Umjetna inteligencija

6. Automatsko zaključivanje

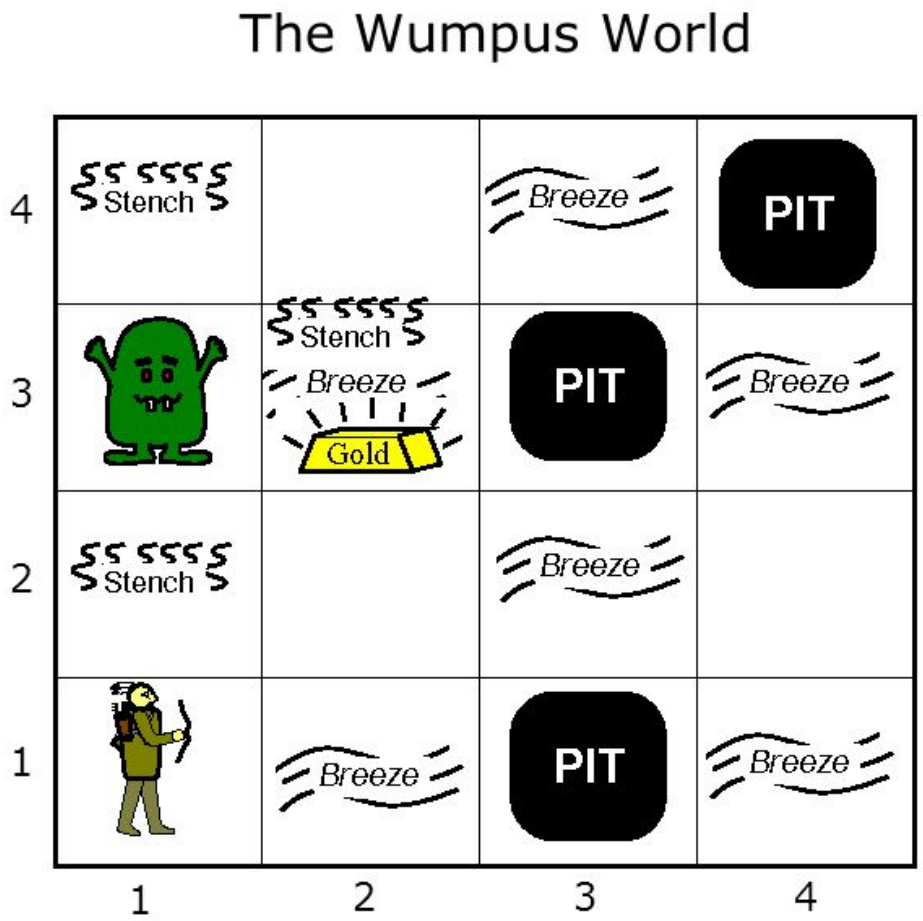
prof. dr. sc. Bojana Dalbelo Bašić
izv. prof. dr. sc. Jan Šnajder

Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

Ak. god. 2019./2020.

2.3

Motivacija: Svijet Wumpusa



Percepcija (činjenice):

$\neg B_{1,1}$

$\neg B_{1,2}$

$B_{2,1}$

$\neg P_{1,1}$

Znanje (pravila):

$B_{2,1} \leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

Novo znanje (zaključci):

$P_{2,2} \vee P_{3,1}$

Motivacija: Carinici i diplomati

Premise

Carinici su pretražili svakoga tko je ušao u zemlju a nije diplomat. Neke krijumčare koji su ušli u zemlju pretražili su samo krijumčari. Niti jedan krijumčar nije diplomat.

Zaključak

Neki su carinici krijumčari.

Zaključak = logička posljedica?

- Zaključak (novoizvedeno znanje) zapravo nije ništa drugo nego **logička posljedica** premisa (postojećeg znanja)
- Dakle, izvođenje novog znanja svodi se na dokazivanje logičkih posljedica postojećeg znanja

$$\text{staro znanje} \models \text{novo znanje}$$

- No, u praksi tu postoji problem...

Problem s dokazivanjem semantičke posljedice

(1) **Netraktabilnost:**

- Moramo provjeriti 2^n interpretacija
- Zamislimo da želimo dokazati $F_1, \ldots, F_{100} \models F_1$. Uz pretpostavku $n = 100$, trebali bismo provjeriti 1.27×10^{30} interpetacija. To je nemoguće!
- Osim toga, u ovom slučaju to je nepotrebno jer je *očigledno* da relacija logičke posljedice vrijedi

(2) **Neodlučivost:**

- U predikatnoj logici broj interpretacija je beskonačan, pa nemamo šanse sve ih ispitati
- Zapravo, FOL je **poluodlučiva** (engl. *semi-decidable*): možemo dokazati valjanost onih formula koje jesu valjane, ali za formule koje nisu valjane ne možemo to uvijek dokazati

- Postoji li rješenje za ove probleme? Da, donekle
- Umjesto da se bavimo interpretacijama i modelima (**semantikom**), trebamo se baviti pravilima zaključivanja (**teorijom dokaza**)

Sadržaj

Sadržaj

Teorija dokaza (engl. *proof theory*)

- Ljudi ne zaključuju na način da dokazuju semantičku posljedicu (ne pokušavaju pronaći interpretaciju koja je istinita za F, a nije za G)!
- Umjesto toga, pokušavamo pokazati kako se G može **izvesti** iz premisa pomoću konačnog broja **pravila zaključivanja** (engl. *inference rules*)
- Svako pravilo zaključivanja treba biti *opravdano* i što jednostavnije
- Pravila zaključivanja omogućuju dobivanje novih formula na temelju zadanih premisa, bez eksplicitnog referenciranja na semantiku logike (istinosne vrijednosti propozicija)

Teorija dokaza (engl. *proof theory*)

Dakle, teorija dokaza nudi dvije prednosti naspram dokazivanja logičke posljedice:

1. **Učinkovitost:** umjesto da iscrpno pretražujemo sve moguće interpretacije, deduktivnu posljedicu možemo brže dokazati (pogotovo ako koristimo pametnu strategiju dokazivanja). Primijetite, međutim, da ne možemo izbjeći neodlučivost FOL-a
2. **Interpretabilnost:** možemo objasniti zašto nešto slijedi iz premisa (pozivajući se na pravila zaključivanja) $\Rightarrow$ dobivamo dokaz

Deduktivna posljedica

Deduktivna posljedica

Formula G je **dedukcija** (engl. *deduction*) ili **deduktivna posljedica** (engl. *deductive consequence*) formula $F_1, F_2, \ldots, F_n$ akko je G moguće **izvesti** (engl. *derive*) iz premisa $F_1, F_2, \ldots, F_n$ pravilima zaključivanja.

Pišemo $F_1, F_2, \ldots, F_n \vdash G$ i čitamo "$F_1, \ldots, F_n$ **izvodi** (engl. *derives*) ili **deduktivno povlači** (engl. *deductively entails*) G".

Teorem

Formula G je **teorem** akko vrijedi $\vdash G$, tj. ako je formula G deduktivno izvediva iz praznog skupa premisa.

- Dokazati $F \vdash G$ je ekvivalentno kao dokazati $\vdash F \to G$
- Zato umjesto o izvođenju dedukcije govorimo o **dokazivanju teorema**

Pravila zaključivanja

- Primjer pravila zaključivanja:
  "Ako su dvije tvrdnje istinite, onda je istinita i njihova konjunkcija"

Pravilo konjunkcije

$$\frac{A \qquad B}{A \wedge B} \qquad \text{ili} \qquad A, B \vdash A \wedge B$$

- Što je sa sljedećim pravilom?

$$A \vee B \vdash A$$

- Intuitivno, prvo pravilo je **semantički ispravno**, a drugo nije
- Da bi pravilo bilo ispravno, deduktivan zaključak koji njime izvodimo mora biti **logička posljedica premisa**

Ispravnost i potpunost pravila

Ispravnost

Pravilo zaključivanja je **ispravno** (engl. *sound*) ako, primijenjeno na skup premisa, izvodi formulu koja je **logička posljedica** tih premisa.

Formalno, pravilo zaključivanja r je ispravno ako i samo ako

$$\text{ako } F_1, \ldots, F_n \vdash_r G \text{ onda } F_1, \ldots, F_n \models G$$

Potpunost

Skup pravila R je **potpun** (engl. *complete*) ako i samo ako je njime moguće izvesti sve logičke posljedice:

$$\text{ako } F_1, \ldots, F_n \models G \text{ onda } F_1, \ldots, F_n \vdash_R G$$

Ispravnost i potpunost povezuju semantiku i teoriju dokaza (povezuju u oba smjera relacije $\vdash$ i $\models$)

Ispravnost i potpunost – primjer

- Dokažimo da je pravilo $F \to G, F \vdash G$ **(modus ponens)** ispravno. Trebamo dokazati $F \to G, F \models G$. Izravan dokaz:

F	G	$F \to G$	$(F \to G) \wedge F$	$\big((F \to G) \wedge F\big) \to G$
$\bot$	$\bot$	$\top$	$\bot$	$\top$
$\bot$	$\top$	$\top$	$\bot$	$\top$
$\top$	$\bot$	$\bot$	$\bot$	$\top$
$\top$	$\top$	$\top$	$\top$	$\top$

- Dokažimo da pravilo $F \to G, G \vdash F$ **(abdukcija)** nije ispravno. Trebamo dokazati $F \to G, G \not\models F$. Izravan dokaz:

F	G	$F \to G$	$(F \to G) \wedge G$	$\big((F \to G) \wedge G\big) \to F$
$\bot$	$\bot$	$\top$	$\bot$	$\top$
$\bot$	$\top$	$\top$	$\top$	$\bot$
$\top$	$\bot$	$\bot$	$\bot$	$\top$
$\top$	$\top$	$\top$	$\top$	$\top$

Automatsko zaključivanje

- U umjetnoj nas inteligenciji zanima kako automatizirati dokazivanje
- Time se bavi područje **automatskog zaključivanja** (engl. *automated reasoning*) ili **automatskog dokazivanja teorema** (engl. *automated theorem proving*, ATP)
- Sustav koji implementira neki postupak dokazivanja naziva se **dokazivač teorema** (engl. *theorem prover*)
- Naravno, to što dokazivači teorema izvode mora biti **semantički ispravno**, dakle opravdivo u semantičkom smislu

Postupci dokazivanja

- U teoriji dokaza razvijeni su različiti **postupci dokazivanja** (engl. *proof methods*)
- Postupak mora biti semantički ispravan, a poželjno je da je i potpun
  - sustavi prirodnog zaključivanja (engl. *natural deduction systems*)
  - aksiomatski (hilbertovski) sustavi
  - sekventni računi (engl. *sequent calculi*)
  - tableau-metoda
  - metoda rezolucije (engl. *resolution method*)
- Mi ćemo se usredotočiti na metodu rezolucije. Ta je metoda ispravna i potpuna te se lako implementira na računalu

Sadržaj

Metoda rezolucije

- **Metoda rezolucije** (metoda razrješavanja) koristi se u propozicijskoj logici i logici prvoga reda
- Metodu je predložio J. A. Robinson 1965. godine
- Metoda se sastoji od samo jednog pravila zaključivanja:

Rezolucijsko pravilo

$$\frac{A \vee F \qquad \neg A \vee G}{F \vee G} \qquad \text{ili} \qquad A \vee F,\ \neg A \vee G \ \vdash\ F \vee G$$

što je ekvivalentno s:

$$\neg F \to A,\ A \to G \ \vdash\ \neg F \to G$$

- Prednost je što radimo sa samo jednim pravilom, a to bitno pojednostavljuje automatsko zaključivanje

Pravilo rezolucije – ispravnost

- Uvjerimo se da je pravilo rezolucije **ispravno**
- Trebamo dokazati da je deduktivna posljedica rezolucijskog pravila također i logička posljedica, tj. trebamo dokazati:

$$A \vee F,\ \neg A \vee G \ \models\ F \vee G$$

- Npr. izravnom metodom:

A	F	G	$\overbrace{A \vee F}^{P}$	$\neg A$	$\overbrace{\neg A \vee G}^{Q}$	$P \wedge Q$	$\overbrace{F \vee G}^{R}$	$(P \wedge Q) \to R$
$\top$	$\top$	$\top$	$\top$	$\bot$	$\top$	$\top$	$\top$	$\top$
$\top$	$\top$	$\bot$	$\top$	$\bot$	$\bot$	$\bot$	$\top$	$\top$
$\top$	$\bot$	$\top$	$\top$	$\bot$	$\top$	$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$	$\top$
$\bot$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$
$\bot$	$\top$	$\bot$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$
$\bot$	$\bot$	$\top$	$\bot$	$\top$	$\top$	$\bot$	$\top$	$\top$
$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\top$	$\bot$	$\bot$	$\top$

Klauzula

- Rezolucijsko pravilo može se primijeniti samo na disjunkcije
- Ako želimo primjenjivati isključivo rezolucijsko pravilo, premise trebaju biti u obliku disjunkcije. Takav oblik nazivamo **klauzula**

Klauzula (engl. *clause*)

Literal je atom ili njegova negacija. **Klauzula** je disjunkcija konačnog broja literala G_i:

$$G_1 \vee G_2 \vee \cdots \vee G_n, \quad n \geq 0$$

Klauzula koja sadrži samo jedan literal naziva se **jedinična klauzula** (engl. *unit clause*).

- Primjeri literala: A, F, $\neg A$, $\neg F$, G, $\neg G$
- Primjeri klauzula: $A \vee F$, $\neg A \vee G$, $A \vee \neg B \vee C \vee \neg D$, F

Rezolucija nad klauzulama

Rezolucijsko pravilo nad PL klauzulama

$$\frac{F_1 \vee \cdots \vee F_i \vee \cdots \vee F_n \qquad G_1 \vee \cdots \vee G_j \vee \cdots \vee G_m}{F_1 \vee \cdots \vee F_{i-1} \vee F_{i+1} \vee \cdots \vee F_n \vee G_1 \vee \cdots \vee G_{j-1} \vee G_{j+1} \vee \cdots \vee G_m}$$

gdje su F_i i G_j **komplementarni literali** (jedan je negacija drugoga).

Premise nazivamo **roditeljske klauzule**, a dedukciju nazivamo **rezolventa**.

- Primjeri:
  $A \vee B \vee \neg C,\ D \vee \neg B \vee E \ \vdash\ A \vee \neg C \vee D \vee E$
  $\neg A \vee B,\ A \ \vdash\ B$
  $A \vee B,\ A \vee \neg B \ \vdash\ A \vee A$
  $A \vee B,\ \neg A \vee \neg B \ \vdash\ B \vee \neg B$
  $A \vee B,\ \neg A \vee \neg B \ \vdash\ A \vee \neg A$

Konjunktivna normalna forma

- Ako su premise klauzule, skup premisa je **konjunkcija klauzula** (premise su implicitno povezane operatorom $\wedge$)
- **Q:** Ograničava li to primjenu rezolucije? **A:** Ne!
- Bilo koju formulu propozicijske logike moguće je prikazati kao konjunkciju klauzula pretvorbom u **konjunktivnu normalnu formu**

Konjunktivna normalna forma (engl. *conjuctive normal form*, CNF)

Formula F je u **konjunktivnoj normalnoj formi** akko je F u obliku

$$F_1 \wedge F_2 \wedge \cdots \wedge F_n$$

pri čemu je F_i oblika

$$G_{i1} \vee G_{i2} \vee \cdots \vee G_{im}$$

gdje su G_{ij} literali (atomi ili njihove negacije).

Pretvorba u CNF

- Svaka se formula može pretvoriti u CNF u četiri slijedna koraka

Pretvorba formule u CNF

(1) Uklanjanje ekvivalencije: $F \leftrightarrow G \equiv (\neg F \vee G) \wedge (\neg G \vee F)$

(2) Uklanjanje implikacije: $F \rightarrow G \equiv \neg F \vee G$

(3) Potiskivanje negacije do atoma: $\neg(F \vee G) \equiv \neg F \wedge \neg G$, $\neg(F \wedge G) \equiv \neg F \vee \neg G$

(4) Primjena distributivnosti: $F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$

Svaki se korak ponavlja sve dok je primjenjiv.
U svim koracima, kad god je to moguće, primijenjuje se ekvivalencija za involuciju $\neg\neg F \equiv F$.

Pretvorba u CNF – primjer

$$(C \vee D) \rightarrow (\neg A \leftrightarrow B)$$

(1) Uklanjanje ekvivalencije:

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

(2) Uklanjanje implikacije:

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

(3) Potiskivanje negacije:

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

(4) Primjena distributivnosti:

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

Klauzalni oblik

- Formula u konjunktivnoj normalnoj formi može se prikazati kao skup klauzula između kojih se implicitno podrazumijeva konjunkcija
- Klauzule se mogu prikazati kao skup literala između kojih se implicitno podrazumijeva disjunkcija
- Dakle, formula se može prikazati kao **skup skupova literala**
- To nazivamo **klauzalni oblik**
- Npr.:
  $(\neg C \vee A \vee B) \wedge (\neg D \vee A \vee B) \wedge (\neg C \vee \neg B) \Rightarrow$
  $\{\{\neg C, A, B\}, \{\neg D, A, B\}, \{\neg C, \neg B\}\}$
- Klauzule se također mogu pisati jedna ispod druge:
  $\neg C \vee A \vee B$
  $\neg D \vee A \vee B$
  $\neg C \vee \neg B$

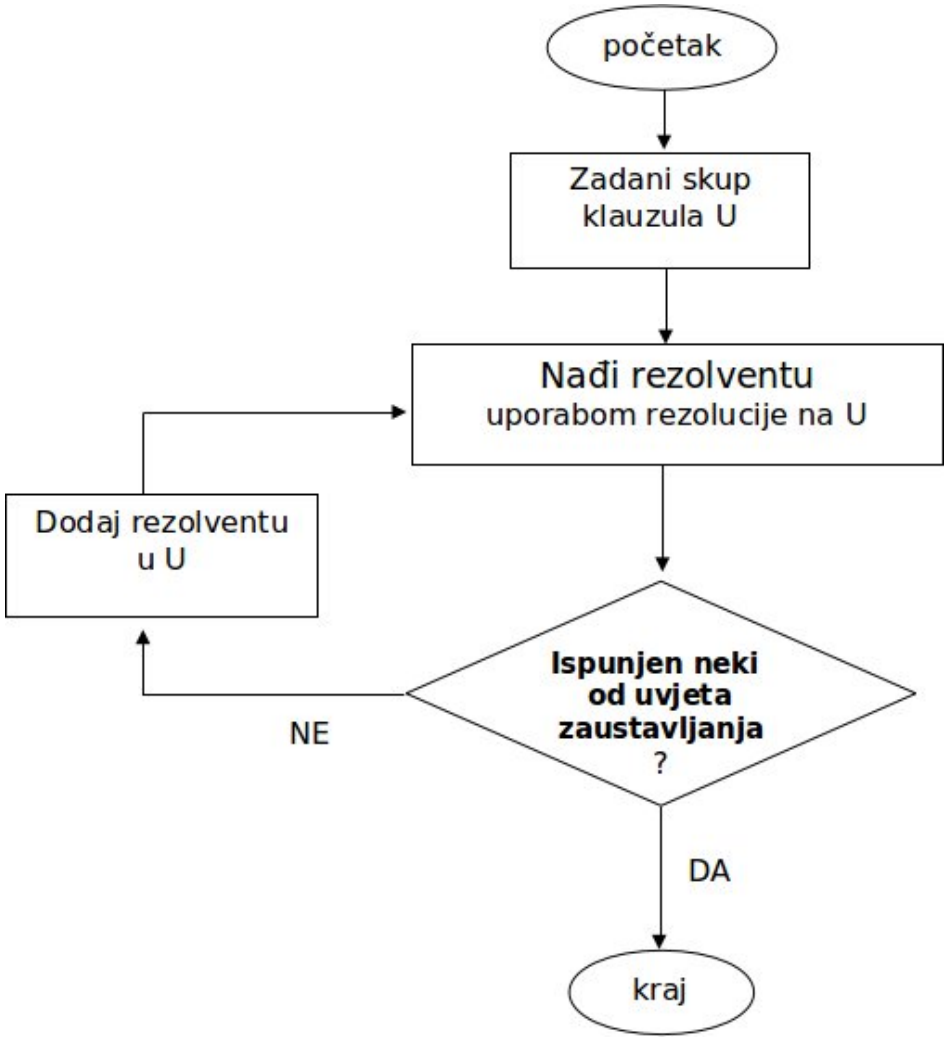
Rezolucija

Postupak se ponavlja sve dok:

(1) izvedena je ciljna formula

(2) ne može se izvesti nova formula

(3) isrcpljeni su računalni resursi



Rezolucija – primjer

- Dokažimo rezolucijom: $A \to B, B \to C, A \vdash C$
- Premise u klauzalnom obliku:
  (1) $\neg A \vee B$
  (2) $\neg B \vee C$
  (3) A
- Rezolucijskim postupkom izvodimo:
  (4) $\neg A \vee C$ (iz 1 i 2)
  (5) C (iz 3 i 4)
- Ili:
  (4') B (iz 1 i 3)
  (5') C (iz 2 i 4')

Nepotpunost rezolucije

- Dokazali smo da je rezolucijsko pravilo ispravno. No je li potpuno?
- Lako je pokazati da rezolucijsko pravilo nije potpuno
- Npr., razmotrimo dedukciju $F \vdash F \vee G$
- Nju ne možemo izvesti rezolucijskim pravilom (zašto?)
- Međutim, vrijedi $F \models F \vee G$ (provjerite!)
- Budući da vrijedi $F \models F \vee G$, a da rezolucijskim pravilom ne možemo deduktivno izvesti $F \vdash F \vee G$, zaključujemo da rezolucijskim pravilom ne možemo dokazati sve logičke posljedice, pa zaključujemo da **rezolucijsko pravilo nije potpuno**

Sadržaj

Izravna rezolucija vs. rezolucija opovrgavanjem

- Rezolucija koju smo do sada primjenjivali je **izravna rezolucija** (engl. *direct resolution*), koja pokušava izvesti G iz $F_1, \ldots, F_n$
- Postoji i **rezolucija opovrgavanjem** (engl. *refutation resolution*), koja pokušava dokazati da je $F_1 \wedge \cdots \wedge F_n \wedge \neg G$ nekonzistentno
- Izravna rezolucija je **nepotpuna**, međutim rezolucija opovrgavanjem je **potpuna**

Rezolucija opovrgavanjem (1)

- Umjesto da dokazujemo $F_1, \ldots, F_n \vdash G$, nastojimo dokazati da je $F_1 \wedge \cdots \wedge F_n \wedge \neg G$ proturječna formula
- Kao poseban slučaj rezolucijskog pravila imamo:

$$\frac{A \qquad \neg A}{\mathrm{NIL}}$$

- NIL označava prauznu klauzulu čija je semantička vrijednost $\bot$
- Ako rezolucijskim zaključivanjem izvedemo klauzulu NIL, onda znači da su premise proturječne (jer je rezolucijsko pravilo ispravno)

Rezolucija opovrgavanjem (2)

- Dokazano je (**ground resolution theorem**) da, uvijek kada je skup klauzula proturječan, rezolucijom možemo izvesti klauzulu NIL
- To znači da uvijek možemo dokazati nekonzistentnost skupa klauzula
- A to znači da možemo dokazati svaku logičku posljedicu. **Q:** Zašto?
- **A:** Zato što $F \models G$ možemo dokazati metodom opovrgavanja tako da dokažemo da je $F \wedge \neg G$ proturječna formula
- A to onda znači da je rezolucija opovrgavanjem potpuna, zato što njome možemo dokazati bilo koju logičku posljedicu
- Dakle, rezolucija opovrgavanjem je **ispravna i potpuna**!

Rezolucija opovrgavanjem – primjer 1

- Pokažimo da rezolucijom opovrgavanjem možemo dokazati $F \vdash F \vee G$:
- Negacija ciljne formule: $\neg(F \vee G) \equiv \neg F \wedge \neg G$
- Skup klauzula:

  (1) F
  (2) $\neg F$
  (3) $\neg G$

- Iz (1) i (2) izvodimo klauzulu NIL
- Dokazali smo da je skup klauzula proturječan, odnosno da je $F \vee G$ deduktivna/logička posljedica premise F

Rezolucija opovrgavanjem – primjer 2

Diplomatski problem

Kao predstavnik protokola, zaduženi ste poslati pozivnice za diplomatski bal koji se održava u ambasadi. Međutim, postoje ograničenja:

(1) Veleposlanik želi da, ako pozovete Tursku, svakako pozovete i UK

(2) Pomoćnik veleposlanika želi da pozovete Tursku ili Argentinu, ili obje.

(3) Zbog nedavnog diplomatskog incidenta, ne možete pozvati i UK i Argentinu.

Koga pozvati?
Dokažimo: *"Ako pozovemo Tursku, nećemo pozvati Argentinu"*

- Logički prikaz problema:
  $(T \to U) \wedge (T \vee A) \wedge \neg(U \wedge A)$
- Trebamo dokazati $T \to \neg A$ (cilj)

Rezolucija opovrgavanjem – primjer 2

- Pretvorba u klauzalni oblik:
  - (1) $U \vee \neg T$
  - (2) $T \vee A$
  - (3) $\neg U \vee \neg A$
- Negacija cilja: $\neg(T \rightarrow \neg A) \equiv \neg(\neg T \vee \neg A) \equiv T \wedge A$
- Nove klauzule:
  - (4) T
  - (5) A
- Rezolucijski postupak:
  - (6) U (iz 1 i 4)
  - (7) $\neg A$ (iz 3 i 6)
  - (8) NIL (iz 5 i 7)

Faktorizacija

- Rezolucija opovrgavanjem je potpuna uz uvjet da su klauzule **faktorizirane**
- Faktorizacija je primjena ekvivalencije $G \vee G \equiv G$ kojom se višekratno pojavljivanje istog literala zamjenuje jednim literalom
- Primjer:
  $\neg A \vee \neg A,\ A \vee A \ \vdash\ A \vee \neg A$
- Skup klauzula je proturječan, a izveli smo valjanu formulu
- **Q:** Je li to ispravno? **A:** Naravno da jest. Valjana formula je logička posljedica bilo koje formule. Ali od toga nemamo koristi.
- Međutim, da smo napravili faktorizaciju, dobili bismo:
  $\neg A,\ A \ \vdash\ \mathrm{NIL}$
- Kako bismo zadržali potpunost, treba primjenjivati faktorizaciju kad god je to moguće

Algoritam rezolucije opovrgavanjem

Algoritam rezolucije opovrgavanjem (za propozicijsku logiku)

function plResolution(F, G)
  $clauses \leftarrow$ cnfConvert($F \wedge \neg G$)
  $new \leftarrow \emptyset$
  loop do
    for each (c_1, c_2) **in** selectClauses($clauses$) **do**
      $resolvents \leftarrow$ plResolve(c_1, c_2)
      if NIL $\in resolvents$ **then return** $true$
      $new \leftarrow new \cup resolvents$
    if $new \subseteq clauses$ **then return** $false$
    $clauses \leftarrow clauses \cup new$

- cnfConvert – pretvara formulu u konjunktivan normalan oblik
- selectClauses – odabire skup parova klauzula za razrješavanje
- plResolve – razrješava roditeljske klauzule i vraća skup rezolventi

Algoritam rezolucije opovrgavanjem – napomene

- Primjena rezolucijskog pravila na par klauzula može dati više rezolventi, pa zato radimo sa skupom rezolventi
- Kako bi se zadržala potpunost, potrebno je uvijek **faktorizirati** sve dobivene rezolvente
- Broj mogućih različitih klauzula je konačan (ako se provodi faktorizacija), pa algoritam sigurno završava u konačnom broju koraka
- Treba voditi računa o tome koji su parovi već bili razriješeni i ne razrješavati ih ponovo
- Izvođenje klauzule NIL iz skupa klauzula zapravo je **problem pretraživanja**: u svakom koraku trebamo odabrati par klauzula koje ćemo razriješiti
- Treba nam **strategija pretraživanja**, koja se u kontekstu rezolucije naziva **rezolucijska strategija**

Rezolucijske strategije

- Dvije vrste **rezolucijskih strategija**:
  - strategije pojednostavljenja (engl. *simplification strategies*)
  - upravljačke strategije (engl. *control strategies*)
- **Strategije pojednostavljivanja** uklanjaju redundantne i nevažne klauzule generirane tijekom postupka dokazivanja čime se sprječava njihovo daljnje nepotrebno razrješavanje
- **Upravljačke strategije** određuju način odabira roditeljskih klauzula
- Rezolucijska strategija treba biti **potpuna**: mora izvesti NIL, ako je skup klauzula nekonzistentan
- To ne treba brkati s potpunošću pravila zaključivanja (općenito, da bi postupak dokazivanja bio potpun, trebamo kombinirati potpuna pravila s potpunom strategijom pretraživanja)

Strategija pojednostavljenja

Strategija brisanja

Uklanjanje **redundantnih** klauzula:

- Klauzula koja je **pokrivena** (engl. *subsumed*) drugom klauzulom može se obrisati
- Prema ekvivalenciji apsorpcije: $F \wedge (F \vee G) \equiv F$
- Ako se u skupu klauzula nađe par klauzula C_1 i C_2 takvih da $C_1 \subseteq C_2$, klauzula C_2 može se obrisati (klauzule su prikazane kao skupovi literala)

Uklanjanje **nevažnih** klauzula:

- Klauzula koja je **valjana** (tautologija) je nevažna (zašto?)
- Ako je rezolventa valjana klauzula, može ju se odmah pobrisati
- Provjera valjanosti klauzule je jednostavna: klauzula je valjana akko sadrži komplementaran par literala F_i i $\neg F_i$

Upravljačke rezolucijske strategije

Strategija zasićenja po razinama (engl. *level saturation strategy*)

- Rezolvente izvodimo razinu po razinu (kao kod pretraživanja u širinu): razrješavamo sve moguće parove klauzula na prvoj razini (početni skup klauzula), zatim na drugoj razini, itd.
- Na i-toj razini, roditeljske klauzule uzimaju se s razina 1 do $(i-1)$
- Ovo je potpuna strategija, ali je vrlo neučinkovita (problem kombinatorne eksplozije)

Upravljačke rezolucijske strategije

Strategija skupa potpore (engl. *set-of-support strategy*, SoS)

- Temelji se na pretpostavci da je skup ulaznih premisa **konzistentan**
- Naime, kada premise ne bi bile konzistentne, iz njih bi logički slijedila bilo koja formula!
- Stoga, kako bismo kod rezolucije opovrgavanjem dokazali nekonzistentnost, moramo kombinirati klauzule ulaznih premisa s klauzulama negiranog cilja, ili s novoizvedenim klauzulama
- **Skup potpore (SoS)**: klauzule dobivene negacijom cilja i sve novoizvedene klauzule
- **Strategija skupa potpore**: barem jedna roditeljska klauzula uvijek dolazi iz SoS
- SoS se povećava kako izvodimo nove klauzule
- Ova je strategija potpuna i u načelu učinkovitija od strategije zasićenja (pogotovo ako je SoS malen)

Sadržaj

Primjer: Carinici i diplomati

Premise

Carinici su pretražili svakoga tko je ušao u zemlju a nije diplomat. Neke krijumčare koji su ušli u zemlju pretražili su samo krijumčari. Niti jedan krijumčar nije diplomat.

Zaključak

Neki su carinici krijumčari.

Jednostavniji primjer

(1) *Svaki student pohađa predavanja.*
(2) *Ivan je student.*
$\vdash$ *Ivan pohađa predavanja.*

Zaključivanje pravilima **prirodnog zaključivanja**:

(1) $\forall x\big(S(x) \to P(x)\big)$
(2) $S(Ivan)$
(3) $S(Ivan) \to P(Ivan)$ (iz 1 pravilom univerzalne instancijacije)
(4) $P(Ivan)$ (iz 2 i 3 pravilom modus ponens)

Univerzalna instancijacija + modus ponens = generalizirani modus ponens

Kako bismo ovo mogli izvesti rezolucijskim pravilom?

Rezolucija u FOL-u – nacrt postupka

1. Pretvoriti formule (premisu i negirani cilj) u klauzalni oblik:

$$\begin{array}{lcl} \forall x\big(S(x) \rightarrow P(x)\big) & \Rightarrow & \neg S(x) \vee P(x) \\ S(Ivan) & \Rightarrow & S(Ivan) \\ P(Ivan) & \Rightarrow & \neg P(Ivan) \end{array}$$

2. Upariti komplementarne literale:

$$\begin{array}{rcll} \neg S(x) & \Leftrightarrow & S(Ivan) & \text{ako } x \leftarrow Ivan \\ P(x) & \Leftrightarrow & P(Ivan) & \text{ako } x \leftarrow Ivan \end{array}$$

   $\Rightarrow$ operacija unifikacije, koja rezultira supstitucijom varijabli

3. Primijeniti dobivene supstituciju i razriješiti klauzule:

$$\begin{array}{lcl} \neg S(x) \vee P(x), S(Ivan) & \vdash & P(Ivan) \\ P(Ivan), \neg P(Ivan) & \vdash & \text{NIL} \end{array}$$

Pretvaranje u klauzalni oblik

- Kao i kod PL, rezolucija u FOL iziskuje pretvaranje formule u **klauzalni oblik**
- Implicitno se podrazumijeva da su:
  - sve varijable u klauzuli **univerzalno kvantificirane**
  - između klauzula je konjunkcija
- Također, sve klauzule trebaju biti **standardizirane** – ne postoje dvije klauzule koje sadrže iste varijable
- Pretvorba u klauzalni oblik provodi se u **10 slijednih koraka**

Pretvaranje u klauzalni oblik

- **Korak 1: Uklanjanje ekvivalencije**

$$F \leftrightarrow G \equiv (\neg F \vee G) \wedge (\neg G \vee F)$$

- **Korak 2: Uklanjanje implikacije**

$$F \rightarrow G \equiv \neg F \vee G$$

- **Korak 3: Smanjivanje dosega operatora negacije** tako da se odnosi na samo jedan atom

$$\neg(F \vee G) \equiv \neg F \wedge \neg G$$
$$\neg(F \wedge G) \equiv \neg F \vee \neg G$$
$$\neg \forall x F(x) \equiv \exists x (\neg F(x))$$
$$\neg \exists x F(x) \equiv \forall x (\neg F(x))$$

- Po potrebi, u svakom od prethodna tri koraka primjenjuje se **involutivnost** $\neg\neg F \equiv F$ za eliminaciju dvostruke negacije

Pretvaranje u klauzalni oblik

- **Korak 4: Preimenovanje varijabli** tako da svaki kvantifikator veže jedinstvenu varijablu

$$(\forall x F(x) \vee \forall x G(x)) \equiv (\forall x F(x) \vee \forall y G(y))$$
$$(\forall x F(x) \vee \exists x G(x)) \equiv (\forall x F(x) \vee \exists y G(y))$$
$$(\exists x F(x) \vee \forall x G(x)) \equiv (\exists x F(x) \vee \forall y G(y))$$
$$(\exists x F(x) \vee \exists x G(x)) \equiv (\exists x F(x) \vee \exists y G(y))$$

$$(\forall x F(x) \wedge \forall x G(x)) \equiv (\forall x F(x) \wedge \forall y G(y))$$
$$(\forall x F(x) \wedge \exists x G(x)) \equiv (\forall x F(x) \wedge \exists y G(y))$$
$$(\exists x F(x) \wedge \forall x G(x)) \equiv (\exists x F(x) \wedge \forall y G(y))$$
$$(\exists x F(x) \wedge \exists x G(x)) \equiv (\exists x F(x) \wedge \exists y G(y))$$

Pretvaranje u klauzalni oblik – skolemizacija

- **Korak 5: Skolemizacija** – zamjena svih egzistencijalno kvantificiranih varijabli Skolem-izrazima
- Ako egzistencijalna varijabla ne ovisi o drugim varijablama: zamjena Skolem-konstantom

$$\exists x \mathrm{SESTRA}(x, Ivan)$$
$$\Rightarrow \quad \mathrm{SESTRA}(\underbrace{Ana}_{\text{Skolem-konstanta}}, Ivan)$$

- Ako egzistencijalna varijabla ovisi o drugim univerzalno kvantificiranim varijablama: zamjena Skolem-funkcijom

$$\forall x \exists y \mathrm{MAJKA}(y, x)$$
$$\Rightarrow \quad \forall x \mathrm{MAJKA}(\underbrace{f(x)}_{\text{Skolem-funkcija}}, x)$$

Pretvaranje u klauzalni oblik – skolemizacija

- Argumenti Skolem-funkcije su sve one univerzalno kvantificirane varijable čiji doseg uključuje doseg egzistencijalno kvanitificirane varijable koja se zamjenjuje

$$\begin{aligned} & \exists u \forall v \forall w \exists x \forall y \exists z F(u, v, w, x, y, z) \\ \Rightarrow \quad & \forall v \forall w \forall y F(a, v, w, f(v, w), y, g(v, w, y)) \end{aligned}$$

Niti jedan od simbola a, f i g ne smije se pojavljivati u izvornoj formuli

Thoraf Albert Skolem (1887–1963)

Norveški matematičar i logičar, jedan od utemeljitelja teorije modela.

Pretvaranje u klauzalni oblik – skolemizacija

- Što je opravdanje za skolemizaciju? Zašto bismo egzistencijalno kvantificiranu varijablu smjeli zamijeniti proizvoljnom konstantom?

$$\exists x \mathrm{SESTRA}(x, Ivan) \overset{???}{\equiv} \mathrm{SESTRA}(Ana, Ivan)$$

- Gornja ekvivalencija općenito ne vrijedi, ali to nije bitno
- Bitno je da **skolemizacija ne utječe na svojstvo nezadovoljivosti formule!**

$$\begin{array}{rl} \text{Ako} & \exists x \mathrm{SESTRA}(x, Ivan) \equiv \bot \\ \text{onda} & \mathrm{SESTRA}(Ana, Ivan) \equiv \bot \end{array}$$

- To znači: ako su premise i negirani cilj proturječni, bit će takvi i nakon skolemizacije
- U kontekstu rezolucije opovrgavanjem to je sve što nam treba

Pretvaranje u klauzalni oblik

- **Korak 6: Preneks-normalan oblik** – premještanje svih univerzalnih kvantifikatora na lijevu stranu fomule, zadržavajući pritom izvorni redoslijed kvantifikatora

$$\forall x F(x) \vee \forall y G(y) \equiv \forall x \forall y (F(x) \vee G(y))$$
$$\forall x F(x) \wedge \forall y G(y) \equiv \forall x \forall y (F(x) \wedge G(y))$$
$$\forall x F(x) \vee H\{x\} \equiv \forall x (F(x) \vee H\{x\})$$
$$\forall x F(x) \wedge H\{x\} \equiv \forall x (F(x) \wedge H\{x\})$$

- Niz kvantifikatora na lijevoj strani naziva se **prefiks**
- Desna strana formule, koja je oslobođena od kvantifikatora, naziva se **matrica**

Pretvaranje u klauzalni oblik

- **Korak 7: Uklanjanje prefiksa**. Preostaje samo matrice, za koju se implicitno podrazumijeva da su sve varijable univerzalno kvantificirane
- **Korak 8: Pretvorba matrice u CNF** korištenjem distributivnost

$$\big(F \vee (G \wedge H)\big) \equiv \big((F \vee G) \wedge (F \vee H)\big)$$
$$\big((F \wedge G) \vee H\big) \equiv \big((F \vee H) \wedge (G \vee H)\big)$$

- **Korak 9: Pretvorba u skup klauzula** uklanjanjem operatora $\wedge$, koji se implicitno podrazumijeva između klauzula
- **Korak 10: Standardizacija klauzula** preimenovanjem varijabli tako da ne postoje klauzule s identičnim varijablama, pomoću:

$$\forall x(F(x) \wedge G(x)) \equiv \forall x \forall y(F(x) \wedge G(y))$$

NB: Ne rade se preminovanja iste varijable unutar iste klauzule! Naime, općenito ne vrijedi:

$$\forall x P(x, x) \not\equiv \forall x \forall y P(x, y)$$
$$\forall x\big(P(x) \vee Q(x)\big) \not\equiv \forall x \forall y\big(P(x) \vee Q(y)\big)$$

Pretvaranje u klauzalni oblik – primjer

$$\forall y \forall z \Big(\exists u \big(P(y,u) \vee P(z,u) \big) \rightarrow \exists u \forall Q(y,z,u) \Big)$$

- Korak 1: Uklanjanje ekvivalencije $\Rightarrow$ OK
- Korak 2: Uklanjanje implikacije

$$\forall y \forall z \big(\neg (\exists u (P(y,u) \vee P(z,u))) \vee \exists u Q(y,z,u) \big)$$

- Korak 3: Smanjivanje dosega operatora negacije

$$\forall y \forall z \big(\forall u (\neg P(y,u) \wedge \neg P(z,u)) \vee \exists u Q(y,z,u) \big)$$

- Korak 4: Preimenovanje varijabli

$$\forall y \forall z \big(\forall u (\neg P(y,u) \wedge \neg P(z,u)) \vee \exists v Q(y,z,v) \big)$$

Pretvaranje u klauzalni oblik – primjer

- Korak 5: Skolemizacija

$$\forall y \forall z \big(\forall u (\neg P(y,u) \wedge \neg P(z,u)) \vee Q(y,z,f(y,z)) \big)$$

- Korak 6: Preneks-normalan oblik

$$\forall y \forall z \forall u \big((\neg P(y,u) \wedge \neg P(z,u)) \vee Q(y,z,f(y,z)) \big)$$

- Korak 7: Uklanjanje prefiksa

$$(\neg P(y,u) \wedge \neg P(z,u)) \vee Q(y,z,f(y,z))$$

- Korak 8: Pretvorba matrice u CNF

$$\big(\neg P(y,u) \vee Q(y,z,f(y,z)) \big) \wedge \big(\neg P(z,u) \vee Q(y,z,f(y,z)) \big)$$

Pretvaranje u klauzalni oblik – primjer

- Korak 9: Pretvorba u skup klauzula

$$\{\neg P(y,u) \vee Q(y,z,f(y,z)), \neg P(z,u) \vee Q(y,z,f(y,z))\}$$

- Korak 10: Standardizacija (uz $u \to v$, $y \to w$, $z \to x$)

$$\{\neg P(y,u) \vee Q(y,z,f(y,z)), \neg P(x,v) \vee Q(w,x,f(w,x))\}$$

Unifikacija

- Operacija **svođenje dva izraza na isti oblik**
- Sve varijable u klauzuli univerzalno su kvantificirane, pa vrijedi pravilo univerzalne instancijacije (eliminacije kvantifikatora $\forall$): ako se varijabla zamijeni bilo kojim izrazom, dobivena formula je logička posljedica izvorne formule

Primjeri

- $S(x) \Rightarrow S(Ivan) \Leftarrow S(Ivan)$, uz supstituciju varijable x izrazom $Ivan$, što označavamo s $\{Ivan/x\}$
- $S(x) \Rightarrow S(z) \Leftarrow S(y)$, uz $\{z/x, z/y\}$
- $Q(x, a) \Rightarrow Q(f(y), a) \Leftarrow Q(f(y), z)$, uz $\{f(y)/x, a/z\}$
- $Q(f(x), x) \Rightarrow Q(f(a), a) \Leftarrow Q(y, a)$, uz $\{f(a)/y, a/x\}$ kao kompoziciju supstitucija, $\{f(x)/y\} \circ \{a/x\} = \{f(a)/y, a/x\}$

- Unifikacija **ne mora uvijek uspijeti!**
  Npr. izraze $P(a)$ i $P(f(x))$ nije moguće unificirati

Supstitucija

Supstitucija

Neka su x_i varijable a t_i izrazi FOL-a. Skup uređenih parova

$$\alpha = \{t_1/x_1, t_2/x_2, \ldots, t_n/x_n\}$$

čini supstituciju (zamjenu) varijabli x_i izrazima t_i, uz uvjet $x_i \neq x_j$ za $i \neq j$ te $t_i \neq x_i$ za $i = 1, \ldots, n$.

- Primjena supstitucije α na neki općeniti izraz K: svako pojavljivanje varijable x_i u K zamjenuje se izrazom t_i
- Dobiveni općeniti izraz označavamo sa $K\alpha$ i kažemo da je $K\alpha$ instanca općenitog izraza K
  Npr. $K = P(x, f(y))$, $\alpha = \{a/x, b/y\}$: $K\alpha = P(a, f(b))$
- Za praznu supstituciju ε vrijedi $K\varepsilon = K$
- **NB:** Samo se varijable mogu supstituirati!

Kompozicija supstitucija

- Kompozicija supstitucija α i β, označena sa $\alpha \circ \beta$, je supstitucija za koju vrijedi $K(\alpha \circ \beta) = (K\alpha)\beta$ za svaki K

Izračun kompozicije supstitucija

Dane su supstitucije:
$\alpha = \{t_1/x_1, t_2/x_2, \ldots, t_n/x_n\}$ i $\beta = \{s_1/y_1, s_2/y_2, \ldots, s_m/y_m\}$.

Konstruiraj skupove:

$$S_1 = \{t_1\beta/x_1, t_2\beta/x_2, \ldots, t_n\beta/x_n, \underbrace{s_1/y_2, s_2, y_2, \ldots, s_m/y_m}_{\beta}\}$$

$$S_2 = \{t_i\beta/x_i \mid t_i\beta/x_i \in S_1, t_i\beta = x_i\}$$
$$S_3 = \{s_i/y_i \mid s_i/y_i \in S_1, y_i \in \{x_1, \ldots, x_n\}\}$$

Kompozicija supstitucija je:

$$\alpha \circ \beta = S_1 \setminus S_2 \setminus S_3$$

Kompozicija supstitucija – primjer

- Neka su zadane supstitucije:

$$\alpha = \{z/u, h(u)/w\}$$
$$\beta = \{a/u, z/w, u/z\}$$

- Izvedimo kompoziciju $\alpha \circ \beta$:

$$\begin{aligned} S_1 &= \{z\beta/u, h(u)\beta/w, a/u, z/w, u/z\} \\ &= \{u/u, h(a)/w, a/u, z/w, u/z\} \\ S_2 &= \{u/u\} \\ S_3 &= \{a/u, z/w\} \\ \alpha \circ \beta &= S_1 \setminus S_2 \setminus S3 = \{h(a)/w, u/z\} \end{aligned}$$

- Pokažimo da za $K = P(u, w, f(z))$ vrijedi $K(\alpha \circ \beta) = (K\alpha)\beta$:

$$\begin{aligned} P(u, w, f(z))(\alpha \circ \beta) &= P(u, h(a), f(u)) \\ \big(P(u, w, f(z))\alpha\big)\beta &= P(z, h(u), f(z))\beta = P(u, h(a), f(u)) \end{aligned}$$

Unifikacija

- Općeniti izrazi K_1 i K_2 mogu se svesti na isti oblik akko postoji supstitucija γ takva da:

$$\boxed{K_1\gamma = K_2\gamma}$$

- Supstitucija γ naziva se unifikator (engl. *unifier*), odnosno kaže se da su K_1 i K_2 unificirani pomoću γ
- Izraz $K_1\gamma$ odnosno $K_2\gamma$ naziva se zajednička instanca
- Dva izraza mogu imati više unifikatora

Primjer

Atomi $P(x)$ i $P(y)$ imaju unifikatore:

- $\gamma_1 = \{b/x, b/y\}$, koji daje zajedničku instancu $P(b)$
- $\gamma_2 = \{z/x, z/y\}$, koji daje zajedničku instancu $P(z)$

Instanca $P(z)$ je općenitija zajednička instanca nego $P(b)$. Zašto?

Najopćenitiji unifikator (MGU)

- Zanimaju nas unifikatori koji daju **što je moguće općenitiju zajedničku instancu**, jer tako osiguravamo općenitost zaključka, a time i potpunost postupka zaključivanja

Najopćenitiji unifikator (engl. *most general unifier, MGU*)

Supstitucija δ je najopćenitiji unifikator (MGU) akko za svaki unifikator γ od K_1 i K_2 postoji supstitucija θ za koju vrijedi $\gamma = \delta \circ \theta$.

- Intuitivno, γ je manje općenit unifikator koji se iz najopćenitijeg unifikatora δ može dobiti dodatnom supstitucijom θ

Primjer

Unifikatori za $P(x)$ i $P(y)$:

- $\delta = \{y/x\}$ (MGU)
- $\gamma = \{b/x, b/y\} = \delta \circ \theta$, gdje $\theta = \{b/y\}$

Algoritam MGU – primjer

- Nađi MGU izraza

$$K_1 = P(g(u), z, f(z)) \qquad K_2 = P(x, y, f(b))$$

- Korak 1:
  - $\{g(u)/x\}$ unificira prve podizraze od K_1 i K_2 koji se ne slažu
  - $K_1\{g(u)/x\} = P(g(u), z, f(z))$
  - $K_2\{g(u)/x\} = P(g(u), y, f(b))$
- Korak 2:
  - $\{y/z\}$ unificira sljedeće podizraze koji se ne slažu
  - kompozicija $\{g(u)/x\} \circ \{y/z\} = \{g(u)/x, y/z\}$
  - $K_1\{g(u)/x, y/z\} = P(g(u), y, f(y))$
  - $K_2\{g(u)/x, y/z\} = P(g(u), y, f(b))$
- Korak 3:
  - $\{b/y\}$ unificira posljednje podizraze koji se ne slažu
  - kompozicija $\{g(u)/x, y/z\} \circ \{b/y\} = \{g(u)/x, b/z, b/y\} = \delta$
  - $K_1\delta = P(g(u), b, f(b))$
  - $K_2\delta = P(g(u), b, f(b))$

Algoritam MGU

- **Ulaz**: dva općenita izraza, K_1 i K_2
- **Izlaz**: najopćenitiji unifikator, ako se K_1 i K_2 mogu unificirati, inače pogreška
- Razmotrit ćemo rekurzivni algoritam **MGUNIFIER** (Luger, Stubblefield, 1993; Shinghal, 1992)
- Algoritam izraze zapisuje u obliku ugniježđenih listi:

$$P(a,b) \Rightarrow [P,a,b]$$
$$P(f(a),g(x,y)) \Rightarrow [P,[f,a],[g,x,y]]$$

- Prvi element liste je glava liste, a ostatak je rep liste

$$K = [P,[f,a],[g,x,y]]$$
$$\text{head}(K) = P$$
$$\text{tail}(K) = [[f,a,],[g,x,y]]$$

Algoritam MGUNIFIER

function $\mathrm{mgUnifier}(K_1, K_2)$
  if $\mathrm{var}(K_1)$ **or** $\mathrm{fun}(K_1)$ **or** $\mathrm{pred}(K_1)$ **or** $K_1 = []$ **or**
    $\mathrm{var}(K_2)$ **or** $\mathrm{fun}(K_2)$ **or** $\mathrm{pred}(K_2)$ **or** $K_2 = []$ **then**
    if $K_1 = K_2$ **then return** $\varnothing$
    if $K_1 = []$ **or** $K_2 = []$ **then return** *fail*
    if $\mathrm{var}(K_1)$ **then**
      if $K_1 \in K_2$ **then return** *fail* **else return** $\{K_2/K_1\}$
    if $\mathrm{var}(K_2)$ **then**
      if $K_2 \in K_1$ **then return** *fail* **else return** $\{K_1/K_2\}$
    return *fail* -- *niti* K_1 *niti* K_2 *nisu varijable*
  else
    $\alpha \leftarrow \mathrm{mgUnifier}(\mathrm{head}(K_1), \mathrm{head}(K_2))$
    if $\alpha =$ *fail* **then return** *fail*
    $\beta \leftarrow \mathrm{mgUnifier}(\mathrm{tail}(K_1)\alpha, \mathrm{tail}(K_2)\alpha)$
    if $\beta =$ *fail* **then return** *fail*
    return $\alpha \circ \beta$

Neuspješna unifikacija

- MGUNIFIER vraća pogrešku u slučajevima kada unifikacija nije moguća
- Npr.:

K_1	K_2	Pogreška kod unifikacije
$P(a)$	$P(b)$	Neslaganje u simbolima konstanti
$P(f(x))$	$P(g(b))$	Neslaganje u sibolima funkcijama
$P(x)$	$Q(y)$	Neslaganje u simbolima predikata
$P(a)$	$P(x, b)$	Neslaganje u broju argumenata
$P(x)$	$P(f(x))$	Supstitucija varijable izrazom koji ju sadržava

Provjera pojavljivanja

- MGUNIFIER radi **provjeru pojavljivanja** (engl. *occurs check*): pojavljuje li se varijabla u izrazu kojim se zamjenjuje ($K_1 \in K_2$ odnosno $K_2 \in K_1$ u pseudokodu)
- Bez ove provjere unifikacija može dati cirkularnu supstituciju koja daje beskonačno ugniježđen izraz
- Npr. $K_1 = P(x, x)$ i $K_2 = P(f(y), y)$

$$\begin{aligned} \alpha = \{f(y)/x\} \Rightarrow \quad K_1\alpha &= P(f(y), f(y)) \\ K_2\alpha &= P(f(y), y) \\ \alpha = \{f(y)/y)\} \Rightarrow \quad K_1\alpha &= P(f(f(\cdots f(y)\cdots)), f(f(\cdots f(y)\cdots))) \\ K_2\alpha &= P(f(f(\cdots f(y)\cdots)), f(\cdots f(y)\cdots)) \end{aligned}$$

- Unifikacijom bez provjere pojavljivanja rezolucija gubi ispravnost
- Npr. moguće je dokazati $\forall x \exists y P(x, y) \vdash \exists y \forall x P(x, y)$, premda $\forall x \exists y P(x, y) \nvDash \exists y \forall x P(x, y)$

Unifikacija literala

- Za rezoluciju će nam trebati unifikacija literala

Unifikacija literala

Dva literala mogu se **unificirati** akko

- oba su negirani atomi ili oba su afirmativni atomi
- ti atomi se mogu unificirati

- Npr. $K_1 = P(x)$ i $K_2 = P(y)$ ili $K_1 = \neg P(x)$ i $K_2 = \neg P(y)$

Komplementarna unifikacija literala

Dva literala mogu se **komplementarno unificirati** akko

- jedan od njih je negirani atom a drugi je afirmativni atom
- ti atomi se mogu unificirati

- Npr. $K_1 = P(x)$ i $K_2 = \neg P(y)$ ili $K_1 = \neg P(x)$ i $K_2 = P(y)$

Sadržaj

Rezolucijsko pravilo za FOL

- Vrlo slično rezoluciji u PL, uz dodatak mehanizma unifikacije
- Rezolucija opovrgavanjem funkcionira na isti način kao u PL
- Roditeljske klauzule trebaju biti standardizirane

Rezolucijsko pravilo nad FOL klauzulama

$$\frac{F_1 \vee \cdots \vee F_i \vee \cdots \vee F_n \qquad G_1 \vee \cdots \vee G_j \vee \cdots \vee G_m}{F_1\delta \vee \cdots \vee F_{i-1}\delta \vee F_{i+1}\delta \vee \cdots \vee F_n\delta \vee G_1\delta \vee \cdots \vee G_{j-1}\delta \vee G_{j+1}\delta \vee \cdots \vee G_m\delta}$$

gdje su F_i i G_j literali koji se mogu **komplementarno unificirati** a δ je njihov najopćenitiji unifikator (MGU).

Rezolventa je disjunkcija svih preostalih literala roditeljskih klauzula uz **primjenu supstitucije** δ na svaki literal.

Razrješavanjem dviju jediničnih klauzula izvodi se **prazna klauzula** NIL.

Primjer 1

- Nađi rezolventu klauzula:

  (1) $P(g(y), x, f(z)) \vee Q(z, b) \vee R(x)$
  (2) $S(x, y) \vee \neg P(x, y, f(a))$

- Klauzule nisu standardizirane: preimenujmo varijable u prvoj klauzuli supstitucijom $\{w/x, u/y\}$
- Standardizirane klauzule:

  (1) $P(g(u), w, f(z)) \vee Q(z, b) \vee R(w)$
  (2) $S(x, y) \vee \neg P(x, y, f(a))$

- Razrješavanjem po komplementarnim literalima uz MGU $\delta = \{g(u)/x, y/w, a/z\}$ dobivamo:

  (3) $S(g(u), y) \vee Q(a, b) \vee R(y)$

Primjer 2

(1) *Svaki student pohađa predavanja.*

(2) *Ivan je student.*

$\vdash$ *Ivan pohađa predavanja.*

Pretvorba u klauzalni oblik i rezolucija opovrgavanjem:

(1) $\neg S(x) \vee P(x)$

(2) $S(Ivan)$

(3) $\neg P(Ivan)$ (negacija cilja)

(4) $\neg S(Ivan)$ (iz 1 i 3 uz $\delta = \{Ivan/x\}$)

(5) NIL (iz 2 i 4 uz $\delta = \varnothing$)

Faktorizacija FOL klauzula

- Kao i u PL, klauzule u FOL moraju biti **faktorizirane** kako bi se sačuvala **potpunost** rezolucije opovrgavanjem
- Primjer gubitka potpunosti zbog neprovođenja faktorizacije:

(1) $P(u) \vee P(w)$
(2) $\neg P(x) \vee \neg P(y)$
(3) $P(w) \vee \neg P(y)$ (iz 1 i 2 uz $\delta = \{u/x\}$)

(slično dobivamo razrješavanjem po drugim literalima)

- Klauzula se može faktorizirati akko ima literale koji se mogu unificirati
- Tako faktorizirana klauzula naziva se **faktor-klauzula**

(1) $P(u) \vee P(w)$
(2) $\neg P(x) \vee \neg P(y)$
(1') $P(w)$ (faktor-klauzula od 1 uz $\delta = \{w/u\}$)
(2') $\neg P(y)$ (faktor-klauzula od 2 uz $\delta = \{y/x\}$)
(3) NIL (iz 1' i 2' uz $\delta = \{w/y\}$)

Faktorizacija FOL klauzula

Faktor-klauzula

Neka klauzula

$$F_1 \vee \cdots \vee \cdots \vee F_i \vee \cdots \vee F_j \vee \cdots \vee F_n$$

sadržava literale F_i i F_j čiji je najopćenitiji unifikator δ. **Faktor-klauzula** ove klauzule je klauzula

$$F_1\delta \vee \cdots \vee F_i\delta \vee \cdots \vee F_n\delta$$

- Npr. za klauzulu

$$P(x, y, f(b)) \vee S(x, y) \vee P(g(u), w, f(z))$$

faktor-klauzula je

$$P(g(u), y, f(b)) \vee S(g(u), y) \qquad (\text{uz } \delta = \{g(u)/x, y/w, b/z\})$$

- **NB:** Jedna klauzula može imati više faktor-klauzula

Rezolucijsko pravilo u FOL – dorada

- Kako bismo **sačuvali potpunost** rezolucije opovrgavanjem, rezoluciju treba provoditi nad faktoriziranim i nad nefaktoriziranim roditeljskim klauzulama (sve kombinacije!)

Rezolucija nad FOL klauzulama

Neka su F i G **roditeljske klauzule** (klauzule koje sadrže literale koji se mogu komplementarno unificirati). **Rezolventa** ovih klauzula je svaka ona klazula dobivena:

(1) razrješavanjem F i G

(2) razrješavanjem F i faktor-klauzule od G

(3) razrješavanjem faktor-klauzule od F i G

(4) razrješavanjem faktor-klauzule od F i faktor-klauzule od G

- **NB:** Ako roditeljska klauzula ima više faktor-klauzula, sve kombinacije treba uzeti u obzir

Potpunost rezolucije opovrgavanjem i neodlučivost FOL

- Kao i kod PL, **rezolucija opovrgavanjem je ispravna i potpuna**
  - Ispravna: ako iz $F \wedge \neg G$ izvede NIL, onda je G logička posljedica od F
  - Potpuna: ako je G logička posljedica od F, onda iz $F \wedge \neg G$ izvodi NIL
- Dakle, logička posljedica i deduktivna posljedica su jedno te isto
- Potpunost je dokazao je J.A. Robinson (1965.), ali je K. Gödel već ranije dokazao postojanje takvog postupka (1929.)
- Međutim, za razliku od PL, FOL **nije odlučiva**!

Neodlučivost (engl. *undecidability*) problema valjanosti u FOL

Ne postoji algoritam koji za svaku formulu F daje "da" ako je F valjana ili "ne" ako F nije valjana.

- Neodlučivost u FOL dokazali su A. Church i A. Turing (1935.)

$\Rightarrow$ ne postoji algoritam dokazivanja svake **logičke posljedice**

- naime, prema **teoremu semantičke dedukcije**, $F \models G$ akko $\models F \rightarrow G$

Poluodlučivost FOL

- Preciznije, FOL je **poluodlučiva**:
  - postoje algoritmi koji daju "da" ako je F valjana, ali ako F nije valjana, algoritam može nikada ne završiti
- Rezolucija opovrgavanjem jedan je takav algoritam
- Zbog poluodlučivosti, moć rezolucije opovrgavanjem je ograničena:
  - Ako G jest logička posljedica od F, postupak će uvijek izvesti NIL
  - Ako G nije logička posljedica F, postupak može nikada ne završiti

Primjer

Vrijedi

$$\forall x\big(\forall y P(y) \to P(x)\big) \not\models \forall x P(x)$$

no rezolucija opovrgavanjem to ne može dokazati (postupak ne završava).

Primjer: Robot i paketi

- Robot dostavlja pakete. Robot zna da su svi paketi u sobi 27 manji od svakog paketa u sobi 28. A i B su paketi. Paket A je u sobi 27 ili u sobi 28, ali robot ne zna u kojoj. Paket B je u sobi 27 i nije manji od paketa A.
- Rezolucije opovrgavanjem pokažimo kako robot može zaključiti da je paket A u sobi 27.

- Prikazivanje znanja:

(1) $\forall x \forall y \Big(\big(P(x) \wedge P(y) \wedge U(x, 27) \wedge U(y, 28) \big) \rightarrow M(x, y) \Big)$

(2) $P(A) \wedge P(B)$

(3) $U(A, 27) \vee U(A, 28)$

(4) $U(B, 27) \wedge \neg M(B, A)$

$\vdash$ $U(A, 27)$

Primjer: Robot i paketi

- Premise i negacija cilja u klauzalnom obliku:

(1) $\neg P(x) \vee \neg P(y) \vee \neg U(x, 27) \vee \neg U(y, 28) \vee M(x, y)$

(2) $P(A)$

(3) $P(B)$

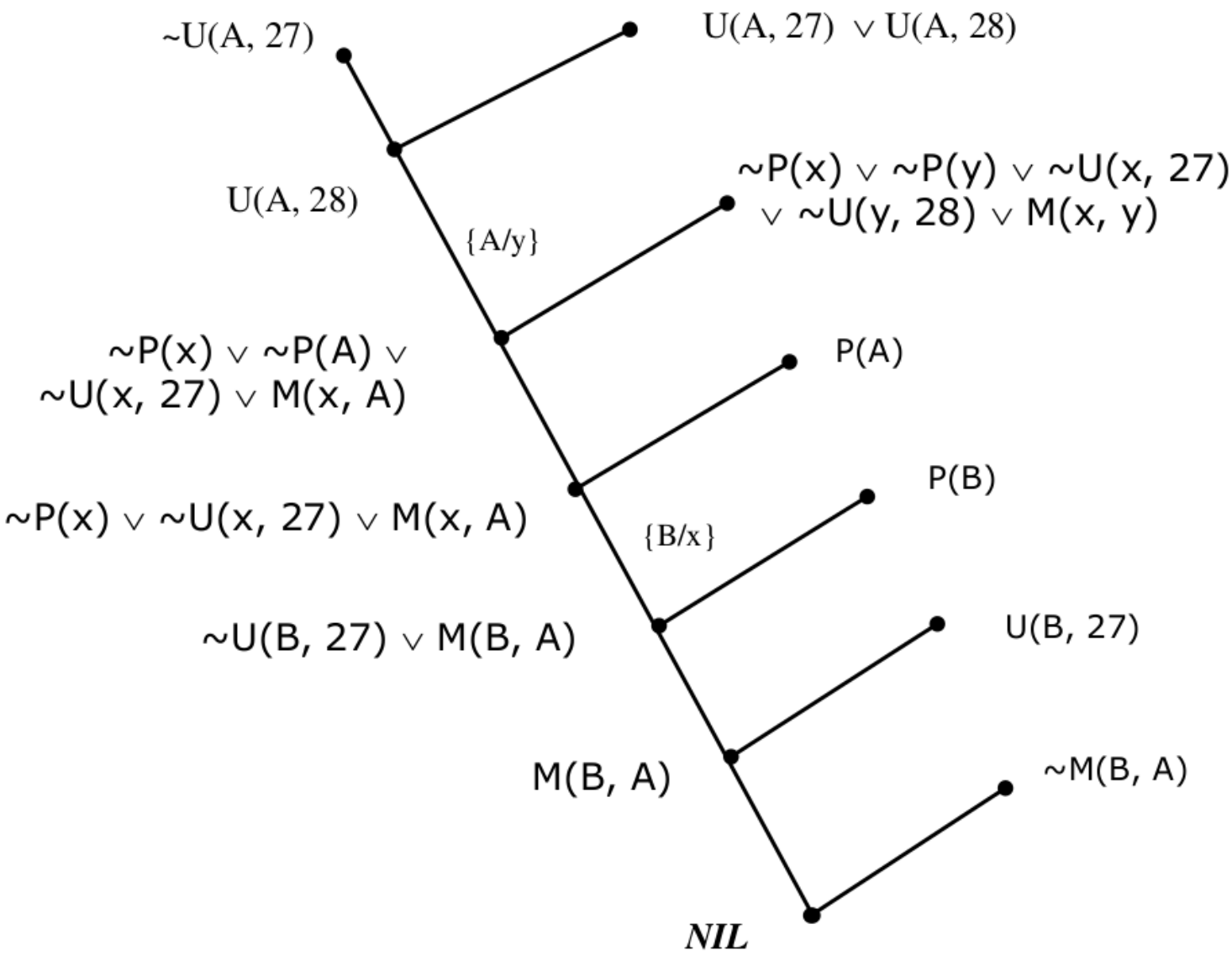
(4) $U(A, 27) \vee U(A, 28)$

(5) $U(B, 27)$

(6) $\neg M(B, A)$

(7) $\neg U(A, 27)$ (negacija cilja)

Primjer: Robot i paketi



Primjer: Carinici i diplomati

Premise

Carinici su pretražili svakoga tko je ušao u zemlju a nije diplomat. Neke krijumčare koji su ušli u zemlju pretražili su samo krijumčari. Niti jedan krijumčar nije diplomat.

Zaključak

Neki su carinici krijumčari.

Primjer: Carinici i diplomati

(1) $\forall x\Big(\big(U(x) \wedge \neg D(x)\big) \rightarrow \exists y\big(C(y) \wedge P(y,x)\big)\Big)$

(2) $\exists x\Big(K(x) \wedge U(x) \wedge \forall y\big(P(y,x) \rightarrow K(y)\big)\Big)$

(3) $\forall x\big(K(x) \rightarrow \neg D(x)\big)$

$\vdash$ $\exists x\big(C(x) \wedge K(x)\big)$

(1) $\neg U(x) \vee D(x) \vee C(f(x))$

(2) $\neg U(z) \vee D(z) \vee P(f(z), z))$

(3) $K(a)$

(4) $U(a)$

(5) $\neg P(y,a) \vee K(y)$

(6) $\neg K(v) \vee \neg D(v)$

(7) $\neg C(w) \vee \neg K(w)$

Primjer: Carinici i diplomati

(8) $\neg U(x) \vee D(x) \vee \neg K(f(x))$ (iz 1 i 7 uz $\delta = \{f(x)/w\}$)

(9) $D(a) \vee \neg K(f(a))$ (iz 4 i 8 uz $\delta = \{a/x\}$)

(10) $\neg P(f(a), a) \vee D(a)$ (iz 5 i 9 uz $\delta = \{f(a)/y\}$)

(11) $\neg U(a) \vee D(a) \vee D(a)$ (iz 2 i 10 uz $\delta = \{a/z\}$)

(12) $D(a)$ (iz 4 i 11 uz $\delta = \varnothing$)

(13) $\neg K(a)$ (iz 6 i 12 uz $\delta = \{a/v\}$)

(14) NIL (iz 3 i 13 uz $\delta = \varnothing$)

Sažetak

- Teorija dokaza koristi **pravila zaključivanja** da bi izvela **deduktivnu posljedicu**, bez eksplicitnog pozivanja na semantiku logike
- Pravila zaključivanja moraju biti **ispravna** i poželjno je da su **potpuna**
- **Rezolucijsko pravilo** je jednostavno pravilo koje je ispravno
- Prije primjene rezolucije, formule treba pretvoriti u **klauzalni oblik**
- Za razrješavanje FOL literala koristimo **unifikaciju**, koja nalazi **supstituciju** varijabli za svođenje dvaju izraza na isti oblik
- **Rezolucijske strategija** pojednostavljuje ili upravlja postupkom dokazivanja (npr. **strategija skupa potpore**)
- **Rezolucija opovrgavanjem** (uz standardizaciju, faktorizaciju i potpunu strategiju) je ispravna i potpuna u PL i FOL
- Zbog **poluodlučivosti** moć rezolucije u FOL je ograničena

Sljedeća tema: Logičko programiranje