B}\vee I2}{\bot}\neg E}{\neg B}\neg I^3}{\neg A \wedge \neg B}\wedge I}{\neg(A \vee B) \Rightarrow \neg A \wedge \neg B}\Rightarrow I^1$$

Teorema (Saglasnost i potpunost prirodne dedukcije)

Formula F je tautologija (tj. $\models F$) ako i samo ako je dokaziva u sistemu prirodne dedukcije za klasičnu logiku (tj. $\vdash_{NK} F$).

Eksplicitni konteksti

Umesto formula, čvorovi stabla dokaza su konteksti (sekventi) oblika $\Gamma \vdash F$, pri čemu je Γ konačan skup pretpostavki.

$$\dfrac{\dfrac{\dfrac{\dfrac{\neg(A \vee B) \vdash \neg(A \vee B) \qquad \dfrac{A \vdash A}{A \vdash A \vee B}\vee I1}{\{\neg(A \vee B), A\} \vdash \bot}\neg E}{\neg(A \vee B) \vdash \neg A}\neg I \qquad \dfrac{\dfrac{\dfrac{\neg(A \vee B) \vdash \neg(A \vee B) \qquad \dfrac{B \vdash B}{B \vdash A \vee B}\vee I2}{\{\neg(A \vee B), B\} \vdash \bot}\neg E}{\neg(A \vee B) \vdash \neg B}\neg I}{\neg(A \vee B) \vdash \neg A \wedge \neg B}}{\vdash \neg(A \vee B) \Rightarrow \neg A \wedge \neg B}\Rightarrow I$$

Pravila sa kontekstima

Negacija

$$\frac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}\ \neg I \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash \neg A}{\Gamma \vdash \bot}\ \neg E$$

Konjunkcija

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge I \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge E1 \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \wedge E2$$

Disjunkcija

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ \vee I1 \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ \vee I2 \qquad \frac{\Gamma \vdash A \vee B \qquad \Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma \vdash C}\ \vee E$$

Pravila sa kontekstima

Implikacija

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \Rightarrow I \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash A \Rightarrow B}{\Gamma \vdash B} \Rightarrow E$$

Logičke konstante

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash A} \bot E \qquad \frac{}{\Gamma \vdash \top} \top I$$

$$\frac{}{\Gamma, A \vdash A} \text{ass}$$

Napomena: pored logičkih pravila, potrebno je dodati još i strukturalna pravila koja brinu o tehničkim detaljima (npr. ponavljanje formula u kontekstima, redosled formula u kontekstima, itd.)

Sistem Isabelle

- Interaktivni dokazivač teorema
- Razvija se skoro 25 godina
- Polson (L. Paulson), Nipkov (T. Nipkow), Vencel (M. Wenzel)
- Podržava različite objektne logike
- Direktno zasnovan na prirodnoj dedukciji
- Formule mogu da se zapisuju bilo u ASCII bilo u lepoj matematičkoj notaciji

Prirodna dedukcija u sistemu Isabelle - pravila

$$
\begin{aligned}
notI &: (P \Longrightarrow False) \Longrightarrow \neg P \\
notE &: [\![\neg P; P]\!] \Longrightarrow R \\
conjI &: [\![P; Q]\!] \Longrightarrow P \wedge Q \\
conjunct1 &: P \wedge Q \Longrightarrow P \\
conjunct2 &: P \wedge Q \Longrightarrow Q \\
conjE &: [\![P \wedge Q; [\![P; Q]\!] \Longrightarrow R]\!] \Longrightarrow R \\
disjI1 &: P \Longrightarrow P \vee Q \\
disjI2 &: Q \Longrightarrow P \vee Q \\
disjE &: [\![P \vee Q; P \Longrightarrow R; Q \Longrightarrow R]\!] \Longrightarrow R \\
impI &: (P \Longrightarrow Q) \Longrightarrow P \longrightarrow Q \\
impE &: [\![P \longrightarrow Q; P; Q \Longrightarrow R]\!] \Longrightarrow R \\
mp &: [\![P \longrightarrow Q; P]\!] \Longrightarrow Q
\end{aligned}
$$

Primena pravila

- Pravila uvođenja primenjuju se sa `apply (rule ime_pravila)`.
- Pravila eliminacije primenjuju se sa `apply (erule ime_pravila)`.

Primer dokaza u sistemu Isabelle

```
lemma "~(A | B) --> ~A & ~B"
apply (rule impI)
apply (rule conjI)
apply (rule notI)
apply (erule notE)
apply (rule disjI1)
apply assumption
apply (rule notI)
apply (erule notE)
apply (rule disjI2)
apply assumption
done
```

Račun sekvenata

- Gencen (G. Gentzen), 1935.
- Razvijen kao pomoćno sredstvo za analizu svojstava prirodne dedukcije.
- Pokazao se kao veoma značajan sam za sebe.
- Naročito pogodan za automatsko rezonovanje.

Sekventi

- U prirodnoj dedukciji, osnovni elemenat u drvetu dokaza je oblika $\Gamma \vdash F$, gde je Γ konačan skup formula, a F je formula.
- Interpretacija objekta $F_1, \ldots, F_n \vdash F$ je da se iz pretpostavki $F_1, \ldots, F_n$ može dokazati formula F.
- Dokazi za $F_1, \ldots, F_n \vdash F$ i $\vdash F_1 \wedge \ldots \wedge F_n \Rightarrow F$ se mogu u prirodnoj dedukciji dopuniti jedan do drugog.
- Postoji određena doza asimetrije u samim pravilima (npr. iako su semantički konjunkcija i disjunkcija dualne operacije, ova dualnost nije jasno vidljiva u pravilima prirodne dedukcije).

Sekventi

- Osnovni objekti računa sekventa su sekventi oblika $\Gamma \vdash \Delta$, gde su Γ i Δ konačne liste (u nekim izlaganjima multiskupovi ili čak skupovi) formula.
- Interpretacija sekventa $F_1, \ldots, F_n \vdash G_1, \ldots G_k$ je da iz svih pretpostavki $F_1, \ldots, F_n$ može dokazati bar jedna od formula $G_1, \ldots G_k$.
- Dokazi za $F_1, \ldots, F_n \vdash G_1, \ldots G_k$ i $\vdash F_1 \wedge \ldots \wedge F_n \Rightarrow G_1 \vee \ldots \vee G_k$ se mogu u računu sekvenata dopuniti jedan do drugog.
- Ipak, intuicionistička varijanta računa sekvenata zabranjuje pojavljivanje više od jedne formule sa desne strane (u skupu Δ) — ovo odgovara prirodnoj dedukciji sa kontekstima.

Pravila (logička)

Konjunkcija

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta} \wedge L \qquad \frac{\Gamma \vdash \Delta, A \qquad \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \wedge B} \wedge R$$

Disjunkcija

$$\frac{\Gamma, A \vdash \Delta \qquad \Gamma, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta} \vee L \qquad \frac{\Gamma \vdash \Delta, A, B}{\Gamma \vdash \Delta, A \vee B} \vee R$$

Negacija

$$\frac{\Gamma \vdash \Delta, A}{\Gamma, \neg A \vdash \Delta} \neg L \qquad \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \Delta, \neg A} \neg R$$

Pravila (logička)

Implikacija

$$\frac{\Gamma \vdash \Delta, A \qquad \Gamma, B \vdash \Delta}{\Gamma, A \Rightarrow B \vdash \Delta} \Rightarrow L \qquad \frac{\Gamma, A \vdash \Delta, B}{\Gamma \vdash \Delta, A \Rightarrow B} \Rightarrow R$$

Logičke konstante

$$\frac{}{\Gamma, \bot \vdash \Delta} \bot L \qquad \frac{}{\Gamma \vdash \Delta, \top} \top R$$

Pretpostavka

$$\frac{}{\Gamma, A \vdash \Delta, A} \; ass$$

Pravila (strukturalna)

Slabljenje (weakening, thinning)

$$\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A}$$

Kontrakcija (contraction)

$$\frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A}$$

Permutovanje (permutation, interchange)

$$\frac{\Gamma', A, B, \Gamma'' \vdash \Delta}{\Gamma', B, A, \Gamma'' \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta', A, B, \Delta''}{\Gamma \vdash \Delta', B, A, \Delta''}$$

Varijante pravila

- Napomenimo da se u literaturi nekad sreću i malo drugačija pravila.
- Na primer, umesto

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta} \wedge L$$

uvode se:

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta} \wedge L_1 \qquad \frac{\Gamma, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta} \wedge L_2$$

- Ovo ne menja suštinski sistem.

Primer izvođenja u računu sekvenata

Primer

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\,ass}{A \vdash A, B}\,weakR}{A \vdash A \vee B}\,\vee R}{A, \neg(A \vee B) \vdash}\,\neg L}{\neg(A \vee B), A \vdash}\,permL}{\neg(A \vee B) \vdash \neg A}\,\neg R \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{B \vdash B}\,ass}{B \vdash B, A}\,weakR}{B \vdash A, B}\,permR}{B \vdash A \vee B}\,\vee R}{B, \neg(A \vee B) \vdash}\,\neg L}{\neg(A \vee B), B \vdash}\,permL}{\neg(A \vee B) \vdash \neg B}\,\neg R$$

$$\dfrac{\neg(A \vee B) \vdash \neg A \qquad \neg(A \vee B) \vdash \neg B}{\neg(A \vee B) \vdash \neg A \wedge \neg B}\,\wedge R$$

$$\dfrac{\neg(A \vee B) \vdash \neg A \wedge \neg B}{\vdash \neg(A \vee B) \Rightarrow \neg A \wedge \neg B}\,\Rightarrow R$$

Sečenje

- Kako bi se dokazi zapisani u prirodnoj dedukciji mogli jednostavnije transformisati u dokaze u računu sekvenata, uvedno je *pravilo sečenja (cut rule)*.

Sečenje (cut)

$$\frac{\Gamma' \vdash \Delta', A \qquad \Gamma'', A \vdash \Delta''}{\Gamma', \Gamma'' \vdash \Delta', \Delta''}\ \text{cut}$$

Primer

Primer

$$\dfrac{\dfrac{[A \wedge B]^1}{A}\wedge E}{A \wedge B \Rightarrow A} \Rightarrow I^1$$

$$\dfrac{\dfrac{\dfrac{}{A \wedge B \vdash A \wedge B}\,ass \qquad \dfrac{\dfrac{}{A \vdash A}\,ass}{A \wedge B \vdash A}\wedge L_1}{A \wedge B \vdash A}\,cut}{\vdash A \wedge B \Rightarrow A} \Rightarrow R$$

Eliminacija sečenja

Teorema (Haputsatz (cut elimination))

Svaki dokaz u računu sekvenata koji koristi sečenje može da se transformiše u dokaz u računu sekvenata koji ne koristi sečenje.

- Veoma kompleksan dokaz (Gencen).
- Nakon eliminacije sečenja dokaz poseduje svojstvo potformule — svaka formula koja se javlja u okviru dokaza je potformula formule koja se dokazuje.
- Izrazito značajno za proces automatizacije pretrage za dokazima.

Teorema (Saglasnost i potpunost računa sekvenata)

Formula F je tautologija (tj. $\models F$) ako i samo ako je dokaziva u sistemu LK^* (tj. $\vdash_{LK^*} F$).

Pregled

Kompaktnost

Teorema (Kompaktnost iskazne logike)

- *Skup iskaznih formula je zadovoljiv akko je svaki njegov konačan skup zadovoljiv.*
- *Skup iskaznih formula je nezadovoljiv akko postoji njegov konačan skup koji je nezadovoljiv.*

Dokaz kompaktnosti za iskaznu logiku

Dokaz (Deduktivni dokaz (skica))

Ukoliko je skup formula nezadovoljiv, njegova nezadovoljivost može biti dokazana u nekom deduktivnom sistemu. U okviru tog dokaza koristi se samo konačno mnogo formula polaznog skupa. Taj konačan podskup formula korišćenih u okviru dokaza je sam za sebe nezadovoljiv.

Moguće je dati i direktan (semantički) dokaz.

Primer primene teoreme kompaktnosti

Teorema kompaktnosti se može iskoristiti kako bi se rezultati sa konačnih domena preneli na beskonačne.

Teorema (Bojenje konačnih grafova)

Svaki planaran graf sa konačnim brojem čvorova se može obojiti sa 4 boje (tako da su susedni čvorovi obojeni različitim bojama).

Teorema (Bojenje grafova)

Svaki planaran graf se može obojiti sa 4 boje (tako da su susedni čvorovi obojeni različitim bojama).

Primer primene teoreme kompaktnosti

Dokaz

4-obojivost se može kodirati u iskaznoj logici. Neka promenljive $p_v^1, p_v^2, p_v^3, p_v^4$ *označavaju da je čvor* v *obojen bojom* $1, 2, 3$ *ili* 4.

- *Svaki čvor je obojen:*

$$p_v^1 \vee p_v^2 \vee p_v^3 \vee p_v^4.$$

- *Svaki čvor je obojen najviše jednom bojom:*

$$\neg(p_v^1 \wedge p_v^2) \wedge \neg(p_v^1 \wedge p_v^3) \wedge \neg(p_v^1 \wedge p_v^4) \wedge \neg(p_v^1 \wedge p_v^2) \wedge \neg(p_v^2 \wedge p_v^3) \wedge \neg(p_v^2 \wedge p_v^4) \wedge \neg(p_v^3 \wedge p_v^4)$$

- *Povezani čvorovi su različito obojeni:*

$$\neg(p_{v_a}^1 \wedge p_{v_b}^1) \wedge \neg(p_{v_a}^2 \wedge p_{v_b}^2) \wedge \neg(p_{v_a}^3 \wedge p_{v_b}^3) \wedge \neg(p_{v_a}^4 \wedge p_{v_b}^4)$$

Ovaj skup formula je konačno zadovoljiv, pa je na osnovu kompaktnosti i zadovoljiv.