Rezolucija za propozicijsku logiku

Smrk, Petra

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Petra Smrk

REZOLUCIJA ZA PROPOZICIJSKU LOGIKU

ZAVRŠNI RAD

Varaždin, 2021.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Petra Smrk

JMBAG: 0016131287

Studij: Informacijski sustavi

REZOLUCIJA ZA PROPOZICIJSKU LOGIKU

ZAVRŠNI RAD

Mentor/Mentorica:

Prof. dr. sc. Sandra Lovrenčić

Varaždin, rujan 2021.

Petra Smrk

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sadržaj

Sažetak

Tema ovog završnog rada je rezolucija za propozicijsku logiku. Kako bi se lakše objasnila rezolucija, na početku je pojašnjena sama propozicijska logika, objašnjene su njena sintaksa i semantika, te su navedene tablice istinitosti koje se koriste pri zaključivanju. Također, objašnjeni su pojmovi tautologije, kontradikcije, implikacije i ekvivalencije koji se koriste pri opisivanju metoda rezolucije. Kako bi se mogla primijeniti rezolucija potrebno je formule pretvoriti u klauzalni oblik, što se postiže primjenom pravila za pretvorbu u klauzalni oblik i to redoslijedom kojim su navedeni.

Nadalje, opisuje se metoda rezolucije, kao i pojam klauzula koji se koristi u primjerima, dokazima i objašnjenjima. Metoda kojom se dolazi do logičkog izvoda pomoću rezolucije naziva se rezolucijsko pravilo koje je potrebno slijediti kako bi rješavanje rezolucijom bilo valjano. Valjanost formule opisuje njeno stanje odnosno je li formula istinita ili lažna. Metateoremima adekvatnosti i potpunosti utvrđuju se ispravnost i potpunost formula.

Na kraju su navedene vrste rezolucija te gdje se rezolucija primjenjuje i kakva je njena korisnost.

Ključne riječi: propozicijska logika; rezolucija; klauzula; dokaz; izvod; rezolventa;

1. Uvod

Ovaj rad bavi se rezolucijom za propozicijsku logiku i cilj je pobliže objasniti rezoluciju kao metodu kojom se primjenom određenih pravila može odrediti zadovoljivost formula. Kako bi se objasnio princip rezolucije, ponajprije je potrebno objasniti osnove propozicijske logike od kojih se polazi. Propozicijska logika kao takva prati niz pravila kojima se dolazi do zaključaka pa su tako tablice istinitosti vrlo važne pri procesu pronalaska izvoda.

Rezolucija koristi klauzule odnosno klauzalni oblik zadanih formula koje treba izvesti do krajnjeg cilja. Pri takvoj pretvorbi potrebno je pratiti svojstva odnosno pravila pretvorbe: rješavanje implikacije i ekvivalencije, negacija, distributivnost te uklanjanje operatora. Nakon pretvorbe može započeti proces rezolucije. Sama rezolucija prilično je intuitivan proces zaključivanja, no pri tom procesu potrebno je paziti da se ne izgubi svojstvo potpunosti. U nekim, kasnije navedenim, tipovima rezolucije dolazi do gubitka tog svojstva potpunosti što rezoluciju čini beskorisnom pa je potrebno pronaći neki drugi pristup, no moguće i je i povećanje učinkovitosti pri korištenju istih. Također, rezolucija ima veliku primjenu u području umjetne inteligencije.

2. Propozicijska logika

Logika sudova to jest propozicijska logika temelji se na pretpostavci kojom se može opisati istinitost činjenica. Činjenice su u obliku izjavnih rečenica, elementarne su i nedjeljive, a još se nazivaju i elementarne propozicije ili atomi. Propozicijska logika definira se sintaksom, semantikom i teorijom dokaza. [1]

Činjenicama se određuje istinitost, dakle one mogu biti istinite ili lažne. Kako bi se istražila složenost istinitosti činjenica se označava sa simbolom (najčešće veliko tiskano slovo) te se ne gleda njeno pravo značenje nego samo istinitost.[9] Na primjer: „Papiga je ptica“ je činjenica i to istinita činjenica, u propozicijskoj logici ta činjenica bit će jednaka nekom simbolu. Dakle „Papiga je ptica“ = F, takvim pristupom dolazi se do temeljne istinitosti bez preispitivanja značenja neke činjenice. U nastavku su detaljnije objašnjene oznake i sintaksa koja se koristi u propozicijskoj logici.

2.1. Sintaksa i semantika

Formalnim jezikom (niz simbola koji tvori formulu) definira se sintaksa, a simboli koje propozicijska logika koristi dio su takozvane abecede, koja je definirana na sljedeći način [7]:

1. Atomi (elementarne propozicije) *A*= {A, B, C, …}, čine skup simbola koji se može prebrojati, simboli su velika tiskana slova.
2. Logički operatori (logički veznici) *V*={¬, ∧, ∨,→, ↔}, a to su unarni operator *negacije* (¬) i binarni operatori *i* (∧), *ili* (∨), *implikacija* (→) i *ekvivalencija* (↔).
3. Zagrade *Z*= { (,) } [7]

Pri definiranju formula javlja se pojam dobro oblikovana formula (eng. *well-formed formula (wff))* iz čijeg se naziva može zaključiti kako formule moraju biti oblikovane kako bi bile valjane.[1]

Takve formule rekurzivno su definirane sljedećim pravilima [1]:

1. Atom je formula.
2. Ako je F formula tada je i (¬F) formula, a zagrade se mogu i izostaviti.
3. Ako su F i G formule tada su formule: (F ∧ G), (F ∨ G), (F → G), (F ↔ G), a zagrade se mogu i izostaviti.
4. Ništa drugo nije dobro oblikovana formula ako nije primijenjeno neko od navedenih pravila (1. - 3.). [1]

Semantika propozicijske logike daje značenje sintaksi, točnije semantika određuje značenje formula, je li ona istinita ili lažna. Dodjeljivanje značenja formulama se naziva interpretacija formule. Ako formula ima n atoma, onda ima 2^n različitih interpretacija.
Istinitost formula određuje se pomoću tablica istinitosti [1]:

Tablica 1: Negacija

F	$\neg F$
⊤	⊥
⊥	⊤

Tablica 2 Konjunkcija

F	G	$F \wedge G$
⊤	⊤	⊤
⊤	⊥	⊥
⊥	⊤	⊥
⊥	⊥	⊥

Tablica 3 Disjunkcija

F	G	$F \vee G$
⊤	⊤	⊤
⊤	⊥	⊤
⊥	⊤	⊤
⊥	⊥	⊥

Tablica 4 Implikacija

F	G	$F \rightarrow G$
⊤	⊤	⊤
⊤	⊥	⊥
⊥	⊤	⊤
⊥	⊥	⊤

Tablica 5 Ekvivalencija

F	G	$F \leftrightarrow G$
⊤	⊤	⊤
⊤	⊥	⊥
⊥	⊤	⊥
⊥	⊥	⊤

2.1.1. Tautologija i kontradikcija

Formula se smatra tautologijom ako i samo ako je ona istinita u svakoj svojoj interpretaciji, a kontradikcijom ako i samo ako je ona neistinita u svakoj svojoj interpretaciji. Ako i samo ako je formula tautologija, negacija te formule je kontradikcija. Isto tako ako i samo ako je formula tautologija ona je i konzistentna. Konzistentne (zadovoljive) formule su one formule koje su istinite u barem jednoj interpretaciji. [2]

Primjer: Ako je dana formula $(F \vee G) \vee (\neg F \vee \neg G)$, tablica istinitosti ove formule je:

Tablica 6 Vlastiti primjer tautologije pomoću tablica istinitosti

F	G	$F \vee G$	$\neg F$	$\neg G$	$\neg F \vee \neg G$	$(F \vee G) \vee (\neg F \vee \neg G)$
⊤	⊤	⊤	⊥	⊥	⊥	⊤
⊤	⊥	⊤	⊥	⊤	⊤	⊤
⊥	⊤	⊤	⊤	⊥	⊤	⊤
⊥	⊥	⊥	⊤	⊤	⊤	⊤

Ovaj primjer prikazuje tautologiju, dakle u svakoj interpretaciji formule ona se javlja kao istinita.

Promjenom operatora konjunkcije u operator disjunkcije u prethodnom primjeru, tablica istinitosti prikazati će kontradikciju formule, znači svaka interpretacija je neistinita.

Primjer: $(F \wedge G) \wedge (\neg F \wedge \neg G)$, a primjer tablice istinitosti izgleda ovako:

Tablica 7 Vlastiti primjer kontradikcije pomoću tablica istinitosti

F	G	$F \wedge G$	$\neg F$	$\neg G$	$\neg F \wedge \neg G$	$(F \wedge G) \wedge (\neg F \wedge \neg G),$
⊤	⊤	⊤	⊥	⊥	⊥	⊥
⊤	⊥	⊥	⊥	⊤	⊥	⊥
⊥	⊤	⊥	⊤	⊥	⊥	⊥
⊥	⊥	⊥	⊤	⊤	⊤	⊥

2.1.2. Implikacija i ekvivalencija

Implikacija ili pogodbeni sud (Tablica 4) ukazuje na sljedeće: atom F implicira atom G, kada je F istinit i G je istinit, s druge strane kada je F lažan implikacija nam ništa ne govori o G. Dakle nužno je da G bude istinit kako bi F bio istinit, ali je dovoljno da F bude istinit kako bi G bio istinit. [2]

Ekvivalencija (Tablica 5) glasi: F je ekvivalentan G ako i samo ako su vrijednosti tih dviju formula jednake (F je istinit i G je istinit ili F je lažan i G je lažan). Zapisuje se još kao i $F \equiv G$.[2]

Primjer: Neka je zadana formula: $\Big(\big((F \vee G) \rightarrow \neg F\big) \leftrightarrow (\neg F \wedge \neg G)\Big)$

Rješenje tablice istinitosti ove formule jei:

Tablica 8 Vlastiti primjer rješavanja formule pomoću tablica istinitosti

F	G	$F \vee G$	$\neg F$	$((F \vee G) \rightarrow \neg F)$	$\neg G$	$\neg F \wedge \neg G$	$((F \vee G) \rightarrow \neg F) \leftrightarrow (\neg F \wedge \neg G))$
⊤	⊤	⊤	⊥	⊥	⊥	⊥	⊤
⊤	⊥	⊤	⊥	⊥	⊤	⊥	⊤
⊥	⊤	⊤	⊤	⊤	⊥	⊥	⊥
⊥	⊥	⊥	⊤	⊤	⊤	⊤	⊤

Iz primjera se može uočiti da ova formula nije tautologija, jer je treća interpretacija neistinita. Također formula nije ni kontradikcija je nije u svakoj interpretaciji neistinita, ali ona je konzistentna jer je u ostalim interpretacijama ona istinita.

2.2. Dokazivanje

Postupak dokazivanja pomoću tablice istinitosti najtočniji je i njime se može doći do rezultata svaki put, no ako se u problemu pojavi više atoma samim time se i tablica mora povećati što postaje prilično nepregledno i neefikasno. Kako bi se izbjeglo pisanje takvih tablica postoje drugi načini kojima se može dokazati istinitost formula. [10] Neki od tih postupaka su: korištenje pravila zaključivanja, izravno zaključivanje, prirodno zaključivanje. Korištenjem pravila zaključivanja zadanim premisama to jest činjenicama se mijenja oblik pa tako postaju nove činjenice kojima se na jednostavniji način dolazi do dokaza.[2]
Jedno od važnijih pravila koje se koristi pri zaključivanju je pravilo *Modus ponens*. Ono se primjenjuje kada postoje dvije premise, jedna od njih je samo atom dok je u drugoj implikacija dva atoma od kojih je jedan onaj iz prve premise. Primjenom tog pravila dobiva se jedna logička posljedica.[2] To izgleda ovako[2]:

1. premisa: F
2. premisa: $F \rightarrow G$

Logička posljedica: G

S druge strane imamo pravilo *Modus tollens* koje također dvije premise pretvara u jednu. U jednoj premisi je negirani atom, a u dugom su dva atoma u implikaciji od kojih je jedan ne

negirani atom iz prve premise. Logička posljedica te dvije premise postaje negirani atom druge premise koji nije u prvoj. Dakle, dokaz će biti sljedećeg oblika[10]:

1. premisa: $\neg G$
2. premisa: $F \rightarrow G$

Logička posljedica: $\neg F$

Pravilo konjunkcije je najjednostavnije, ono dvije istinite premise spaja u jednu premisu s operatorom za konjunkciju[2]:

1. premisa: F
2. premisa: G

Logička posljedica: $F \wedge G$

Isto tako postoji pravilo koje eliminira konjunkciju i pojednostavljuje premisu, iz jedne premise mogu nastati dvije logičke posljedice.[10]

1. premisa: $F \wedge G$

Logička posljedica: F ili

Logička posljedica: G

Kako postoji eliminacija i uvođenje konjunkcije tako postoje i pravila uvođenja i eliminiranja disjunkcije, ona se uvode prema istom principu kao i pravila za konjunkciju[10]:

Uvođenje disjunkcije:

1. premisa: F
2. premisa: G

Logička posljedica: $F \vee G$

Eliminacija disjunkcije:

1. premisa: $F \vee G$

Logička posljedica: F ili

Logička posljedica: G

Pravilo ulančavanja odnosno silogizam spaja dvije implikacije u jednu pod uvjetom da obje premise imaju zajednički atom[10]:

$$\text{Uvođenje disjunkcije:}$$
$$1.\ \text{premisa: } F \to G$$
$$2.\ \text{premisa: } G \to H$$
$$\text{Logička posljedica: } F \to H$$

Izravnim zaključivanjem dolazi se do dokaza pomoću slijeda činjenica. Činjenice se zapisuju jedna ispod druge, isto tako ispod činjenica se zapisuju i njihovi dokazi, a zadnja činjenica je ona koja treba biti dokazana.[10] Za primjer neka su zadane premise $F \to G$ i F na kojima se može primijeniti jedno od gore navedenih pravila:

$1.\ F \to G$	premisa
$2.\ F$	premisa
$3.\ G$	modus ponens, 1,2

Kod prirodnog zaključivanja uz skup zadanih rečenica to jest činjenica postoji i ciljna činjenica odnosno ona koju treba dokazati da vrijedi. Taj dokaz postiže se primjenom pravila za zaključivanje pomoću kojih se izvode nove činjenice iz zadanih činjenica. [10]

Primjer: Neka su zadane činjenice: Ana gleda film. Ako Ana gleda film položila je ispit. Ako je Ana položila ispit, uspješno je završila akademsku godinu.
Iz činjenica je potrebno prepoznati atome:

$$F = \text{Ana gleda film}$$
$$G = \text{Ana je položila ispit}$$
$$H = \text{Ana je uspješno završila akademsku godinu}$$

Potrebno je dokazati da je Ana uspješno završila akademsku godinu odnosno treba dokazati da vrijedi H. Prvo treba zapisati premise jednu ispod druge te pomoću pravila zaključivanja izvesti nove formule pa tako i ciljnu formulu.

Na ovom primjeru to će izgledati ovako:

$1.\ F$	premisa
$2.\ F \to G$	premisa
$3.\ G \to H$	premisa
$4.\ G$	modus ponens, 1,2
$5.\ H$	modus ponens, 3,4

Ovime je dokazano da je H logička posljedica zadanih premisa.

3. Rezolucija

Metodu rezolucije razvio je John Alan Robinson 1965. godine, a danas se ta metoda koristi za automatsko zaključivanje (engl. *Automated Reasoning)*. Ovom metodom se određuje zadovoljivost formula. [3]

Metoda koristi atome (literale) i njihove negacije, a temeljni objekt je klauzula odnosno disjunkcija literala. [4] Ako klauzula ima samo jedan literal, ona se naziva jedinična klauzula, a ako je klauzula prazna označava se sa NIL. Rezolucija se može koristiti isključivo na formulama koje imaju oblik konjunkcija klauzula. [2]

Klauzule su sljedećeg oblika: $\{p\}, \{\neg p\}, \{\neg p, q\}$. [5]

3.1. Pretvorba u klauzalni oblik

Za pretvorbu formula u klauzalni oblik potrebno je slijediti pravila za pretvorbu. Pravila se primjenjuju redoslijedom kojim su navedena [5]:

1. Rješavanje implikacija i ekvivalencija:

$$F \rightarrow G \text{ slijedi } \neg F \vee G$$
$$F \leftarrow G \text{ slijedi } F \vee \neg G$$
$$F \leftrightarrow G \text{ slijedi } (\neg F \vee G) \wedge (F \vee \neg G)$$

2. Negacije:

$$\neg\neg F \text{ slijedi } F$$
$$\neg(F \wedge G) \text{ slijedi } \neg F \vee \neg G$$
$$\neg(F \vee G) \text{ slijedi } \neg F \wedge \neg G$$

3. Distributivnost:

$$F \vee (G \wedge H) \text{ slijedi } (F \vee G) \wedge (F \vee H)$$
$$(F \wedge G) \vee H \text{ slijedi } (F \vee H) \wedge (G \vee H)$$
$$F \vee (F_1 \vee \ldots \vee F_n) \text{ slijedi } F \vee F_1 \vee \ldots \vee F_n$$
$$(F_1 \vee \ldots \vee F_n) \vee F \text{ slijedi } F_1 \vee \ldots \vee F_n \vee F$$
$$F \wedge (F_1 \wedge \ldots \wedge F_n) \text{ slijedi } F \wedge F_1 \wedge \ldots \wedge F_n$$
$$(F_1 \wedge \ldots \wedge F_n) \wedge F \text{ slijedi } F_1 \wedge \ldots \wedge F_1 \wedge F$$

4. Uklanjanje operatora:

$$F_1 \vee \ldots \vee F_n \text{ slijedi } \{F_1, \ldots, F_n\}$$
$$F_1 \wedge \ldots \wedge F_n \text{ slijedi } \{F_1\}, \ldots, \{F_n\}$$

Primjer: Neka je zadana jednostavna formula $(F \vee G) \rightarrow H$ koju treba pretvoriti u klauzalni oblik.
Prvi korak je rješavanje implikacije. Nakon primjene prvog pravila formula će izgledati ovako:

$$\neg(F \vee G) \vee H$$

Zatim se primjenjuju negacije:

$$(\neg F \wedge \neg G) \vee H$$

Nakon primjene pravila za negacije primjenjuju se pravila za distributivnost:

$$(\neg F \vee H) \wedge (\neg G \vee H)$$

I kao zaključno, uklanjaju se operatori kako bi se dobio klauzalni oblik na kojem se može primijeniti metoda rezolucije:

$$\{\neg F, H\}, \{\neg G, H\}$$

3.2. Pravilo rezolucije

Kako bi se primijenilo pravilo rezolucije, prvo je potrebno pretvoriti zadanu formulu u konjunktivnu normalnu formu.
Konjunktivna normalna forma je konjunkcija disjunkcija literala. Kako bi se došlo do takvog oblika formule potrebno je primijeniti pravila za pretvorbu u klauzalni oblik i to pravilnim redoslijedom kako su navedena.[7]
Primjer: Neka je zadano $A \leftrightarrow (B \vee C)$

$$\big(A \rightarrow (B \vee C)\big) \wedge \big((B \vee C) \rightarrow A\big) \equiv$$
$$(\neg A \vee (B \vee C) \wedge (\neg(B \vee C) \vee A) \equiv$$
$$\big(\neg A \vee (B \vee C)\big) \wedge \big((\neg B \wedge \neg C) \vee A\big) \equiv$$
$$(\neg A \vee B \vee C) \wedge \big((\neg B \vee A) \wedge (\neg C \vee A)\big) \equiv$$
$$(\neg A \vee B \vee C) \wedge (\neg B \vee A) \wedge (\neg C \vee A)$$

Nakon pretvorbe formule, njena konjunktivna normalna forma je:

$$(\neg A \vee B \vee C) \wedge (\neg B \vee A) \wedge (\neg C \vee A)$$

S druge strane postoji i disjunktivna normalna forma, ona je disjunkcija konjunkcija literala, dakle suprotna konjunkitvnoj normalnoj formi. Primjenom istih pravila kao i u pretvorbi u konjunktivnu normalnu formu dobiva se i disjunktivna normalna forma, a razlika je u tome što je cilj dobiti disjunkciju konjunkata.[7]
Primjer: $A \leftrightarrow (B \vee C)$

$$\big(A \rightarrow (B \vee C)\big) \wedge \big((B \vee C) \rightarrow A\big) \equiv$$
$$(\neg A \vee (B \vee C) \wedge (\neg(B \vee C) \vee A) \equiv$$
$$\big(\neg A \vee (B \vee C)\big) \wedge \big((\neg B \wedge \neg C) \vee A\big) \equiv$$
$$(\neg A \vee B \vee C) \wedge \big((\neg B \wedge \neg C) \vee A\big) \equiv$$
$$\big((\neg A \vee B \vee C) \wedge A\big) \vee \big((\neg B \wedge \neg C) \wedge (\neg A \vee B \vee C)\big) \equiv$$
$$(A \wedge B) \vee (A \wedge C) \vee (\neg A \wedge \neg B \wedge \neg C)$$

Disjunktivna normalna forma danog primjera je: $(A \wedge B) \vee (A \wedge C) \vee (\neg A \wedge \neg B \wedge \neg C)$

Pravilo rezolucije je pravilo zaključivanja kojom se dolazi do rezolvente (izvod koji se dobiva razrješavanjem dvije klauzule). Rezolucijsko pravilo nalaže da ako dvije klauzule sadrže literale koje su komplementarni, dakle jedna sadrži istinit literal, a druga isti taj literal samo lažan, dolazi se do rezolvente tako da se ta dva komplementarna literala razriješe i nisu sadržana u novoj klauzuli. Zbog toga se ovo pravilo naziva i pravilo razrješavanja. [5]
Princip kojim se dolazi do izvoda pomoću rezolucije za propozicijsku logiku jednostavan je i na neki način intuitivan. Neka su zadane dvije klauzule {F, G} i {¬F, H}, prva klauzula sadrži literal F, a druga ¬F. Ako je F u prvoj klauzuli lažan, G mora biti istinit, s druge strane ako je F istinit u drugoj klauzuli onda je H istinit. Dakle, G ili H su istiniti pa dolazimo do izvoda koji glasi {G, H}. [5]

$$\frac{\{F, G\} \quad \{\neg F, H\}}{\{G, H\}}$$

Isto tako ako se u dvije klauzule pojavljuje isti literal, on se u izvodu neće ponoviti. Izvod će izgledati ovako[5]:

$$\frac{\{\neg F, G\} \quad \{F, G\}}{\{G\}}$$

Također, ako se u jednoj klauzuli pojavi više literala, a u drugoj samo jedan literal i to negacija nekog literala koji se pojavljuje u prvoj klauzuli broj literala u novonastaloj klauzuli bit će manji nego u prvoj klauzuli[5]:

$$\frac{\{F, G, H\} \quad \{\neg F\}}{\{G, H\}}$$

Poseban slučaj se pojavljuje kada dvije klauzule koje se razrješavaju imaju po jedan literal, a oni su suprotni jedan drugome. Nastaje prazna klauzula [2]:

$$\frac{\{F\} \quad \{\neg F\}}{\{\} \ (\text{NIL})}$$

3.3. Valjanost rezolutivnog izvoda

Formule se smatraju valjanima ako i samo ako su one istinite u svakoj interpretaciji. Valjanosti formula u propozicijskoj logici mogu se odrediti konačnim brojem koraka. Iako se lako može dokazati valjanost formule, javlja se jedan problem, a to je da se u postupku dokazivanja povećava broj koraka u skladu s brojem elementarnih propozicija.[1]
Rezolucijom za propozicijsku logiku ne može se dobiti cjelovito rješenje, odnosno ne može se doći do svih izvoda. Kako bi se ovaj problem zaobišao, potrebno je uzeti klauzalne oblike formula to jest premise, ako one postoje, i negiranu ciljnu klauzulu te pomoću njih izvesti praznu klauzulu. Na sljedećem primjeru bez premisa prikazan je taj postupak[6]:

$$\text{Ciljna formula: } (F \to (G \to F))$$
$$\text{Negiranje ciljne klauzule: } \neg(F \to (G \to F))$$
$$\text{Rješavanje implikacija: } \neg(\neg F \vee \neg G \vee F)$$
$$\text{Negacije: } \neg\neg F \wedge \neg\neg G \wedge \neg F$$
$$F \wedge G \wedge \neg F$$
$$\text{Distributivnost: } F \wedge G \wedge \neg F$$
$$\text{Uklanjanje operatora: } \{F\}, \{G\}, \{\neg F\}$$

Iz klauzalnog oblika dobivamo praznu klauzulu ({*F*} i {¬*F*} tvore praznu klauzulu) i time je dokazana valjanost formule.[6]

U slučaju da su premise i ciljna formula zadane taj postupak će izgledati ovako[6]:

$$\text{Zadane premise: } (F \to G) \text{ i } (G \to H)$$
$$\text{Ciljna formula: } (F \to H)$$
$$1.\{\neg F, G\}$$
$$2.\{\neg G, H\}$$
$$3.\{F\} - \text{negacija ciljne formula}$$
$$4.\{\neg H\} - \text{negacija ciljne formula}$$
$$5.\{G\} - 3{,}1\ (\{F\}\{\neg F\} \text{ tvore praznu klauzulu})$$
$$6.\{H\} - 5{,}2\ (\{G\}\{\neg G\} \text{ tvore praznu klauzulu})$$
$$7.\{\} - 6{,}4\ (\{H\}\{\neg H\} \text{ tvore praznu klauzulu})$$

Na kraju je dobivena prazna klauzula što dokazuje da je ciljna formula logička posljedica zadanih premisa. [6]

Primjer: Neka su zadane premise: $(B \to A), \big((A \vee C) \to D\big), (D \to E)$ i $(\neg B \to C)$, treba dokazati da vrijedi E, odnosno ono nam predstavlja ciljnu formulu. Primjenom pravila za pretvorbu u klauzalni oblik i rezolucijskog pravila izvod izgleda ovako:

$1.\{A, \neg B\}$	
$2.\{\neg A, D\}$	
$3.\{\neg C, D\}$	
$4.\{\neg D, E\}$	
$5.\{B, C\}$	
$6.\{\neg E\}$	
$7.\{\neg B, D\}$	1,2
$8.\{D, C\}$	7,5
$9.\{D\}$	8,3
$10.\{E\}$	9,4
$11.\{\}$	

Primjena rezolucijskog pravila na neku formulu *F* rezultira izvodom prazne klauzule, u tom slučaju metateroem adekvatnosti takvu formulu opisuje kao nezadovoljavajućom. Ako je neka formula *F* nezadovoljavajuća, metateorem potpunosti rezolucijskog pravila govori kako će rezolventa formule *F* biti prazna klauzula ukoliko je na toj formuli *F* primijenjena metoda rezolucije. U nastavku se navode oba metateorema. [6]

Metateorem adekvatnosti rezolucijskog pravila: „Neka je *D* skup disjunkta i neka niz disjunkta $D_1,\ldots,D_m$: $\perp$ predstavlja rezolutivni izvod identički lažnog disjunkta $\perp$ iz *D*. Tada je D kontradiktoran skup disjunkta.“. [7]

Metateorem potpunosti rezolucijskog pravila: „Neka je *D* kontradiktoran skup disjunkta tada postoji rezolutivni izvod identički lažnog disjunkta $\perp$ iz *D.*“ [7]

4. Usporedba različitih tipova rezolucije

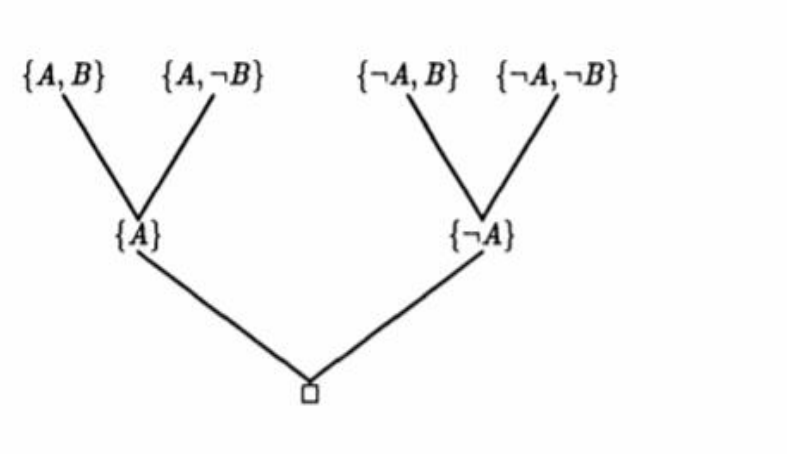
Različiti tipovi rezolucije koriste se kako bi se povećala učinkovitost u određenim slučajevima. Tipovi rezolucije odnose se na heuristička pravila kojima je određen način traženja rješenja. Pomoću nekih moguće je brže doći do rezolutivnog izvoda, no važno je da se primjenom ovih rezolucija ne izgubi svojstvo potpunosti. Ako se izgubi svojsto potpunosti, rezolucija nam nije korisna.[3] Tipovi rezolucije koji će u nastavku biti objašnjeni i uspoređeni su: linearna rezolucija, skup potpore za rezoluciju (engl. *Set of Support Resolution*), SLD-rezolucija, jedinična rezolucija. Kako bi se vidjela razlika u provođenju navedenih rezolucija, one su primijenjene na primjeru iz poglavlja o valjanosti rezolutivnog izvoda.

4.1.1. Linearna rezolucija

Linearna rezolucija je, kao što i samo ime govori, linearno zaključivanje prazne klauzule. Linearnog je oblika, na vrhu je polazna klauzula iz koje se dalje granaju ostale klauzule pomoću rezolucije. Svaka rezolventa rezultat je prethodne rezolvente i neke druge klauzule.[8]
Primjer: Neka je zadan skup klauzula $D = \{\{A, B\}, \{A, \neg B\}, \{\neg A, B\}, \{\neg A, \neg B\}\}$. Običnom rezolucijom do izvoda se dolazi u tri koraka[9]:

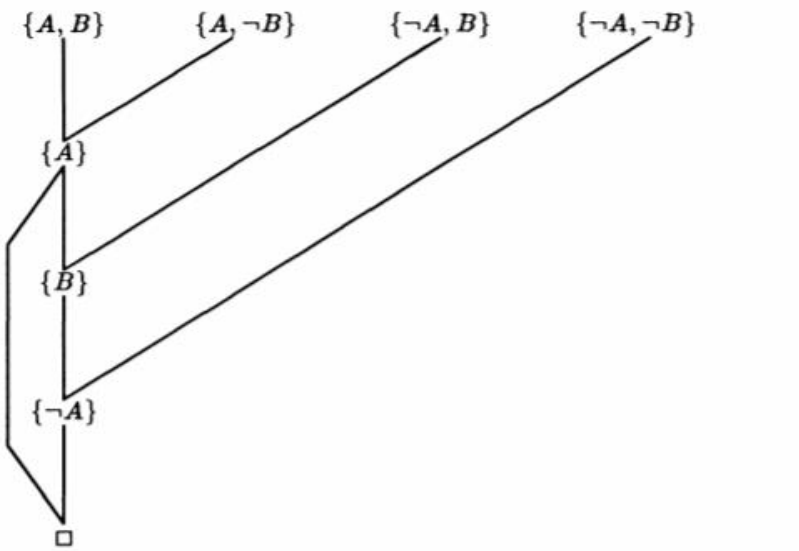
$1.\{A, B\}$	
$2.\{A, \neg B\}$	
$3.\{\neg A, B\}$	
$4.\{\neg A, \neg B\}$	
$5.\{A\}$	$1,2$
$6.\{\neg A\}$	$3,4$
$7.\{\ \}$	$5,6$

Grafički prikaz ovog izvoda prikazan je na sljedećoj slici:



Slika 1 Izvod pomoću rezolucije [9]

Linearnom rezolucijom skupa klauzula *D* do izvoda se dolazi u četiri koraka kako je prikazano na slici 2.



Slika 2 Izvod linearnom rezolucijom [9]

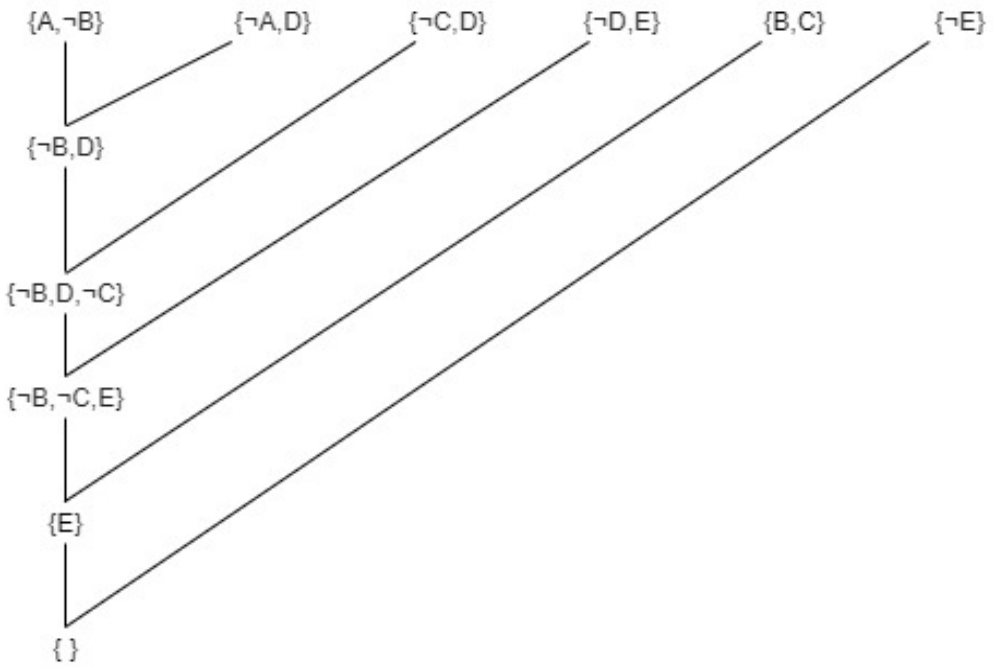
Dakle, iako se linearnom rezolucijom brže se dolazi do rezolvente (zbog manjeg područja traženja) ona se sastoji od više koraka to jest dužeg dokaza.[3] Ovaj tip rezolucije je potpun, što znači da za svaki neistinit skup klauzula *D* postoji neka klauzula C sadržana u skupu *D* na temelju koje se dolazi do rezolvente.[9]

Primjer: U primjeru iz poglavlja o valjanosti rezolutivnog izvoda dobivene su sljedeće premise: $\{A, \neg B\}, \{\neg A, D\}, \{\neg C, D\}, \{\neg D, E\}, \{B, C\}, \{\neg E\}$. Linearna rezolucija na ovom primjeru izgleda ovako:

1. $\{A, \neg B\}$	
2. $\{\neg A, D\}$	
3. $\{\neg C, D\}$	
4. $\{\neg D, E\}$	
5. $\{B, C\}$	
6. $\{\neg E\}$	
7. $\{\neg B, D\}$	1,2
8. $\{\neg B, D, \neg C\}$	7,3
9. $\{\neg B, \neg C, E\}$	8,4
10. $\{E\}$	9,5
11. $\{\ \}$	10,6

Kao što je ranije objašnjeno kako bi se došlo do prazne klauzule, kod linearne rezolucije, počinje se sa prve dvije klauzule u ovom slučaju to su $\{A, \neg B\}$ i $\{\neg A, D\}$. Izvod te dvije klauzule je $\{\neg B, D\}$ te ona automatski postaje „roditeljska“ klauzula i sa sljedećom zadanom klauzulom $\{\neg C, D\}$ izvodi novu „roditeljsku“ klauzulu $\{\neg B, D, \neg C\}$. Ta nova klauzula s 4. klauzulom $\{\neg D, E\}$ tvori novu roditeljsku klauzulu $\{\neg B, \neg C, E\}$ koja s predzadnjom zadanom klauzulom $\{B, C\}$ tvori klauzulu koja jse sastoji od jednog literala $\{E\}$. S posljednjom klauzulom se razrješava novonastala klauzula i tako se dobiva prazna klauzula { }. Ovakva metoda rezolucije prilično je intuitivna i proces nije složen, dapače vrlo je jednostavan jer se klauzule rješavaju po redu kako su i zapisane.

Grafički prikaz linearne rezolucije na ovom primjeru izgleda ovako:



Slika 3 Izvod linearnom rezolucijom (vlastiti primjer)

U ovom grafičkom prikazu vidljivo je da se do izvoda došlo u 5 koraka i jasno se vidi kako se razrješavanjem zadanih klauzula i novonastalih klauzula dolazi do novih izvoda koji se dalje koriste u izvođenju.

4.1.2. Skup potpore za rezoluciju

Skup potpore za rezoluciju je metoda u kojoj se određuje takozvana podrška odnosno potpora pri rješavanju klauzula. Kada je skup klauzula istinit on se rješava tako da se dvije klauzule međusobno razriješe, tako da se negirane rezolvente mogu zanemariti bez gubitka potpunosti rezolucije. Ako imamo neki skup Δ i podskup Γ, podskup Γ bit će skup podrške skupu Δ ako i samo ako je Δ- Γ zadovoljavajuće odnosno istinito [8].

Primjer: Ako su zadane klauzule: $\{F, G\}, \{\neg F, H\}, \{\neg G, H\}, \{\neg H\}$, metoda skup potpore za rezoluciju izgleda ovako:

1.	$\{F, G\}$	
2.	$\{\neg F, H\}$	
3.	$\{\neg G, H\}$	
4.	$\{\neg H\}$	potpora
5.	$\{\neg F\}$	2,4
6.	$\{\neg G\}$	3,4
7.	$\{G\}$	1,5
8.	$\{\ \}$	6,7

U gore navedenom primjeru kao skup podrške određena je klauzula *H*. Ta klauzula se rješava s klauzulama 2 i 3 te kao rezultat nastaju klauzule 5 i 6 koje se mogu razriješiti sa prvom klauzulom što rezultira u dobivanju prazne klauzule. Ovaj način rješavanja rezolucijom nije koristan ako se ne može odrediti dobar skup podrške. [8]

Primjer: Ako se ova metoda iskoristi na primjeru koji se koristio u prethodnoj metodi izgledat će ovako:

$1.\{A, \neg B\}$	
$2.\{\neg A, D\}$	
$3.\{\neg C, D\}$	
$4.\{\neg D, E\}$	
$5.\{B, C\}$	
$6.\{\neg E\}$	potpora
$7.\{\neg D\}$	6,4
$8.\{\neg A\}$	7,2
$9.\{\neg C\}$	7,3
$10.\{B\}$	9,5
$11.\{A\}$	10,1
$12.\{\ \}$	11,8

U ovom primjeru kao potpora određena je 6. klauzula. Izvodom klauzule 6. $\{\neg E\}$ i klauzule 4. $\{\neg D, E\}$ nastaje nova klauzula $\{\neg D\}$ koja je član skupa potpore zbog toga što joj je roditeljska klauzula potpora. Nadalje, svaka klauzula kojoj je 7. klauzula roditeljska također je član u skupu potpore. Tako da su sve novonastale klauzule u ovom slučaju u skupu potpore jer su im roditeljske klauzule također u tom skupu. Ovaj proces je složeniji za razliku od linearne rezolucije zbog toga što se treba dobro odrediti potporna klauzula kako bi se realizirala rezolucija. Ako se ne pronađe valjana potpora, neće se moći ostvariti prazna klauzula na kraju izvoda. Važno je da je potporna klauzula podskup neke zadane klauzule, te da svaka njena klauzula „dijete“ također bude podskup neke klauzule kako bi se ovom metodom došlo do traženog izvoda. U ovom primjeru može se uočiti da su svi izvodi literali zbog toga što je svaki član skupa potpore jedan literal i razrješavanjem sa zadanim klauzulama dolazi do izvoda koji također sadržavaju po jedan literal.

4.1.3. Jedinična rezolucija

Jediničnom rezolucijom dolazi se do jedinične rezolvente kojoj su roditeljske klauzule jedinične klauzule odnosno one klauzule koje se sastoje od samo jednog literala. U jediničnom izvodu svi izvodi su jedinične rezolvente, a jedinična rezolucija je jedinični izvod prazne klauzule.

Primjer [8]:

$1.\{F, G\}$
$2.\{\neg F, H\}$
$3.\{\neg G, H\}$
$4.\neg H$

$5.\{\neg F\}$ 2,4
$6.\{\neg G\}$ 3,4
$7.\{G\}$ 1,5
$8.\{F\}$ 1,6
$9.\{H\}$ 3,7
$10.\{\ \}$ 6,7

U primjeru se može vidjeti kako se zadane klauzule s dva literala rješavaju s jediničnim klauzulama kako bi nastale nove jedinične klauzule.

U ovoj metodi prve dvije klauzule mogu se razriješiti tako da se odmah dođe do rezolvente {*G,H*}, ali tako se ne stvaraju nove jedinične klauzule. Ipak kada se klauzule rješavaju jediničnom rezolucijom rezolutivni izvod imat će manje literala nego što to ima klauzula „roditelj“ što pospješuje pronalazak prazne klauzule, a i učinkovitost. Jedinična rezolucija je potpuna samo za Hornove klauzule. Hornove klauzule su klauzule u kojima je sadržan samo jedan istinit literal. Ako se ne koriste Hornove klauzule postoji mogućnost da se jediničnom rezolucijom ne može doći do ciljne, prazne klauzule.[8]

Slijedi prikaz ove metode na primjeru na kojemu su prikazane i metode skupa potpore rezolucije i linearne rezolucije.

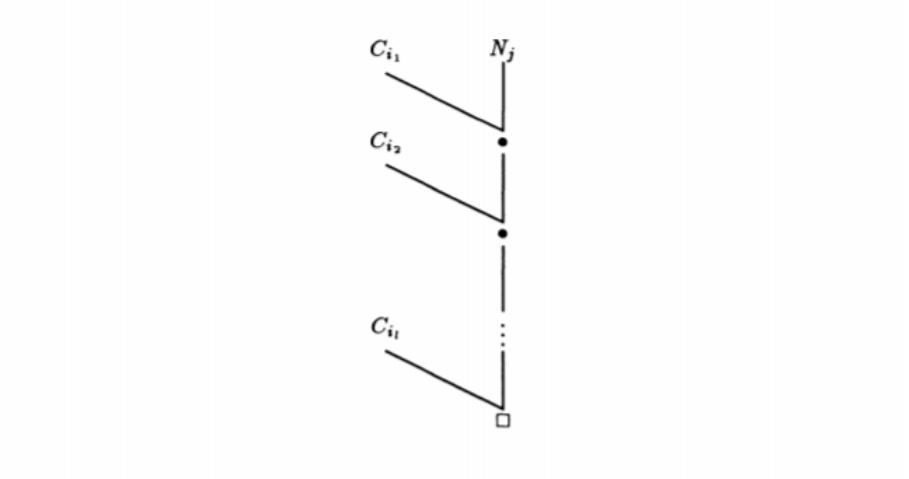
1. $\{A, \neg B\}$	
2. $\{\neg A, D\}$	
3. $\{\neg C, D\}$	
4. $\{\neg D, E\}$	
5. $\{B, C\}$	
6. $\{\neg E\}$	
7. $\{\neg D\}$	6,4
8. $\{\neg A\}$	7,2
9. $\{\neg C\}$	7,3
10. $\{B\}$	9,5
11. $\{A\}$	10,1
12. $\{D\}$	11,2
13. $\{E\}$	12,4
14. $\{\neg B\}$	8,1
15 $\{C\}$	14,5
16. $\{\}$	15,9

Ova metoda nije idealna prvenstveno zbog toga što je ona potpuna za Hornove klauzule, one u kojima je naviše jedan literal istinit. Ako se ne koriste Hornove klauzule postoji mogućnost da se ne dođe do traženog rješenja, stoga su druge metode adekvatnije za takve slučajeve. Isto tako ova metoda nije idealna ako se izvod dobiva iz više klauzula kao u primjeru gore.
Broj koraka u rezoluciji se povećava, jer svaka izvedena klauzula mora biti jedinična to jest sadržavati samo jedan literal kako bi se naposlijetku došlo do prazne klauzule.
Ovaj proces nije složen sam po sebi jer se proces obavlja s jednim literalom, ali proces može biti nepotrebno dug.

4.1.4. SLD rezolucija

SLD-rezolucija je linearna rezolucija s funkcijom odabira za zadane klauzule. Ovaj tip rezolucije definiran je samo za Hornove klauzule. Kao što je ranije spomenuto Hornove klauzule su one koje sadrže samo jedan istinit literal. [3]
SLD- rezolucija ima poseban oblik, osnovna klauzula mora biti negativna (ciljna klauzula) te u svakom sljedećem koraku rezolucije sporedna klauzula mora biti pozitivna ulazna klauzula. Na primjer, neka je *D* skup Hornovih klauzula $\{C_1,C_2,\ldots,C_N,N_1,N_2,\ldots,N_M\}$ od kojih su $C_1,\ldots,C_n$

zadane klauzule dok su klauzule $N_1,\ldots,$Nm ciljne klauzule. [9] Dokaz ima sljedeći oblik: osnovna klauzula je N_j na koju se dodaju postavljene klauzule $C_1,\ldots,C_n$.[3] Grafički to izgleda ovako:



Slika 4 Izvod SLD-rezolucijom [9]

Na grafičkom prikazu točke prikazuju ciljne klauzule, odnosno one koje ne sadrže pozitivne literale.
SLD-rezolucija je adekvatna i potpuna za Hornove klauzule.[3]

Na primjeru iz prijašnjih poglavlja ova metoda izgleda ovako:

$1.\{A, \neg B\}$	
$2.\{\neg A, D\}$	
$3.\{\neg C, D\}$	
$4.\{\neg D, E\}$	
$5.\{B, C\}$	
$6.\{\neg E\}$	
$7.\{\neg D\}$	6,4
$8.\{\neg A\}$	7,2
$9.\{\neg B\}$	8,1
$10.\{C\}$	9,5
$11.\{D\}$	10,3
$12.\{\ \}$	11,7

Kod ove metode potrebno je u svakom novom izvodu koristiti novi izvod kao roditeljsku klauzulu, kao i u običnoj linearnoj rezoluciji. U ovoj metodi javlja se funkcija selekcije to jest odabir kojim se bira druga roditeljska klauzula. Počinje se sa negiranim ciljem, u ovom slučaju to je $\{\neg E\}$ druga klauzula kojom se dolazi do izvoda, a bira se po tome postoji li u njoj takav literal. Razrješavanjem nastaje klauzula koja se koristi dalje sve dok se ne dođe do kraja izvoda.

Iako ova metoda funkcionira po istom principu kao i obična linearna rezolucija, složenija je zbog spomenute funkcije odabira, jer proces traje dulje nego u običnoj metodi linearne rezolucije.

U prikazanom primjeru ova metoda dovodi do valjanog izvoda, no kao što je spomenuto ona je potpuna za Hornove klauzule pa nije adekvatna za izvođenje svih klauzula. Metoda jedinične rezolucije isto je potpuna za Hornove klauzule, no SLD-rezolucija je za razliku od nje puno brža zbog toga što ima manje izvoda.

4.1.5. Usporedba rezolucija

Kada se usporede ove četiri metode rezolucije (uspoređene su prema primjeru na kojem su primijenjene sve četiri rezolucije) najbolja metoda je linearna jer je najbrža, provedena je u 5 koraka, dok je za jediničnu rezoluciju bilo potrebno najviše koraka zbog toga što se u svakom izvodu mora doći do klauzule s jednim literalom. Time se povećava broj potrebnih koraka do prazne klauzule. Linearna rezolucija također je i najjednostavnija, uz nju je ovdje i jedinična rezolucija koja, iako ima više koraka, ima jednostavan proces izvođenja prazne klauzule. Skup potpore za rezoluciju je s druge strane složenija metoda zbog toga što nije lako iz prve odrediti potpornu klauzulu iz koje će nastati skup potpore i samim time proces traje duže. SLD rezolucija je složeniji tip linearne rezolucije zbog funkcije odabira koja usporava proces zaključivanja. Što se tiče potpunosti, linearna rezolucija i skup potpore za rezoluciju su potpune metode za sve klauzule, dok su jedinična rezolucija i SLD rezolucija potpune samo za Hornove klauzule. U tablici 9. prikazana su svojstva rezolucija.

Tablica 9 Usporedba rezolucijskih metoda

Vrsta rezolucije	Broj koraka	Jednostavnost	Priprema	Potpunost
Linearna rezolucija	5	Jednostavna	-	Potpuna
Skup potpore za rezoluciju	6	Složena	Potrebno je pronaći potpornu klauzulu kako bi se provela rezolucija	Potpuna
Jedinična rezolucija	10	Jednostavna	-	Potpuna samo za Hornove klauzule
SLD - rezolucija	6	Složena	-	Potpuna samo za Hornove klauzule

5. Primjeri primjene i korisnosti rezolucije

Rezolucija se koristi za zaključivanje u umjetnoj inteligenciji. Sam naziv ovog rada govori da je rezolucija prisutna u propozicijskoj logici. Ta metoda se također koristi i u predikatnoj logici. Rezolucija se može koristiti u dokazivanju teorema, pa je tako često implementirana u programima koji „rješavaju“ teoreme. [3]

U umjetnoj inteligenciji (engl. Artificial Intelligence) metoda rezolucije koristi se u procesu zaključivanja. Umjetna inteligencija sadržava takozvanu bazu znanja (engl. kowledge base). Takvo znanje predstavlja znanje o svijetu koje umjetna inteligencija koristi samo što ono poprima oblik propozicija iz kojih metodom rezolucije nastaju nove propozicije kojima se umjetna inteligencija kasnije može služiti. [11]

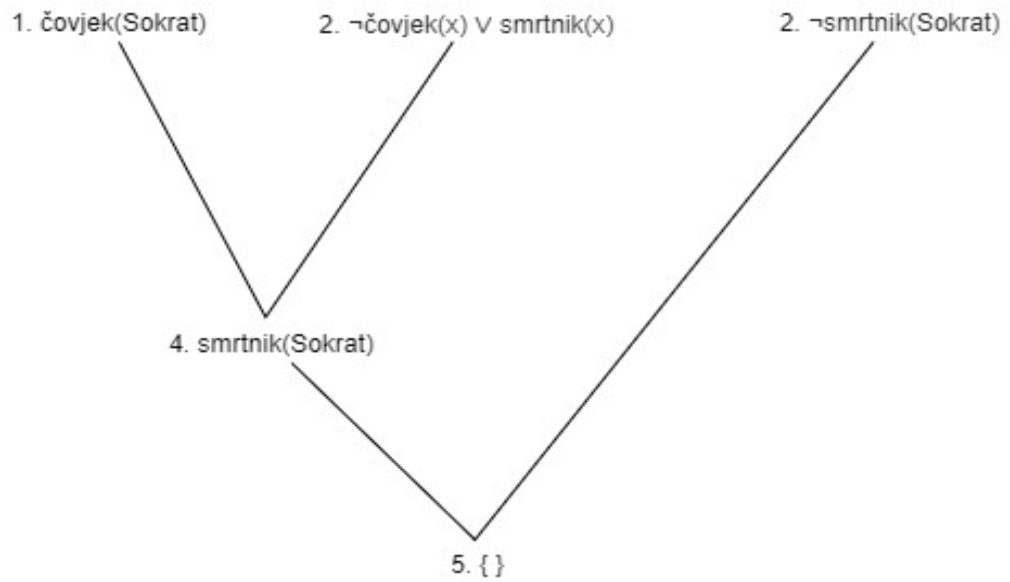
Rezolucija u predikatnoj logici odvija se u koracima koji su slični koracima rezoluciji za propozicijsku logiku. Prvo je potrebno činjenice pretvoriti u premise predikatne logike, zatim premise pretvoriti u konjunktivnu normalnu formu (CNF), treći korak je negiranje cilja i kao posljednje potrebno je primijeniti unifikaciju kako bi se proces rezolucije završio. [12]
U najpoznatijem primjeru zaključivanja je zadana sljedeća činjenica: Svi ljudi su smrtnici i Sokrat je bio čovjek. Potrebno je dokazati da je Sokrat smrtnik. [13]
Nakon pretvorbe u konjunktivnu normalnu formu premise će imati sljedeći oblik:

1. čovjek(Sokrat)
2. ¬čovjek(x) ∨ smrtnik(x)
3. ¬smrtnik(Sokrat)

Moguća su dva rješenja:

1. čovjek(Sokrat)
2. ¬čovjek(x) ∨ smrtnik(x)
3. ¬smrtnik(Sokrat)
4. smrtnik(Sokrat)
5. { }

Do 4. izvoda se dolazi spajanjem 1. i 2. premise te do 5. izvoda razrješavanjem 3. premise i 4. izvoda. Grafički prikaz ovog rješenja izgleda ovako:



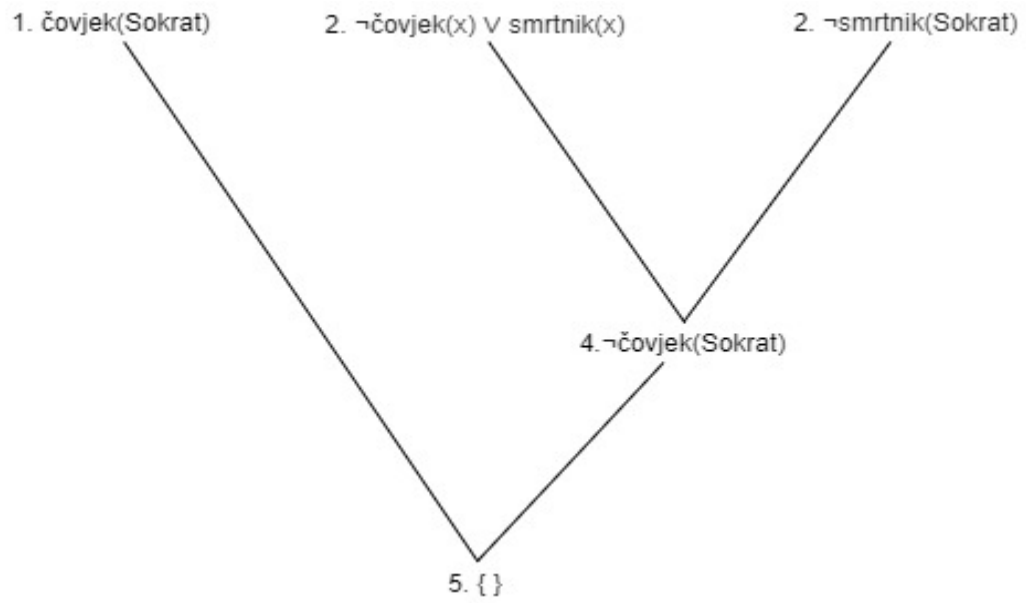
Slika 5 Rezolucija u predikatnoj logici (prvo rješenje)

Drugo moguće rješenje:

1. čovjek(Sokrat)
2. ¬čovjek(x) ∨ smrtnik(x)
3. ¬smrtnik(Sokrat)
4. ¬čovjek(Sokrat)
5. { }

U ovom slučaju 4. izvod nastaje iz 2. i 3. premise, a 5. izvod razrješavanjem 1. premisom i 4. izvodom. Pravila razrješavanja su slična i intuitivna kao pravila za rezoluciju kod propozicijske logike.

Grafički prikazi za ovaj primjer izgledat će ovako:



Slika 6 Rezolucija u predikatnoj logici (drugo rješenje)

Temelj logički programskog jezika Prolog je predikatna logika i vrsta klauzula koje koristi pri zaključivanju su Hornove klauzule. [15] U prethodnom poglavlju objašnjena je SLD-rezolucija koja se koristi u programskom jeziku Prolog (= *Programming in Logic*). Prolog koristi samo Hornove klauzule pa je tako SLD-rezolucija pogodna zbog svog slijednog računanja.[9]
Primjer: Kada bi se u Prologu dokazao prijašnji primjer da je Sokrat smrtnik to bi izgledalo ovako:

```
čovjek('Sokrat').
smrtnik(X) :- čovjek(X).
smrtnik('Sokrat').
```

Metoda rezolucije se također koristi u određenim dokazivačima teorema (engl. theorem provers). Jedan od takvih dokazivača je Prover9 koji je nasljednik dokazivača Otter (koji je isto koristio rezoluciju, no on se danas više ne koristi). Dokazivač Prover9 zajedno s Mace4 pretraživačem čini kombinaciju programa koji pronalaze dokaze. Prover9 koristi rezoluciju kako bi došao do ciljnog dokaza.[14]

Još jedan primjer dokazivača koji se temelji na rezoluciji je dokazivač pod nazivom „Geo“. Sustav koristi geometrijske formule i formule logike prvog reda kao ulaz pri čemu formule logike prvog reda mijenja u geometrijske formule.[14]

6. Zaključak

Kao što je na početku spomenuto, rezolucija je intuitivna metoda zaključivanja koja se primjenjuje u raznim područjima. Proces rezolucije isključivo radi s klauzulama, stoga je važno poznavati pretvorbu u klauzalni oblik. Proces rezolucije završen je kada se dođe do ciljne klauzule odnosno prazne klauzule koja označava da je izvod valjan.

Metateoremi adekvatnosti i potpunosti za rezolucijsko pravilo važni su jer ukazuju na ispravnost i potpunost formula nad kojima se vrši rezolucija.

Postoje brojne vrste rezolucije no uz njih se mogu javljati i neki problemi koji ih čine beskorisnima. Ranije je navedeno koliko je potpunost važna za proces rezolucije pa je pri povećanju efikasnosti i bržem pronalasku rezolventnih klauzula važno paziti i na svojstvo potpunosti kako bi rezolucija bila korisna te kako bi se došlo do traženog cilja, a to je prazna klauzula. Najefikasnija od uspoređenih metoda rezolucije pokazala se linearna rezolucija jer je za dani primjer imala najmanji broj koraka što znači da je i najbrža, a sam proces je jednostavan i intuitivan jer se klauzule razrješavaju redom kojim su i zapisane.

Metoda rezolucije primjenjiva je u raznim područjima, a najviše u zaključivanju umjetnoj inteligenciji. Koristi se i u logičkom programskom jeziku PROLOG koji koristi SLD rezoluciju kao metodu zaključivanja. Također, neki od dokazivača teorema (engl. theorem provers) koriste rezoluciju kao metodu zaključivanja. Iako rezolucija ima veću ulogu u logici prvog reda, ona je primjenjiva i u propozicijskoj logici kao metoda zaključivanja to jest dokazivanja.

Popis literature

[1] Bojana Dalbelo Bašić, Jan Šnajder, (2008.), *UMJETNA INTELIGENCIJA Zaključivanje uporabom propozicijske i predikatne logike -zbirka zadataka-*, preuzeto: 06.07.2021 s https://www.fer.unizg.hr/_download/repository/DalbeloBasic-Snajder_UI_Logika.pdf

[2] S. Ribarić; B. Dalbelo Bašić, (23.04.2001.), *Inteligentni sustavi*, preuzeto: 06.07.2021. s http://www.zemris.fer.hr/predmeti/is/nastava/web-propozicijska-logika.pdf

[3] Ralitsa Dardjonova, (2020), *Resolution in FOL,* preuzeto: 06.07.2021. s https://www21.in.tum.de/teaching/sar/SS20/3.pdf

[4] Jan Krajiček, (1995.), *Bounded Arithmetic, Propositional Logic and Complexity Theory*, pristupano: 07.07.2021. dostupno na:

https://books.google.hr/books?hl=en&lr=&id=6XkgKydE0Z8C&oi=fnd&pg=PP1&dq=propositional+logic+&ots=RAk4mvzWhx&sig=GvGp85GJbg-1WUA_c7v1LLfh7SM&redir_esc=y#v=onepage&q&f=false

[5] *Propositional resolution* (bez datuma) u Stanford introduction to logic, pristupano: 07.07.2021. dostupno na:
http://logic.stanford.edu/intrologic/notes/chapter_05.html

[6] *Propositional resolution, Chapter 5* (bez datuma), pristupano: 08.07.2021. dostupno na: https://u.math.biu.ac.il/~margolis/Math_Bedida2_5768/Resolution%20in%20Propositional%20Logic.pdf

[7] Čubrilo, M. Matematička logika za ekspertne sisteme. Informator, Zagreb, 1989.

[8] *Resolution* (bez datuma) u Stanford introduction to logic, pristupano: 08.07.2021. dostupno na:
http://intrologic.stanford.edu/textbook/chapter_10.html

[9] Uwe Schöning, (1989), *Logic for Computer Scientists*, pristupano: 09.07.2021. dostupno na:
http://tinman.cs.gsu.edu/~raj/8710/f16/UweSchoning/UweSchoningBook.pdf

[10] *Propositional Logic* (bez datuma) u Internet Encyclopedia of Philospohy, pristupano: 02.09.2021.
https://iep.utm.edu/prop-log/#H5

[11] *CS50's Introduction to Artificial Intelligence with Python* (bez datuma, pristupano: 13.09.2021.
https://cs50.harvard.edu/ai/2020/notes/1/

[12] *Resolution in FOL* (bez datuma) u javaTpoint, pristupano: 02.09.2021.
https://www.javatpoint.com/ai-resolution-in-first-order-logic

[13] *Resolution Theorem Proving* (bez datuma) u Computational Creativity, pristupano: 02.09.2021. dostupno na:
http://ccg.doc.gold.ac.uk/ccg_old/teaching/artificial_intelligence/lecture9.html

[14] M. Saqib Nawaz, Moin Malik, Yi Li, Meng Sun and M. Ikram Ullah Lali, (06.12.2019), *A Survey on Theorem Provers in Formal Methods*, pristupano: 13.09.2021., dostupno na:
https://arxiv.org/pdf/1912.03028.pdf

[15] Siniša Šegvić, *Uvod u programski jezik Prolog, Inteligentni sustavi*, pristupano: 14.09.2021., dostupno na:
http://www.zemris.fer.hr/~ssegvic/pubs/prolog.pdf

Popis slika

Popis tablica