Logická reprezentácia a usudzovanie

Znalostný agent

- rozumný agent, ktorého konanie sa bude zakladať na znalostiach
- znalostný agent získa schopnosť riešiť nový druh problémov naučením alebo získaním nových znalostí o prostredí
- jazyk pre znalostný agent: zápis poznatkov + odvodzovanie
- v matematickej (formálnej) logike sa skúmajú spôsoby *odvodzovania* dôsledkov zo známych axióm, t.j. logické usudzovanie
- *báza poznatkov*
  - množina zápisov faktov o svete vo forme *viet* v jazyku pre reprezentáciu poznatkov

Znalostný agent

- operácie nad bázou poznatkov:
  - pridanie novej vety
  - pýtanie sa na niečo známe
- určovanie toho, čo vyplýva z bázy poznatkov, zabezpečuje **odvodzovací stroj**

```
function ZNALOSTNÝ-AGENT(vnem) returns akcia
        static: BP, báza poznatkov
                t, počítadlo, ktoré indikuje čas; na začiatku 0

    PRIDAJ (BP, VYTVOR-VETU-O-VNEME (vnem, t))
    akcia ← ODPOVEDAJ(BP, VYTVOR-DOPYT-NA-AKCIU (t))
    PRIDAJ (BP, VYTVOR-VETU-O-AKCII (akcia, t))
    t ← t + 1
return akcia
```


Znalostný agent - úrovne abstrakcie opisu

- **znalostná úroveň** (čo vie)
  - agent zabezpečujúci riadenie vozidla môže vedieť, že do časti Bratislava Petržalka sa možno dostať z časti Staré Mesto po Starom moste
- **logická úroveň** (zápis vo vetách)
  - *spája* (*Starý most, Staré Mesto, Petržalka*).
- **implementačná úroveň** (technické vybavenie agenta)
  - `"spaja(stary_most,stare_mesto, petrzalka)"`
  - reťazec `"stary_most"` v dvojrozmernej tabuľke prepojení častí Bratislavy

Logika a reprezentácia poznatkov

- jazyk je prostriedok na vyjadrenie faktov okolitého sveta. To, ktorá *syntaktická* jednotka jazyka (ktorá veta) vyjadruje ktorý fakt, určuje *sémantika* jazyka.
- *proces usudzovania* musí byť taký, že nové fakty vyplývajú z doteraz známych faktov.
  - z danej množiny faktov **vyplýva** nový fakt
  - z množiny viet (reprezentujúcich danú množinu faktov) **logicky vyplýva** nová veta (reprezentujúca nový fakt).

Logika a reprezentácia poznatkov

- **odvodzovanie** (v logike) - spôsob skúmania, či z množiny faktov vyplýva iný fakt
  - musí zachovávať *pravdivosť*
  - *úplné*, ak nájde *dôkaz* každej vety logicky vyplývajúcej z množiny formúl
- **dôkaz**
  - odvodzovacími pravidlami vytvorená postupnosť formúl nad množinou pôvodných a v predchádzajúcich krokoch odvodených formúl

Logika a reprezentácia poznatkov

- báza poznatkov i jednotlivé fakty sú súčasťou okolitého sveta
- množina formúl i jednotlivé formuly sú súčasťou reprezentácie sveta
- vzťah medzi nimi:
  - **syntax** jazyka pre reprezentáciu poznatkov
  - **sémantika** jazyka pre reprezentáciu poznatkov

Logika a reprezentácia poznatkov

- **interpretácia** je zobrazenie, ktoré priraďuje formulám fakty
- interpretovaním formuly sa z nej stáva **výrok** (ne- / pravdivý)
- stav sveta, pre ktorý je nejaká formula pravdivá pri danej interpretácii, predstavuje **model** (tej formuly)
- formula je:
  - **splniteľná**, ak existuje interpretácia, pri ktorej je pravdivá (má model)
  - **nesplniteľná**

Model formuly - príklad

Nech je daná formula $\forall x \exists y P(x, y)$ a interpretácia *I* predikátového symbolu *P* je takáto:

***P* ∈ N^2 je relácia menší (<), kde *N* je množina prirodzených čísel.**

Potom interpretácia *I* je modelom tejto formuly, lebo v *I* formula vyjadruje pravdivé tvrdenie:
"*pre každé prirodzené číslo existuje prirodzené číslo, ktoré je od neho väčšie*".

Na druhej strane *I* nie je modelom formuly $\exists y \forall x P(x, y)$,
ktorej slovenská parafráza je takáto:
"*existuje prirodzené číslo také, že všetky prirodzené čísla sú od neho menšie*".

Logika a reprezentácia poznatkov

- **tautológia** - formula pravdivá pri ľubovoľnej interpretácii
- **model množiny formúl -** interpretácia, ktorá je modelom každej formuly z tejto množiny
- postup určenia, či formula logicky vyplýva z nejakej množiny formúl:
  - preskúšať všetky interpretácie
  - zúžiť interpretácie na používané v báze poznatkov
  - *znalostný agent rozhoduje* nie na základe sémantiky formúl, ale len *na základe syntaxe*

Logika a reprezentácia poznatkov

- **odvodzovanie**
  - formálny a mechanický proces
  - neobmedzené rozsahom použitých poznatkov (počtom a zložitosťou formúl)
- jazyk *logiky* (syntax aj sémantika) a teória dôkazu
  - študuje pravidlá odvodzovania dôsledkov vyplývajúcich z množiny viet

Logika a reprezentácia poznatkov

- logika v UI
  - výroková
  - predikátová
  - temporálna (čas)
  - modálna („možno platí, že“)
  - viachodnotová
  - fuzzy (neostrá, rozmazaná)

Výroková logika

- syntax (BNF - Backusov Naurov tvar)

formula	→	*jednoduchá_formula* | *zložená_formula*
jednoduchá_formula	→	`True | False | P | Q | …`
zložená_formula	→	(*formula*) | *formula spojka formula* | ¬*formula*
spojka	→	∧ | ∨ | ⇔ | ⇒

- terminálne symboly
  - Logické konštanty True a False,
  - výrokové premenné P, Q, R a pod.,
  - spojky ∧, ∨, ⇔, ⇒ (dvojmiestne funktory), ¬ (jednomiestny funktor)
  - zátvorky (,)
- neterminálny symbol *formula*
- **literál**

Výroková logika

- sémantika
  - interpretácia - ohodnotenie pravdivostnou hodnotou
  - pravdivostné hodnoty *Pravda, Lož*
  - (logické konštanty `True`, `False`)
- pravdivostné tabuľky

P	Q	¬P	P ∧Q	P ∨Q	P ⇒Q	P ⇔Q
Lož	*Lož*	*Pravda*	*Lož*	*Lož*	*Pravda*	*Pravda*
Lož	*Pravda*	*Pravda*	*Lož*	*Pravda*	*Pravda*	*Lož*
Pravda	*Lož*	*Lož*	*Lož*	*Pravda*	*Lož*	*Lož*
Pravda	*Pravda*	*Lož*	*Pravda*	*Pravda*	*Pravda*	*Pravda*

Výroková logika – príklad

- P ⇒ (Q ⇒ P)

P	Q	(Q ⇒ P)	P ⇒ (Q ⇒ P)
Lož	*Lož*	*Pravda*	*Pravda*
Lož	*Pravda*	*Lož*	*Pravda*
Pravda	*Lož*	*Pravda*	*Pravda*
Pravda	*Pravda*	*Pravda*	*Pravda*

- Tautológia

Výroková logika - odvodzovacie pravidlá

- pravidlo odlúčenia (modus ponens)

$$\frac{\alpha,\ \alpha \Rightarrow \beta}{\beta}$$

- pravidlo vovedenia dizjunkcie

$$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n}$$

- pravidlo odstránenia konjunkcie

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}{\alpha_i}$$

- pravidlo odstránenia dvojitej negácie

$$\frac{\neg\neg\alpha}{\alpha}$$

- pravidlo vovedenia konjunkcie

$$\frac{\alpha_1, \alpha_2, \ldots, \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}$$

- pravidlo jednotkovej rezolvencie

$$\frac{\alpha \vee \beta,\ \neg\beta}{\alpha}$$

Výroková logika - odvodzovacie pravidlá

- Pravidlo rezolvencie $\dfrac{\alpha \vee \beta,\ \neg\beta \vee \gamma}{\alpha \vee \gamma}$

α	β		α ∨ β		
Lož	*Lož*		*Lož*		
Lož	*Lož*		*Lož*		
Lož	*Pravda*		*Pravda*		
Lož	*Pravda*		*Pravda*		
Pravda	*Lož*		*Pravda*		
Pravda	*Lož*		*Pravda*		
Pravda	*Pravda*		*Pravda*		
Pravda	*Pravda*		*Pravda*		

Výroková logika - odvodzovacie pravidlá

- Pravidlo rezolvencie $\dfrac{\alpha \vee \beta,\ \neg\beta \vee \gamma}{\alpha \vee \gamma}$

	β	γ		¬β ∨ γ	
	Lož	*Lož*		*Pravda*	
	Lož	*Pravda*		*Pravda*	
	Pravda	*Lož*		*Lož*	
	Pravda	*Pravda*		*Pravda*	
	Lož	*Lož*		*Pravda*	
	Lož	*Pravda*		*Pravda*	
	Pravda	*Lož*		*Lož*	
	Pravda	*Pravda*		*Pravda*	

Výroková logika - odvodzovacie pravidlá

- Pravidlo rezolvencie $\dfrac{\alpha \vee \beta,\ \neg\beta \vee \gamma}{\alpha \vee \gamma}$

			$\alpha \vee \beta$	$\neg\beta \vee \gamma$	$\alpha \vee \gamma$
			Lož	*Pravda*	*Lož*
			Lož	*Pravda*	*Pravda*
			Pravda	*Lož*	*Lož*
			Pravda	*Pravda*	*Pravda*
			Pravda	*Pravda*	*Pravda*
			Pravda	*Pravda*	*Pravda*
			Pravda	*Lož*	*Pravda*
			Pravda	*Pravda*	*Pravda*

Výroková logika - odvodzovacie pravidlá

- Pravidlo rezolvencie $\dfrac{\alpha \vee \beta,\ \neg\beta \vee \gamma}{\alpha \vee \gamma}$

α	β	γ	$\alpha \vee \beta$	$\neg\beta \vee \gamma$	$\alpha \vee \gamma$
Lož	*Lož*	*Lož*	*Lož*	*Pravda*	*Lož*
Lož	*Lož*	*Pravda*	*Lož*	*Pravda*	*Pravda*
Lož	*Pravda*	*Lož*	*Pravda*	*Lož*	*Lož*
Lož	*Pravda*	*Pravda*	*Pravda*	*Pravda*	*Pravda*
Pravda	*Lož*	*Lož*	*Pravda*	*Pravda*	*Pravda*
Pravda	*Lož*	*Pravda*	*Pravda*	*Pravda*	*Pravda*
Pravda	*Pravda*	*Lož*	*Pravda*	*Lož*	*Pravda*
Pravda	*Pravda*	*Pravda*	*Pravda*	*Pravda*	*Pravda*

Výroková logika - odvodzovanie

- použitie odvodzovacích pravidiel
- dôkaz formuly

```
function VÝROKOVÝ-ZNALOSTNÝ-AGENT(vnem) returns akcia
   static: BP, báza poznatkov
           t, počítadlo, ktoré indikuje čas; na začiatku 0

   PRIDAJ (BP, VYTVOR-VETU-O-VNEME (vnem, t))
   for each akcia in zoznam možných akcií do
         if ODPOVEDAJ (BP,VYTVOR-DOPYT-NA-AKCIU (t, akcia)) then
                t ← t + 1
                return akcia
         end
```


Predikátová logika

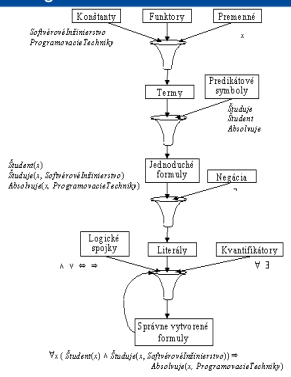
- syntax (BNF - Backusov Naurov tvar)

formula	→	*jednoduchá_formula* | *formula spojka formula* | *kvantifikátor premenná*, … *formula* | ¬ *formula* | (*formula*)
jednoduchá_formula	→	*predikátový_symbol*(*term*,…) | *term* = *term*
term	→	*funktor*(*term*,…) | *konštanta* | *premenná*
spojka	→	∧ | ∨ | ⇒ | ⇔
kvantifikátor	→	∀ | ∃
konštanta	→	*Identifikátor*
premenná	→	*Identifikátor*
predikátový_symbol	→	*identifikátor*
funktor	→	*identifikátor*

Predikátová logika

- sémantika:
  - interpretácia konštanty jej priraďuje objekt okolitého sveta
  - interpretáciou predikátového symbolu je relácia medzi objektmi okolitého sveta
  - interpretáciou funktora je funkcia nad objektmi okolitého sveta
- term
- jednoduchá formula

Predikátová logika



Predikátová logika – kvantifikátory

- **univerzálny kvantifikátor ∀**
  - „Všetky cesty vedú do Ríma.“
  - “pre ľubovoľný objekt *x*, ak *x* je cesta, tak *x* vedie do Ríma”.
  - $\forall x\ Cesta(x) \Rightarrow VedieDoRíma(x)$

    $Cesta(ViaAppia) \Rightarrow VedieDoRíma(ViaAppia) \wedge$

    $Cesta(ViaSalaria) \Rightarrow VedieDoRíma(ViaSalaria) \wedge$

    $Cesta(ViaCassia) \Rightarrow VedieDoRíma(ViaCassia) \wedge$

    $Cesta(ViaAurelia) \Rightarrow VedieDoRíma(ViaAurelia) \wedge \ldots$

- ak je formula pravdivá, vypovedá to, čo tvorí pravú stranu implikácie, iba o objektoch, ktoré spĺňajú ľavú stranu

Predikátová logika – kvantifikátory

- **existenčný kvantifikátor ∃**
  - „Do Ríma sa dá dôjsť po ceste.“
  - “existuje cesta *x* taká, že *x* vedie do Ríma”.
  - $\exists x\ Cesta(x) \wedge VedieDoRíma(x)$

    $Cesta(ViaAppia) \wedge VedieDoRíma(ViaAppia) \vee$

    $Cesta(ViaSalaria) \wedge VedieDoRíma(ViaSalaria) \vee$

    $Cesta(ViaAurelia) \wedge VedieDoRíma(ViaAurelia) \vee \ldots$

- **vnorené kvantifikátory**
  - ak dva kvantifikátory v jednej formule vovádzajú tú istú premennú, “patrí” príslušný výskyt bližšiemu

    $\forall x\ [Cesta(x) \vee \exists x\ VedieDo(Rím, x)]$

    alebo

    $\forall x\ [Cesta(x) \vee \exists y\ VedieDo(Rím, y)]$

Predikátová logika – kvantifikátory

- vzťah medzi existenčným a všeobecným kvantifikátorom
  - “všetci ľudia nemajú krídla”

    “neexistuje taký človek, ktorý má krídla”
  - $\forall$ č $\neg$ *Má* (č , *Krídla*)

    $\neg\exists$ č *Má* (č , *Krídla*)

- pravidlá pre kvantifikátory

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

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

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

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

Predikátová logika

- reprezentácia poznatkov
  - formuly (axiómy, definície)

    PRIDAJ(*BP*, $\forall s\ \forall v$ *StarýRodič*(*s*, *v*) $\Leftrightarrow \exists r$ *Rodič*(*s*,*r*)$\wedge$ *Rodič*(*r*, *v*))
  - dopyty

    ODPOVEDAJ(*BP*, $\exists s$ *StarýRodič*(*s*, *Lenny*))

- reprezentácia zvláštnych druhov poznatkov
  - vyjadrenie zmien stavu sveta

Príklad – reprezentácia poznatkov v predikátovej logike

- Každý, kto je spôsobilý na jazdu a má auto, môže jazdiť. Na jazdu je spôsobilý ten, kto má vodičský preukaz. Peter má vodičský preukaz a požičal si auto. Tí, čo jazdia, nemajú problémy s dopravou. Dá sa nájsť niekto, kto nemá problémy s dopravou?

- **Opis problému:**

  $\forall$x spôsobilý(x) $\wedge$ má_auto(x) $\Rightarrow$ jazdí(x)

  $\forall$x má_vodičský_preukaz(x) $\Rightarrow$ spôsobilý(x)

  $\exists$x = Peter má_vodičský_preukaz(x) $\wedge$ má_auto(x)

  $\forall$x jazdí(x) $\Rightarrow$ nemá_problémy_s_dopravou(x)

  Otázka:

  $\exists$x nemá_problémy_s_dopravou(x)

Predikátová logika

- situačný počet
  - vlastnosť/vzťah meniaci sa v čase sa vyjadruje predikátom s argumentom navyše

    JeNa (*RobotArnold*, *Pitvor*, S_1)
  - dôsledok:

    JeNa (*RobotArnold*, *Pitvor*, S_1) $\wedge \neg$*JeNa* (*RobotArnold*, *Pitvor*, S_2)
  - zmena stavu sveta:

    $\neg$*JeNa* (*RobotArnold*, *Pitvor*, S_1) $\Rightarrow$

    JeNa (*RobotArnold*, *Pitvor*, *ĎalšiaSituácia* (*Vstúp*, S_1))
  - riešenie problémov - *dôkaz* existenčnej cieľovej formuly a *postupnosť akcií*

Axiómy v situačnom počte

- **fakty (začiatočný stav S_0)**
  "robot Arnold sa nachádza v pitvore, debna B leží na debne A, debny B a C sú voľné."
  JeNa(RobotArnold, Pitvor, S_0)
  LežíNa(B, A, S_0)
  VoľnýVrch(B, S_0) ∧ VoľnýVrch(C, S_0)
- **všeobecné tvrdenia**
  ∀x ∀y ∀s [LežíNa(x, y, s) ⇒ ¬LežíNa(y, x, s)]
- **akcie a ich účinky**
  ∀x ∀s [VoľnýVrch(x, s) ⇒ Zdvihnutý(x, ĎalšiaSituácia(Zdvihni(x), s)]
- **rámcové axiómy**
  ∀x ∀y ∀s [[VoľnýVrch(x, s) ∧ VoľnýVrch(y, s) ∧ x ≠ y] ⇒ VoľnýVrch(y, ĎalšiaSituácia(Zdvihni(x), s))]
- **cieľ**
  ∃s [LežíNa(B, A, s) ∧ LežíNa(C, B, s)]

31

Odvodzovanie v predikátovej logike 1.rádu

- **Substitúcia**
  - SUBST ({*x*/*Cesta61*, *y*/*Západ*}, *Smeruje*(*x*, *y*)) = *Smeruje*(*Cesta61*, *Západ*)
- **Odstránenie univerzálneho kvantifikátora**
  - Pre ľubovoľnú formulu α, premennú v a konštantný výraz g

$$\frac{\forall\, v\ \alpha}{\text{SUBST}(\{v/g\},\ \alpha)}$$

- **Odstránenie existenčného kvantifikátora**
  - pre ľubovoľnú formulu α, premennú v a konštantu k takú, že sa nenachádza inde v báze poznatkov

$$\frac{\exists\, v\ \alpha}{\text{SUBST}(\{v/k\},\ \alpha)}$$

32

Odvodzovanie v predikátovej logike 1.rádu

- **Vovedenie existenčného kvantifikátora**
  - pre ľubovoľnú formulu α, premennú v takú, že sa nenachádza v α a konštantný výraz g

$$\frac{\alpha}{\exists\, v\ \text{SUBST}(\{g/v\},\ \alpha)}$$

- **Zovšeobecnené pravidlo odlúčenia**
  - Nech p_1, p_1', q sú jednoduché formuly také, že existuje substitúcia θ taká, že SUBST(θ, p_i') = SUBST(θ, p_i) pre všetky i

$$\frac{p_1',\ p_2',\ \ldots,\ p_n',\ (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{\text{SUBST}(\theta,\ q)}$$

- **Hornove formuly**
  - formula v báze poznatkov je buď jednoduchá formula alebo implikácia, ktorá má na ľavej strane konjunkciu jednoduchých formúl a jedinú jednoduchú formulu na pravej strane

33

Príklad odvodenia (dôkazu)

Predpokladajme, že platia axiómy
(A1) Každá manželka obdivuje svojho manžela
(A2) Dorota je manželka
treba dokázať, že Dorota obdivuje svojho manžela

Manželka(*x*), *Obdivuje*(*x*, *y*), *Manžel*(*x*), *Dorota*
cieľ: *Obdivuje*(*Dorota*, *Manžel*(*Dorota*))

axiómy:
(1) ∀x (*Manželka*(*x*) ⇒ *Obdivuje*(*x*, *Manžel*(*x*)))
(2) *Manželka*(*Dorota*)
odstránime univerzálny kvantifikátor na (1)
(3) *Manželka*(*Dorota*) ⇒ *Obdivuje*(*Dorota*, *Manžel*(*Dorota*))
pravidlo odlúčenia na (2) a (3)
(4) *Obdivuje*(*Dorota*, *Manžel*(*Dorota*))

Dorota vo zvolenom axiomatickom systéme obdivuje svojho manžela

34

Odvodzovanie v predikátovej logike 1.rádu

- **unifikačný algoritmus**
  UNIFY(*p*, *q*) = θ taká,
  že SUBST(θ, *p*) = SUBST (θ, *q*)
- **unifikátor θ**
  UNIFY (*Vedie*(*x*, *Západ*), *Vedie*(*y*, *z*))
  = {*y*/*x*, *z*/*Západ*}
  = {*y*/*x*, *z*/*Západ*, *u*/*Cesta61*}
  = {*y*/*Cesta61*, *z*/*Západ*, *x*/*Cesta61*}
  = …
- **najvšeobecnejší unifikátor**
  – Najmenej zužuje možnosti náhrady premenných

35

Unifikačný algoritmus

```
function UNIFY(x, y) returns substitúcia, ktorou sa stanú x a y totožné,
    ak existuje VLASTNÝ-UNIFY(x, y, {})

function VLASTNÝ-UNIFY(x, y, θ) returns substitúcia,
    ktorou sa stanú x a y totožné, ak existuje (pre dané θ)
        inputs: x, premenná / konštanta / zložená formula
          y, premenná / konštanta / zložená formula
          θ, dosiaľ vytvorená substitúcia

      if θ = neúspech then return neúspech
      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 ZLOŽENÁ?(x) and ZLOŽENÁ?(y) then
      return VLASTNÝ-UNIFY(ARGUMENTY[x], ARGUMENTY[y],
                           VLASTNÝ-
    UNIFY(OPERÁTOR[x],OPERÁTOR[y],θ))
      else if ZOZNAM?(x) and ZOZNAM?(y) then
      return VLASTNÝ-UNIFY (ZVYŠOK[x], ZVYŠOK [y],
                           VLASTNÝ-UNIFY (PRVÝ[x], PRVÝ [y], θ))
      else return neúspech
```

36

Unifikačný algoritmus

function UNIFY-VAR(*var*, *x*, *θ*) **returns** substitúcia
inputs: *var*, premenná
x, ľubovoľná formula
θ, dosiaľ vytvorená substitúcia

if {*var*/*val*} ∈ *θ* **then return** VLASTNÝ-UNIFY(*val*, *x*, *θ*))
else if {*x*/*val*} ∈*θ* **then return** VLASTNÝ-UNIFY(*var*, *val*, *θ*))
else if *var* sa vyskytuje hocikde v *x* **then return** neúspech
else return { *x*/*var*} ∪ *θ*

Odvodzovanie v predikátovej logike 1.rádu

- **dopredné zreťazenie**
  - vychádza sa z formúl v báze poznatkov a odvodzujú sa nové dôsledky, ktoré môžu poslúžiť na odvodzovanie ešte ďalších dôsledkov
- **spätné zreťazenie**
  - vychádza sa z formuly, ktorá sa má dokázať, hľadajú sa implikácie, ktoré by ju umožnili odvodiť a pre nájdené implikácie sa pokračuje pokusmi dokázať ich predpoklady

Príklad - Odvodzovanie v predikátovej logike 1.rádu

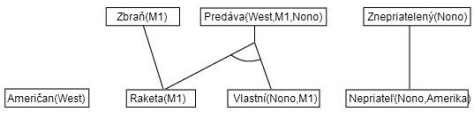
- Zákon v USA hovorí, že ak Američan predáva zbrane znepriateleným národom, je to zločin. Krajina Nono, nepriateľ USA vlastní rakety, ktoré jej predal plukovník West – americký občan.
- Dokážte, že plukovník West je zločinec.

Príklad - Odvodzovanie v predikátovej logike 1.rádu

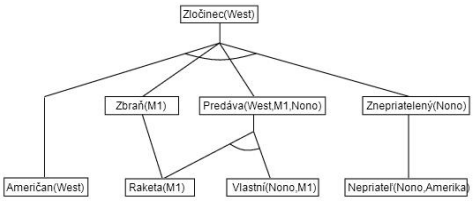
- ... Američan, ktorý predáva zbrane znepriateleným národom je zločinec:
  Američan(x) ∧ Zbraň(y) ∧ Predáva(x,y,z) ∧ Znepriatelený(z) ⇒ Zločinec(x)
- Nono … vlastní rakety, t.j., ∃x Vlastní(Nono,x) ∧ Raketa(x):
  Vlastní(Nono,M_1) a Raketa(M_1)
- … všetky rakety boli predané plukovníkom Westom
  Raketa(x) ∧ Vlastní(Nono,x) ⇒ Predáva(West,x,Nono)
- Rakety sú zbrane:
  Raketa(x) ⇒ Zbraň(x)
- Nepriateľ Ameriky je znepriatelený národ:
  Nepriateľ(x,America) ⇒ Znepriatelený(x)
- West je Američan …
  Američan(West)
- Krajina Nono je nepriateľom Ameriky …
  Nepriateľ(Nono,America)

Dopredné zreťazenie - príklad

Dopredné zreťazenie - príklad



Dopredné zreťazenie - príklad



Odvodzovanie v predikátovej logike 1.rádu

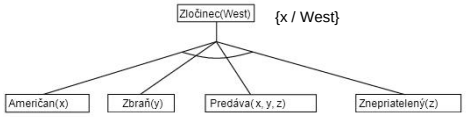
procedure DOPREDNÉ-ZREŤAZENIE (*BP*, *p*)
  if existuje v *BP* formula, ktorá je premenovaním *p* **then return**
  pridaj *p* do *BP*
  for each ($p1 \wedge ... \wedge pn \Rightarrow q$) **in** *BP*
    také, že pre nejaké *i* UNIFY(*pi*, *p*) = θ je úspešné **do**
      NÁJDI-A-ODVOĎ (*BP*, [*p*1, …, *pi*-1, *pi*+1, …, *pn*], *q*, θ)
  end

procedure NÁJDI-A-ODVOĎ (*BP*, *predpoklady*, *dôsledok*, θ)

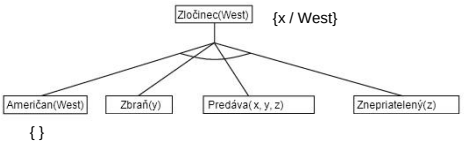
  if *predpoklady* = [ ] **then**
    DOPREDNÉ-ZREŤAZENIE (*BP*, SUBST(θ, *dôsledok*))
  else for each *p'* **in** *BP* také, že
      UNIFY (*p'*, SUBST(θ, PRVÝ(*predpoklady*)))= θ2 **do**
    NÁJDI-A-ODVOĎ (*BP*, ZVYŠOK(*predpoklady*), *dôsledok*, ZLOŽ(θ, θ2))
end

Spätné zreťazenie - príklad

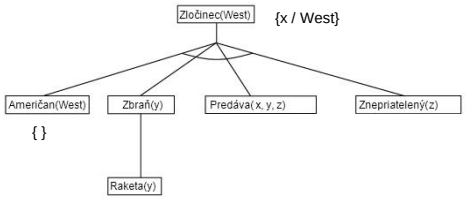
Spätné zreťazenie - príklad



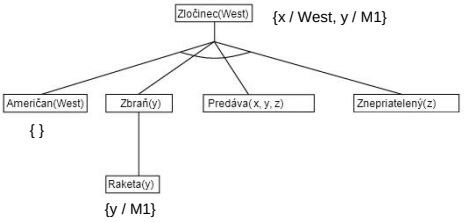
Spätné zreťazenie - príklad



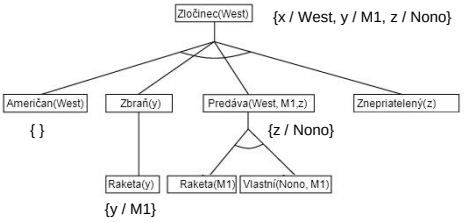
Spätné zreťazenie - príklad



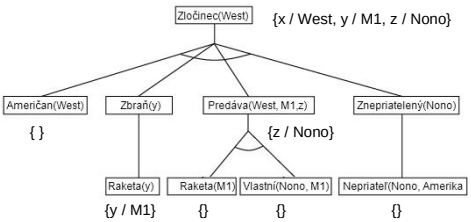
Spätné zreťazenie - príklad



Spätné zreťazenie - príklad



Spätné zreťazenie - príklad



Odvodzovanie v predikátovej logike 1.rádu

function SPÄTNÉ-ZREŤAZENIE(*BP*, *q*) **returns** množina substitúcií
  SPÄTNÉ-ZREŤAZENIE-ZOZNAMU(*BP*, [*q*], {})

function SPÄTNÉ-ZREŤAZENIE-ZOZNAMU(*BP*, *qzoznam*, θ)
  inputs: *BP*, báza poznatkov
    qzoznam, konjunktov tvoriacich dopyt (po aplikácii θ)
    θ, súčasná substitúcia
  static: *odpovede*, množina substitúcií, na začiatku prázdna

  if *qzoznam* je prázdny **then return** θ
  q ← PRVÝ(*qzoznam*)
  for each *qi'* **in** *BP* také, že θi ← UNIFY(*q*, *qi'*) úspešné **do**
    pridaj ZLOŽ(θ, θi) do *odpovede*
  end
  for each formula (*p*1 ∧ ... ∧ *pn* ⇒ *qi'*) **in** *BP*
      taká, že θi ← UNIFY(*q*, *qi'*) úspešné **do**
    odpovede ← SPÄTNÉ-ZREŤAZENIE-ZOZNAMU(*BP*,
      SUBST(θi, [*p*1, ..., *pn*], ZLOŽ(θ, θi)) ∪ *odpovede*
  end
return zjednotenie SPÄTNÉ-ZREŤAZENIE-ZOZNAMU (*BP*,ZVYŠOK(*qzoznam*), θ)
    pre každé θ ∈ *odpovede*

pravidlá pre reprezentáciu znalostí

- AK ... TAK (IF… THEN) pravidlá možno použiť na reprezentovanie znalostí, napr:
  - ak prší tak zmokneš
- pravidlá môžu byť aj odporúčaniami, napr:
  - ak prší tak by si mal mať pršiplášť

pravidlové odvodzovacie systémy

spôsob, akým vyjadrí expert nejakú znalosť, nesie so sebou dôležitú informáciu, napr:
ak osoba má horúčku a cíti bolesť v žalúdku tak môže mať infekciu.
v predikátovej logike sa to dá vyjadriť ako:

∀x (má_horúčku(x) & bolesť_v_žalúdku(x) → má_infekciu(x))

ak sa takáto formula skonvertuje do klauzulárneho tvaru (pozri ďalej), stratíme časť obsahu, lebo rovnakú reprezentáciu majú aj iné ekvivalentné formuly napr:

(i) má_horúčku(x) & ~ má_infekciu(x) → ~ bolesť_v_žalúdku(x)
(ii) ~má_infekciu(x) & bolesť_v_žalúdku(x) → ~ má_horúčku(x)

všimnime si, že:
- (i) a (ii) sú logicky ekvivalentné s pôvodnou vetou
- stratili hlavnú informáciu v nej obsiahnutú.

produkčný systém

- hlavná myšlienka za dopredným/spätným produkčným systémom je:
  - využiť výhodu tvaru implikácie, v akom formuluje expert pravidlá
  - použiť túto informáciu, aby pomohla dosiahnuť cieľ.
- typicky v produkčných systémoch majú formuly tvar:
  - pravidlá
  - fakty
- produkčné pravidlá sú vyjadrenia v tvare implikácie.
  - vyjadrujú špecifické znalosti o probléme.
- fakty sú tvrdenia, ktoré nie sú vyjadrené implikáciou
  - vyjadrujú sa formulou s použitím negácie, súčinu a súčtu.

55

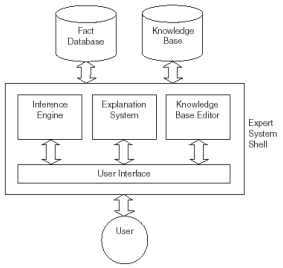
pravidlový produkčný systém

- produkčný systém používa znalosti v tvare pravidiel na to, aby poskytoval diagnózu, radu, odporúčanie na základe vstupných údajov.
- pozostáva z databázy pravidiel (báza znalostí), databázy faktov a odvodzovacieho stroja (mechanizmu), ktorý robí odvodenia nad faktami pomocou pravidiel.

56

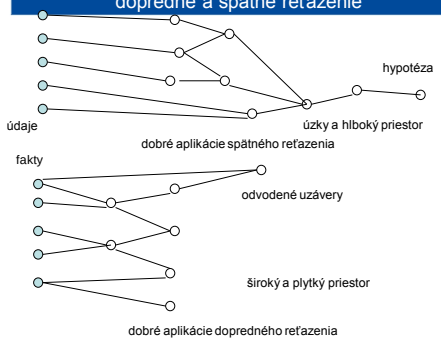
architektúra expertného systému

- báza znalostí (Knowledge base): databáza pravidiel (doménové znalosti).
- vysvetľovací systém (explanation system): vysvetľuje rozhodnutia, ktoré systém robí.
- rozhranie na používateľa (user interface): pomocou neho interaguje používateľ s expertným systémom.
- editor bázy znalostí (knowledge base editor): umožňuje používateľovi editovať znalosti v báze znalostí.



57

dopredné a spätné reťazenie



58

produkčný systém

dopredne reťazený

všetka komunikácia prostredníctvom pracovnej pamäti

1. [zisťovanie podobnosti - matching] nájdi všetky pravidlá, ktorých podmienková časť je splnená (pri súčasnom stave pracovnej pamäti)
2. [skončenie] ak nie je ani jedno pravidlo aplikovateľné, tak skonči
3. [rozriešenie konfliktov – conflict resolution] ak sú viac než jedno pravidlo aplikovateľné, vyber jedno (s najvyššou prioritou)
   - metapravidlá
4. [vykonanie- execution] vykonaj (odpáľ) vybrané pravidlo. vykonanie zmení obsah pracovnej pamäti.
   - pridaj, vymaž údaj, vykonaj v/v, ...
5. opakuj od 1.

59

Odvodzovanie v predikátovej logike 1.rádu

- ? úplnosť
  - odvodzovanie, založené iba na zovšeobecnenom pravidle odlúčenia, je *neúplné*
- pravidlo zovšeobecnenej rezolvencie

Pre literály p_i a q_j, pričom platí UNIFY(p_j, $\neg q_k$) = θ:

$$\frac{p_1 \vee \ldots \vee p_j \vee \ldots \vee p_m \qquad q_1 \vee \ldots \vee q_k \vee \ldots \vee q_n}{\text{SUBST}(\theta, (p_1 \vee \ldots \vee p_{j-1} \vee p_{j+1} \vee \ldots \vee p_m \vee q_1 \vee \ldots \vee q_{k-1} \vee q_{k+1} \vee \ldots q_n))}$$

Pre atómy p_i, q_i, r_i, s_i, pričom platí UNIFY(p_j, q_k) = θ:

$$\frac{p_1 \wedge \ldots \wedge p_j \wedge \ldots \wedge p_{n_1} \Rightarrow r_1 \vee \ldots \vee r_{n_2} \qquad s_1 \wedge \ldots \wedge s_{n_3} \Rightarrow q_1 \vee \ldots \vee q_k \vee \ldots \vee q_{n_4}}{\text{SUBST}(\theta, (p_1 \wedge \ldots \wedge p_{j-1} \wedge p_{j+1} \wedge \ldots \wedge p_{n_1} \wedge s_1 \wedge \ldots \wedge s_{n_3} \Rightarrow r_1 \vee \ldots \vee r_{n_2} \vee q_1 \vee \ldots \vee q_{k-1} \vee q_{k+1} \vee \ldots \vee q_{n_4}))}$$

60

Odvodzovanie v predikátovej logike 1.rádu

- **rezolvenčný dôkaz**
  - sporom
    ($BP \wedge \neg P \Rightarrow Lož$) $\Leftrightarrow$ ($BP \Rightarrow P$)
- **klauzulárny tvar**
  - klauzula je disjunkcia niekoľkých literálov (nemusí byť ani jeden)
  - literál je atóm alebo negácia atómu
  - všetky premenné v klauzule sú implicitne univerzálne kvantifikované
  - medzi všetkými klauzulami je implicitná konjunkcia

Prepis formuly do klauzulárneho tvaru

1. odstránenie ekvivalencie

$$\frac{F \Leftrightarrow G}{(\neg F \vee G) \wedge (\neg G \vee F)}$$

2. odstránenie implikácie

$$\frac{F \Rightarrow G}{\neg F \vee G}$$

3. zmenšenie rozsahu negácie

$$\frac{\neg(F \vee G)}{\neg F \wedge \neg G} \quad \frac{\neg \forall x\, F}{\exists x\, \neg F} \quad \frac{\neg\neg F}{F}$$

$$\frac{\neg(F \wedge G)}{\neg F \vee \neg G} \quad \frac{\neg \exists x\, F}{\forall x\, \neg F}$$

Prepis formuly do klauzulárneho tvaru

4. premenovanie premenných
5. odstránenie existenčných kvatifikátorov

   Skolemov výraz (konštanta / funkcia, presnejšie funktor)

   $\exists u \forall v \forall w \exists x \forall y \exists z\, F(u,v,w,x,y,z)$,po náhrade

   $\forall v \forall w \forall y\, F(b, v, w, f(v, w), y, g(v, w, y))$
6. presun kvantifikátorov doľava (pozn.: $H\{x\}$ - formula H neobsahuje prem. X)

$$\frac{\forall x\, F(x) \vee \forall y\, G(y)}{\forall x\, \forall y\, (F(x) \vee G(y))} \quad \frac{\forall x\, F(x) \vee H\{x\}}{\forall x\, (F(x) \vee H\{x\})}$$

$$\frac{\forall x\, F(x) \wedge \forall y\, G(y)}{\forall x\, \forall y\, (F(x) \wedge G(y))} \quad \frac{\forall x\, F(x) \wedge H\{x\}}{\forall x\, (F(x) \wedge H\{x\})}$$

Prepis formuly do klauzulárneho tvaru

7. odstránenie prefixu
8. prepis do konjuktívneho tvaru

$$\frac{F \vee (G \wedge H)}{(F \vee G) \wedge (F \vee H)} \quad \frac{F \wedge (G \vee H)}{(F \wedge G) \vee (F \wedge H)}$$

9. zápis konjukcie klauzúl ako množiny
10. normalizácia premenných v klauzulách

$$\frac{\forall x\, (F(x) \wedge G(x))}{\forall x\, \forall y\, (F(x) \wedge G(y))}$$

Príklad - Prepis formuly do klauzulárneho tvaru

Každého, kto má rád všetky zvieratá, má niekto rád.

$\forall x\, [\forall y\, Zviera(y) \Rightarrow Má\_rád(x,y)] \Rightarrow [\exists y\, Má\_rád(y,x)]$

1. Odstránenie implikácie

$\forall x\, [\neg \forall y\, \neg Zviera(y) \vee Má\_rád(x,y)] \vee [\exists y\, Má\_rád(y,x)]$

2. Zmenšenie rozsahu negácie:

$\forall x\, [\exists y\, \neg(\neg Zviera(y) \vee Má\_rád(x,y))] \vee [\exists y\, Má\_rád(y,x)]$

$\forall x\, [\exists y\, \neg\neg Zviera(y) \wedge \neg Má\_rád(x,y)] \vee [\exists y\, Má\_rád(y,x)]$

$\forall x\, [\exists y\, Zviera(y) \wedge \neg Má\_rád(x,y)] \vee [\exists y\, Má\_rád(y,x)]$

Príklad - Prepis formuly do klauzulárneho tvaru

3. Premenovanie premenných

$\forall x\, [\exists y\, Zviera(y) \wedge \neg Má\_rád(x,y)] \vee [\exists z\, Má\_rád(z,x)]$

4. Odstránenie existenčných kvantifikátorov

$\forall x\, [Zviera(F(x)) \wedge \neg Má\_rád(x,F(x))] \vee Má\_rád(G(x),x)$

5. Odstránenie prefixu:

$[Zviera(F(x)) \wedge \neg Má\_rád(x,F(x))] \vee Má\_rád(G(x),x)$

6. Prepis do konjuktívneho tvaru :

$[Zviera(F(x)) \vee Má\_rád(G(x),x)]$
$\wedge\, [\neg Má\_rád(x,F(x)) \vee Má\_rád(G(x),x)]$

Procedúra odvodzovania s pravidlom rezolvencie

P - dokazovaná formula

- prepis $F_1, F_2, \ldots, F_n$ a $\neg P$ do klauzulárneho tvaru vznikne vstupná množina *U*
- opakujúce sa rezolvovanie
  - *výber* dvoch klauzúl z *U*, ktoré sa dajú rezolvovať
  - ak treba, *štandardizácia* týchto klauzúl
  - *odvodenie* rezolventy
  - *pridanie* rezolventy do *U*

  pokiaľ (t.j. skonči ak):
  - rezolventa je prázdna klauzula *(P je teoréma)*
  - neexistujú dve klauzuly v U, ktoré by sa dali rezolvovať, alebo by sa dala odvodiť nová rezolventa *(P nie je teoréma)*
  - vyčerpali sa vopred určené výpočtové zdroje

Príklad rezolvenčného dôkazu

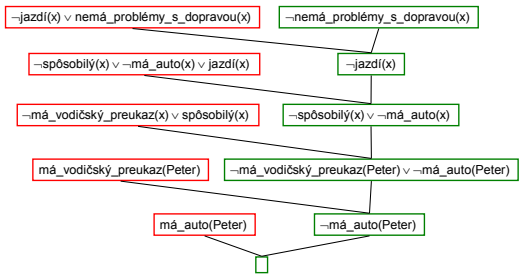
Každý, kto je spôsobilý na jazdu a má auto, môže jazdiť. Na jazdu je spôsobilý ten, kto má vodičský preukaz. Peter má vodičský preukaz a požičal si auto. Tí, čo jazdia, nemajú problémy s dopravou. Dá sa nájsť niekto, kto nemá problémy s dopravou?

Formuly jazyka predikátovej logiky prvého rádu:
$\forall x$ spôsobilý(x) $\wedge$ má_auto(x) $\Rightarrow$ jazdí(x)
$\forall x$ má_vodičský_preukaz(x) $\Rightarrow$ spôsobilý(x)
$\exists x$ = Peter má_vodičský_preukaz(x) $\wedge$ má_auto(x)
$\forall x$ jazdí(x) $\Rightarrow$ nemá_problémy_s_dopravou(x)
$\exists x$ nemá_problémy_s_dopravou(x)

Úprava do klauzulárneho tvaru (pre jednoduchosť nasledujúceho obrázku tu ešte pred posledným premenovaním premenných) :

¬spôsobilý(x) ∨ ¬má_auto(x) ∨ jazdí(x)	(K1)	
¬má_vodičský_preukaz(x) ∨ spôsobilý(x)	(K2)	(napr. x na x1)
má_vodičský_preukaz(Peter)	(K3)	
má_auto(Peter)	(K4)	
¬jazdí(x) ∨ nemá_problémy_s_dopravou(x)	(K5)	(napr. x na x2)
¬nemá_problémy_s_dopravou(x)	(K6)	(napr. x na x3)

Príklad rezolvenčného dôkazu



Procedúra odvodzovania s pravidlom rezolvencie

- zefektívnenie:
  rezolvenčné stratégie
  - usporadúvajúce

    hĺbkové saturovanie, preferencia najmenšieho počtu literálov, jednotková preferencia
  - odsekávajúce

    vylúčenie jalových klauzúl, vylúčenie tautológií, zahrnutie
  - obmedzujúce

    podporná množina, vstupná rezolvencia, lineárna rezolvencia
- *úplnosť* stratégie definícia – otázka na jednotlivé stratégie

Rezolvenčné stratégie

- **Usporadúvajúce stratégie**
  - **Stratégia hĺbkového saturovania**

    Predpisuje najprv odvodzovať všetky možné rezolventy v hĺbke n a až potom v hĺbke (n+1).
  - **Stratégia preferencie najmenšieho počtu literálov**

    Uprednostňuje spomedzi dvojíc rezolvovateľných klauzúl tú, ktorej súčet dĺžok je najmenší.
  - **Stratégia jednotkovej preferencie**

    Stratégia uprednostňuje použitie rezolvenčného pravidla tak, že jedna z klauzúl je len jeden literál (jednotková klauzula).

Rezolvenčné stratégie

- **Odsekávajúce stratégie**
  - **Stratégia vylúčenia jalových klauzúl**

    Literál L je jalový, ak sa vyskytuje v množine klauzúl, ale komplementárny unifikovateľný literál ¬L sa v nej nevyskytuje. Stratégia predpisuje vylúčiť všetky jalové literály zo vstupnej množiny.
  - **Stratégia vylúčenia tautológií**

    Predpisuje vylúčiť zo vstupnej množiny klauzuly, ktoré sú tautológie, napr. F(x) ∨ G(x,y) ∨ ¬F(x).
  - **Stratégia zahrnutia**

    Klauzula J je zahrnutá v klauzule I, ak existuje taká substitúcia θ, že všetky literály v klauzule SUBST(θ, I) sa vyskytujú v klauzule J. Napr. F(A) je špecifickejšia ako F(x) a preto je v nej zahrnutá.

    Na začiatku sa vylúčia všetky zahrnuté klauzuly zo vstupnej množiny, ak sa neskôr odvodí rezolventa, zahrnutá v nejakej klauzule, vylúči sa tiež.

Rezolvenčné stratégie

- **Obmedzujúce stratégie**
  - **Stratégia podpornej množiny**

    Najprv sa identifikuje podporná množina (klauzuly, ktoré vznikli z negovanej dokazovanej formuly). Každé použitie rezolvenčného pravidla vezme jednu klauzulu z podpornej množiny a rezolventu potom zase pridá do podpornej množiny.

  - **Stratégia vstupnej rezolvencie**

    Predpisuje vziať vždy ako jednu z klauzúl na rezolvovanie klauzulu, ktorá vznikla zo vstupných formúl.

  - **Stratégia lineárnej rezolvencie**

    Predpisuje rezolvovať dve klauzuly nielen v prípade, ak je jedna zo vstupnej množiny klauzúl, ale aj v prípade, ak je jedna predchodkyňou druhej v strome dôkazu.