Ekspertski sistemi

PREDSTAVLJANJE ZNANJA U FORMALNOJ LOGICI

Predstavljanje znanja

- Mora se najpre znati da se apstraktno predstavlja znanje i zatim kako se ono koristi u sistemu zaključivanja.
- Predstavljanje znanja obuhvata:
  - strukturu (model) koja se zahteva da opiše elemente znanja (formalna logika, semantičke mreže, okviri, pravila)
  - interpretativni proces koji se zahteva u korišćenju znanja

Predstavljanje znanja

- faktičko znanje - znanja koja opisuju neki element u posmatranoj oblasti (statičko stanje), ništa o dinamičkim aktivnostima povezanim sa objektom (voda je u tečnom stanju),
- proceduralno znanje - opisuju neke dinamičke akcije povezane sa elementima domena (ako voda curi iz česme, zameniti gumicu na česmi)

Proceduralna pravila

IF (skup uslova)

THEN (akcije koje treba preduzeti).

- Ovo je primer sintetičkih sistema.

IF (životinja leti & leže jaja)

THEN (to je ptica)

- U IF delu se nalaze uslovi, a u THEN delu činjenice. Ovakvi sistemi se nazivaju analitički sistemi

Kriterijumi predstave znanja

- transparentnost - mera u kojoj se može identifikovati smešteno znanje
- eksplicintnost - mera direktnog predstavljanja znanja
- prirodnost - mera prirodne predstave znanja (znanja u njegovoj prirodnoj formi)
- efikasnost - mera u kojoj se zadata struktura može iskoristiti za predstavu celokupnog znanja ekspertskog sistema
- modularnost - mera u kojoj se delovi znanja mogu skladištiti nezavisno jedno od drugog

Strategijske šeme predstavljanja znanja

- Deklarativne - akumulacija statičkih činjenica sa ograničenim informacijama o tome kako ih koristiti
- Proceduralne - naglasak na dinamičkim pravilima koja opisuju procedure za korišćenje znanja, sa malim uskladištenjem direktno na činjenice

Deklarativna predstava znanja - karakteristike

- transparentnost - znanja je smešteno u eksplicitnoj i nedvosmislenoj formi. Znanje se lako revidira zbog transparentnosti
- efikasno uskladištenje - svaki deo znanja se skladišti samo jedanput i ako se koristi na više različitih načina
- fleksibilnost - znanje se može smestiti na nižem nivou uz dobijanje povećane fleksibilnosti
- direktno zaključivanje - direktna statička predstava dozvoljava eksplicitno, direktno, matematici slično zaključivanje.

Propozicionalna logika

- ☺ Propozicionalna logika je deklarativna
- ☺ Propozicionalna logika dozovoljava parcijalne/disjunktivne/negirane informacije
  - (za razliku od većine struktura i baza podataka)
- Za propozicionalna logiku važi:
  - značenje $B_{1,1} \wedge P_{1,2}$ se dobija na osnovu značenja $B_{1,1}$ i $P_{1,2}$
- ☺ Propozicionalna logika je nezavisna od konteksta

  (za razliku od prirodnih jezika gde postoji zavisnost od konteksta)
- ☹ Propozicionalna logika je veoma izražajno limitirana
  - (za razliku od prirodnih jezika)
  - Ne može “rupe prouzrokuju strujanje u susednim sobama“
    - Osim pisanja posebnog izraza za svaki slučaj

Logika prvog reda

- Propozicionalna logika pretpostavlja da se svet sastoji od činjenica,
- Logika prvog reda (kao prirodni jezici) pretpostavlja da se svet sastoji od
  - Objekata: ljudi, kuće, brojevi, boje, igre, …
  - Relacija: veći od, deo od, brat od, između, …
  - Funkcija: otac, najbolji prijatelj, plus …

Predikatska logika prvog reda

Azbuka je skup objekata od kojih se prave iskazi u formalnom jeziku:

- konstante
- promenljive
- funkcije
- predikati
- veznici
- kvantifikatori
- ograničivači (zagrade, zarezi)

Konstante

- Predstavljaju specifičan element u domenu. Prikazuju se preko simbloičkog imena i predstavljaju objekte, apstrakcije (ideje, stanovišta, skupovi podataka). Obeležavaju se velikim slovima.
- DJERDAP - hidroelektrana, klisura
- KAPITALIZAM - ideološka pozicija

Promenljive

- Predstavljaju član skupa elemenata domena ne specifirajući pri tome neki poseban element. Obeležavaju se malim slovima.
- bolest - nespecificirana bolest
- težina - bez specifikacije

Funkcije

- Identifikuje element domena kao jedinstven rezultat preslikavanjem elemenata u domenu:

  ime_funkcije(argumenti)

- Argumenti predstavljaju izraze preko kojih se identifikuje element domena (promenljiva, konstanta, funkcija).
- otac(PETAR) - jedinstveni element koji je otac PETRA
- otac(otac(PETAR)) - jedinstveni element koji je otac PETROVOG oca - deda

Predikat

- Koristi se za predstavljanje relacija unutar domena. Označava da je neki element povezan sa drugim u domenu na određeni način. Vrednosti su TRUE i FALSE.
- Pri označavanju koriste se velika slova. Zajedno sa izrazima koji identifikuju povezane elemente koriste se za stvaranje atomskih formula ili atoma kao osnovnih elemenata u predikatskoj logici.
- PTICA(albatros)
- NIŽI(PETAR, MILAN)
- Kada se biraju imena predikata treba se opredeliti na ona koja imaju mnemoničko značenje. Prvi primer ne bi trebao da glasi P(albatros).

Veznici

- Za izgradnju složenijih formula koriste se veznici, i to $\wedge$, $\vee$, $\rightarrow$, $\cong$, $\sim$.
- $\wedge$ - LETI (albatros) $\wedge$ LEŽE_JAJA(albatros) – koristi se za indikaciju da svaka komponenta u formuli mora da bude istinita, da bi cela formula bila istinita. Cela se formula naziva konjunkcija, a svaka komponenta konjunkt.
- $\vee$ - KUPIO(PETAR, džemper) $\vee$ KUPIO(PETAR, pulover) - koristi se za formule čija istinitost zavisi od bilo koje komponente. Svaka komponenta se naziva disjunkt, a ceo izraz disjunkcija.
- $\rightarrow$ - UKLJUČEN(prekidač) $\wedge$ PRIKLJUČEN(kabl) $\rightarrow$ RADI(uređaj) - veznik kojim se gradi if-then konstrukcija, naziva se implikacija. Ako je preduslov tačan, podrazumeva se da je i posledica istinita.
- $\cong$ - označava logičku ekvivalenciju dve formule. X $\cong$ Z označava da su stanja istinitosti leve i desne strane ekvivalentne.
- $\sim$ - veznik kojim se menja istinitost komponente, TRUE postaje FALSE, a FALSE TRUE

Kvantifikatori

- Univerzalni kvantifikator $\forall X$ se koristi da označi da je formula istinita za sve vrednosti pridružene promenljive.

$$\forall X\ [PERJE(X) \rightarrow PTICA(X)]$$

- Egzistencijalni kvantifikator $\exists X$, koji se koristi da označi da postoji bar jedno X za koje je formula istinita.

$$\exists X\ [PTICA(X)]$$

- Opseg važenja kvantifikatora je formula koja sledi. Promenljive koje se pojavljuju u kvantifikatorima se nazivaju vezane, inače su slobodne.

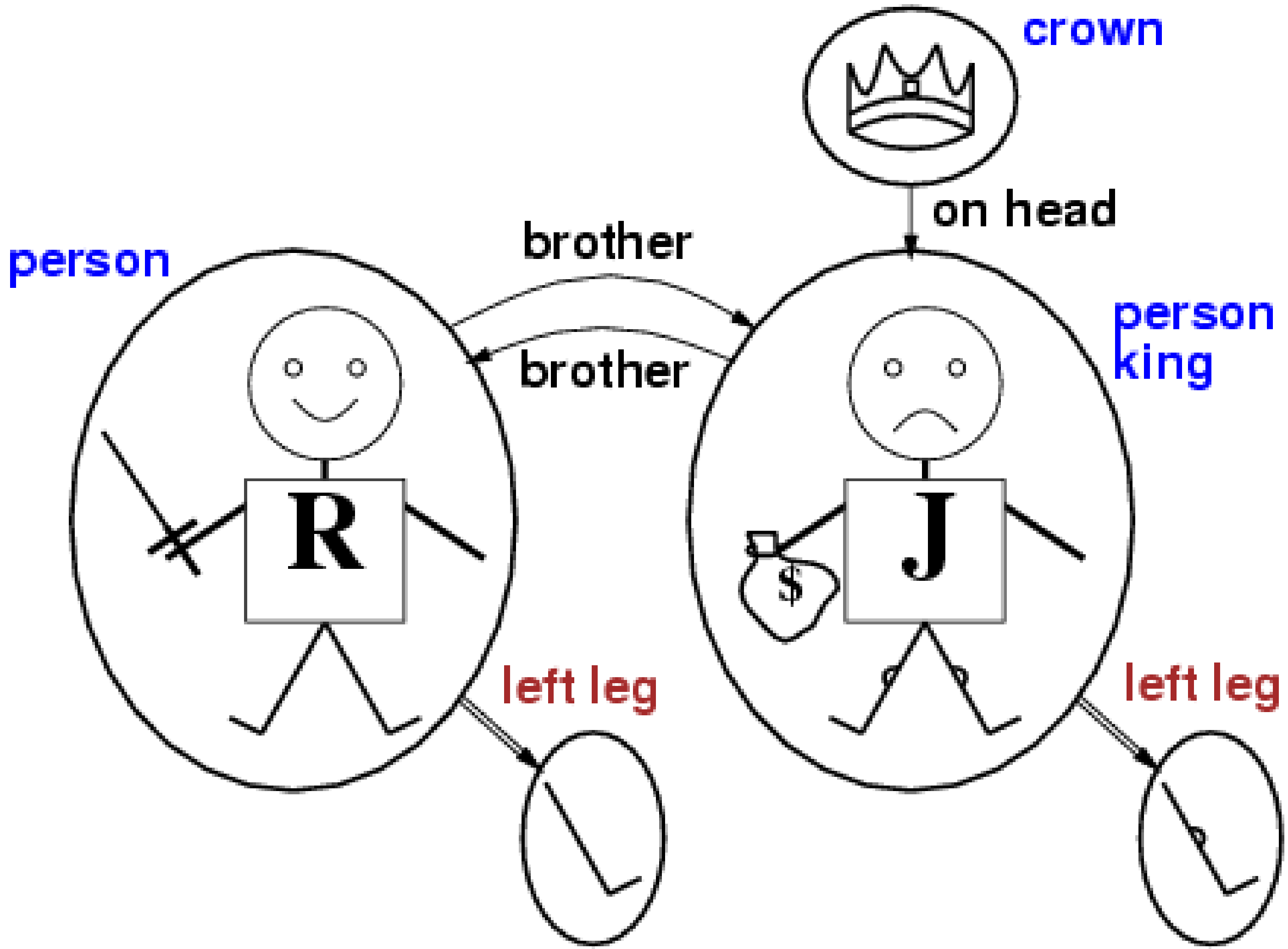
Osobine kvantifikatora

- $\forall x\ \forall y$ je isto kao i $\forall y\ \forall x$
- $\exists x\ \exists y$ je isto kao i $\exists y\ \exists x$

- $\exists x\ \forall y$ nije isto kao i $\forall y\ \exists x$
- $\exists x\ \forall y$ Loves(x,y)
  - “Postoji osoba koja voli svakog na svetu”
- $\forall y\ \exists x$ Loves(x,y)
  - “Svako na svetu je voljen od strane bar jedne osobe”

- Dualnost:
- $\forall x$ Likes(x,IceCream) $\neg\exists x\ \neg$Likes(x,IceCream)
- $\exists x$ Likes(x,Broccoli) $\neg\forall x\ \neg$Likes(x,Broccoli)

Primer


crown
on head
brother
brother
person
person
king
R
J
$
left leg
left leg

Dobro formirana formula WFF

- Atomska formula je WFF.
- Atomske formule povezane sa $\wedge$, $\vee$, $\rightarrow$, $\sim$ su WFF (na primer $\sim A$, $A \wedge B$, $B \vee C$, $C \rightarrow D$).
- Atomske formule okružene kvantifikatorima su takođe WFF
- Postoje specijalni slučajevi WFF:
  – WFF u kojoj su vezane promenljive i nazivaju se sentence,
  – WFF koja se sastoji od disjunkcije literala naziva se klauzula.
  – direktno, matematici slično zaključivanje

Primer

- Svet ljudoždera: opažaj mirisa i strujanja (nema sjaja) u *t=5*:

  `Tell`(KB,Percept([Smell,Breeze,None],5))
  `Ask`(KB,∃a BestAction(a,5))

- Da li će baza znanja dovesti do najbolje akcije u *t=5*?

- Odgovor: *da*, {*a/Shoot*} ← substitucija, zamena

- Dat je izraz *S* i zamena σ,
- *S*σ je rezultat primene zamene σ u okviru *S*;
  S = Pametniji(x,y)
  σ = {x/supruga,y/suprug}
  *S*σ = Pametniji(supruga,suprug)

- `Ask`(KB,S) vraća neke/sve σ takve da je KB $\models$ σ

Baza znanja primera

- Percepcija
  - $\forall t,s,b$ Percept([s,b,Glitter],t) $\Rightarrow$ Glitter(t)

- Refleksno ponašanje
  - $\forall t$ Glitter(t) $\Rightarrow$ BestAction(Grab,t)

Otkrivanje skrivenih osobina

- $\forall$x,y,a,b *Susedni*([x,y],[a,b]) $\Leftrightarrow$ [a,b] $\in$ {[x+1,y], [x-1,y],[x,y+1],[x,y-1]}

Osobine soba:

- $\forall$s,t *At*(Agent,s,t) $\wedge$ Breeze(t) $\Rightarrow$ Breezy(s)

Oseća se strujanje u sobama pored rupa:

- Dijagnostička pravila---zaključiti o uzroku na osnovu efekta

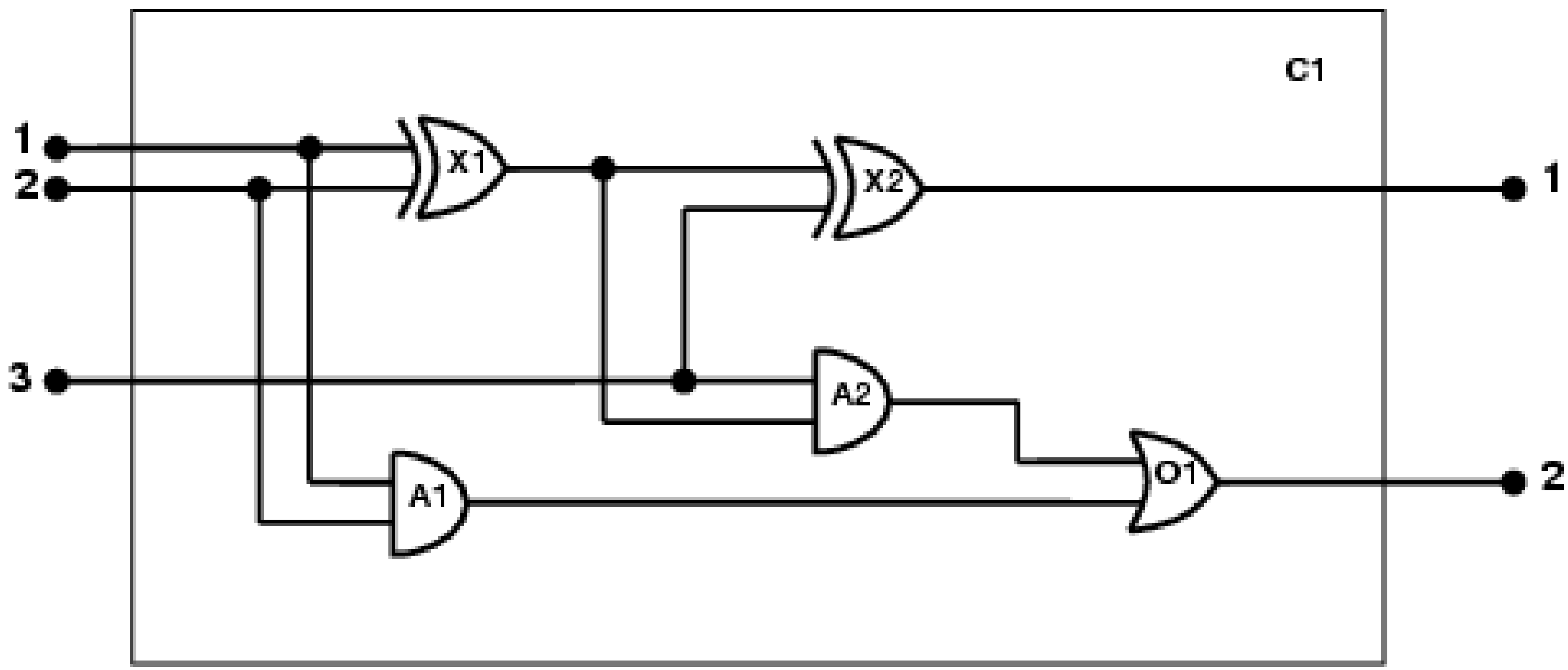
  $\forall$s Breezy(s) $\Rightarrow$ [$\exists$r Susedni(r,s) $\wedge$ Pit(r)]

- Uslovna pravila--- zaključiti o efektu na osnovu uzorka

  $\forall$r Pit(r) $\Rightarrow$ [$\forall$s Adjacent(r,s) $\Rightarrow$ Breezy(s)]

Domen logičkih kola

Potpuni sabirač



Domen logičkih kola

1. Identifikovati zadatak
   – Da li kolo radi ispravno? (verifikacija)
2. Primeniti relevantno znanje
   – Mreža od žica i elemenata; Tipovi elemenata (AND, OR, XOR, NOT)
   – Irelevantno: veličina, oblik, boja, cena
3. Rečnik pojmova
   – Alternative:

     $Type(X_1) = XOR$

     $Type(X_1, XOR)$

     $XOR(X_1)$

Domen logičkih kola

4. Prikazati znanje u okviru domena
   - $\forall t_1,t_2$ Connected(t_1, t_2) $\Rightarrow$ Signal(t_1) = Signal(t_2)
   - $\forall t$ Signal(t) = 1 $\vee$ Signal(t) = 0
   - 1 ≠ 0
   - $\forall t_1,t_2$ Connected(t_1, t_2) $\Rightarrow$ Connected(t_2, t_1)
   - $\forall g$ Type(g) = OR $\Rightarrow$ Signal(Out(1,g)) = 1 $\Leftrightarrow$ $\exists n$ Signal(In(n,g)) = 1
   - $\forall g$ Type(g) = AND $\Rightarrow$ Signal(Out(1,g)) = 0 $\Leftrightarrow$ $\exists n$ Signal(In(n,g)) = 0
   - $\forall g$ Type(g) = XOR $\Rightarrow$ Signal(Out(1,g)) = 1 $\Leftrightarrow$ Signal(In(1,g)) ≠ Signal(In(2,g))
   - $\forall g$ Type(g) = NOT $\Rightarrow$ Signal(Out(1,g)) ≠ Signal(In(1,g))

Domen logičkih kola

5. Definisati specifična stanja problema

$Type(X_1) = XOR$ | $Type(X_2) = XOR$
$Type(A_1) = AND$ | $Type(A_2) = AND$
$Type(O_1) = OR$

$Connected(Out(1,X_1),In(1,X_2))$
$Connected(Out(1,X_1),In(2,A_2))$
$Connected(Out(1,A_2),In(1,O_1))$
$Connected(Out(1,A_1),In(2,O_1))$
$Connected(Out(1,X_2),Out(1,C_1))$
$Connected(Out(1,O_1),Out(2,C_1))$

$Connected(In(1,C_1),In(1,X_1))$
$Connected(In(1,C_1),In(1,A_1))$
$Connected(In(2,C_1),In(2,X_1))$
$Connected(In(2,C_1),In(2,A_1))$
$Connected(In(3,C_1),In(2,X_2))$
$Connected(In(3,C_1),In(1,A_2))$

Domen logičkih kola

6. Postaviti pitanja za procedure nasleđivanje

   Koje su moguće vrednosti ulaza potpunog sabirača?

   $\exists i_1,i_2,i_3,o_1,o_2$ Signal(In(1,C_1)) = $i_1 \wedge$ Signal(In(2,C_1)) = $i_2 \wedge$ Signal(In(3,C_1)) = $i_3 \wedge$ Signal(Out(1,C_1)) = $o_1 \wedge$ Signal(Out(2,C_1)) = o_2

7. Osvežiti bazu znanja

   Moguće je zaboraviti na činjenice kao što je

   $1 \neq 0$

Tehnike znanja formalne logike

- Razviti razumevanje znanja.
- Formulisati znanje iskazano na prirodnom jeziku.
- Razložiti iskaze na njihove komponentne delove.
- Izabrati simbole da predstavljaju elemente i veze u svakoj komponenti.
- Izgraditi WFF sistem koristeći izabrane simbole, koje predstavljaju iskaze.

Primer

- Svaki glasač x ili podržava zakonski predlog ZP ili ga odbacuje:
- $\forall$x (GLASAČ(x) $\rightarrow$ ([PODRŽAVA(x, ZP) $\vee$ ODBACUJE(x, ZP)] $\wedge$ ~[PODRŽAVA(x, ZP) $\wedge$ ODBACUJE(x, ZP)])
- Dobijena je formula oblika (A $\vee$ B) $\wedge$ ~(A $\wedge$ B)

Zaključivanje u formalnoj logici

- aksiome,
- teoreme,
- procedure dokazivanja,
- zdravo pravilo zaključivanja

Aksiome i teoreme

$$\forall X\ [PERJE(X) \rightarrow PTICA(X)]$$

$$PERJE(golubovi)$$

- Obe ove formule su istinite, te se nazivaju aksiome.
- Ako se želi dokazati na osnovu aksioma da mora da važi

$$PTICA(golubovi)$$

- Teoreme u odnosu na prethodno definisane aksiome.

Zaključivanje

Procedura dokazivanja

- Način na koji se dokazuje teorema naziva se procedura dokazivanja.

Zdravo pravilo zaključivanja

- Procedure dokazivanja koriste manipulacije, koje se nazivaju zdrava pravila zaključivanja, koje generišu nove WFF iz starih tako da ako su stari izrazi TRUE, garantovano je da će i novi izrazi biti TRUE.

Modus ponens

Ako postoji aksioma oblika

$E1 \rightarrow E2$ i postoji aksioma E1, tada E2 logički sledi.

- **Ako je E2 teorema koju treba dokazati, onda je sve gotovo. Ako nije E2 se dodaje grupi teorema. Povećavajući listu aksiome, može se eventualno pokazati daje neka teorema tačna.**
- **Ako je E1:**

$$E1 = [P1 \wedge (P1 \rightarrow P2)] \rightarrow P2$$

- **Gde je P1 – dim, a P2 – vatra, onda se može zaključiti o prisustvu vatre opažajući dim.**

Rezolucija

Ako postoji aksioma oblika E1 $\vee$ E2 i postoji druga aksioma oblika ~E2 $\vee$ E3, tada logički sledi i E1 $\vee$ E3. Dobijeni izraz se naziva rezolvent.

Rezolucija se može proveriti preko sledećih primera:

- E2 = TRUE, ako važi prethodni izraz tada je ~E2 = FALSE, pa sledi da E3 mora da bude jednako TRUE, na osnovu čega se može zaključiti da je rezolvent istinit.
- Ako je E2 = FALSE, tada zbog prve aksiome E1 mora biti TRUE, što opet dokazuje da je rezolvent istinit, dok god su E1 $\vee$ E2 i ~E2 $\vee$ E3 istiniti.
- Na osnovu rezolucije sledi da u izrazu može biti prozvoljan broj disjunkta u bilo kojoj aksiomi, s tim što jedna askioma mora sadržavati negaciju nekog disjunkta u drugoj aksiomi.

Rezolucija - primer

- **Dokazati PTICA(golubovi), na osnovu aksioma $\forall$X [PERJE(X) $\rightarrow$ PTICA(X)] i PERJE(golubovi).**
- **Dokaz: Ako važi $\forall$X [PERJE(X) $\rightarrow$ PTICA(X)], važi i za X = golubovi, pa se može smatrati PERJE(golubovi) $\rightarrow$ PTICA(golubovi), a na osnovu zakona ekvivalencije dobijamo ~ PERJE(golubovi) $\vee$ PTICA(golubovi). Ako se uporede druga aksioma i dobijeni izraz može se zaključiti da su dobijene dve rezolucije na koje se može primeniti rezolucija, i kao rezultat se dobija zaključak da važi PTICA(golubovi).**

Modus tolens

- **Ako postoji aksioma oblika E1 → E2 i postoji aksioma oblika ~E2, tada logički sledi ~E1.**
- **Prethodni primer u kome je dokazano da važi PTICA(golubovi), na osnovu aksioma ∀X [PERJE(X) → PTICA(X)] i PERJE(golubovi), može se dokazati i pomoću ovog pravila zaključivanja.**

Razlaganje aksioma

- **izabrati dve aksiome koje sadrže isti disjunkt, jedanput u pozitivnoj, a jedanput u negativnoj formi,**
- **formirati novu aksiomu (rezolvent) disjunkcijom svih literala iz polaznih aksioma.**
- **Iterativna primena ove tehnike je osnova rezolucije**
- Pristup da negacija rezultata teoreme, ne može da bude istinita. Ovaj princip sa naziva dokazivanje opovrgavanjem, odnosno resolution by refutation

Dokazivanje opovrgavanjem

- pretpostaviti da je negacija teoreme istinita i dodati je postojećim aksiomama,
- pokazati da aksiome i pretpostavljena negacija teoreme zajedno pretpostavlju nešto da je istinito, a što sigurno ne može da bude istinito,
- zaključiti da pretpostavljena negacija teoreme ne može da bude istinita, jer vodi ka kontradikciji,
- zaključiti da teorema mora da bude istinita, jer pretpostavljena negacija teoreme ne može da bude istinita.

Primer

- Dokazati: PTICA(golubovi), na osnovu aksioma $\forall$X [PERJE(X) $\rightarrow$ PTICA(X)] i PERJE(golubovi).
- Dokaz: Ako važi $\forall$X [PERJE(X) $\rightarrow$ PTICA(X)], važi i za X = golubovi, pa se može smatrati PERJE(golubovi) $\rightarrow$ PTICA(golubovi), odnosno ~PERJE(golubovi) $\vee$ PTICA(golubovi).
- Dodajući negaciju na teoremu koju treba dokazati: ~ PERJE(golubovi) $\vee$ PTICA(golubovi), PERJE(golubovi) i ~PTICA(golubovi),
- Razlažući prvu i drugu aksiomu dobija se ~PERJE(golubovi) $\vee$ PTICA(golubovi), PERJE(golubovi), PTICA(golubovi) i ~PTICA(golubovi) –kontradikcija.
- Sledi ne važi ~PTICA(golubovi)

Kako do CNF

- eliminisati implikacije,
- premestiti negacije na atomske formule,
- ukloniti egzistencijalne kvantifikatore. Mora da postoji jedan argument za svaku univerzalno kvantifikovanu promenljivu čiji opseg sadrži egzistencijalnu kvantifikovanu promenljivu i koju treba zameniti funkcijom.
- preimenovati promenljive, kad je neophodno, tako da nema dve promenljive sa istim imenom. Preimenovati duplikate u svakom izrazu tako da kvantifikator ima jedinstveno ime.
- pomeriti univerzalni kvantifikator ulevo,
- distribuirati disjunkcije duž literala,
- eliminisati konjunkcije, mada to nije prava eliminacije, već se svaki deo konjunkcije posmatra kao aksioma,
- preimenovati sve promenljive, tako da nema dve iste promenljive,
- ukloniti univerzalne kvantifikatore, usvaja se da su promenljive univerzalni kvantifikatori

Primer: CNF

1. Cigla je na nečemu što nije piramida.
2. Nema ništa na čemu je cigla, a da je to isto takođe na cigli.
3. Nema ništa drugo što nije cigla, a isto je i što cigla.

$$\forall x \{ \text{Cigla}(x) \Rightarrow \{ \exists y [\text{Na}(x,y) \wedge \neg \text{Piramida}(y)] \wedge$$

$$\neg\exists y [\text{Na}(x,y) \wedge \text{Na}(y,x)] \wedge$$

$$\wedge \forall y [\neg\text{Cigla}(y) \Rightarrow \neg\text{Jednako}(x,y)] \}\}$$

Primer: CNF

$E_1 \Rightarrow E_2$ transformiše se u $\neg E_1 \vee E_2$

Sledi

$\forall x$ [$\neg$Cigla(x) $\vee$ ($\exists y$ [Na(x,y) $\wedge$ $\neg$ Piramida(y)] $\wedge$ $\neg\exists y$ [Na(x,y) $\wedge$ Na(y,x)] $\wedge$ $\wedge$ $\forall y$ [$\neg$($\neg$Cigla(y)) $\vee$ $\neg$Jednako(x,y)]])]

Primer: CNF

2. 'Spuštanje' negacija do atomskih formula

($\neg(E_1 \wedge E_2)$ transformiše se u $\neg E_1 \vee \neg E_2$,

$\neg(E_1 \vee E_2)$ transformiše se u $\neg E_1 \wedge \neg E_2$,

$\neg(\neg E_1)$ transformiše se u E_1 ,

$\neg\forall x\ [\ E_1(x)\ ]$ transformiše se u $\exists x\ [\ \neg E_1(x)\ ]$,

$\neg\exists x\ [\ E_1(x)\ ]$ transformiše se u $\forall x\ [\ \neg E_1(x)\ ]$)

Sledi

$\forall x\ [\ \neg Cigla(x) \vee (\exists y\ [\ Na(x,y) \wedge \neg Piramida(y)\ ] \wedge \forall y\ [\ \neg Na(x,y) \vee \neg Na(y,x)\ ] \wedge$

$\wedge\ \forall y\ [Cigla(y) \vee \neg Jednako(x,y)\ ]\ )]$

Primer: CNF

$\forall x\ \exists y\ [\ Na(x,y) \wedge \neg Piramida(y)\ ].$

$Na(x,\Phi(x)) \wedge \neg Piramida(\Phi(x))$

$\forall x\ [\ \neg Cigla(x) \vee (\ \ Na(x,Drži(x)) \wedge \neg Piramida(Drži(x))\ \ \wedge$
$\forall y\ [\ \neg Na(x,y) \vee \neg Na(y,x)\ ]$
$\wedge\ \forall y\ [Cigla(y) \vee \neg Jednako(x,y)\ ])]$

Primer: CNF

4. Preimenovanje promenljivih tako da svakom kvantifikatoru odgovara posebna promenljiva

$\forall x\ [\ \neg Cigla(x) \vee (\ \ Na(x,Drži(x)) \wedge \neg Piramida(Drži(x))\ \ \wedge \forall y\ [\ \neg Na(x,y) \vee \neg Na(y,x)\ ]$

$\wedge\ \forall z\ [Cigla(z) \vee \neg Jednako(x,z)\ ])]$

Primer: CNF

5. Premeštanje svih univerzalnih kvantifikatora na levu stranu bez promene njihovog redosleda

$\forall x\ \forall y\ \forall z\ [\ \neg Cigla(x) \vee (\ Na(x,Drži(x)) \wedge \neg Piramida(Drži(x))\ \wedge$

$\wedge\ [\ \neg Na(x,y) \vee \neg Na(y,x)\ ] \wedge [Cigla(z) \vee \neg Jednako(x,z)\ ])]$

Primer: CNF

6. (E1 $\wedge$ E2) $\vee$ E3 transformiše se u (E1 $\vee$ E3) $\wedge$ (E2 $\vee$ E3))

Sledi

$\forall x\ \forall y\ \forall z$ [($\neg$Cigla(x) $\vee$ Na(x,Drži(x))) $\wedge$ ($\neg$Cigla(x) $\vee$ $\neg$Piramida(Drži(x))) $\wedge$

$\wedge$ ($\neg$Cigla(x) $\vee$ $\neg$Na(x,y) $\vee$ $\neg$ Na(y,x)) $\wedge$

$\wedge$ ($\neg$Cigla(x) $\vee$ Cigla(z) $\vee$ $\neg$ Jednako(x,z)))]

Primer: CNF

7. $\forall x\,[\, \neg Cigla(x) \vee Na(x, Drži(x))\,]$

$\forall x\,[\, \neg Cigla(x) \vee \neg Piramida(Drži(x))\,]$

$\forall x\, \forall y\,[\, \neg Cigla(x) \vee \neg Na(x,y) \vee \neg Na(y,x)\,]$

$\forall x\, \forall z\,[\, \neg Cigla(x) \vee Cigla(z) \vee \neg Jednako(x,z))\,]$

Primer: CNF

$\forall x$ [$\neg$Cigla(x) $\vee$ Na(x,Drži(x))]

$\forall u$ [$\neg$Cigla(u) $\vee$ $\neg$Piramida(Drži(u))]

$\forall v\ \forall y$ [$\neg$Cigla(v) $\vee$ $\neg$Na(v,y) $\vee$ $\neg$Na(y,v)]

$\forall w\ \forall z$ [$\neg$Cigla(w) $\vee$ Cigla(z) $\vee$ $\neg$ Jednako(w,z))]

Primer: CNF

9. $\neg$Cigla(x) $\vee$ Na(x,Drži(x))

$\neg$Cigla(u) $\vee$ $\neg$Piramida(Drži(u))

$\neg$Cigla(v) $\vee$ $\neg$Na(v,y) $\vee$ $\neg$Na(y,v)

$\neg$Cigla(w) $\vee$ Cigla(z) $\vee$ $\neg$Jednako(w,z))

Teoreme i rezolucija

Postupak za dokazivanje teoreme koristeći rezoluciju glasi:

- negirati teoremu koju treba dokazati i dodati rezultat na listu aksioma,
- dovesti listu aksioma u klauzulnu formu,
- dok se ne dobije prazna klauzula NIL, ili dok se ne dobije par klauzula koje se ne mogu razlagati, naći razlaganje klauzula, primeniti i dodavati rezultat na listu klauzula,
- ako se dobije prazna klauzula, obavestiti da je teorema istinita u suprotnom, obavestiti da je neistinita.

Primer: jabuke I kruške

Na osnovu sledećih pretpostavki

1. $\forall x\ [\neg \text{Jednako}(x, x+1)]$
2. Jednako(2,3)

rezolucijom izvesti zaključak:

Sve jabuke su kruške.

Primer: jabuke i kruške

$\forall x$ [Jabuka(x) $\Rightarrow$ Kruška(x)]

1. $\neg$Jednako(x,x+1)

2. Jednako(2,3)

3'. Jabuka(C)

3''. $\neg$Kruška(C)

1., 2. $\xrightarrow{x=2}$ NIL

Unifikacija

- Proces unifikacije pomaže kada treba odrediti da li dva literala u klauzulama koje se razlažu mogu da se učine identičnim, t.j. da se ponište.
- Unifikacija se usredsređuje na zamenu promenljivih, funkcijskih izraza za promenljive u literale.
- Primerak zamene je literal koji proizilazi iz takve zamene

Procedura unifikacije:

- predstaviti svaki predikat kao listu u kojoj je predikatski simbol prvi element iz koga slede njegovi argumenti tačno po redosledu,
- napustiti ako dve liste nisu iste dužine,
- obaviti poređenje elemenata iz lista koristeći sledeće pravilo:
  - predikatski i funkcionalni simboli, kao i konstante moraju da budu tačno upareni,
  - za promenljive, izvesti uparivanje putem zamene, posebno, kada se naiđe na promenljivu, zameniti je kao i sva njena sledeća dešavanja u listi, odgovarajućim elementima iz druge liste; ograničenje u pogledu uparivanja je da promenljiva ne može da bude zamenjena izrazom koji sadrži istu promenljivu
- dva predikata su unificirana ako se svi elementi poklapaju; rutina se može pozivati rekurzivno, radi procene elemenata liste, koji su ugnježđeni listovi

function UNIFY(x, y, θ) **returns** a substitution to make x and y identical
  inputs: x, a variable, constant, list, or compound
        y, a variable, constant, list, or compound
        θ, the substitution built up so far (optional, defaults to empty)

  if θ = failure **then return** failure
  else if $x = y$ **then return** θ
  else if VARIABLE?(x) **then return** UNIFY-VAR(x, y, θ)
  else if VARIABLE?(y) **then return** UNIFY-VAR(y, x, θ)
  else if COMPOUND?(x) **and** COMPOUND?(y) **then**
    return UNIFY(ARGS[x], ARGS[y], UNIFY(OP[x], OP[y], θ))
  else if LIST?(x) **and** LIST?(y) **then**
    return UNIFY(REST[x], REST[y], UNIFY(FIRST[x], FIRST[y], θ))
  else return failure

---

function UNIFY-VAR(var, x, θ) **returns** a substitution
  inputs: var, a variable
        x, any expression
        θ, the substitution built up so far

  if $\{var/val\} \in \theta$ **then return** UNIFY(val, x, θ)
  else if $\{x/val\} \in \theta$ **then return** UNIFY(var, val, θ)
  else if OCCUR-CHECK?(var, x) **then return** failure
  else return add $\{var/x\}$ to θ