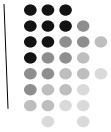
Klasična iskazna logika

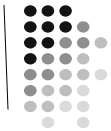
Algebra i logika u računarstvu

Normalne forme



- **Definicija**. *Literal* je atomska formula ili njena negacija. Formula je u *konjuktivnoj normalnoj formi* (KNF), ako je oblika $D_1 \wedge D_2 \wedge \ldots \wedge D_n$ pri čemu je svaki D_i disjunkcija literala oblika $D_i = L_{i,1} \vee L_{i,2} \vee \ldots \vee L_{i,m_i}$.
  Formula je u *disjunktivnoj normalnoj formi* (DNF), ako je oblika $K_1 \vee K_2 \vee \ldots \vee K_n$ pri čemu je svaki K_i konjukcija literala oblika $K_i = L_{i,1} \wedge L_{i,2} \wedge \ldots \wedge L_{i,m_i}$
- **Teorema**. Za svaku formulu *A* postoje njoj ekvivalentne *KNF(A)* i *DNF(A).*

Normalne forme

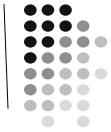


```
Procedure NormalnaForma
Begin
```

1. Eliminisati veznik $\leftrightarrow$ koristeći ekvivalenciju
   $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$
2. Eliminisati veznik $\rightarrow$ koristeći ekvivalenciju
   $A \rightarrow B \equiv \neg A \vee B$
3. Dok god je moguće ponavljati primenu DeMorganovih zakona $\neg(A \wedge B) \equiv \neg A \vee \neg B$ i $\neg(A \vee B) \equiv \neg A \wedge \neg B$
4. Dok god je moguće ponavljati primenu zakona dvojne negacije $\neg\neg A \equiv A$
5. a) Za dobijanje KNF ponavljati primenu zakona distributivnosti $\vee$ prema $\wedge$
   b) Za dobijanje DNF ponavljati primenu zakona distributivnosti $\wedge$ prema $\vee$

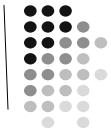
```
End.
```

Normalne forme



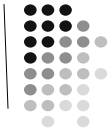
- **Definicija.** *Kanonska (potpuna) DNF(A)* formule A koja nije kontradikcija i koja sadrži iskazna slova $p_1,\ldots,p_n$ je formula oblika $\bigvee_{I_v(A)=1}\left(p_1^v \wedge \ldots \wedge p_n^v\right)$, $p_i^v = \begin{cases} p_i & , v(p_i)=1 \\ \neg p_i & , v(p_i)=0 \end{cases}$
- **Definicija.** *Kanonska (potpuna) KNF(A)* formule A koja nije tatutologija i koja sadrži iskazna slova $p_1,\ldots,p_n$ je formula oblika $\bigwedge_{I_v(A)=0}\left(p_1^v \vee \ldots \vee p_n^v\right)$, $p_i^v = \begin{cases} \neg p_i & , v(p_i)=1 \\ p_i & , v(p_i)=0 \end{cases}$
- Primeri

Metod rezolucije



- **Definicija**. Literali L_1 i L_2 su komplementarni, ako je jedan od njih atomska formula, a drugi njena negacija. Klauza je disjunkcija literala. Prazna klauza $\varnothing$ ne sadrži ni jedan literal.
- Ako je I proizvoljna interpretacija, tada je
  1. $I(\varnothing) = 0$,
  2. $D = \{C_i \mid i = 1,2,\ldots,n\}, I(D) = 1 \Leftrightarrow \forall i, I(C_i) = 1$ gde su C_i klauze
  3. $S_1 \simeq S_2 \Leftrightarrow I(S_1) = I(S_2)$ gde su S_1 i S_2 skupovi klauza.

Metod rezolucije

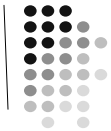


- **Definicija**. Neka su C_1 i C_2 klauze i neka su L_1 i L_2 komplementarni literali, takvi da se L_1 nalazi u C_1, a L_2 u C_2. *Rezolventa* klauza C_1 i C_2 po literalima L_1 i L_2 je klauza $Res(C_1, C_2, L_1, L_2) = (C_1 \setminus \{L_1\}) \cup (C_2 \setminus \{L_2\})$.

$$RR: \frac{A \vee p, B \vee \neg p}{A \vee B}$$

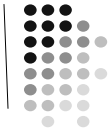
- **Teorema**. Neka su C_1 i C_2 klauze i R njihova rezolventa. Tada je $\{C_1, C_2\} \simeq \{C_1, C_2, R\}$
- **Definicija**. Dokaz za klauzu C iz skupa klauza $\{A_1, A_2, \ldots\}$ je niz klauza $C_1, C_2, \ldots, C_m$ gde je C_i bilo neka od klauza A_j, bilo klauza dobijena primenom RR pravila na neke dve prethodne klauze u dokazu i $C_m = C$.

Metod rezolucije



- Neka je *E* skup klauza, tada sa *R(E)* označavamo skup svih klauza iz *E* i svih klauza dobijenih primenom RR-a na klauze iz *E*.
- Neka je *F* skup klauza, $Res_0(F)$=F, za $i \geq 0$, $Res_{i+1}(F) = R\left(Res_i(F)\right)$ i $Res^*(F) = \bigcup_i Res_i(F)$.
- **Teorema**. Skup klauza je kontradiktoran ako i samo ako je prazna kaluza u skupu *Res*(F)*.
- Neka je *I* proizvoljna interpretacija, tada je $I(A) = 1 \Leftrightarrow I(\neg A) = 0$, tj. formula A je zadovoljiva pri *I* ako je iz $KNF(\neg A)$ primenom RR-a izvodi $\varnothing$. Takođe, $B_1, B_2, \ldots, B_n \vdash A \Leftrightarrow B_1, B_2, \ldots, B_n, \neg A \overset{RR}{\vdash} \varnothing$

Metod rezolucije



```
Procedure IskaznaRezolucija
Begin
  A prevesti u KNF(¬A)(= F := {D1, D2, ..., Dn})
  Res0(F) := F
  Res1(F) := R(Res0(F))
  i:=1
  while (Resi(F) ≠ Resi-1(F)) and (∅ ∉ Resi(F)) do
    i:=i+1
    Resi(F) := R(Resi-1(F))
  if  ∅ ∈ Resi(F)  then  Formula je zadovoljiva
                   else  Formula nije zadovoljiva
End
```

Metod rezolucije

- Metoda rezolucije je po svojoj prirodi mehanička, odnosno orijentisana ka računarskom izvršavanju, što se lako uočava kod primene na veće skupove formula kada se pojavljuje ogroman broj klauza.
- Tokom rezolviranja od klauza roditelja se često dobijaju klauze koje nisu u polaznom skupu, niti su deo nekih klauza polaznog skupa, što ima za posledicu potencijalno eksplozivni rast broja klauza.
- Rezolucija se može elegantno primeniti samo ako je moguće proizvoljne formule prevesti u oblik konjunktivne normalne forme, što nije slučaj sa većinom logika.
- Metod rezolucije predstavlja osnov mehanizma izvođenje na kome je baziran programski jezik *Prolog*.
- Primeri

Metod tabloa

- **Definicija**. Neka se iskazni jezik sastoji od prebrojivog skupa Φ iskaznih slova, logičkih veznika $\neg$, $\wedge$, $\vee$ i $\rightarrow$ i zagrada. Neka su T i F novi formalni simboli koji se ne nalaze u iskaznom jeziku. Ako je A iskazna formula, TA i FA su *označene formule*. *Konjugat* ozmačene formule TA (u oznaci $\overline{TA}$) je formula FA i obrnuto $\overline{FA} = TA$.
- Za proizvoljnu interpretaciju I, $I(TA) = I(A)$ i $I(FA) = I(\neg A)$.
- Sve označene neatomske fomule dele su na α i β formule.

α	α_1	α_2
$TA \wedge B$	TA	TB
$FA \vee B$	FA	FB
$FA \rightarrow B$	TA	FB
$T\neg A$	FA	FA
$F\neg A$	TA	TA

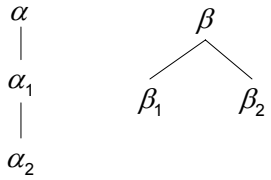
β	β_2	β_1
$FA \wedge B$	FA	FB
$TA \vee B$	TA	TB
$TA \rightarrow B$	FA	TB

Metod tabloa

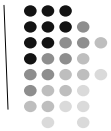
- **Teorema**. Za proizvoljnu interpretaciju I važi:
  - $I(\alpha) = \text{T}$ ako i samo ako $I(\alpha_1) = I(\alpha_2) = \text{T}$
  - $I(\beta) = \text{T}$ ako i samo ako $I(\beta_1) = \text{T}$ ili $I(\beta_2) = \text{T}$
  - Označena formula je tačna pri interpretaciji I ako i samo ako je njen konjugat netačan.

Metod tabloa

- Analitički tablo za formulu A je uređeno binarno stablo čiji čvorovi sadrže označene formule i koje se konstruiše prema sledećim pravilima:
  - U korenu drveta je formula FA
  - Neka je čvor Y kraj neke grane do tog momenta konstruisanog drveta. Zavisno od formula na grani koja sadrži Y, nastavak konstrukcije je moguće izvesti na dva načina:
    - α-pravilo: ako je neka α formula na grani, tada se grana produžava sa dva čvora od kojih jedan sadrži α_1 formulu, a drugi α_2 formulu,
    - β-pravilo: ako je neka β formula na grani, tada se grana u čvoru Y grana, pri čemu levi naslednik sadrži β_1, a desni β_2 formulu.

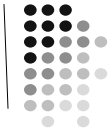


Metod tabloa



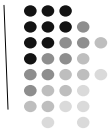
- Grana tabloa je *zatvorena* ako se na njoj nalazi neka formula i njen konjugat.
- Tablo je *zatvoren* ako mu je zatvorena svaka grana.
- *Dokaz* za (neoznačenu) formulu *A* je zatvoren tablo za *A*.
- Formula je teorema ako postoji tablo koji je dokaz za nju.
- Grana je *završena* ako su odgovarajuća pravila konstrukcije primenjena na sve čvorove sa te grane.
- Tablo je *završen* ako mu je svaka grana završena ili zatvorena.
- Grana tabloa je *zadovoljiva* pri nekoj interpretaciji ako zadovoljava sve označene formule koje se nalaze u čvorovima te grane.
- Tablo je *zadovoljiv* ako postoji interpertacija takva da je bar jedna grana tabloa zadovoljiva pri *I*.

Metod tabloa



- **Teorema. (Korektnost)** Ako formula *A* ima tablo dokaz, ona je valjana.
- **Teorema.** Svaka završena grana tabloa koja nije zatvorena je zadovoljiva.
- **Teorema. (Kompletnost)** Ako je formula *A* tautologija, svaki završeni tablo za *A* mora biti zatvoren.

Metod tabloa



- Metoda analitičkih tabloa se može isprogramirati, recimo, na sledeći način. Formula *A* se predstavlja u obliku binarnog drveta čiji čvorovi sadrže logičke veznike, a listovi iskazna slova formule.
- Tablo se konstruiše kao binarno drvo čiji čvorovi sadrže pokazivače na delove drveta formule i oznake *T* i *F*, zavisno od toga koji prefiks ima formula koja odgovara čvoru.
- Ako se radi sekvencijalno, konstrukcija tabloa se odvija grana po grana u skladu sa pravilima izvođenja. Posle primene svakog pravila izvođenja proverava se da li je grana zatvorena ili završena. Ako je grana zatvorena, prelazi se na sledeću granu ukoliko takva postoji. Ako je grana završena, a nije zatvorena, polazna formula nije valjana i eventualno se može prikazati njen kontramodel. Ako su sve grane zatvorene, formula je valjana.
- Memorijska reprezentacija svake grane podseća na stek na koji upravljački program dodaje nove čvorove, odnosno sa koga oslobađa nepotrebne čvorove prilikom prelaska na novu granu. Pogodnost ovakvog postupka je malo memorijsko zauzeće pošto nema kopiranja podformula formule *A*.
- U paralelnom izvođenju je pogodno na svaki raspoloživi procesor poslati poseban deo formule koji odgovara jednoj ili više grana, što se može uraditi prilikom primene β pravila kada se leva grana zadržava na aktuelnom procesoru, a desna šalje na neki slobodni procesor.