Automatsko rezonovanje – beleške sa predavanja Uvod

Filip Marić

*Matematički fakultet,
Univerzitet u Beogradu

Proletnji semestar 2011.

Pregled

1 O kursu automatskog rezonovanja

2 O automatskom rezonovanju

O kursu

Predavanja: dr Filip Marić
Vežbe: Milan Banković
Kurs na master studijama informatike
Nedeljni fond časova: $2 + 3$
Godišnji fond časova: $28 + 42$

Poeni

max	min	Vrsta aktivnosti
10	4	Priprema za čas
5	2	Aktivno prisustvo
15	5	Projekat
20	5	Kolokvijum
30	10	Pismeni deo ispita
20	8	Usmeni deo ispita

Literatura

1. John Harrison, *Handbook of Practical Logic and Automated Reasoning*.
2. Predrag Janičić, *Matematička logika u računarstvu*.
3. Alan Robinson, Andrei Voronkov, eds. *Handbook of Automated Reasoning*.
4. Alan Bundy, *The Computer Modelling of Mathematical Reasoning*

Pregled

1 O kursu automatskog rezonovanja

2 O automatskom rezonovanju

rezonovanje - zaključivanje, rasuđivanje, dedukcija, izvođenje ispravnih zaključaka na osnovu postojećih pretpostavki...

automatsko - prati precizno definisane postupke, može ga izvoditi i mašina (računar), ...

- Rezonovanje leži u osnovi bavljenja matematikom i predmet proučava uglavnom matematičko rezonovanje.
- Grana matematike koja se bavi formalizacijom matematičkog rezonovanja je matematička logika.
- Automatsko rezonovanje se može smatrati oblašću veštačke inteligencije.

Postupci otkrivanja dokaza?

- Matematička logika se uglavnom bavi opravdavanjem ispravnosti matematičkih argumenata i dokaza, dok se načini njihovog otkrivanja uglavnom zanemaruju i prepuštaju *intuiciji*, *iskustvu*, *sreći*.
- Postupci korišćeni da bi se došlo do samih dokaza često se ne prikazuju (npr. dokazi u analizi koji počinju sa *neka je* $\delta < \frac{2\varepsilon}{3}$), već se prikazuje samo ono što čitaoca treba da uveri da je navedeni iskaz tačan.
- Automatsko rezonovanje ima obavezu da proširi prethodni pogled karakterističan za matematičku logiku i da naglasak stavi na precizan opis postupaka otkrivanja i pronalaženja dokaza.

Šta je logičko rezonovanje?

- Vera da je neki iskaz (tvrđenje) tačan može da leži na različitim osnovama (npr. rekao nam je neko kome verujemo, iskaz je saglasan rezultatima eksperimenta koji smo izveli, . . .). — Ovo ostavlja crv sumnje!
- Logičko rezonovanje pokušava da opravda iskaze dajući argumente tj. dedukujući iskaze iz nekih drugih, elementarnijih iskaza.
- Ukoliko je argumentacija ispravna, tačnost nekog iskaza svodi se na tačnost pretpostavki korišćenih pri arugmentaciji.
- Argumentacija se smatraju ispravnom samo ukoliko zadovoljava određenu formu, bez obzira na sadržaj iskaza na koje se odnosi.
- Logički argumenti su dokazi (tačnosti?) iskaza.
- Ipak, da bi se mogla zaključiti tačnost, logika nije dovoljna, tj. potrebno je načiniti i korake koji nisu čisto logički.

Šta je logičko rezonovanje?

Svi ljudi su smrtni. Sokrat je čovek.	Svi X su Y. a je X.	Svi prosti brojevi su parni. 7 je prost broj.
Dakle, Sokrat je smrtan.	Dakle, a je Y	Dakle, 7 je paran.

Sva tri argumenta se smatraju logički ispravnim, bez obzira što su u trećem primeru neki iskazi netačni (pri uobičajenom tumačenju korišćenih termina).

Svi Atinjani su Grci.
Sokrat je Atinjanin.

Dakle, Sokrat je smrtan.

Ovaj argument nije logički ispravan, bez obzira što je izvedeni iskaz tačan (pri uobičajenom tumačenju korišénih termina).

Sintaksa i semantika.

- Matematičko i logičko rezonovanje često biva značajno jednostavnije ukoliko se uvede precizan jezik u kome se iskazi izražavaju.
- Jezici koji se koriste u matematici su obično dominantno simbolički (npr. $x + 3 \geq z^2$ naspram ,,zbir x i 3 je veći ili jednak kvadratu od z).
- Simboli se kombinuju i grade se izrazi koji označavaju neke matematičke objekte.
- Poželjno je jasno razlikovati simbole i izraze od onoga šta oni označavaju, tj. od njima pridruženog značenja.

Sintaksa i semantika

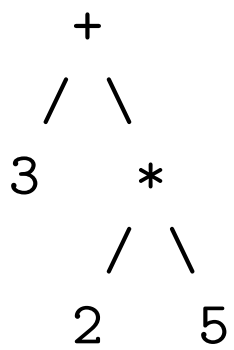
- Dva simbola 12 uobičajeno označavaju prirodan broj dvanaest. Izraz $x + y$ uobičajeno označava primenu operacije sabiranja na x i y. Oba izraza $2 + 3$ i $3 + 2$ uobičajeno označavaju broj 5.
- Simboli $)4 + \cdot(5$ obično ne predstavljaju ispravan izraz i ne označava ništa.
- Izraz $2 + 3$ može da označava i broj 0 ukoliko se uzme da simbol $+$ označava operaciju sabiranja po modulu 5.

Sintaksa i semantika

- Sintaksa jezika definiše skup dopuštenih simbola i skup ispravno zapisanih izraza.
- Semantika se bavi tumačenjem značenja (interpretiranjem) simbola i izraza jezika.

Konkretna i apstraktna sintaksa

- Izrazi su obično predstavljeni niskama nekih karaktera koje predstavljaju konretnu sintaksu (npr. $3 + 2 \cdot 5$ ili 3+2*5).
- Najvažniji aspekt izraza je njihova struktura predstavljena kroz apstraktnu sintaksu. Struktura se najčešće predstavlja u obliku drveta.



- Apstraktna sintaksa nije opterećena tehničkim detaljima poput pravila korišćenja zagrada, prioriteta operatora, asocijativnosti operatora i slično.

Simbolizam i formalizam

- Pogodna simbolika olakšava manipulaciju izrazima. Pravila manipulacije se jednostavno definišu i često primenjuju mehanički (bez razmišljanja zašto su ta pravila opravdana).

Dok sabiramo dva prirodna broja ne razmišljamo o opravdanosti uobičajenog algoritma koji se primenjuje, pa čak ni o definiciji korišćenog dekadnog zapisa brojeva.

- Formalizam shvata matematiku kao čisto sintaksno-deduktivnu ,,igru" nad simbolički zapisanim objektima.
- Formalna ,,igra" može da se igra i automatski, tj. mogu da je izvode i računari.

Objektni jezik i logika; metajezik i logika

- Često se javlja potreba da se izvrši prikaz i rezonuje o svojstvima formalnih logičkih sistema.
- U okviru logika, rezonuje se o različitim matematičkim teorijama, međutim, kako je moguće rezonovati o samoj logici?
- Za to se obično opet koriste simbolički jezici i matematička logika, pri čemu su oni različiti od formalizma koji se prikazuje i o kome se rezonuje.
- Metajezik je jezik koji se koristi za opis drugog objektnog jezika. Metalogika je logika koja se koristi da bi se rezonovalo o svojstvima neke formalne objektne logike.
- U standardnim udžbenicima obično je metajezik govorni jezik, dok je metalogika intuitivna logika (obično višeg reda).

Automatsko rezonovanje – beleške sa predavanja Iskazno rezonovanje

Filip Marić

*Matematički fakultet,
Univerzitet u Beogradu

Proletnji semestar 2011.

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

1 Sintaksa i semantika iskazne logike. Tautologičnost, zadovoljivost.

2 Istinitosne tablice

3 Zamena

4 Normalne forme (NNF, KNF, DNF). Definiciona KNF (Cajtin).

5 DPLL procedura.

6 Metod iskazne rezolucija. DP procedura.

7 Metod tabloa

8 Deduktivni sistemi za iskaznu logiku

9 Kompaktnost

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 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 ga nižem ($\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$) i sa levom asocijativnošću operatora $\wedge$ i $\vee$.

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

Važi $v \vDash F$ *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* $F_1 \equiv F_2'$ *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

1 Sintaksa i semantika iskazne logike. Tautologičnost, zadovoljivost.

2 Istinitosne tablice

3 Zamena

4 Normalne forme (NNF, KNF, DNF). Definiciona KNF (Cajtin).

5 DPLL procedura.

6 Metod iskazne rezolucija. DP procedura.

7 Metod tabloa

8 Deduktivni sistemi za iskaznu logiku

9 Kompaktnost

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

1 Sintaksa i semantika iskazne logike. Tautologičnost, zadovoljivost.

2 Istinitosne tablice

3 Zamena

4 Normalne forme (NNF, KNF, DNF). Definiciona KNF (Cajtin).

5 DPLL procedura.

6 Metod iskazne rezolucija. DP procedura.

7 Metod tabloa

8 Deduktivni sistemi za iskaznu logiku

9 Kompaktnost

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

1 Sintaksa i semantika iskazne logike. Tautologičnost, zadovoljivost.

2 Istinitosne tablice

3 Zamena

4 Normalne forme (NNF, KNF, DNF). Definiciona KNF (Cajtin).

5 DPLL procedura.

6 Metod iskazne rezolucija. DP procedura.

7 Metod tabloa

8 Deduktivni sistemi za iskaznu logiku

9 Kompaktnost

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 je pravilo koje se primenjuje instanca tautologije, 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.

$$\neg\bot \equiv \top$$
$$\neg\top \equiv \bot$$
$$P \wedge \bot \equiv \bot$$
$$\bot \wedge P \equiv \bot$$
$$P \wedge \top \equiv P$$
$$\top \wedge P \equiv P$$
$$P \vee \bot \equiv P$$
$$\bot \vee P \equiv P$$
$$P \vee \top \equiv \top$$
$$\top \vee P \equiv \top$$

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

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. 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

1 Sintaksa i semantika iskazne logike. Tautologičnost, zadovoljivost.

2 Istinitosne tablice

3 Zamena

4 Normalne forme (NNF, KNF, DNF). Definiciona KNF (Cajtin).

5 DPLL procedura.

6 Metod iskazne rezolucija. DP procedura.

7 Metod tabloa

8 Deduktivni sistemi za iskaznu logiku

9 Kompaktnost

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*	$[-6, \vert 1]$
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*	$[-6, \vert 1, 2, \vert 7, -3, \vert 4, -5]$
$M \not\models \neg\, F_0$, $(\text{vars } M) = (\text{vars } F_0)$	*SAT*	$[-6, \vert 1, 2, \vert 7, -3, \vert 4, -5]$

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*	$[\mid +6]$
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

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

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

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

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

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

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*	$[\mid +6]$
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*	$[\vert +6]$
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*	$[\mid +6]$
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*	$[\vert +6]$
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*	$[\vert +6]$
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*	$[\mid +6]$
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

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

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

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

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*	$[-6, \vert +1]$
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*	$[-6, \vert +1]$
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*	$[-6, \vert +1, +2, -3, \vert 4, -5, \vert +7]$
$M \not\models_{\neg} F$, (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

1 Sintaksa i semantika iskazne logike. Tautologičnost, zadovoljivost.

2 Istinitosne tablice

3 Zamena

4 Normalne forme (NNF, KNF, DNF). Definiciona KNF (Cajtin).

5 DPLL procedura.

6 Metod iskazne rezolucija. DP procedura.

7 Metod tabloa

8 Deduktivni sistemi za iskaznu logiku

9 Kompaktnost

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?

- Važano 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

1 Sintaksa i semantika iskazne logike. Tautologičnost, zadovoljivost.

2 Istinitosne tablice

3 Zamena

4 Normalne forme (NNF, KNF, DNF). Definiciona KNF (Cajtin).

5 DPLL procedura.

6 Metod iskazne rezolucija. DP procedura.

7 Metod tabloa

8 Deduktivni sistemi za iskaznu logiku

9 Kompaktnost

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.

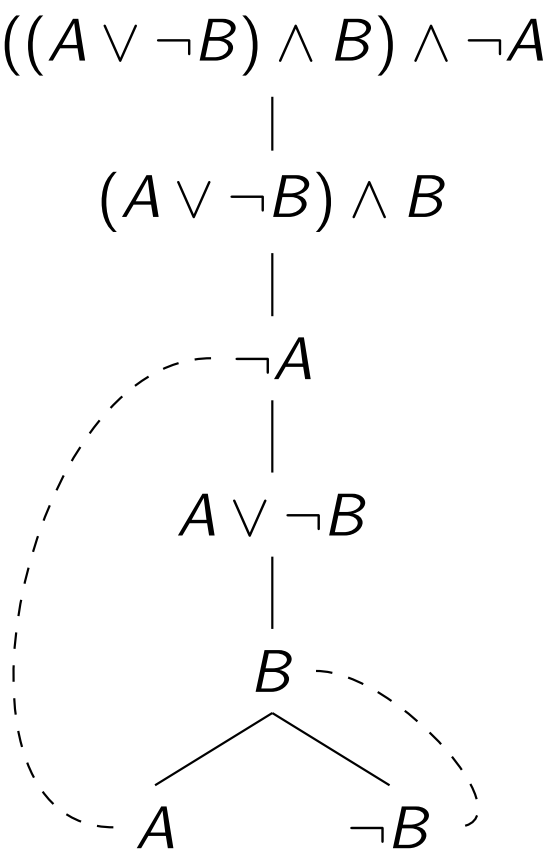
1. Konstrukcija tabloa kreće od stabla koje u svom jedinom svom čvoru (korenu) sadrži formulu čija se zadovoljivost ispituje.
2. U svakom koraku, moguće je stablo proširiti naniže primenom nekog od datih pravila na neku formulu koja se nalaze u putanji od korena do lista koji se proširuje, a na koju 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.

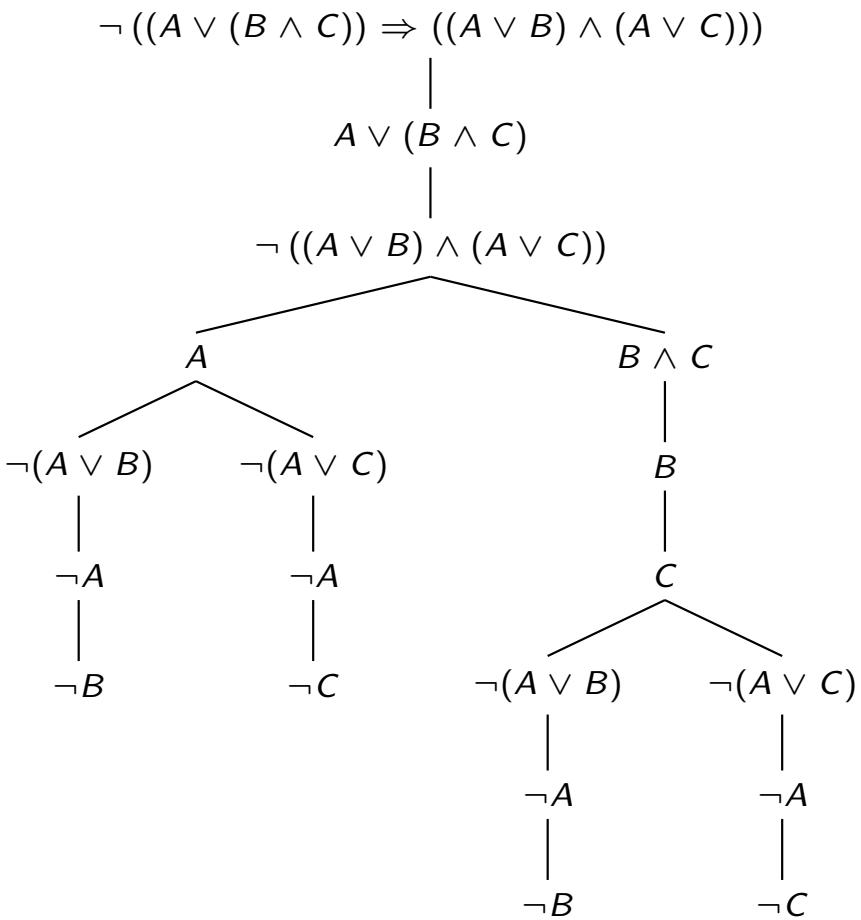
Primer

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

1 Sintaksa i semantika iskazne logike. Tautologičnost, zadovoljivost.

2 Istinitosne tablice

3 Zamena

4 Normalne forme (NNF, KNF, DNF). Definiciona KNF (Cajtin).

5 DPLL procedura.

6 Metod iskazne rezolucija. DP procedura.

7 Metod tabloa

8 Deduktivni sistemi za iskaznu logiku

9 Kompaktnost

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

$$\Gamma \vdash_H F.$$

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

$$\dfrac{A}{A \vee B}\ \vee I1 \qquad \dfrac{B}{A \vee B}\ \vee I2 \qquad \dfrac{A \vee B \qquad \begin{array}{c}[A]^1\\ \vdots\\ C\end{array} \qquad \begin{array}{c}[B]^2\\ \vdots\\ C\end{array}}{C}\ \vee E^{1,2}$$

Pravila

Implikacija

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

Logičke konstante

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

Klasična naspram intuicionističke logike

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

Klasična pravila

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

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 ND (tj. $\vdash_{ND} F$).

Eksplicitni konteksti

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

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

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

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

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) 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

1 Sintaksa i semantika iskazne logike. Tautologičnost, zadovoljivost.

2 Istinitosne tablice

3 Zamena

4 Normalne forme (NNF, KNF, DNF). Definiciona KNF (Cajtin).

5 DPLL procedura.

6 Metod iskazne rezolucija. DP procedura.

7 Metod tabloa

8 Deduktivni sistemi za iskaznu logiku

9 Kompaktnost

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

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.

Automatsko rezonovanje – beleške sa predavanja Rezonovanje u logici prvog reda

Filip Marić

*Matematički fakultet,
Univerzitet u Beogradu

Proletnji semestar 2011.

Pregled

1 Uvod.

2 Sintaksa i semantika logike prvog reda

3 Normalne forme

4 Erbranova teorema.

5 Rezolucija

Unutrašnja struktura iskaza

- Iskazna logika iskaze smatra elementarnim objektima i ne zalazi u njihovu internu strukturu.

Primer

- *Iskazi Sokrat je čovek. i Aristotel je čovek se smatraju različitim.*
- *Njihova untrašnja struktura je slična: . . . je čovek.*

- Poželjno je izgraditi bogatiju logiku koja bi analizirala i unutrašnju strukturu iskaza.

Odnosi među objektima

- Iskazi obično govore o svojstvima objekata i odnosima među objektima.

Primer

- *Sokrat je čovek* — *čovek(Sokrat).*
- *Broj* 3 *je paran* — *paran*(3).
- *Broj* 3 *je manji od broja* 5 — *manji*$(3, 5)$ *ili* $3 < 5$

- U zapisu iskaza, objekti su predstavljeni simbolima konstanti a svojstva i odnosi relacijskim simbolima.

Kombinovanje iskaza

- Iskaze je moguće kombinovati na isti način kao u iskaznoj logici.

Primer

- *Ako je broj* 4 *paran, onda je broj* 4 *veći od* 2 — $paran(4) \Rightarrow 4 > 2$.
- *Broj* 15 *je paran ili je deljiv brojem* 3 — $paran(15) \vee 3 \mid 15$.

Funkcije

- Moguće je govoriti o objektima koji su jednoznačno određeni na osnovu nekih datih objekata.

Primer

- *Sokratova majka je žena.* — *žena(majka(Sokrat)).*
- *Zbir brojeva* 2 *i* 6 *je paran* — *paran*$(2+6)$.

žena i paran označavaju svojstva, dok su majka i $+$ *označavaju funkcije.*

- U zapisu iskaza funkcije se označavaju funkcijskim simbolima.

Kvantifikacija

- U nekim slučajevima želimo da kažemo da svi objekti imaju neko svojstvo ili su u nekom odnosu.
- U nekim slučajevima želimo da kažemo da neki objekti imaju neko svojstvo ili su nekom odnosu.
- U slučaju da objekata o kojima govorimo ima konačno mnogo, moguće je napraviti konjunkciju ili disjunkciju iskaza koji govore o pojedinačnim objektima.
- Kako bi se izricanje ovakvih tvrdnji olakžalo uvode se promenljive i kvantifikatori.

Kvantifikacija

Primer

- *Svi ljudi su smrtni* — $\forall x.čovek(x) \Rightarrow smrtan(x)$.
- *Ne postoji savršen čovek* — $\neg\exists x.čovek(x) \wedge savršen(x)$.
- *Svi parni brojevi su deljivi sa 2* — $\forall n.paran(n) \Rightarrow 2 \mid n$.
- *Sledbenici nekih parnih brojeva su deljivi sa 3* — $\exists n.paran(n) \wedge 3 \mid (n+1)$.

Kvantifikatori se primenjuju na promenljive koje učestvuju u izrazima ravnopravno sa simbolima konstanti.

Kvantifikacija prvog reda

- U okviru logike prvog reda, kvantifikacija se vrši samo po primitivnim objektima — funkcijski i relacijski simboli se ne mogu kvantifikovati. Npr. rečenica da neki objekat a ima sva moguća svojstva ($\forall p.p(a)$) nije rečenica logike prvog reda.
- Logike višeg reda dopuštaju kvantifikaciju funkcijskih (odnosno relacijskih) simbola.

Šta se dešava ukoliko se kvantifikator izostavi?

- Svi do sada posmatrani iskazi predstavljali su rečenice.
- Ima li smisla u izrazima koristiti promenljive koje nisu pod dejstvom kvantifikatora?
- U tom slučaju, istinitosna vrednost iskaza nije jednoznačno određena već zavisi od vrednosti koje promenljive uzimaju.

Primer

- *Istinitosna vrednost tvrđenja* $\forall x.\ x > 0$ *ne zavisi od vrednosti promenljive* x*.*
- *Istinitosna vrednost tvrđenja* $x > 0$ *zavisi od vrednosti koju promenljiva* x *predstavlja.*

Pregled

1 Uvod.

2 Sintaksa i semantika logike prvog reda

3 Normalne forme

4 Erbranova teorema.

5 Rezolucija

Jezik (signatura)

- U zapisu formula logike prvog reda, pored logičkih simbola, učestvuju:
  - simboli konstanti
  - funkcijski simboli
  - relacijski (predikatski) simboli
- Svakom simbolu je pridružena arnost — broj argumenata na koji se funkcijski ili relacijski simbol primenjuje.
- Simboli konstanti se mogu shvatiti kao funkcijski simboli arnosti 0.
- Atomički iskazi (iskazna slova) se mogu shvatiti kao relacijski simboli arnosti 0.
- Neki binarni funkcijski ili relacijski simboli se obično pišu infiksno (npr. umesto $+(3, 2)$, pišemo $3 + 2$).

Jezik (signatura)

Definicija (Jezik (signatura))

Jezik $\mathcal{L}$ *čini skup funkcijskih simbola* Σ*, skup relacijskih simbola* Π *i funkcija* $ar : (\Sigma \cup \Pi) \to \mathbb{N}$ *koja svakom simbolu dodelju arnost.*

Jezik — primeri

- Jezik je obično određen matematičkom teorijom koja se formalizuje.

Primer

- *Ukoliko se priča o aritmetici, jezik je obično* $(0, 1, +, -, *, <, \leq, \ldots)$. *Npr.* $0 < 1 + 1$.
- *Ukoliko se priča o geometriji, jezik je obično čisto relacijski* (*incidentno*, *između*, *podudarno*, ...)

Sintaksa

- Sintaksu logike prvog reda definišemo u nekoliko faza:
  - Termovi
  - Atomičke formule
  - Formule
- Sa semantičkog stanovišta, termovi označavaju objekte nekog domena, dok formule imaju istinitosne vrednosti (tačno, netačno).

Termovi

- Termovi (jezika $\mathcal{L}$) su izrazi izgrađeni primenom funkcijskih simbola na konstante i promenljive.

Definicija

Skup termova (jezika $\mathcal{L}$) je najmanji skup koji zadovoljava:

- *Svaka promenljiva je term.*
- *Ako je c simbol konstante (jezika $\mathcal{L}$), onda je c term.*
- *Ako su $t_1, \ldots, t_k$ termovi, i f funkcijski simbol (jezika $\mathcal{L}$) arnosti k, onda je i $f(t_1, \ldots, t_k)$ takođe term.*

Termovi — primeri

Primer

Neka je $\mathcal{L}$ jezik sa konstantnim simbolima a i b, funkcijskim simbolom f arnosti 2 i g arnosti 1.
Neki od termova jezika $\mathcal{L}$ su:

- $a,\ b,\ x,\ y$
- $f(a,a),\ f(x,b),\ g(a)$
- $f(x,g(a)),\ f(f(a,b),g(x)),\ \ldots$

Implementacija u funkcionalnom jeziku

Obično se u apstraktnoj sintaksi ne vrši provera ispravnosti terma (arnosti funkcija) već se takva provera vrši prilikom parsiranja (obrade konkretne sintakse).

```
datatype term = Var string
              | Fn string  "term list"
```

Atomičke formule

Atomičke formule se grade primenom relacijskih simbola na termove.

Definicija

Atomičke formule (jezika $\mathcal{L}$) su najmanji skup koji zadovoljava da ako je ρ relacijski simbol jezika $\mathcal{L}$ arnosti k, i $t_1, \ldots, t_k$ termovi (jezika $\mathcal{L}$), onda je $\rho(t_1, \ldots, t_k)$ atomička formula.

Atomičke formule — primeri

Primer

- $0 < 1,\ 2 + x \geq 5,\ x = 3$
- $\mathit{paran}(3),\ 4 \mid f(x)$
- $\mathit{između}(A, B, C), \mathit{podudarno}(A, B, A_1, B_1)$
- $\rho(x \circ 0, 0)$

Implementacija u funkcionalnom jeziku

Slično, provera arnosti relacija se ne vidi u apstraktnoj sintaksi.

```
datatype atom = R string "term list"
```

Formule

Formule se grade od atomičkih formula primenom logičkih veznika i kvantifikatora.

Definicija

Skup formula je najmanji skup koji zadovoljava:

- *Atomičke formule su formule*
- *Ako je A formula onda je $\neg(A)$ formula*
- *Ako su A i B formule, onda su i $(A) \wedge (B)$, $(A) \vee (B)$, $(A) \Rightarrow (B)$, $(A) \Leftrightarrow (B)$ formule.*
- *Ako je A formula, a x promenljiva onda su i $\forall x.(A)$ i $\exists x.(A)$ formule.*

Skraćeni zapis

- Prilikom zapisa formula, usvaja se prioritet veznika i u skladu sa tim se mogu izostaviti zagrade.

Primer

$$\forall x.\ f(x) < 3 \wedge x = 5 \Rightarrow (\exists y.\ g(y) = x)$$

je skraćeni zapis za

$$\forall x.\ (((f(x) < 3) \wedge (x = 5)) \Rightarrow (\exists y.\ g(y) = 7))$$

- Uzastopni kvantifikatori se kondenzuju.

Primer

$$\forall xyz.\ \rho(f(x, y), z)$$

je skraćeni zapis za

$$\forall x.\ \forall y.\ \forall z.\ \rho(f(x, y), z)$$

Implementacija u funkcionalnom jeziku

```
datatype formula =
      TRUE
    | FALSE
    | Atom atom
    | 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)
    | Forall string formula
    | Exists string formula
```

Slobodna i vezana pojavljivanja promenljivih

Pojavljivanja promenljive su vezana ako su pod dejstvom nekog kvantifikatora, a slobodna inače.

Definicija

- *Svako pojavljivanje promenljive u okviru atomičke formule je slobodno.*
- *Sva slobodna pojavljivanja promenljvih u formuli A su slobodna i u $\neg A$.*
- *Sva slobodna pojavljivanja promenljvih u formulama A i B su slobodna i u $A \wedge B$, $A \vee B$, $A \Rightarrow B$ i $A \Rightarrow B$.*
- *Sva slobodna pojavljivanja promenljive različita od x u formuli A su slobodna i u $\forall x.\ A$ i $\exists x.\ A$.*

Sva pojavljivanja promenljive u formuli koja nisu slobodna su vezana.

Slobodna i vezana pojavljivanja — primer

Primer

- *U formuli* $p(x, y) \Rightarrow (\forall x.\ q(x))$, *prvo pojavljivanje promenljive* x *i pojavljivanje promenljive* y *su slobodna, dok je drugo pojavljivanje promenljive* x *vezano.*
- *U formuli* $\exists x.\ p(x) \wedge (\forall x.\ q(x))$, *oba pojavljivanja promenljive* x *su vezana, pri čemu je pojavljivanje u okviru* $p(x)$ *pod dejstvom univerzalnog, a pojavljivanje u okviru* $q(x)$ *pod dejstvom egzistencijalnog kvantifikatora.*

Slobodne promenljive — implementacija

```
primrec fvt where
  "fvt (Var x) = {x}"
| "fvt (Fn args) = Union (map fvt args)"

fun fv where
  "fv False = {}"
| "fv True = {}"
| "fv (Atom (R r args)) = Union (map fvt args)"
| "fv (Not p) = fv p"
| "fv (p And q) = (fv p) Un (fv q)"
| "fv (p Or q) = (fv p) Un (fv q)"
| "fv (p Imp q) = (fv p) Un (fv q)"
| "fv (p Iff q) = (fv p) Un (fv q)"
| "fv (Forall x f) = (fv f) - {x}"
| "fv (Exists x f) = (fv f) - {x}"
```

Rečenice. Bazne formule

Definicija

- *Formula je rečenica akko nema slobodnih promenljivih.*
- *Formula je bazna akko nema promenljivih.*

Interpretacija

Obično u matematici, kada se napiše formula, implicitno se podrazumeva na koji skup objekata se formula odnosi (koje objekte konstante označavaju, šta su domeni promenljivih, koje funkcije i koje relacije su označene relacijskim i funkcijskim simbolima).

Primer

$$\forall x.\ 6|x \Rightarrow \neg paran(x+1)$$

Podrazumeva se da 6 *označava prirodan broj šest,* 1 *označava prirodan broj* 1*, da* $+$ *označava funkciju sabiranja dva prirodna broja, da* $|$ *označava relaciju deljivosti na skupu prirodnih brojeva dok paran označava svojstvo parnosti prirodnih brojeva.*

Interpretacija — domeni

Primer

- *Formula* $\forall x.\ \exists y.y + 1 = x$ *je tačna ako je domen skup celih brojeva, a netačna ako je domen skup prirodnih brojeva.*
- *Formula* $\forall x\ y.\ x < y \Rightarrow \exists z.\ (x < z \wedge z < y)$ *je tačna ako je domen skup realnih brojeva a netančna ako je dome skup celih brojeva.*

$\mathcal{L}$-strukture (modeli)

Definicija

Neka je dat jezik $\mathcal{L}$. $\mathcal{L}$-strukturu (model) $\mathfrak{D}$ čini:

- *Neprazan skup objekata (domen) D*
- *Za svaki simbol konstante c, njegova interpretacija $c_{\mathfrak{D}} \in D$.*
- *Za svaki funkcijski simbol f arnosti k, njegova interpretacija $f_{\mathfrak{D}} : D^k \to D$.*
- *Za svaki relacijski simbol ρ arnosti k, njegova interpretacija $\rho_{\mathfrak{D}} \subseteq D^k$.*

Na dalje ćemo obično pretpostavljati da je jezik $\mathcal{L}$ fiksiran pa ćemo umesto $\mathcal{L}$ struktura govoriti samo struktura (ili domen).

Totalnost funkcija

U definiciji modela, podrazumeva se da su sve funkcije totalne i jednoznačne, tj. da su njihove jednoznačno definisane za sve vrednosti argumenata iz domena D.

Tipovi (sorte)

- Primetimo da se svi objekti u okviru formule interpretiraju u okviru jedinstvenog domena, tj. da se ne razlikuju tipovi (sorte) objekata.
- Postoje varijacije logike (višesortne logike, logike višeg reda, . . .) koje dopuštaju uvođenje tipova (sorti) i tada se objekti svakog pojedinačnog tipa uzimaju iz posebnog domena.

Primer

- *Za svake dve tačke postoji prava koja ih sadrži.*
- $\forall A : t.\ \forall B : t.\ \exists l : p.\ A \in l \wedge B \in l$
- $\forall A.\ \forall B.\ t(A) \wedge t(B) \Rightarrow \exists l.\ p(l) \wedge A \in l \wedge B \in l$

Istinitosna vrednost rečenica

- Istinitosna vrednost rečenica zavisi samo od domena i interpretacije simbola (tj. određena je jednoznačno strukturom).
- Definicija ide rekurzivno po strukturi formule.
- Problem: Nije dovoljno posmatrati samo istinitosnu vrednost rečenica, jer se uklanjanjem kvantifikatora dobija formula sa slobodnim promenljivim čija vrednost nije određena jednoznačno dok se ne odredi vrednost promenljive.
  Npr. vrednost formule $\forall x.\ x > 0$ zavisi od vrednosti formule $x > 0$, pri čemu je potrebno dodatno naglasiti koje vrednosti x se posmatraju.
- Zbog toga se posmatra vrednost formule u datoj strukturi $\mathfrak{D}$ za datu valuaciju promenljivih v.

Vrednost termova

Vrednost terma u strukturi $\mathfrak{D}$ pri valuaciji v, definiše se rekurzivno po strukturi terma.

Definicija

Vrednost terma

- *Ako je term t promenljiva x, onda je njegova vrednost vrednost promenljive x u valuaciji v, tj. $\mathfrak{D}_v(t) = v(x)$.*
- *Ako je term t konstantni simbol c, onda je njegova vrednost interpretacija simbola c u strukturi $\mathfrak{D}$, tj. $\mathfrak{D}_v(t) = c_{\mathfrak{D}}$.*
- *Ako je term t oblika $f(t_1, \ldots, t_k)$, onda je njegova vrednost jednaka rezultatu primene funkcije koja je interpretacija simbola f u strukturi $\mathfrak{D}$ na vrednosti termova $t_1, \ldots t_k$, tj. $\mathfrak{D}_v(t) = f_{\mathfrak{D}}(\mathfrak{D}_v(t_1), \ldots, \mathfrak{D}_v(t_k))$.*

Implementacija u funkcionalnom jeziku

```
fun termval where
  "termval (domain, func, pred) v (Var x) = v x"
| "termval (domain, func, pred) v (Fn f args) =
        (func f) (map (termval (domain, func, pred) v) args)"
```

Zadovoljenje (tačnost)

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

- Atomička formula $\rho(t_1, \ldots, t_k)$ je tačna u strukturi $\mathfrak{D}$ pri valuaciji v (tj. $(\mathfrak{D}, v) \models \rho(t_1, \ldots, t_k)$) akko važi $\rho_{\mathfrak{D}}(\mathfrak{D}_v(t_1), \ldots, \mathfrak{D}_v(t_k))$, gde je $\mathfrak{D}_v(t)$ funkcija vrednosti terma u strukturi $\mathfrak{D}$ pri valuaciji v.
- Konstanta $\top$ je tačna u svakoj strukturi i valuaciji ($(\mathfrak{D}, v) \models \top$). Konstanta $\bot$ je netačna u svakoj strukturi i valuaciji ($(\mathfrak{D}, v) \not\models \bot$).
- Formula oblika $\neg F$ je tačna u strukturi $\mathfrak{D}$ pri valuaciji v akko je formula F netačna u strukturi $\mathfrak{D}$ pri valuaciji v (tj. $(\mathfrak{D}, v) \models \neg F$ akko $(\mathfrak{D}, v) \not\models F$).
- Formula oblika $F_1 \wedge F_2$ je tačna u strukturi $\mathfrak{D}$ pri valuaciji v akko su obe formule F_1 i F_2 tačne u strukturi $\mathfrak{D}$ pri valuaciji v (tj. $(\mathfrak{D}, v) \models F_1 \wedge F_2$ akko $(\mathfrak{D}, v) \models F_1$ i $(\mathfrak{D}, v) \models F_2$).

- Formula oblika $F_1 \vee F_2$ je tačna u strukturi $\mathfrak{D}$ pri valuaciji v akko je bar jedna od formula F_1 i F_2 tačna u strukturi $\mathfrak{D}$ pri valuaciji v (tj. $(\mathfrak{D}, v) \models F_1 \vee F_2$ akko $(\mathfrak{D}, v) \models F_1$ ili $(\mathfrak{D}, v) \models F_2$).
- Formula oblika $F_1 \Rightarrow F_2$ je tačna u strukturi $\mathfrak{D}$ pri valuaciji v akko su je formula F_1 netačna ili je formula F_2 tačna u strukturi $\mathfrak{D}$ pri valuaciji v (tj. $(\mathfrak{D}, v) \models F_1 \Rightarrow F_2$ akko $(\mathfrak{D}, v) \not\models F_1$ ili $(\mathfrak{D}, v) \models F_2$).
- Formula oblika $F_1 \Leftrightarrow F_2$ je tačna u strukturi $\mathfrak{D}$ pri valuaciji v akko su formule F_1 i F_2 istovremeno tačne ili istovremeno netačne u strukturi $\mathfrak{D}$ pri valuaciji v (tj. $(\mathfrak{D}, v) \models F_1 \Rightarrow F_2$ akko $(\mathfrak{D}, v) \models F_1$ i $(\mathfrak{D}, v) \models F_2$ ili $(\mathfrak{D}, v) \not\models F_1$ i $(\mathfrak{D}, v) \not\models F_2$).

- Formula oblika $\exists x.F$ je tačna u strukturi $\mathfrak{D}$ pri valuaciji v akko postoji valuacija v' dobijena od v samo izmenom vrednosti promenljive x takva da je F tačna u strukturi $\mathfrak{D}$ pri valuaciji v' (tj. $(\mathfrak{D}, v) \models \exists x.\ F$ ako postoji v' tako da $(\mathfrak{D}, v') \models F$). Drugim rečima, $(\mathfrak{D}, v) \models \exists x.\ F$ akko postoji element a iz domena D, tako da $(\mathfrak{D}, v(x \mapsto a)) \models F$.
- Formula oblika $\forall x.F$ je tačna u strukturi $\mathfrak{D}$ pri valuaciji v akko za svaku valuaciju v' dobijenu od v samo izmenom vrednosti promenljive x važi da je F tačna u strukturi $\mathfrak{D}$ pri valuaciji v' (tj. $(\mathfrak{D}, v) \models \forall x.\ F$ ako za svaku v' važi $(\mathfrak{D}, v') \models F$). Drugim rečima $(\mathfrak{D}, v) \models \forall x.\ F$ akko za svaki element a iz domena D, važi $(\mathfrak{D}, v(x \mapsto a)) \models F$.

Formalna definicija

```
  fun holds where
  "holds (domain, func, pred) v FALSE = False"
| "holds (domain, func, pred) v TRUE = True"
| "holds (domain, func, pred) v (Atom (R r args)) =
     (pred r) (map (termval (domain, func, pred) v) args)"
| "holds (domain, func, pred) v (Not F) =
     (~ holds (domain, func, pred) v F)"
| "holds (domain, func, pred) v (F1 And F2) =
     (holds (domain, func, pred) v F1 & holds (domain, func, pred) v F2)"
| "holds (domain, func, pred) v (F1 Or F2) =
     (holds (domain, func, pred) v F1 | holds (domain, func, pred) v F2)"
| "holds (domain, func, pred) v (F1 Imp F2) =
     (holds (domain, func, pred) v F1 -->  holds (domain, func, pred) v F2)"
| "holds (domain, func, pred) v (F1 Iff F2) =
     (holds (domain, func, pred) v F1 = holds (domain, func, pred) v F2)"
| "holds (domain, func, pred) v (Forall x p) =
     (ALL a : domain. holds (domain, func, pred) (v (x := a)) p)"
| "holds (domain, func, pred) v (Exists x p) =
     (EXISTS a : domain. holds (domain, func, pred) (v (x := a)) p)"
```

Problem određivanja tačnosti formule

- U iskaznoj logici, u trenutku kada je data valuacija, vrednost formule je (trivijalno) moguće odrediti.
- U logici prvog reda, iako je data valuacija i interpretacija, vrednost formule je komplikovano odrediti. Računarsku implementaciju na osnovu definicije je moguće odrediti samo u slučaju konačnih domena D (u slučaju beskonačnog domena, kod kvantifikacije bi bilo potrebno razmatrati vrednost potformule u besknačno mnogo različitih valuacija).

Tačnost rečenica ne zavisi od valuacije

Stav

Ako se valuacija v i v' poklapaju za sve slobodne promenljive formule F, onda $(\mathfrak{D}, v) \models F$ ako i samo ako $(\mathfrak{D}, v') \models F$.

Dokaz

Indukcijom po strukturi formule F.

- *Ako je F oblika $\top$ ili $\bot$, tvrđenje trivijalno važi.*
- *Ako je F oblika $\neg F'$, onda se skup slobodnih promenljivih za F i F' poklapa. $(\mathfrak{D}, v) \models F$ akko $(\mathfrak{D}, v) \not\models F'$. Po induktivnoj hipotezi, ovo važi akko $(\mathfrak{D}, v') \not\models F'$ akko $(\mathfrak{D}, v') \models F$.*

Tačnost rečenica ne zavisi od valuacije

Dokaz

- *Ako je F oblika $F' \wedge F''$, onda je skup slobodnih promenljivih formule F jednak uniji skupova slobodnih promenljivih formula F' i F'', pa se valuacije v i v' poklapaju i na skupovima slobodnih promenljivih formula F' i F'', pa se na ove formule sme primeniti induktivna hipoteza. $(\mathfrak{D}, v) \models F$ važi akko važi $(\mathfrak{D}, v) \models F'$ i $(\mathfrak{D}, v) \models F''$. Na osnovu i.h. ovo važi akko $(\mathfrak{D}, v') \models F'$ i $(\mathfrak{D}, v') \models F''$, a ovo važi akko $(\mathfrak{D}, v') \models F$.*
- *Ostali iskazni veznici se analogno razmatraju.*

Tačnost rečenica ne zavisi od valuacije

Dokaz

- *Ako je F oblika $\forall x.\ F'$, onda je skup slobodnih promenljivih formule F jednak skupu slobodnih promenljivih formule F', bez promenljive x. $(\mathfrak{D}, v) \models F$ važi akko za svako a iz D, $(\mathfrak{D}, v(x \mapsto a)) \models F'$. Valuacije $v(x \mapsto a)$ i $v'(x \mapsto a)$ se poklapaju na skupu slobodnih promenljivih formule F' te na osnovu i.h. $(\mathfrak{D}, v(x \mapsto a)) \models F'$ važi akko $(\mathfrak{D}, v'(x \mapsto a)) \models F'$, što važi akko $(\mathfrak{D}, v') \models F$.*
- *Slučaj kada je F oblika $\exists x.\ F'$ se razmatra analogno.*

Tačnost rečenica ne zavisi od valuacije

Posledica

Vrednost rečenica ne zavisi od valuacije, već zavisi samo od interpretacije nelogičkih simbola. Preciznije, za svake dve valuacije v i v' i rečenicu F, $(\mathfrak{D}, v) \models F$ važi ako i samo ako $(\mathfrak{D}, v') \models F$.

Semantika — primeri

Primer

Neka je dat jezik $\mathcal{L} = (0, 1, +, \cdot, =)$. *Posmatrajmo domen* $D = \mathbb{N}$. *Neka je struktura* $\mathfrak{D}_{\mathbb{N}}$ *određena na sledeći način:*

1. *Simbol* 0 *se interpretira prirodnim brojem* 0
2. *Simbol* 1 *se interpretira prirodnim brojem* 1
3. *Simbol* $+$ *se interpretira operacijom sabiranja prirodnih brojeva*
4. *Sibmol* $\cdot$ *se interpretira operacijom množenja prirodnih brojeva.*
5. *Simbol* $=$ *se interpretira jednakošću prirodnih brojeva.*

Primer

Sledeće rečenice su tačne u $\mathfrak{D}_{\mathbb{N}}$*:*

- $\neg(0 = 1)$
- $\forall x.\ \neg(x + 1 = 0)$
- $\forall x\ y.\ x \cdot (y + 1) = x \cdot y + x$
- $\forall x.\ x = 0 \vee \neg(x = 0)$
- $\neg(\forall x.\ x = 0) \Leftrightarrow (\exists x.\ \neg(x = 0))$

Sledeće rečenice su netačne u $\mathfrak{D}_{\mathbb{N}}$*:*

- $\forall x.\ \exists y.\ y + 1 = x$
- $\forall x.\ \exists y.\ x + y = 0$

Vrednost formula zavisi od valuacije:

- *Formula* $x + 1 = 1$ *je tačna u valuacijama u kojima je vrednost* x *prirodan broj nula, a netačna u valuacijama u kojima je vrednost* x *različita od prirodnog broja nula.*

Semantika — primeri

Primer

Neka je dat jezik $\mathcal{L} = (0, 1, +, \cdot, =)$. *Posmatrajmo domen* $D_{bool} = \{\top, \bot\}$. *Neka je struktura* $\mathfrak{D}_{bool}$ *određena na sledeći način:*

1. *simbol* 0 *se interpretira elementom* $\bot$,
2. *simbol* 1 *se interpretira elementom* $\top$,
3. *simbol* $+$ *se interpretira funkcijom koja elemente* x *i* y *iz* D_{bool} *slika u* $\top$ *akko je* $x \wedge y$ *tačna tj.*

$+$	$\top$	$\bot$
$\top$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$

4. *simbol* $\cdot$ *se interpretira funkcijom koja elemente* x *i* y *iz* D_{bool} *slika u* $\top$ *akko je* $\neg(x \Leftrightarrow y)$ *tačna, tj.*

$\cdot$	$\top$	$\bot$
$\top$	$\bot$	$\top$
$\bot$	$\top$	$\bot$

5. *simbol* $=$ *se interpretira kao jednakost*

```
definition bool_interp where
"bool_interp == let
        domain = {True, False};
        funcs  = % f args. case (f, args) of
                    (''0'', []) => True |
                    (''1'', []) => False |
                    (''+'', [x, y]) => x & y |
                    (''*'', [x, y]) => ~(x = y);
        rels   =  % r args. case (r, args) of
                    (''='', [x::bool, y]) => x = y in
        (domain, funcs, rels)"
```

Semantika — primeri

Primer

Sledeće rečenice su tačne u $\mathfrak{D}_{bool}$.

- $\forall x.\ x = 0 \vee x = 1.$
- $\forall x.\ \exists y.\ x + y = 0.$
- $\forall x.\ x = 0 \vee \neg(x = 0)$
- $\neg(\forall x.\ x = 0) \Leftrightarrow (\exists x.\ \neg(x = 0))$

Sledeće rečenice su netačne u $\mathfrak{D}_{bool}$.

- $0 = 1$
- $\forall x.\ \neg(x + 1 = 0)$

Semantika — primeri

Primer

Da bi $\forall x.\ \exists y.\ x + y = 0$ *bila tačna u valuaciji* v*, potrebno je da:*

1. *formula* $\exists y.\ x + y = 0$ *bude tačna u valuaciji* $v(x \mapsto \top)$
2. *formula* $\exists y.\ x + y = 0$ *bude tačna u valuaciji* $v(x \mapsto \bot)$.

1. *Da bi formula* $\exists y.x + y = 0$ *bila tačna u valuaciji* $v(x \mapsto \top)$*, potrebno je da postoji element* d *iz* D *takav da je* $x + y = 0$ *tačna u valuaciji* $v(x \mapsto \top, y \mapsto d)$*. Zaista, formula* $x + y = 0$ *je tačna za* $d = \bot$*, tj. u valuaciji* $v' = v(x \mapsto \top, y \mapsto \bot)$*. Zaista, vrednost oba terma* $x + y$ *i* 0 *u* v' *je* $\bot$*, te su oni jednaki.*
2. *Da bi formula* $\exists y.x + y = 0$ *bila tačna u valuaciji* $v(x \mapsto \bot)$*, potrebno je da postoji element* d *iz* D *takav da je* $x + y = 0$ *tačna u valuaciji* $v(x \mapsto \bot, y \mapsto d)$*. Zaista, formula* $x + y = 0$ *je tačna za, npr.* $d = \bot$*, tj. u valuaciji* $v' = v(x \mapsto \top, y \mapsto \bot)$*. Zaista, vrednost oba terma* $x + y$ *i* 0 *u* v' *je* $\bot$*, te su oni jednaki.*

Semantika — primeri

Primer

Neka je dat jezik $\mathcal{L} = (0, 1, +, \cdot, =)$. *Za* $n \geq 2$, *posmatrajmo domen* $D_{mod_n} = \{0, 1, \ldots n-1\}$. *Neka je struktura* $\mathfrak{D}_{mod_n}$ *određena na sledeći način:*

1. *simbol* 0 *se interpretira brojem* 0,
2. *simbol* 1 *se interpretira brojem* 1,
3. *simbol* $+$ *se interpretira funkcijom koja sabira brojeve* x *i* y *iz* D *po modulu* n,
4. *simbol* $\cdot$ *se interpretira funkcijom koja množi elemente* x *i* y *po modulu* n,
5. *simbol* $=$ *se interpretira kao jednakost.*

Semantika — primeri

- Rečenica $\forall x.\ x = 0 \vee \neg(x = 0)$ je tačna u svim $\mathfrak{D}_{mod_n}$.
- Rečenica $\neg(\forall x.\ x = 0) \Leftrightarrow (\exists x.\ \neg(x = 0))$ je tačna u svim $\mathfrak{D}_{mod_n}$.
- Rečenica $\forall x.\ x = 0 \vee x = 1$ je tačna u $\mathfrak{D}_{mod_2}$, a netačna u $\mathfrak{D}_{mod_3}$.
-
- Rečenica $\forall x.\ \neg(x = 0) \Rightarrow \exists y.\ x \cdot y = 1$ je tačna u $\mathfrak{D}_{mod_n}$ akko je n prost broj.

Valjanost

- Pojam valjane formule donekle odgovara pojmu iskazne tautologije.

Definicija

Formula je valjana ako tačna u svakoj valuaciji, pri svakoj interpretaciji.

- Naravno, rečenica je valjana ako je tačna pri svakoj interpretaciji.

Stav

- *Rečenica je valjana akko je tačna pri svakoj interpretaciji.*
- *Formula F koja ima slobodne promenljive $x_1, \ldots x_n$ je valjana akko je valjano njeno univerzalno zatvorenje $\forall x_1, \ldots, x_n.\ F$.*

Valjane formule — primeri

Primer

- *Rečenica* $\forall x.\ x = 0 \vee \neg(x = 0)$ *je valjana.*
- *Rečenica* $\neg(\forall x.\ x = 0) \Leftrightarrow (\exists x.\ \neg(x = 0))$.
- $(\forall x.P(x)) \Rightarrow P(a)$ *je valjana, dok formula* $P(x) \Rightarrow P(a)$ *ni formula* $\forall x.P(x) \Rightarrow P(a)$ *nije.*

Valjane formule — primeri

Primer (Paradoks pijanca)

Rečenica $\exists x.\ P(x) \Rightarrow \forall y.P(y)$ *je valjana.*
Time je tačna i sledeća interpretacija: ,,Postoji čovek, takav da ako on pije, onda svi piju".
Zaista, ako postoji čovek koji ne pije, on je taj traženi jer je u tom slučaju premisa netačna. Sa druge strane, ako svi piju, onda je konkluzija tačna pa bilo ko može biti traženi čovek (pretpostavka o nepraznosti domena je ključna kako bi se izbegao slučaj da nijedan čovek ne postoji).

Zadovoljivost

- Važno: definicije zadovoljivosti se značajno razlikuju po literaturi!
- U slučaju rečenica, stvar je jednostavna.

Definicija

Rečenica je zadovoljiva akko postoji interpretacija u kojoj je tačna.

- Problem nastaje prilikom definicije zadovoljivosti za formule sa slobodnim promenljima.
- Neke knjige izbegavaju definiciju zadovoljivosti za formule sa slobodnim promenljivim.
- Negde se (suštinski) daje naredna definicija:

Definicija

Formula je zadovoljiva akko postoji interpretacija u kojoj je tačna.

pri čemu je potrebno precizirati šta znači da je formula sa

Zadovoljenost (tačnost pri fiksiranoj interpretaciji).

Definicija

Formula F je zadovoljena interpretacijom $\mathfrak{D}$ akko postoji valuacija v tako da je formula tačna u valuaciji v pri interpretaciji $\mathfrak{D}$. Formula je zadovoljiva akko postoji interpretacija koja je zadovoljava.

Definicija

Formula F zadovoljena interpretacijom $\mathfrak{D}$ akko je za svaku valuaciju v formula tačna u valuaciji v pri interpretaciji $\mathfrak{D}$. Formula je zadovoljiva akko postoji interpretacija koja je zadovoljava.

Zadovoljivost (tačnost pri fiksiranoj interpretaciji)

Stav

U kontekstu prve definicije, formula F koja ima slobodne promenljive $x_1, \ldots x_n$ je zadovoljiva akko je zadovoljivo njeno egzistencijalno $\exists x_1, \ldots, x_n.\ F$.

Stav

U kontekstu druge definicije, formula F koja ima slobodne promenljive $x_1, \ldots x_n$ je zadovoljiva akko je zadovoljivo njeno univerzalno zatvorenje $\forall x_1, \ldots, x_n.\ F$.

Zadovoljenost (tačnost pri fiksiranoj interpretaciji)

Stav

Rečenica F je valjana akko je $\neg F$ nezadovoljiva.

U svetlu druge definicije, prethodni stav ne mora važi za formule koje nisu rečenice!

Primer

Formula $P(x) \vee P(y)$ nije valjana (nije tačna u onoj valuaciji koja promenljivim x i y dodeljuje isti objekat koji nema svojstvo P) dok je njena negacija $\neg P(x) \wedge P(y)$ nezadovoljiva (nije tačna u valuacijama koje dodeljuju istu vrednost promenljivim x i y).

Logičke posledice

Definicija

Formula F je logička posledica skupa formula Γ (što označavamo sa $\Gamma \models F$) akko za svaku interpretaciju $\mathfrak{D}$ važi da ako $\mathfrak{D}$ zadovoljava svaku formulu iz Γ, onda zadovoljava i F (tj. ako je svaki model za skup Γ istovremeno i model za formulu F).

Primer

- *Rečenica $smrtan(Sokrat)$ je logička posledica rečenica $čovek(Sokrat)$ i $\forall x.\ čovek(x) \Rightarrow smrtan(x)$.*

Logičke posledice

Stav

Ako su $G_1, \ldots, G_n$ i F rečenice, onda $\{G_1, \ldots, G_n\} \models F$ akko je $G_1 \wedge \ldots \wedge G_n \Rightarrow F$ valjana.

U svetlu druge definicije, prethodni stav ne mora da važi ako nisu u pitanju rečenice!

Primer

Važi $P(x) \models P(a)$, ali formula $P(x) \Rightarrow P(a)$ nije valjana. Naime, ako $\mathfrak{D}$ onda za svaku valuaciju v, $P(x)$ važi. Samim tim, važi i za onu valuaciju koja x slika u onaj objekat koji je interpretacija konstante a. Sa druge strane, $P(x) \Rightarrow P(a)$ nije valjana jer je netačna u strukturi i valuaciji koja x-u dodeljuje različitu vrednost od objekta kojim je interpretirana konstanta a, pri čemu P ne važi za tu vrednost, dok važi za objekat kojim je interpretirana konstanta a.

Logička ekvivalentnost

Definicija

Formule F i G su logički ekvivalentne akko svaka interpretacija koja zadovoljava F zadovoljava i G i obratno.

Zamena

Definicija (Zamena promenljive u termu)

Term $t[x \to t']$ je zamena promenljive x u termu t termom t' ako je dobijen tako što se na mesto svakog pojavljivanja promenljive x u termu t postavi term t'.

Zamena promenljive u termu — implementacija

```
primrec tsubst x t' t where
  "tsubst x t' (Var x')  =
       (if x = x' then t' else t)"
| "tsubst x t' (Fn f args) =
       Fn f (map (tsubst x t') args)"
```

Zamena promenljive u termu — svojstva

Stav

Neka je v' valuacija dobijena od v tako što se promenljivoj x dodeljuje vrednost terma t' u valuaciji v, tj. $v' = v(x := \mathfrak{D}_v(t'))$. Onda je

$$\mathfrak{D}_v(t[x \to t']) = \mathfrak{D}_{v'}(t)$$

Zamena promenljive u formuli

- Postojanje slobodnih i vezanih promenljivih čini zamenu promenljive u okviru formule komplikovanijom operacijom.

Primer

- *Zamena promenljive x termom t ne bi trebalo da promeni formulu $\forall x.\ x = x$.*
- *Zamena promenljive x promenljivom y u formuli $\exists\ y.\ y + 1 = x$ je moguća tek nakon preimenovanja (tzv. α konverzije) vezane promenljive y, tj. vrednost zamene može biti formula $\exists y'.\ y' + 1 = y$. Direktna zamena dovodi do $\exists y.\ y + 1 = y$, što nije ono što se želi.*

Zamena promenljive u formuli — implementacija

```
primrec subst where
  subst x t False = False | subst x t True = True
| subst x t (Atom (R r args)) = Atom (R r (map (tsubst x t) args))
| subst x t (Not F) = Not (subst x t F)
| subst x t (F1 And F2) = (subst x t F1) And (subst x t F2)
| subst x t (F1 Or F2)  = (subst x t F1) Or (subst x t F2)
| subst x t (F1 Imp F2) = (subst x t F1) Imp (subst x t F2)
| subst x t (F1 Iff F2) = (subst x t F1) Iff (subst x t F2)
| subst x t (ALL a F) = substquant x t ALL a F
| subst x t (EX a F) = substquant x t EX a F

definition substquant where
"substquant x t Q a F ==
   if x = a then Q a F
   else let a' = fresh_var x t a F in
      Q a' (subst x t (subst a a' F))"
```

Zamena promenljive u formuli — primeri

Primer

- $(\forall x.\ P(x))[x \to y] = \forall x.P(x)$
- $(\forall z.\ P(x,z))[x \to y] = \forall z'.P(y,z')$ – *preimenovanje nije neophodno u ovom primeru*
- $(\forall y.\ P(x,y))[x \to y] = \forall y'.P(y,y')$ – *preimenovanje jeste neophodno u ovom primeru*

Broj preimenovanja se može smanjiti. Jedan od kriterijuma za ispitivanje da li je neophodno vršiti preimenovanje je da li term t sadrži promenljivu a.

Zamena promenljive u formuli — svojstva

Stav

Neka je v' valuacija dobijena od v tako što se promenljivoj x dodeljuje vrednost terma t u valuaciji v, tj. $v' = v(x := \mathfrak{D}_v(t))$. Onda je

$$(D, v) \vDash F[x \to t] \ akko \ (D, v') \vDash F$$

Zamene formule u formuli

Pored zamene promenljive u formuli, moguće je definisati i zamenu potformule u okviru formule.

Primer

Rezultat zamene formule $\neg(P(x) \vee Q(x))$ *formulom* $\neg P(x) \wedge \neg Q(x)$ *u formuli* $\forall x.\ \neg(P(x) \vee Q(x)) \wedge R(x)$ *je formula* $\forall x.\ (\neg P(x) \wedge \neg Q(x)) \wedge R(x)$.

Zamene iskaznih slova formulama logike prvog reda

Takođe, moguće je (čisto sintaksno) definisati i zamene iskaznih slova u iskaznim formulama

Primer

Rezultat zamene P *sa* $p(x)$ *i* Q *sa* $\forall y.\ y > 0$ *u formuli* $P \vee Q$ *je* $p(x) \vee (\forall y.\ y > 0)$.

Pregled

1 Uvod.

2 Sintaksa i semantika logike prvog reda

3 Normalne forme

4 Erbranova teorema.

5 Rezolucija

Negaciona normalna forma

Negaciona normalna forma se definiše po uzoru na iskaznu logiku.

Definicija

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

Prenex normalna forma

Definicija

Formula je prenex normalnoj formi ako je oblika

$$Q_1\ x_1.\ \ldots\ Q_n\ x_n.\ F,$$

pri čemu su Q_i kvantifikatori $\forall$ ili $\exists$, a formula F ne sadrži kvantifikatore.

Formula koja je u NNF se može prevesti u prenex normalnu formu primenama sledećih ekvivalencija:

$$
\begin{array}{llll}
(\forall x.A) \wedge B & \equiv & (\forall x.\ A \wedge B) & \qquad (\exists x.A) \wedge B \equiv (\exists x.\ A \wedge B) \\
(\forall x.A) \vee B & \equiv & (\forall x.\ A \vee B) & \qquad (\exists x.A) \vee B \equiv (\exists x.\ A \vee B) \\
B \wedge (\forall x.A) & \equiv & (\forall x.\ B \wedge A) & \qquad B \wedge (\exists x.A) \equiv (\exists x.\ B \wedge A) \\
B \vee (\forall x.A) & \equiv & (\forall x.\ B \vee A) & \qquad B \vee (\exists x.A) \equiv (\exists x.\ B \vee A)
\end{array}
$$

pri čemu, ako se promenljiva x javlja slobodna u B, potrebno je izvršiti njeno preimenovanje (α-konverziju) u formuli $\forall x.\ A$.

Skraćenice

U nekim slučajevima, moguće je koristiti i naredna pravila koja smanjuju broj kvantifikatora.

$$(\exists x.\ A) \vee (\exists x.B) \equiv (\exists x.A \vee B)$$
$$(\forall x.\ A) \wedge (\forall x.B) \equiv (\forall x.A \wedge B)$$

Obratiti pažnju da naredna pravila ne predstavljaju ekvivalencije i ne mogu se koristiti za prevođenje u prenex normalnu formu.

$$(\exists x.\ A) \wedge (\exists x.B) \equiv (\exists x.A \wedge B)$$
$$(\forall x.\ A) \vee (\forall x.B) \equiv (\forall x.A \vee B)$$

Teorema Prenex

Teorema

Za svaku formulu postoji formula koja je u prenex normalnoj formi i koja joj je ekvivalentna.

Skolemizacija

Definicija

Formula je u Skolemovoj normalnoj formi ukoliko je oblika

$$\forall x_1. \ \ldots \ \forall x_n. \ F,$$

Postupak svođenja u Skolemovu normalnu formu naziva se skolemizacija.
Obično (mada ne i obavezno) se skolemizacija na formule koje su u prenex normalnoj formi.

Ekvizadovoljivost pri skolemizaciji

- Osnovni korak skolemizacije je uklanjanje egzistencijalnih kvantifikatora.
- Egzistencijalni kvantifikatori se ne mogu ukloniti, a da se zadrži ekvivalentnost.
- Ipak, moguće je ukloniti egzistencijalne kvantifikatore, i zadržati zadovoljivost, što je za većinu primena dovoljno.

Skolemizacija — primeri

Primer

Posmatrajmo formulu $\exists x.\ p(x)$ jezika $\mathcal{L} = \{p\}$ i formulu $p(c)$, gde je c novi simbol konstante (ne pripada jeziku $\mathcal{L}$). Neka je $\mathcal{L}' = \mathcal{L} \cup \{c\}$.

- *Ukoliko je rečenica $\exists x.\ p(x)$ zadovoljiva, postoji $\mathcal{L}$-struktura $\mathfrak{D}$ takva da $\mathfrak{D} \models \exists x.\ p(x)$. To znači da u domenu D postoji element $\hat{x}$ takav važi $p_{\mathfrak{D}}(\hat{x})$. Ukoliko posmatramo $\mathcal{L}'$-strukturu $\mathfrak{D}'$ dobijenu od $\mathfrak{D}$ dodatnim interpretiranjem simbola c elementom $\hat{x}$ (tj. $c_{\mathfrak{D}'} = \hat{x}$), važi $\mathfrak{D}' \models p(c)$, te je i rečenica $p(c)$ zadovoljiva.*
- *Ukoliko je $\mathfrak{D}'$ $\mathcal{L}'$-struktura takva da važi $\mathfrak{D}' \models p(c)$, tada važi $p(c_{\mathfrak{D}'})$, te važi da $\mathfrak{D}' \models \exists x.\ p(x)$.*

Skolemizacija — primeri

Primer

Posmatrajmo formulu $\forall x.\ \exists y.\ p(x,y)$ *jezika* $\mathcal{L} = \{p\}$ *i formulu* $\forall x.\ p(x, f(x))$, *gde je* f *novi funkcijski simbol (ne pripada jeziku* $\mathcal{L}$*). Neka je* $\mathcal{L}' = \mathcal{L} \cup \{f\}$.

- *Ukoliko je rečenica* $\forall x.\ \exists y.\ p(x,y)$ *zadovoljiva, postoji* $\mathcal{L}$*-struktura* $\mathfrak{D}$ *takva da za svaki element* $\hat{x} \in D$ *postoji element* $\hat{y} \in D$ *takav važi* $p_{\mathfrak{D}}(\hat{x}, \hat{y})$. *Dakle (pod pretpostavkom aksiome izbora), postoji funkcija* $\hat{f} : D \to D$ *koja elementima* $\hat{x}$ *dodeljuje odgovarajuće* $\hat{y}$. *Ukoliko posmatramo* $\mathcal{L}'$*-strukturu* $\mathfrak{D}'$ *dobijenu od* $\mathfrak{D}$ *dodatnim interpretiranjem simbola* f *funkcijom* $\hat{f}$ *(tj.* $f_{\mathfrak{D}'} = \hat{f}$*), važi* $\mathfrak{D}' \vDash \forall x.p(x, f(x))$, *te je ova rečenica zadovoljiva.*
- *Ukoliko je* $\mathfrak{D}'$ $\mathcal{L}'$*-struktura takva da važi* $\mathfrak{D}' \vDash \forall x.\ p(x, f(x))$, *tada za svaki element* $\hat{x} \in D$ *važi* $p(\hat{x}, f_{\mathfrak{D}'}(\hat{x}))$, *te važi da* $\mathfrak{D}' \vDash \forall x.\ \exists y.\ p(x,y)$.

Postupak skolemizacije

Procedura skolemizacije se zasniva na uzastopnoj primeni sledećih transformacija:

- Ako je formula oblika $\exists x.A$, bira se novi simbol konstante c (koji ne pripada jeziku $\mathcal{L}$) i formula se zamenjuje formulom $A[x \to c]$.
- Ako je formula oblika $\forall x_1 \ldots \forall x_n.\exists x.\ A$, bira se novi funkcijski simbol f (koji ne pripada jeziku $\mathcal{L}$ i formula se zamenjuje formulom $A[x \to f(x_1, \ldots, x_n)]$.

Skolemizacija i logika višeg reda

Naredna ekvivalencija važi u logici drugog reda (u logici prvog reda nije dopuštena kvantifikacija nad funkcijama):

$$\forall x.\ \exists y.\ p(x, y) \equiv \exists f.\ \forall x.\ p(x, f(x))$$

Opštije

$$\forall x_1 \ldots \forall x_n.\ \exists y.\ p(x_1, \ldots, x_n, y) \equiv \exists f.\ \forall x_1 \ldots \forall\, x_n.\ p(x_1, \ldots, x_n, f(x))$$

Teorema o skolemizaciji

Teorema

Formula dobijena skolemizacijom formule F je ekvizadovoljiva formuli F.

Dokaz

Teorema se dokazuje na način sličan opisanom u prethodna dva primera.

Oslobađanje univerzalnih kvantifikatora?

- Da li je moguće ukloniti univerzalne kvantifikatore tako da formula ostane ekvizadovoljiva?
- Zavisi od definicije zadovoljive formule.
- Ukoliko se koristi prva definicija (postoji interpretacija i postoji valuacija) nije moguće osloboditi se univerzalnih kvantifikatora.
- Ukoliko se koristi druga definicija (postoji interpretacija tako da je za svaku valuaciju) univerzalni kvantifikatori se mogu samo izostaviti i dobija se ekvizadovoljiva formula.

Formule bez kvantifikatora

Definicija

Formula je formula bez kvantifikatora (eng. quantifier free) ukoliko je oblika

$$\forall x_1. \ \ldots \ \forall x_n. \ F.$$

Često se, po konvenciji, za ove formule navodi samo matrica F, dok se blok kvantifikatora izostavlja — ovo je naročito opravdano u svetlu druge definicije zadovoljivosti.

Stav

Za svaku formulu postoji formula bez kvantifikatora koja joj je ekvizadovoljiva.

Prethodni stav ne važi za valjanost.

Definicija

- *KNF i DNF formula bez kvantifikatora logike prvog reda se definišu analogno iskaznoj logici.*
- *Formula je u klauzalnoj formi ukoliko je oblika*

$$\forall x_1. \ \ldots \ \forall x_n. \ F,$$

pri čemu je F u KNF.

Teorema o klauzalnoj formi

Teorema

Za svaku formulu postoji formula u klauzalnoj formi koja joj je ekvizadovoljiva.

Dokaz

Na formulu se primeni NNF, Prenex, Skolemizacija i konverzija matrice u KNF (npr. Cajtinovom transformacijom).

Dokazivanje pobijanjem

- Jedan od pristupa dokazivanja valjanosti neke rečenice je dokazivanje pobijanjem koje podrazumeva dokazivanje nezadovoljivosti negacije polazne rečenice.
- Klauzalna forma se najčešće koristi u svetlu dokazivanja pobijanjem.

Pregled

1 Uvod.

2 Sintaksa i semantika logike prvog reda

3 Normalne forme

4 Erbranova teorema.

5 Rezolucija

Erbranova teorema

- Erbran (fr. Herbrand) 1930. — tragovi u ranijim radovima Skolema i Gedela
- U logici prvog reda nije odlučivo (ne postoji opšti postupak) da li je neka formula valjanja (Entscheidungsproblem, postavio Hilbert, negativno razrešili Čerč i Tjuring).
- Međutim, Erbranova teorema daje jedan od načina da se pokaže poluodlučivost: postoji algoritam koji za svaku valjanu formulu može da pokaže da je valjana.
- Erbranova teorema uspostavlja veze između logike prvog reda i iskazne logike.

Iskazni pogled na formule bez kvantifikatora

Formule bez kvantifikatora se mogu tretirati kao iskazne formule (gde se na mesto iskaznih slova koriste atomi logike prvog reda).

Primer

Posmatrajmo formulu $x > 0 \vee \neg(x > 0)$ *kao iskaznu formulu nad skupom atoma* $\{x > 0\}$. *Iskazne valuacije proglašavaju ovaj atom ili za tačan ili za netačan. U oba slučaja polazna formula je tačna pa je iskazna tautologija. Sa druge strane, formula* $x > 0 \vee \neg(x > 0)$ *je i valjana (u logici prvog reda). Zaista, za svaku valuaciju* v *i svaku interpretaciju* $\mathfrak{D}$ *važi da* $(\mathfrak{D}, v) \vDash x > 0 \vee \neg(x > 0)$. *Valuacija* v *definiše vrednost atomičke formule* $x > 0$, *međutim, kakva god ona bila, na osnovu semantike logike prvog reda, sledi da je formula tačna.*
Formula $p \vee \neg p$ *je iskazna tautologija, a prethodna formula se može dobiti zamenom promenljive u tautologiji. Ovakve formule se nazivaju i izvodima tautologija (ili tautologijama prvog reda).*

Veza između valjanosti i (iskazne) tautologičnosti za formule bez kvantifikatora

Stav

Formula bez kvantifikatora je valjana akko je iskazna tautologija.

Dokaz

Jednostavno se dokazuje da, ako je formula bez kvantifikatora iskazna tautologija, onda je ona valjana (indukcijom po strukturi formule prateći definicije semantike u iskaznoj logici i logici prvog reda).

Dokaz

Drugi smer je komplikovaniji. Pretpostavimo da je formula bez kvantifikatora F valjana i pokažimo da je iskazna tautologija. Fiksirajmo iskaznu valuaciju $\bar{v}$. Pokazaćemo da je moguće izgraditi $\mathcal{L}$-strukturu $\mathcal{D}$ i valuaciju prvog reda v tako da je $\bar{v} \models F$ akko važi $(\mathfrak{D}, v) \models F$. Dovoljno je pokazati da tvrđenje važi za svaki atom $R(t_1, \ldots, t_n)$ formule F, tj. da važi $\bar{v} \models R(t_1, \ldots, t_n)$ akko $(\mathfrak{D}, v) \models R(t_1, \ldots, t_n)$ i nastavak dokaza sledi indukcijom.

Dokaz

Model $\mathfrak{D}$ gradimo od sintaksnog materijala, tj. domen D je skup termova jezika $\mathcal{L}$. Interpretacije simbola konstanti su oni sami (tj. $c_{\mathfrak{D}} = c$), a interpretacije funkcijskih simbola se grade tako da je $f_{\mathfrak{D}}(t_1, \ldots t_n) = f(t_1, \ldots, t_n)$. Vrednost svakog terma u ovako definisanoj valuaciji je on sâm (tj. $\mathfrak{D}_v(t) = t$). Dakle, različiti termovi imaju obavezno različite vrednosti. Sada je moguće definisati interpretaciju svakog relacijskog simbola R tako što je $R_{\mathfrak{D}}(t_1, \ldots, t_n)$ tačno akko $\bar{v} \vDash R(t_1, \ldots, t_n)$.
Neka valuacija v dodeljuje svakoj promenljivoj nju samu (tj. $v(x) = x$).
Ovako definisani $\mathfrak{D}$ i v zadovoljavaju traženo svojstvo. Zaista $(\mathfrak{D}, v) \vDash R(t_1, \ldots, t_n)$ akko $R_{\mathfrak{D}}(\mathfrak{D}_v(t_1), \ldots, \mathfrak{D}_v(t_n))$ akko $R_{\mathfrak{D}}(t_1, \ldots, t_n)$ akko $\bar{v} \vDash R(t_1, \ldots, t_n)$.
Tvrđenje dalje jednostavno sledi indukcijom po strukturi formule F (prateći definicije semantike u iskaznoj logici i logici prvog reda).

Kanonski modeli

- Termovski modeli izgrađeni na način kao u prethodnom dokazu se veoma često koristi.
- Ovakvi modeli nazivaju se *kanonski modeli (kanonske interpretacije)*.

Veza između zadovoljivosti i iskazne zadovoljivosti formula bez kvantifikatora

- Kod formula bez kvantifikatora, valjanosti u logici prvog odgovara iskazna tautologičnost, međutim, važnije pitanje je da li postoji slična veza za zadovoljivost (svakoj formuli možemo pridružiti ekvizadovoljivu formulu bez kvantifikatora, a ne ekvivaljanu).
- Za rečenice, tj. bazne formule (formule bez promenljivih), odnos je jednostavan.

Stav

Bazna formula bez kvantifikatora je zadovoljiva akko je iskazno zadovoljiva.

- Za formule sa slobodnim (univerzalno kvantifikovanim) promenljivim, odnos nije tako jednostavan.

Veza između zadovoljivosti i iskazne zadovoljivosti formula bez kvantifikatora

- Jedan smer je i dalje trivijalan.

Stav

Ako je formula bez kvantifikatora zadovoljiva, ona je i iskazno zadovoljiva.

Dokaz

Ako F ne bi bila iskazno zadovoljiva, onda bi $\neg F$ bila tautologija i zato bi bila valjana formula pa F ne bi mogla biti zadovoljiva.

Veza između zadovoljivosti i iskazne zadovoljivosti formula bez kvantifikatora

- Drugi smer ne važi.

Primer

Formula $P(x) \wedge \neg P(y)$ *je iskazno zadovoljiva u valuaciji u kojoj je* $P(x)$ *tačno, a* $P(y)$ *netačno. Međutim formula* $\forall x.\ \forall y.P(x) \wedge \neg P(y)$ *nije zadovoljiva formula logike prvog reda.*

Veza između zadovoljivosti i iskazne zadovoljivosti formula bez kvantifikatora

- Formule bez kvantifikatora su u suštini univerzalno kvantifikovane.
- Univerzalno kvantifikovane formule predstavljaju suštinski ,,beskonačne konjunkcije" baznih formula.
- Npr. $\forall x.\ p_1(x, c_1) \vee p_2(f(x), c_2)$ predstavlja konjunkciju

  $(p_1(a_1, c_1) \vee p_2(f(a_1), c_2)) \wedge (p_1(a_2, c_1) \vee p_2(f(a_2), c_2)) \wedge \ldots,$

  gde a_1, a_2, ... predstavljaju različite elemente domena.

Veza između zadovoljivosti i iskazne zadovoljivosti formula bez kvantifikatora

- Cilj je ispitati (ne)zadovoljivost formule.
- Ispostaviće se da je u tu svrhu, umesto proizvoljnih domena, dovoljno posmatrati (u neku ruku najopštije) kanonske modele.
- U prethodnom primeru, elementi kanonskog modela su identifikovani termovima c_1, c_2, $f(c_1)$, $f(c_2)$, $f(f(c_1))$, $f(f(c_2))$, ...
- Formula $\forall x.\ p_1(x, c_1) \vee p_2(f(x), c_2)$ predstavlja konjunkciju

$$\begin{aligned} (p_1(c_1, c_1) \vee p_2(f(c_1), c_2)) \quad & \wedge \\ (p_1(c_2, c_1) \vee p_2(f(c_2), c_2)) \quad & \wedge \\ (p_1(f(c_1), c_1) \vee p_2(f(f(c_1)), c_2)) \quad & \wedge \quad \ldots \end{aligned}$$

Erbranov univerzum

- Domen kanonskih interpretacija je bilo koji skup termova koji sadrži sve konstante jezika $\mathcal{L}$ i zatvoren je za sve funkcijske simbole jezika $\mathcal{L}$.
- Iz praktičnih razloga, poželjno je koristiti što manje domene.
- Pokazaće se da je za ispitivanje zadovoljivosti dovoljno posmatrati skup svih baznih termova jezika $\mathcal{L}$.

Erbranov univerzum

- Skup svih baznih termova jezika $\mathcal{L}$ nazivamo Erbranov univerzum i označavamo sa $H(\mathcal{L})$.
- Ukoliko jezik ne sadrži ni jednu konstantu, u njega se veštački umeće novi simbol konstante (npr. c) kako Erbranov unverzum ne bi bio prazan.

Primer

Ako je jezik $\mathcal{L} = \{c, f\}$, *onda je Erbranov univerzum* $H = \{c, f(c), f(f(c)), \ldots\}$. *Ako je jezik* $\mathcal{L} = \{+, \cdot\}$, *onda je Erbranov univerzum*
$\{c, c + c, c \cdot c, c + c + c, c + c \cdot c, c \cdot c + c, c \cdot c \cdot c, \ldots\}$.

Erbranov univerzum formule i njene bazne instance

- Erbranov univerzum formule $H(F)$ je Erbranov univerzum jezika sačinjenog od simbola koji se javljaju u toj formuli.
- Bazne instance formule, dobijaju se kao rezultat zamene promenljivih elementima njenog Erbranovog univerzuma.

Veza između zadovoljivosti i iskazne zadovoljivosti formula bez kvantifikatora

Ključni kriterijum kojim se uspostavlja veza između zadovoljivosti u logici prvog reda i iskazne zadovoljivosti za formule bez kvantifikatora dat je narednom teoremom:

Teorema (Erbran)

Formula bez kvantifikatora F je zadovoljiva akko je skup svih njenih baznih instanci iskazno zadovoljiv. Preciznije, formula F oblika $\forall x_1 \ \ldots \ x_n.\phi(x_1, \ldots, x_n)$ je zadovoljiva akko je zadovoljiv skup $\{\phi[x_1 \to t_1] \ldots [x_n \to t_n] \mid t_1, \ldots, t_n \in H(F)\}$.

Dokaz

Radi pojednostavljivanja zapisa, bazne instance $\phi[x_1 \to t_1] \ldots [x_n \to t_n]$ *ćemo kraće označavati sa* $F[x_i \to t_i]$. *Ako je* F *zadovoljiva ona važi u nekom modelu* $\mathfrak{D}$ *i u svakoj valuaciji* v. *Neka je iskazna valuacija* $\bar{v}$ *takva da za svaki atom* a *važi* $\bar{v} \vDash a$ *akko* $(\mathfrak{D}, v) \vDash a$. *Pokažimo da* $\bar{v}$ *zadovoljava sve bazne instance formule* F.
Neka je i proizvoljna instancijacija, tj. preslikavanje promenljivih x_i *u bazne termove* t_i *iz* H. *Posmatrajmo baznu instancu* $F[x_i \to t_i]$. *Na osnovu definicije* $\bar{v}$, *važi da* $\bar{v} \vDash F[x_i \to t_i]$ *akko* $(\mathfrak{D}, v) \vDash F[x_i \to t_i]$. *Neka je* v' *valuacija dobijena od* v *tako što se svakoj promenljivoj* x_i *dodeljuje vrednost terma* t_i *u valuaciji* v. *Tada* $(\mathfrak{D}, v) \vDash F[x_i \to t_i]$ *akko* $(\mathfrak{D}, v') \vDash F$. *Međutim, ovo je tačno jer je* $\mathfrak{D}$ *model za* F, *a vrednost* F *ne zavisi od valuacije.*

Dokaz

Obratno, neka valuacija $\bar{v}$ zadovoljava sve bazne instance formule F. Neka je $\mathfrak{D}$ kanonska interpretacija nad Erbranovim univerzumom određena valuacijom $\bar{v}$ i neka je v proizvoljna valuacija prvog reda koja preslikava promenljive u elemente Erbranovog univerzuma. Potrebno je pokazati da $(\mathfrak{D}, v) \models F$. Pokažimo da važi $(\mathfrak{D}, v) \models F$ akko $\bar{v} \models F[x_i \to v(x_i)]$ (tvrđenje odatle sledi jer je $F[x_i \to v(x_i)]$ bazna instanca koja je tačna u $\bar{v}$). Dovoljno je dokazati tvrđenje za atomičke formule $R(t_1, \ldots, t_n)$ (dokaz dalje sledi indukcijom po strukturi formule F). Važi:
$(\mathfrak{D}, v) \models R(t_1, \ldots, t_n)$ *akko* $R_{\mathfrak{D}}(\mathfrak{D}_v(t_1), \ldots, \mathfrak{D}_v(t_n))$ *akko*
$R_{\mathfrak{D}}(t_1[x_i \to v(x_i)], \ldots, t_n[x_i \to v(x_i)])$ *akko*
$\bar{v} \models R(t_1[x_i \to v(x_i)], \ldots, t_n[x_i \to v(x_i)])$ *akko*
$\bar{v} \models R(t_1, \ldots, t_n)[x_i \to v(x_i)]$.

Teorema (Erbran)

Erbranova interpetacija $\mathfrak{D}$ zadovoljava formulu bez kvantifikatora F akko zadovoljava sve njene bazne instance.

Dokaz

Pretpostavimo da $\mathfrak{D}$ zadovoljava F. Neka je i
Neka je i proizvoljna instancijacija, tj. preslikavanje promenljivih x_i u bazne termove t_i iz H. Posmatrajmo baznu instancu $F[x_i \to t_i]$. Neka je v' valuacija dobijena od v tako što se svakoj promenljivoj x_i dodeljuje vrednost terma t_i u valuaciji v. Tada $(\mathfrak{D}, v) \models F[x_i \to t_i]$ akko $(\mathfrak{D}, v') \models F$. Međutim, ovo je tačno jer je $\mathfrak{D}$ model za F, a vrednost F ne zavisi od valuacije.

Dokaz

Obratno, neka $\mathfrak{D}$ zadovoljava sve bazne instance. Neka je v proizvoljna valuacija koja preslikava promenljive x_v u termove t_v Erbranovog univerzuma. Neka je v' valuacija koja svakoj promenljivoj x_v dodeljuje vrednost terma t_v u $\mathfrak{D}$, tj. $\mathfrak{D}_v(t_v)$. Međutim, pošto je $\mathfrak{D}$ kanonska, onda je $\mathfrak{D}_v(t_v) = t_v$ pa je $v' = v$. Dakle, važi $(\mathfrak{D}, v) \models F$ akko $(\mathfrak{D}, v') \models F$ akko $(\mathfrak{D}, v) \models F[x_v \to t_v]$. Tvrđenje dalje sledi jer je $F[x_v \to t_v]$ bazna instanca i tačna je u $\mathfrak{D}$.

Mehanizacija Erbranove teoreme

- Ispitivanje zadovoljivosti se svodi na ispitivanje zadovoljivosti skupa svih baznih instanci.
- Ovaj skup može biti beskonačan.
- Ipak, kompakntost obezbeđuje potrebnu konačnost

Teorema

Formula bez kvantifikatora je nezadovoljiva akko postoji konačan nezadovoljiv podskup njenih baznih instanci.

- Procedure Erbranovog tipa nabrajaju skupove baznih instanci dodajući nove instance.
  - Ukoliko se utvrdi iskazna nezadovoljivost tekućeg skupa instanci, polazna formula je nezadovoljiva.
  - Ukoliko se iscrpe sve bazne instance, a nezadovoljivost se ne utvrdi, polazna formula je zadovoljiva.
  - Moguće je da procedura beskonačno dodaje nove bazne instance (u slučaju da je polazna formula zadovoljiva) i da se ne zaustavlja.
- Najčuvenija je Gilmorova procedura (prva implementirana) koja instance prevodi u DNF i tako ispituje njihovu zadovoljivost.

Erbranova teorema — primeri

Primer

Pokažimo da je rečenica $\forall x.\ \exists y.P(x,y) \Rightarrow \exists y.\ \forall x.\ P(x,y)$ *valjana. Pobijanjem, potrebno je pokazati da je njena negacija nezadovoljiva. Nakon NNF transformacije, dobija se formula* $\forall x.\ \exists y.\ P(x,y) \wedge \forall y.\ \exists x.\neg P(x,y)$. *Prenex trasformacija daje njoj ekvivalentnu formulu* $\forall z.\exists x.\exists y.\ P(z,y) \wedge \neg P(x,z)$. *Skolemizacijom se dobija formula bez kvantifikatora* $\forall z.\ P(z,c_1) \wedge \neg P(c_2,z)$ *koja je zadovoljiva akko je polazna formula valjana.*
Na osnovu Erbranove teoreme, prethodna formula je zadovoljiva akko je iskazno zadovoljiva formula

$$P(c_1,c_1) \wedge \neg P(c_2,c_1) \wedge P(c_2,c_1) \wedge P(c_2,c_2),$$

što nije slučaj.

Primer

Paradoks berberina: Nije moguće da postoji berberin koji brije sve one koji sami sebe ne briju:

$$\neg(\exists b.\ \forall x.\ brije(b, x) \Leftrightarrow \neg brije(x, x))$$

Da bi se pokazalo da je prethodna formula valjana, pokažimo da je njena negacija nezadovoljiva. Nakon svođenja u normalnu formu, dobija se:

$$\forall x.\ (\neg brije(c, x) \vee \neg brije(x, x)) \wedge (brije(c, x) \vee brije(x, x))$$

Jedina Erbranova instanca ove formule je:

$$(\neg brije(c, c) \vee \neg brije(c, c)) \wedge (brije(c, c) \vee brije(c, c)),$$

koja je očigledno iskazno nezadovoljiva.

Erbranova teorema – primeri

Primer

Pokažimo da je formula $\exists x.\ P(x) \Rightarrow \forall y.\ P(y)$ *valjana. Primenimo metodu pobijanja i transformacije normalnih formi.*

$$\neg(\exists x.\ P(x) \Rightarrow \forall y.\ P(y))$$

$$\forall x.\ P(x) \wedge (\exists y.\ \neg P(y))$$

$$\forall x.\ \exists y.\ (P(x) \wedge \neg P(y))$$

$$\forall x.\ P(x) \wedge \neg P(f(x))$$

Konjunkcija naredne dve instance je iskazno nezadovoljiva:

$$(P(c) \wedge \neg P(f(c))) \wedge (P(f(c)) \wedge \neg P(f(f(c))))$$

Erbranova teorema – primeri

Primer

Dokažimo da je formula $\forall x.\ \exists y.\ q(x,y)$ *logička posledica formula* $\forall x.\ \exists y.p(x,y)$ *i* $\forall x.\ \forall y.\ p(x,y) \Rightarrow q(x,y)$. *Potrebno je dokazati da je*

$$((\forall x.\ \exists y.p(x,y)) \wedge (\forall x.\ \forall y.\ p(x,y) \Rightarrow q(x,y))) \Rightarrow \forall x.\ \exists y.\ q(x,y)$$

valjana formula. NNF negacije prethodne formule je:

$$(\forall x.\ \exists y.\ p(x,y)) \wedge (\forall x.\ \forall y.\ \neg p(x,y) \vee q(x,y)) \wedge (\exists x.\ \forall y.\ \neg q(x,y))$$

Prenex transformacija daje:

$$\exists x.\ \forall z.\ \exists y.\ \forall w.\ p(z,y) \wedge (\neg p(z,w) \vee q(z,w)) \wedge \neg q(x,z)$$

Nakon skolemizacije, dobija se

$$\forall z.\ \forall w.\ p(z,f(z)) \wedge (\neg p(z,w) \vee q(z,w)) \wedge \neg q(c,z)$$

Erbranova teorema – primeri

Primer

Instance

$$p(c, f(c)) \wedge (\neg p(c, f(c)) \vee q(c, f(c))) \wedge \neg q(c, c)$$

i

$$p(f(c), f(f(c))) \wedge (\neg p(f(c), c) \vee q(f(c), c)) \wedge \neg q(c, f(c))$$

su zajedno nezadovoljive.

Primer

Primetimo, da je nezadovoljivost bilo moguće detektovati i jednostavnije da prilikom prenex transformacije nije vršena ,,ušteda" broja kvantifikatora.

$$\exists x.\ \forall x'.\ \exists y.\ \forall x''.\ \forall y'.\ \forall y''.\ p(x', y) \wedge (\neg p(x'', y') \vee q(x'', y')) \wedge \neg q(x, y''),$$

tj., nakon Skolemizacije

$$\forall x'.\ \forall x''.\ \forall y'.\ \forall y''.\ p(x', f(x')) \wedge (\neg p(x'', y') \vee q(x'', y')) \wedge \neg q(c, y'')$$

Tada bi postojala jedna jedina Erbranova instanca koja je nezadovoljiva:

$$p(c, f(c)) \wedge (\neg p(c, f(c)) \vee q(c, f(c))) \wedge \neg q(c, f(c)))$$

Erbranova teorema — zasebno instanciranje klauza

Prethodno zapažanje važi i generalno. Naime, uvođenjem novih univerzalnih kvantifikatora, moguće je postići da su slobodne promenljive u svim klauzama razdvojene (npr. $\forall x.P(x) \wedge Q(x)$ je ekvivalentno sa $\forall x'.\forall x''.P(x') \wedge P(x'')$.

Teorema

Neka je $F \equiv \forall x_1 \ldots x_n.C_1(x_1, \ldots, x_n) \wedge \ldots C_k(x_1, \ldots, x_n)$ *formula u klauzalnoj normalnoj formi. Formula je zadovoljiva ako i samo ako je zadovoljiv skup svih baznih instanci pojedinačnih klauza iskazno zadovoljiv, tj. ako je zadovoljiv skup* $\{C_i[x_1 \to t_1] \ldots [x_n \to t_n] \mid t_1, \ldots, t_n \in H(F), 1 \leq i \leq k\}$.

Pregled

1 Uvod.

2 Sintaksa i semantika logike prvog reda

3 Normalne forme

4 Erbranova teorema.

5 Rezolucija

- Osnovni problem Erbran/Gilmorovog metoda dokazivanja je bio kako pronaći bazne instance koje daju nezadovoljivost.
- Faza generisanja skupova baznih instanci i faza ispitivanja njihove nezadovoljivosti su potpuno razdvojene.
- Da li je moguće nekako objediniti ove dve faze?

Automatsko rezonovanje – beleške sa predavanja Rezonovanje u logici prvog reda sa jednakošću

Filip Marić

*Matematički fakultet,
Univerzitet u Beogradu

Proletnji semestar 2011.

Pregled

1 Normalni modeli

2 Aksiome jednakosti

3 Birkhofov sistem

4 Kongruentno zatvorenje

5 Prezapisivanje

Poseban tretman jednakosti

- Do sada je simbol $=$ bio tretiran kao bilo koji drugi predikatski simbol.

Primer

Za očekivati je da je formula

$$\forall x\ y\ z.\ x = y \wedge y = z \Rightarrow f(x) = f(z)$$

valjana, međutim, ona to nije. Po do sada izloženom, ova formula se ni po čemu ne razlikuje od formule

$$\forall x\ y\ z.\ p(x, y) \wedge p(y, z) \Rightarrow p(f(x), f(z)),$$

za koju je jasno da nije valjana (npr. netačna je kada se f interpretira identičkom funkcijom, a p nekom netranzitivnom relacijom).

Poseban tretman jednakosti

- Ipak, uloga jednakosti je centralna u matematici tako da je poželjno posmatrati samo one interpretacije u kojima se simbol '=' interpretira upravo relacijom jednakosti.
- Iako se prethodno opisane procedure jednostavno modifikuju tako da u obzir uzmu samo ovakve modele, moguća je i izrada efikasnijih procedura, specijalizovane za rezonovanje u prisustvu jednakosti.

Normalne interpretacije

Definicija

Ako jezik $\mathcal{L}$ sadrži simbol $=$, za $\mathcal{L}$-strukturu (model, interpretaciju) kažemo da je normalna ako se simbol $=$ interpretira relacijom jednakosti odgovarajućeg domena.

Pregled

1 Normalni modeli

2 Aksiome jednakosti

3 Birkhofov sistem

4 Kongruentno zatvorenje

5 Prezapisivanje

Aksiome jednakosti

S obzirom da je relacija jednakosti relacija ekvivalencije, u svim normalnim modelima naredne formule su tačne.

$$\begin{array}{rcl} \text{refleksivnost} & : & \forall x.\ x = x \\ \text{simetričnost} & : & \forall x\ y.\ x = y \Leftrightarrow y = x \\ \text{tranzitivnost} & : & \forall x\ y\ z.\ x = y \wedge y = z \Rightarrow x = z \end{array}$$

Takođe, za svaki n-arni funkcijski simbol f odnosno svaki n-arni predikatski simbol p tačne su i formule kongruentnosti

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

$$\forall x_1 \ldots x_n\ y_1 \ldots y_n.\ x_1 = y_1 \wedge \ldots \wedge x_n = y_n \Rightarrow p(x_1, \ldots, x_n) \Leftrightarrow p(y_1, \ldots, y_n)$$

Navedene formule nazivamo aksiomama jednakosti. Oznaka $eqax(\mathcal{L})$ označava aksiome jednakosti jezika $\mathcal{L}$.

- Svaki normalni model zadovoljava aksiome jednakosti.
- Međutim, postoje modeli koji nisu normalni, a opet zadovoljavaju aksiome jednakosti.

Primer

Ukoliko se jezik $\mathcal{L} = \{+, \cdot, 0, 1, =\}$ *interpretira uobičajeno na skupu prirodnih brojeva, a simbol* $=$ *relacijom* $x \equiv y (mod\ 2)$, *model zadovoljava aksiome jednakosti, a nije normalan.*

Veza između normalnih modela i modela aksioma jednakosti

Stav

Formula F jezika $\mathcal{L}$ ima normalan model akko F i $eqax(\mathcal{L})$ imaju model.

Dokaz

Ako F ima normalan model on je model u kome su i F i $eqax(\mathcal{L})$ tačne.
Obratno, ako postoji model za F i $eqax(\mathcal{L})$ može se definisati relacija $\sim$ na domenu D takva da je $x \sim y$ akko $x =_{\mathcal{M}} y$, tj. kada su x i y jednaki po interpretaciji M. Pošto u M važe $eqax(\mathcal{L})$ relacija $\sim$ je relacija ekvivalencije. Traženi normalni model se može jednostavno izgraditi nad klasama ekvivalencije relacije $\sim$ kao domenom.

Veza između normalnih modela i modela aksioma jednakosti

Stav

- *Formula F je zadovoljiva u nekom normalnom modelu akko je formula $F \wedge eqax(\mathcal{L})$ zadovoljiva.*
- *Formula F važi u svim normalnim modelima akko je formula $eqax(\mathcal{L}) \Rightarrow F$ valjana.*

Smanjenje broja aksioma jednakosti

- Prethodni stavovi ostaju na snazi i ako se broj aksioma jednakosti smanji.
- Npr. aksioma simetričnosti se može dokazati iz aksiome refleksivnosti i kongruentnosti za jednakost.

Pregled

1 Normalni modeli

2 Aksiome jednakosti

3 Birkhofov sistem

4 Kongruentno zatvorenje

5 Prezapisivanje

Sintaksno-deduktivni sistemi za jednakost — Birkhofova pravila

$$\frac{}{\Delta \vdash t = t}\ \text{refl}$$

$$\frac{\Delta \vdash s = t}{\Delta \vdash t = s}\ \text{sym}$$

$$\frac{\Delta \vdash s = t \quad \Delta \vdash t = u}{\Delta \vdash s = u}\ \text{trans}$$

$$\frac{\Delta \vdash s_1 = t_1 \quad \ldots \quad \Delta \vdash s_n = t_n}{\Delta \vdash f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n)}\ \text{cong}$$

$$\frac{s = t \in \Delta}{\Delta \vdash s = t}\ \text{ax}$$

$$\frac{\Delta \vdash s = t}{\Delta \vdash (s = t)[x \to a]}\ \text{inst}$$

Birkofova teorema

Teorema

$\Delta \models s = t$, tj. jednačina $s = t$ važi u svim normalnim modelima skupa jednačina Δ akko i samo ako $\Delta \vdash s = t$, tj. ako se $s = t$ može izvesti iz Δ primenom Birkofovih pravila.

Pregled

1 Normalni modeli

2 Aksiome jednakosti

3 Birkhofov sistem

4 Kongruentno zatvorenje

5 Prezapisivanje

Odlučivanje baznih jednakosti

- Prilikom primene Birkofovih pravila problem predstavlja pravilo instancijacije (nije jasno kako odlučiti koju instancijaciju $x \to t$ upotrebiti).
- Ukoliko su sve jednakosti bazne (nemaju slobodnih promenljivih), onda pravilo instancijacije nije potrebno koristiti, što čini ovaj fragment odlučivim.
- Procedure odlučivanja za bazne jednakosti se obično zasnivaju na kongruentnom zatvorenju.

Kongruentne relacije

Definicija

Relacija $\sim$ je kongruencija na skupu D u odnosu na jezik $\mathcal{L}$ ako je relacija ekvivalencije na domenu D saglasna (kongruentna) sa svim funkcijskim simbolima jezika $\mathcal{L}$, tj. za svako f važi da ako $s_1 \sim t_1$, $\ldots$, $s_n \sim t_n$, tada $f(s_1, \ldots, s_n) \sim f(t_1, \ldots, t_n)$.
Kongruentno zatvorenje relacije je najmanja kongruencija koja sadrži polaznu relaciju.

Teorema o kongruentnom zatvorenju

Teorema

Neka su s_i, t_i, s i t bazni termovi i neka je D skup svih ovih termova i svih njihovih podtermova. Neka je $\sim$ kongruentno zatvorenje relacije $\{(s_1, t_1), \ldots, (s_n, t_n)\}$. Tada

$$\{s_1 = t_1, \ldots, s_n = t_n\} \models s = t$$

akko

$$\{s_1 = t_1, \ldots, s_n = t_n\} \vdash s = t$$

akko

$$s \sim t$$

Teorema o kongruentnom zatvorenju

Dokaz

Prva i druga stavka su ekvivalentne na osnovu Birkhofove teoreme. Iz druge stavke sledi treća . . . . Iz treće stavke sledi druga . . . .

Nelson-Openov algoritam za određivanje kongruentnog zatvorenja

- Postupak se zasniva na postepenom proširenju relacije kongruencije počevši od prazne relacije i dodavanjem jedne po jedne jednakosti.
- Kongruencije se predstavljaju klasama ekvivalencije (korišćenjem *union-find* strukture).
- Prilikom spajanja klasa vrši se analiza termova u kojima elementi tih klasa učestvuje i na osnovu kongruentnosti spajaju se odgovarajući termovi. Ukoliko se npr. klase koje sadrže termovi s i t spajaju, a postoje npr. termovi $f(s, g(t))$ i $f(t, g(s))$, potrebno je spojiti i klase kojima oni pripadaju.

Nelson-Openov algoritam za određivanje kongruentnog zatvorenja

- Neka je $E = \{s_1 = t_1, \ldots, s_n = t_n\}$ skup baznih jednakosti. Neka je T skup termova zatvoren za podtermove koji sadrži sve bazne termove s_i, t_i (i možda još neke druge termove i njihove podtermove).
- Neka $use(t)$ označava skup svih termova skupa D čiji je podterm term t. Ovi skupovi se mogu unapred izračunati.
- Neka $find(t)$ vraća kanonskog predstavnika klase ekvivalencije kojoj pripada term t.
- Neka $union(s, t)$ spaja klasu ekvivalencije terma s i klasu ekvivalencije terma t.
- Neka $cong(s, t)$ označava da je s oblika $f(s_1, \ldots, s_n)$ i t oblika $f(t_1, \ldots, t_n)$ pri čemu je $find(s_i) = find(t_i)$.

Nelson-Openov algoritam za određivanje kongruentnog zatvorenja

```
function cc(E, T)
begin
    foreach t ∈ T  find(t) := t.
    foreach s_i = t_i ∈ E
        merge(s_i, t_i)
end

function merge(s, t)
begin
    T_s = ⋃{use(u) | find(u) = find(s)}
    T_t = ⋃{use(u) | find(u) = find(t)}
    union(s, t)
    foreach s' ∈ T_s, t' ∈ T_t
        if (find(s') ≠ find(t') ∧ cong(s', t'))
        merge(s', t')
end
```

Nelson-Openov algoritam za određivanje kongruentnog zatvorenja — primer

Primer

Odredimo kongruentno zatvorenje za skup jednakosti $E = \{x = y, y = z\}$ *i skup termova* $\{x, y, z, f(x), f(z)\}$.
use je određen sa

$$\{x \mapsto \{f(x)\}, y \mapsto \{\}, z \mapsto \{f(z)\}, f(x) \mapsto \{\}, f(z) \mapsto \{\}\}$$

Krećemo od jednočlanih klasa $\{\{x\}, \{y\}, \{z\}, \{f(x)\}, \{f(z)\}\}$,
tj. find je određen sa:

$$\{x \mapsto x, y \mapsto y, z \mapsto z, f(x) \mapsto f(x), f(z) \mapsto f(z)\}$$

Primer

- *merge*(x, y) – $T_x = \{f(x)\}$, $T_y = \{\}$. *Nakon union*(x, y), *dobijaju se klase* $\{\{x, y\}, \{z\}, \{f(x)\}, \{f(z)\}$, *tj. find je određen sa* $\{x \mapsto x, y \mapsto x, z \mapsto z, f(x) \mapsto f(x), f(z) \mapsto f(z)\}$. *Petlja je prazna.*
- *merge*(y, z) – $T_y = \{f(x)\}$, $T_z = \{f(z)\}$. *Nakon union*(y, z), *dobijaju se klase* $\{\{x, y, z\}, \{f(x)\}, \{f(z)\}$, *tj. find je određen sa* $\{x \mapsto x, y \mapsto x, z \mapsto x, f(x) \mapsto f(x), f(z) \mapsto f(z)\}$. $f(x)$ *i* $f(z)$ *su kongruentni pa se poziva:*
  - *merge*$(f(x), f(z))$ – $T_{f(x)} = \{\}$, $T_{f(z)} = \{\}$. *Nakon union*$(f(x), f(z))$ *dobijaju se klase* $\{\{x, y, z\}, \{f(x), f(z)\}$, *tj. find je određen sa* $\{x \mapsto x, y \mapsto x, z \mapsto x, f(x) \mapsto f(x), f(z) \mapsto f(x)\}$. *Petlja je prazna.*

Odlučivanje univerzalnog fragmenta EUF

- Razmatrajmo validnost formula oblika $\forall x_1 \ldots x_n.\ P(x_1, \ldots, x_n)$, gde P ne sadrži drugih predikatskih simbola osim $=$ (funkcijski simboli se mogu javljati).
- Prethodni fragment se ponekad naziva EUF (Equality with Uninterpreted Functions).
- Negacijom i skolemizacijom se dobija bazna formula oblika $P'(c_1, \ldots, c_n)$.
- Prevođenjem u DNF (ili primenom naprednijih tehnika poput SMT), ispitivanje zadovoljivosti se svodi na ispitivanje zadovoljivosti konjunkcija oblika

$$s_1 = t_1 \wedge \ldots \wedge s_n = t_n \wedge s'_1 \neq t'_1 \wedge \ldots s'_{n'} \neq t'_{n'}$$

Odlučivanje univerzalnog fragmenta EUF

- Odeđuje se kongruentno zatvorenje za skup jednakosti $E = \{s_1 = t_1, \ldots, s_n = t_n\}$ i skup termova T dobijen od $\{s_1, t_1, \ldots s_n, t_n, s'_1, t'_1, \ldots, s'_{n'}, t'_{n'}\}$ i svih njihovih podtermova.
- Konjunkcija je zadovoljiva akko je $find(s'_i) \neq find(t'_i)$, za svako $1 \leq i \leq n'$.

Odlučivanje univerzalnog fragmenta EUF – primer

Primer

Pokažimo da formula

$$\forall x.\ f^3(x) = x \ \wedge\ f^5(x) = x \ \Rightarrow\ f^2(x) = x$$

važi u svim normalnim modelima, pri čemu je $f^i(x)$ *skraćeni zapis za* $f(f(\ldots f(x)))$*, gde se* f *primenjuje* i *puta. Negiranjem polazne formule i skolemizacijom dobija se formula*

$$f^3(c) = c \wedge f^5(c) = c \wedge f^2(c) \neq c.$$

Posmatrajmo izvršvanje Nelson-Oppen-ovog algoritma. Skup jednakosti $E = \{f^3(c) = c, f^5(c) = c\}$*, a skup termova* $T = \{c, f(c), f^2(c), f^3(c), f^4(c), f^5(c)\}$.

Odlučivanje univerzalnog fragmenta EUF – primer I

- $merge(f^3(c), c)$ – $T_{f^3(c)} = \{f^4(c), f^5(c)\}$, $T_c = \{f(c), f^2(c), f^3(c), f^4(c), f^5(c)\}$. Nakon izvršetka $union(f^3(c), c)$ dobijaju se klase $\{\{c, f^3(c)\}, \{f(c)\}, \{f^2(c)\}, \{f^4(c)\}, \{f^5(c)\}\}$. $f^4(c)$ i $f(c)$ su kongruentni pa se poziva:
  - $merge(f^4(c), f(c))$ – $T_{f^4(c)} = \{f^5(c)\}$, $T_{f(c)} = \{f^2(c), f^3(c), f^4(c), f^5(c)\}$. Nakon izvršetka $union(f^4(c), f(c)$ dobijaju se klase $\{\{c, f^3(c)\}, \{f(c), f^4(c)\}, \{f^2(c)\}, \{f^5(c)\}\}$. $f^5(c)$ i $f^2(c)$ su kongruentni pa se poziva:
    - $merge(f^5(c), f^2(c))$ – $T_{f^5(c)} = \{\}$, $T_{f^2(c)} = \{f^3(c), f^4(c), f^5(c)\}$. Nakon izvršetka $union(f^4(c), f(c)$ dobijaju se klase $\{\{c, f^3(c)\}, \{f(c), f^4(c)\}, \{f^2(c), f^5(c)\}\}$. Petlja je prazna.

Odlučivanje univerzalnog fragmenta EUF – primer II

Ostali parovi iz $T_{f^4(c)}$ i $T_{f(c)}$ ili nisu kongruentni ili su već u istoj klasi.

- $merge(f^5(c), c)$ — $T_{f^5(c)} = \{f^3(c), f^4(c), f^5(c)\}$, $T_c = \{f(c), f^2(c), f^3(c), f^4(c), f^5(c)\}$. Nakon $union(f^5(c), c)$ dobijaju se klase $\{\{c, f^2(c), f^3(c), f^5(c)\}, \{f(c), f^4(c)\}\}$.
  Termovi $f^3(c)$ i $f(c)$ su kongruentni pa se poziva
  - $merge(f^3(c), f(c))$ –
    $T_{f^3(c)} = \{f(c), f^2(c), f^3(c), f^4(c), f^5(c)\}$,
    $T_{f(c)} = \{f^2(c), f^3(c), f^4(c), f^5(c)\}$.
    Nakon $union(f^3(c), f(c)$ dobijaju se klase $\{\{c, f(c), f^2(c), f^3(c), f^4(c), f^5(c)\}\}$. Svi termovi su istoj klasi pa nema daljih rekurzivnih poziva.

Svi termovi su istoj klasi pa nema daljih rekurzivnih poziva.

Odlučivanje univerzalnog fragmenta EUF – primer

Primer

Dakle, pošto negirana formula sadrži različitost $f^2(c) \neq f(c)$, *a na osnovu kongruentnog zatvorenja važi* $f^2(c) = f(c)$, *ona je nezadovoljiva, te polazna formula*

$$\forall x.\ f^3(x) = x \ \wedge\ f^5(x) = x \ \Rightarrow\ f^2(x) = x$$

važi u svim normalnim modelima.

Pregled

1 Normalni modeli

2 Aksiome jednakosti

3 Birkhofov sistem

4 Kongruentno zatvorenje

5 Prezapisivanje

Pojam prezapisivanja

- Dokazi jednakosti oblika $s = t$ se (neformalno) obično sprovode tako što što se započne od terma s, a onda se na njega (odnosno njegove podtermove) primenjuju transformacije na osnovu datih jednakosti $s_i = t_i$, vršeći pri tom pogodne instancijacije.
- Transformacije se obično vrše tako da se dobije jednostavniji term (npr. $x \cdot x^{-1}$ se zamenjuje sa 1 — zamena u suprotnom smeru je obično neintuitivna).
- Pojam jednostavnijeg terma je suptilan. Npr. $(x + y) \cdot (w + z)$ je kraći od $xw + xz + yw + yz$ ali se drugi smatra jednostavnijim jer omogućava dalja skraćivanja.
- Korišćenje transformacija termova na osnovu usmerenih jednakosti naziva se prezapisivanje.

Definicija

Pravilo prezapisivanja termova je usmerena jednakost oblika $l \to r$ gde su l i r termovi takvi da l nije promenljiva i r ne uvodi nove slobodne promenljive u odnosu na l (skup slobodnih promenljivih terma r je podskup skupa slobodnih promenljivih terma l).

Definicija

Ako je $l \to r$ pravilo prezapisivanja termova kažemo da se term t na osnovu ovog pravila prezapisuje u term t' ako postoji podterm terma t koji je instanca terma l, takav da se njegovom zamenom instancom terma r (pri istoj instancijaciji) dobija term t'.

Definicija

Sistem prezapisivanja R termova je skup pravila prezapisivanja termova. Kažemo da $t \to_R t'$ ako postoji neko pravilo $l \to r \in R$. Relacija $\to_R$ je relacija prezapisivanja termova.

Primer primene prezapisivanja

Primer

Prezapisivanjem terma $(a + a) + (b + b)$ *na osnovu pravila* $x + x \to 2x$ *mogu se dobiti* $2a + (b + b)$ *i* $(a + a) + 2b$. *Term* $2a + 2b$ *se može dobiti nakon dva koraka.*

- Prezapisivanje ima svoje teoretsko opravdanje (zasnovano na Birkhofovoj teoremi).
- Svaki lanac prezapisivanja se može konvertovati u formalni dokaz u Birkhofovom sistemu.
- Važi i obratno.

Teorema

Ako je $\rightarrow_R$ relacija dobijena na osnovu sistema prezapisivanja R tada za svaka dva terma s i t važi $s \rightarrow_R t$ ako i samo ako $R \models s = t$.

Kanonski sistemi

- Ponekad, sistemi prezapisivanja imaju svojstvo da se svi ,,ekvivalentni" termovi svode na istu normalnu formu — term na koji dalje nije moguće primeniti ni jedno pravilo.
- Ovakvi sistemi se nazivaju kanonski ili konvergentni.
- U slučaju jednakosnog rezonovanja na osnovu datih jednakosti E, obično se kaže da su s i t ekvivalentni ako $E \models s = t$.
- Ovakvi sistemi se mogu koristiti kao procedure odlučivanja ekvivalentnosti dva terma u odnosu na dati sistem jednakosti (proveri se da li se nakon ,,normalizacije" dobija ista normalna forma).

Primer kanonskog sistema

Primer

$$\{m+0 = m, 0+n = n, m+S(n) = S(m+n), S(m)+n = S(m+n)\}$$

Utvrđivanje jednakosti dva terma na osnovu ovog sistema ne zahteva nikakvu kreativnost. Npr.

$$S(0) + S(S(0)) \to S(0 + S(S(0))) \to S(S(0))$$

Takođe,

$$\begin{aligned} S(0) + S(S(0)) \quad \to \quad & S(S(0) + S(0)) \to S(S(S(0) + 0)) \to \\ & S(S(S(0 + 0))) \to S(S(S(0))) \end{aligned}$$

Naravno, efikasnost zavisi od redosleda primene pravila, međutim, normalna forma do koje će se stići ne zavisi.

Primeri ne-kanonskih sistema — nezaustavljanje

Primer

Neka je dato pravilo $x + y \to y + x$. *Term* $1 + 2$ *nema normalnu formu. Zaista, postoji izvođenje:*

$$1 + 2 \to 2 + 1 \to 1 + 2 \to 2 + 1 \to \ldots$$

Kaže se da sistem nema svojstvo zaustavljanja.

Primeri ne-kanonskih sistema — nekonfluentnost

Primer

$$\{x + 0 \to x, s(x + y) \to x + s(y)\}$$

Term $s(x + 0)$ se primenom prvog pravila može prezapisati u $s(x)$, a primenom drugog pravila u $x + s(0)$. Nijedan od ova dva terma nije moguće dalje preazapisivati.

Dobijene normalne forme nisu jednake. Kaže se da sistem nema svojstvo konfluentnosti.

Primeri ne-kanonskih sistema — nekonfluentnost

Primer

$$\{x \cdot (y + z) \to x \cdot y + x \cdot z, (x + y) \cdot z \to x \cdot z + y \cdot z\}$$

$$\begin{aligned}(a + b) \cdot (c + d) &\to a \cdot (c + d) + b \cdot (c + d) \\ &\to (a \cdot c + a \cdot d) + b \cdot (c + d) \\ &\to (a \cdot c + a \cdot d) + (b \cdot c + b \cdot d)\end{aligned}$$

$$\begin{aligned}(a + b) \cdot (c + d) &\to (a + b) \cdot c + (a + b) \cdot d \\ &\to (a \cdot c + b \cdot c) + (a + b) \cdot d \\ &\to (a \cdot c + b \cdot c) + (a \cdot d + b \cdot d)\end{aligned}$$

Apstraktni sistemi za prezapisivanje

- Najznačajni sistemi za prezapisivanje su sistemi za prezapisivanje termova.
- Ipak, ponekad se posmatraju opštiji sistemi, tzv. apstraktni sistemi za prezapisivanje.
- Relacija prezapisivanja R se posmatra na proizvoljnom skupu X (ne obavezno skupu termova) i obično se zapisuje infiksno $x \to y$. Ova relacija je apstraktno data i ne definiše se na osnovu sistema prezapisivanja.
- Oznaka $\to^+$ označava tranzitivno zatvorenje relacije $\to$, oznaka $\to^*$ njeno refleksivno i tranzitivno zatvorenje, dok oznaka $\leftrightarrow^*$ označava njeno simetrično, refleksivno i tranzitivno zatvorenje.

Zaustavljanje

Definicija

Relacija $\rightarrow$ je zaustavljajuća (dobro zasnovana) ako ne postoji beskonačan lanac $t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow \ldots$.

Oblici konfluentnosti

Definicija

- *Relacija* $\rightarrow$ *ima svojstvo dijamanta ako kada* $x \rightarrow y$ *i* $x \rightarrow y'$ *tada postoji* z *tako da* $y \rightarrow z$ *i* $y' \rightarrow z$.
- x *i* y *su spojivi što označavamo sa* $x \downarrow y$ *ako postoji* z *tako da* $x \rightarrow^* z$ *i* $y \rightarrow^* z$.
- *Relacija je konfluentna ako kada* $x \rightarrow^* y$ *i* $x \rightarrow^* y'$ *tada* $y \downarrow y'$. *Ekvivalentno,* $\rightarrow^*$ *ima svojstvo dijamanta.*
- *Relacija je slabo konfluentna ako kada ako kada* $x \rightarrow y$ *i* $x \rightarrow y'$ *tada* $y \downarrow y'$.
- *Relacija ima Čerč-Roserovo svojstvo ako kada god* $x \leftrightarrow^* y$, *tada postoji* $x \downarrow y$.

Oblici konfluentnosti

Stav

- *Ako relacija ima svojstvo dijamanta ona je slabo konfluentna.*
- *Ako je relacija konfluentna ona je slabo konfluentna.*
- *Ako relacija ima svojstvo dijamanta ona je ona konfluentna.*
- *Relacija ima Čerč Roserovo svojstvo akko je konfluentna.*

Oblici konfluentnosti

Slabo konfluentna relacija ne mora biti konfluentna.

Primer

Relacija $b \to a, b \to c, c \to b, c \to d$ je slabo konfluentna, ali nije konfluentna.

Ipak, važi naredna teorema.

Teorema (Njuman)

Ako je relacija $\to$ zaustavljajuća i slabo konfluentna, ona je i konfluentna.

Ispitivanje zaustavljanja

- Jedan od fundamentalnih problema je kako za dati sistem prezapisivanja termova utvrditi da li je zaustavljajući.
- Problem ispitivanja zaustavljanja je neodlučiv!
- Osnovni metod je pronalaženje dobro zasnovanog uređenja $\succ$ takvog da iz $t \to_R t'$ sledi da $t \succ t'$.
- Još je bolje posmatrati dobro-zasnovano $\succ$ uređenje takvo da za svako pravilo $l \to r$ važi $l \succ r$. Međutim, da bi se iz ovoga moglo garantovati zaustavljanje potrebno je da uređenje $\succ$ ima i dodatna svojstva data u sledećoj definiciji.

Problemi poređenja po veličini

- Problem kod obezbeđivanja zaustavljanja predstavljaju pravila poput $(x \cdot y) \cdot z \to x \cdot (y \cdot z)$ i $x \cdot (y + z) \to x \cdot y + x \cdot z$ jer se u njima veličina terma sa desne strane ne menja.
- Ovo ukazuje da samo poređenje veličine termova nije dobar pristup.

Uređenje svođenja

Definicija

Relacija $\succ$ je uređenje prezapisivanja ako je tranzitivno, irefleksivno i zatvoreno u odnosu na instancijacije i kongruencije, tj.

- *irefleksivno – Ni za jedan term t ne važi $t \succ t$,*
- *tranzitivno – Ako je $s \succ t$ i $t \succ u$, tada $s \succ u$.*
- *stabilno – Ako je $s \succ t$ tada je $s[x \to t'] \succ t[x \to t']$.*
- *monotono – Ako je $s_i \succ t_i$ za neko i, tada $f(s_1, \ldots, s_i, \ldots, s_n) \succ f(t_1, \ldots, t_i, \ldots, t_n)$.*

Dobro zasnovana uređenja prezapisivanja nazivaju se uređenja svođenja.

Uređenja svođenja

Lema (Manna, Ness)

Ako je $\succ$ uređenje svođenja i za svako $l \to r \in R$ važi $l \succ r$, tada je $\to_R$ zaustavljajuća.

Dokaz

Dovoljno je dokazati da iz $s \to_R t$ sledi $s \succ t$ — pošto je $\succ$ dobro zasnovano tvrđenje sledi. Iz $s \to_R t$ sledi da postoji $l \to r \in R$ i instanca $\bar{l}$ terma l koja je podterm terma s, takva da kada se on zameni odgovarajućom instancom $\bar{r}$ terma r dobija se term t. Pošto je $l \succ r$, na osnovu svojstva instancijacije važi da je $\bar{l} \succ \bar{r}$. Na osnovu uzastopne primene svojstva kongruentnosti dobija se i da je $s \succ t$.

Uređenja pojednostavljivanja

Specijalnu vrstu uređenja svođenja čine uređenja pojednostavljivanja.

Definicija

Relacija $\succ$ je uređenje pojednostavljivanja ako je irefleksivno, tranzitivno, stabilno, monotono i ima sledeće svojstvo podtermova:

- *svojstvo podtermova* – $f(\ldots, t_i, \ldots) \succ t_i$.

Stav

Deršovic Svako uređenje pojednostavljivanja je dobro uređenje te je i uređenje svođenja.

Rekurzivna uređenja staze

- U mnogim slučajevima se uređenja definišu induktivno nad strukturom termova što ih čini monotonim i stabilnim.
- Često je pristup da se prvo izgradi uređenje $\succeq$ na skupu svih funkcijskih simbola jezika $\mathcal{L}$, a da se zatim ono upotrebi kako bi se izgradilo uređenje pojednostavljivanja na skupu termova.

Rekurzivna uređenja staze

- Da bi dobijena uređenja bila uređenja pojednostavljivanja, potrebno je da zadovoljavaju svojstva podtermova, prvo tako što se proverava je da li je eventualno jedan od njih podterm drugoga.
- Ukoliko se utvrdi da nijedan od termova nije podterm drugoga, tada se njihov odnos određuje na osnovu njihovih vodećih funkcijskih simbola.
- Term sa ,,većim" vodećim simbolom se proglašava većim (uz određene uslove za argumente), dok se u slučaju jednakih vodećih simbola, prelazi na poređenje nizova argumenata.
- Uređenja izgrađena na ovaj način se nazivaju uređenjima rekurzivne staze (recursive path orderings).
- Primeri takvih konstrukcija su uređenje multiskup staze i uređenje leksikografske staze.

Definicija

Neka je data signatura $\mathcal{L}$, i neka je $\succeq$ (kvazi)uređenje skupa funkcijskih simbola Σ. Uređenje leksikografkse staze (lexicographic path ordering) $>_{lpo}$ se definiše rekurzivno sledećim skupom pravila:

1. $$\frac{\exists i.\ s_i \geq_{lpo} t}{f(s_1, \ldots, s_m) >_{lpo} t}$$
2. $$\frac{f \succ g, \quad \forall i.\ f(s_1, \ldots, s_m) \geq_{lpo} t_i}{f(s_1, \ldots, s_m) >_{lpo} g(t_1, \ldots, t_n)}$$
3. $$\frac{f \approx g, \quad \forall i.\ f(s_1, \ldots, s_m) \geq_{lpo} t_i, \quad (s_1, \ldots, s_m) >_{lpo}^{lex} (t_1, \ldots, t_n)}{f(s_1, \ldots, s_m) >_{lpo} g(t_1, \ldots, t_n)}$$
4. $$\frac{}{t \geq_{lpo} t}$$
5. $$\frac{v \in Vars(t) \quad t \neq v}{t >_{lpo} v}$$

Uređenje leksikografske staze

Stav

Svako uređenje leksikografske staze je uređenje pojednostavljivanja, pa samim tim i uređenje svođenja.

Uređenje leksikografske staze – primeri

Primer

Korišćenjem uređenja leksikografske staze pokažimo da je zaustavljajući sistem

$$\begin{aligned} x + 0 &\to x \\ -0 &\to 0 \\ -(x + y) &\to (-x) + (-y) \end{aligned}$$

Neka je kvazi uređenje simbola $- \succ + \succ 0$.

Primer

Tada je:

- $x + 0 >_{lpo} x$ *na osnovu pravila 5. jer je* $x \in Var$
- $-0 >_{lpo} 0$ *na osnovu 1. jer je na osnovu 4.* $0 \geq_{lpo} 0$
- $-(x + y) >_{lpo} (-x) + (-y)$ *na osnovu pravila 2. jer je* $- \succ +$ *i pošto je, kao prvo,* $-(x + y) \geq_{lpo} -x$ *i pošto je, kao drugo,* $-(x + y) \geq_{lpo} -y$*. Prvo zaista važi na osnovu pravila 3. ukoliko je* $x + y >_{lpo}^{lex} x$ *a ovo je ispunjeno jer je* $x + y >_{lpo} x$ *na osnovu pravila 5. Na isti način se pokazuje i drugi uslov.*

Uređenje leksikografske staze – primeri

Primer

Zaustavljanje pravila

$$(x \cdot y) \cdot z \to x \cdot (y \cdot z)$$

se može pokazati uređenjem leksikografske staze. Zaista na osnovu pravla 3. treba pokazati da je $(x \cdot y, z) >_{lpo}^{lex} (x, y \cdot z)$ *što je ispunjeno jer je* $x \cdot y >_{lpo} x$ *na osnovu pravila 5.*

- Problem ispitivanja konfluentnosti je u opštem slučaju neodlučiv.
- Ipak, postoje specijalni slučajevi koji su odlučivi:

Stav

- *Pitanje konfluentnosti baznih sistema za prezapisivanje je odlučivo.*
- *Pitanje konfluentnosti zaustavljajućih sistema za prezapisivanje je odlučivo.*

- Naglasimo da zaustavljanje nije neophodno za konfluentnost:

Primer

Sistem $R = \{a \to b, b \to a, a \to c\}$ *je konfluentan, ali nije zaustavljajući.*

Kritični parovi

Definicija (Kritični par)

Neka su $l_1 \to r_1$ i $l_2 \to r_2$ dva pravila koja nemaju zajedničkih promenljivih (ovo se može uvek postići preimenovanjem). Neka je l_1' podterm terma l_1 koji nije promenljiva i neka je θ najopštiji unifikator termova l_1' i l_2. Term $l_1[l_1' \to \theta(l_2)]$ određuje kritični par $\langle \theta(r_1), \theta(l_1)[\theta(l_1') \to \theta(r_2)] \rangle$.

Primer

Neka je $l_1 \to r_1 \equiv s(x + y) \to x + s(y)$, a $l_2 \to r_2 \equiv x + 0 \to x$. Podterm $l_1' \equiv x + y$ se može unifikovati sa $x + 0$ unifikatorom $\theta = \{y \mapsto 0\}$. Term $s(x + 0)$ određuje kritični par $\langle x + s(0), s(x) \rangle$.

Knut-Bendiksova teorema

Značaj kritičnih parova dolazi iz sledeće teoreme:

Teorema

Sistem za prezapisivanje je lokalno konfluentan ako i samo ako su mu svi kritični parovi povezivi tj. ako važi $u_1 \downarrow u_2$*, za svaki kritični par* $\langle u_1, u_2 \rangle$

Provera kritičnih parova i ispitivanje konfluentnosti

- Da bi se otkrili svi kritični parovi datog sistema za prezapisivanje, svako pravilo se kombinuje sa svakim drugim (uključujući i svoju preimenovanu verziju).
- Pošto u slučaju konačnog sistema za prezapisivanje termova kritičnih parova ima konačno mnogo, da bismo utvrdili da li je sistem lokalno konfluentan, dovoljno je ispitati povezivost konačno mnogo parova termova.
- U slučaju zastavljajućeg sistema, povezivost kritičnih parova se efektivno može ispitati što daje proceduru odlučivanja za pitanje konfluentnost konačnih, zaustavljajućih sistema za prezapisivanje termova.

Knut-Bendiksova procedura upotpunjavanja

- Iako se ispitivanjem kritičnih parova ponekad ustanovi da određeni sistem za prezapisivanje termova R nije konfluentan, u nekim slučajevima se, dodavanjem određenih pravila, sistem može učiniti konfluentnim.
- Neka je $\langle u_1, u_2 \rangle$ kritični par koji nije poveziv, tj. postoje različite normalne forme $\bar{u}_1$ i $\bar{u}_2$ termova u_1 i u_2.
- Jednakost $\bar{u}_1 = \bar{u}_2$ je posledica sistema R jer je $\bar{u}_1 \overset{*}{\leftrightarrow}_R \bar{u}_2$ i zbog toga dodavanje pravila $\bar{u}_1 \to \bar{u}_2$ ili $\bar{u}_2 \to \bar{u}_1$ ne menja odgovarajuću jednakosnu teoriju.
- U ovom proširenom sistemu, par $\langle u_1, u_2 \rangle$ postaje poveziv.
- Da bi sistem ostao zaustavljajući, potrebno je da je $\bar{u}_1 \succ \bar{u}_2$ ili $\bar{u}_2 \succ \bar{u}_1$ u odgovarajućem uređenju svođenja $\succ$.

if ($\exists(s = t) \in E \wedge s \neq t \wedge s \not> t \wedge t \not> s$)
    return fail;
$R_0 = \{l \to r \mid (l = r) \in E \cup E^{-1} \wedge l > r\}$
do
    $R_{i+1} = R_i$
    forall ($\langle u_1, u_2 \rangle \in CP(R_i)$)
        $\bar{u}_1 = u_1 \downarrow$; $\bar{u}_2 = u_2 \downarrow$;
        if ($\bar{u}_1 \neq \bar{u}_2 \wedge \bar{u}_1 \not> \bar{u}_2 \wedge \bar{u}_2 \not> \bar{u}_1$)
            return fail;
        if ($\bar{u}_1 > \bar{u}_2$)
            $R_{i+1} = R_{i+1} \cup \{\bar{u}_1 \to \bar{u}_2\}$
        else if ($\bar{u}_2 > \bar{u}_1$)
            $R_{i+1} = R_{i+1} \cup \{\bar{u}_2 \to \bar{u}_1\}$
        i := i+1;
while ($R_i \neq R_{i-1}$);
return R_i;