IV130 Přínosy a rizika inteligentních systémů

Logičtí aktéři, vyvozování

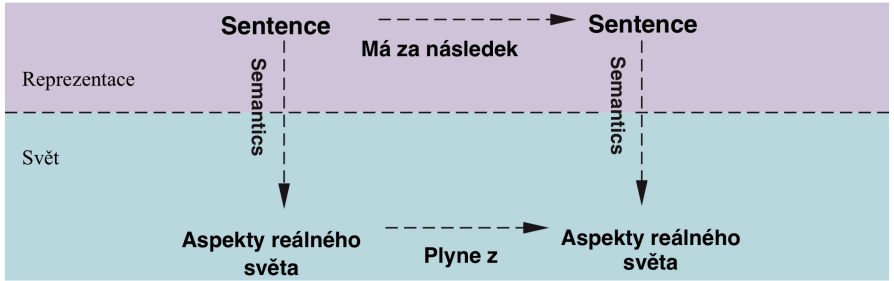
31. března 2023

Logičtí (znalostní) aktéři

- Aktéři si vytvářejí **reprezentaci světa**
- Z reprezentace světa vyvozují nové informace
- Tyto informace dále využívají ke svým akcím
- Logičtí (znalostní) aktéři dávají dohromady obecné znalosti s konkrétními znalostmi z pozorování světa (smyslových vstupů)
- **Odvozování** může odhalovat skryté vlastnosti světa (prostředí)
- **Reprezentace znalostí** jako množina vět (sentencí) v *jazyce reprezentace znalostí*
- Sentence bez odvození se nazývá **axiómem**
- Operace TELL přidává tvrzení do **znalostní báze**
- Operace ASK ze znalostní báze vybírá odpověď, která **vyplývá z obsahu** znalostí báze spolu s tím, co do ní přidaly operace TELL
- **Znalostní bázi** si udržuje aktér, na počátku může obsahovat nějaké obecné znalosti

Logika znalostního aktéra

- Vhodná **syntax** pro vyjadřování sentencí
- **Sémantika** definující význam sentencí, zejména pak jejích pravdivost vzhledem k možnému světu (stavu světa, **modelu**)
- Pokud je sentence pravdivá v modelu, pak říkáme, že model splňuje tuto sentenci (nebo je to model té sentence)
- Ze sentence α plyne sentence β, pokud každý model splňující α splňuje také β
- Vyplývání lze využít k odvozování důsledků; Vyvozování je formální cesta, jak tyto důsledky hledat
- Korektní vyvozování odvozuje pouze takové sentence, které vyplývají z výchozích
- Obsahuje-li znalostní báze pravdivé sentence o reálném světě, dává korektní vyvozování pouze sentence, které jsou rovněž pravdivé:



Sentence jsou fyzické konfigurace aktéra, vyvozování je proces konstrukce nových fyzických konfigurací ze starých. Logické vyvozování musí zajistit, že nové konfigurace reprezentují aspekty světa, které plynou z toho, co reprezentovala stará konfigurace.

Obrázky a schémata z Russell-Norwig: AI A Modern Approach, 4th ed., 2021

Výroková logika

- Z hlediska logických operací jsou vyhodnocení sentencí v tomto jazyce dány pravdivostními tabulkami:

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

Pravdivostní tabulky pro pět logických spojek (operací).

- Odvozování generováním důsledků enumerace pravdivostní tabulky, tzv. ověřování modelů (model checking)
- Odvozovací pravidla, zejména pak rezoluční metoda

Dokazování ve výrokové logice

- **Dokazování teorémů** vyvozování přímo na úrovni sentencí
- Logické ekvivalence sentencí

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{komutativita } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{komutativita } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{asociativita } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{asociativita } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{eliminace dvojí negace} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contrapozice} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{eliminace implikace} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{eliminace ekvivalence} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{De Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{De Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivita } \wedge \text{ přes } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivita } \vee \text{ přes } \wedge
\end{aligned}
$$

Standardní logické ekvivalence. Symboly α, β , a γ označují libovolné sentence

- **Odvozovací pravidla:**
- Modus ponens $\alpha \Rightarrow \beta,\ \alpha \ \models \beta$
- Eliminace konjunkce: $\alpha \wedge \beta \models \alpha$
- Pravidla na bázi logických ekvivalencí

Odvozování ve výrokové logice

- Věta (sentence) je **splnitelná**, pokud je pravdivá v nějakém modelu. *Příklad: A ∨ B, C*
- Věta (sentence) je **nesplnitelná**, pokud není pravdivá v žádném modelu. *Příklad: A ∧ ¬A*
- Logický důsledek lze převést na testování splnitelnosti, protože KB ⊨ α právě tehdy, když je (KB ∧ ¬α) nesplnitelná. (Důkaz se vede pomocí zamítnutí nebo sporem: Ověřování, zda α je logickým důsledkem KB lze tedy převést na problém testování splnitelnosti (KB ∧ ¬α).)
- Výrokové formule v konjunktivní normální formě (CNF):

➢ *literál* je výroková proměnná nebo její negace

➢ *Klauzule* je disjunkce literálů

❖ Formule v CNF je konjunkce klauzulí

- Každou formuli lze převést do CNF.

Rezoluční odvozování ve výrokové logice

- Rezoluční algoritmus dokazuje nesplnitelnost formule (KB ∧ ¬α) převedené do CNF opakovanou aplikací **rezolučního pravidla**,
- To ze dvou klauzulí obsahujících *komplementární literály* (P a ¬P)
- vytvoří novou klauzuli:

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

- ℓ_i a m_j jsou komplementární literály.

- Příklad:

$$\frac{P \vee \neg Q \vee R, \qquad \neg P \vee Q}{\neg Q \vee Q \vee R}$$

- Algoritmus končí pokud
- nelze přidat žádnou další klauzuli, a pak ¬ KB ⊨ α), nebo
- vznikla prázdná klauzule, a pak KB ⊨ α
- Jde o úplný a korektní algoritmus
- Úplné je i používání jedné z klauzulí z *oporné množiny* T ⊆ S takové, že S−T je bezesporná (např. pochází z bezesporných axiómů).

Rezoluční odvozování ve výrokové logice

Rezoluční důkaz prázdné klauzule $\square$ *z množiny klauzulí*

$$S = \{\{P, R\}, \{Q, \neg R\}, \{\neg Q\}, \{\neg P\}\}$$

je strom z obr. 1.

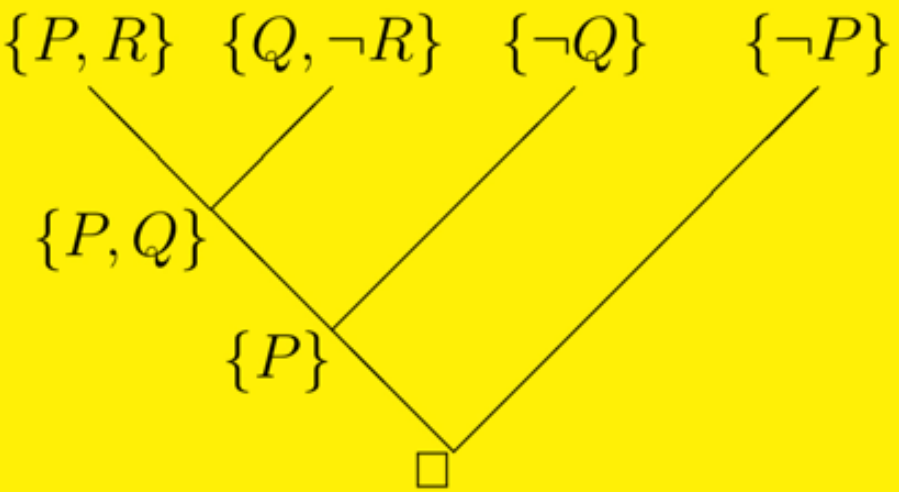


Figure 1: Rezoluční důkaz.

Hornovy klauzule

- Klauzule odpovídající implikacím $P_1 \wedge \ldots \wedge P_n \rightarrow Q$, kde žádný z $P_1, \ldots, P_n, Q$ není negativní
- V množinovém zápise mají klauzule tvar $\{ Q, \neg P_1, \ldots, \neg P_n \}$,
- Hornova klauzule s právě jedním pozitivním literálem se nazývá *programová klauzule*. Namísto $\{ Q, \neg P_1, \ldots, \neg P_n \}$ bývá v kontextu logického programování zvykem ji zapisovat ve tvaru: $Q \leftarrow P_1, \ldots, P_n$
- Taková klauzule se nazývá *pravidlem* pro $n > 0$; nebo *faktem* pro $n = 0$ (její zápis je pak Q zvýrazněně $Q \leftarrow$)
- Hornova klauzule bez pozitivních literálů, tj. $\{ \neg P_1, \ldots, \neg P_n \}$ či $\leftarrow P_1, \ldots, P_n$, se nazývá *cílem* (*cílovou klauzulí*).
- Cíle a fakta spolu interagují při vytváření důkazů nesplnitelnosti množin Hornových klauzulí: bez obou z nich je lib. množina Hornových klauzulí splnitelná.

- Pro Hornovy klauzule jsou úplnými metodami *jednotková rezoluce* (aspoň jedna z použitých klauzuilí musí být jednoprvková) i *lineární rezoluce* (alespoň jedna z klauzulí je ze vstupní množiny klauzulí, tj. KB bez dalších odvozování).

Hornovy klauzule

- Gramatika pro Hornovy klauzule:

$$
\begin{aligned}
\mathit{CNFSentence} &\rightarrow \mathit{Klauzule}_1 \wedge \cdots \wedge \mathit{Klauzule}_n \\
\mathit{Klauzule} &\rightarrow \mathit{Literal}_1 \vee \cdots \vee \mathit{Literal}_m \\
\mathit{Fakt} &\rightarrow \mathit{Symbol} \\
\mathit{Literal} &\rightarrow \mathit{Symbol} \mid \neg \mathit{Symbol} \\
\mathit{Symbol} &\rightarrow P \mid Q \mid R \mid \ldots \\
\mathit{HornovaKlauzule} &\rightarrow \mathit{DefiniteClauseForm} \mid \mathit{GoalClauseForm} \\
\mathit{DefiniteClauseForm} &\rightarrow \mathit{Fakt} \mid (\mathit{Symbol}_1 \wedge \cdots \wedge \mathit{Symbol}_l) \Rightarrow \mathit{Symbol} \\
\mathit{GoalClauseForm} &\rightarrow (\mathit{Symbol}_1 \wedge \cdots \wedge \mathit{Symbol}_l) \Rightarrow \mathit{False}
\end{aligned}
$$

Gramatika pro konjunktivní normální formu, Hornovy klauzule and definite klauzule.
CNF klauzule jako třeba $\neg A \vee \neg B \vee C$ lze v definite klauzulární formě zapsat jako as $A \wedge B \Rightarrow C$.

Hornovy klauzule - dopředné řetězení

- **Dopředné řetězení**:
- Z dostupných faktů odvozujeme pomocí Hornových klauzulí všechny možné závěry dokud vznikají nové fakty resp. dokud nedokážeme platnost dotazu.
- U každé klauzule si pamatujeme počet předpokladů, který se zmenší, když je předpoklad dokázán.
- Klauzule s nula (zbývajícími) předpoklady slouží k dokázání hlavy klauzule.
- Postup řízený daty v KB.

- Jde o korektní a úplný algoritmus pro Hornovy klauzule
- Lineární časová složitost v rozměru báze znalostí KB

- **Zpětné řetězení**:
- Pro daný dotaz hledáme rozdělení pomocí Hornovy klauzule na poddotazy, které se řeší stejným způsobem.
- Postup řízený dotazem

Logiky za výrokovou logikou

- Zachycení vlastností objektů, relací, funkcí a vlastností
- Ontologické zakotvení jazyka logiky – co se předpokládá o stavebních kamenech vyjadřování a vyvozování

Jazyky	Ontologické zakotvení (co ve světě existuje)	Epistemiologické zakotvení (co aktér věří o faktech)
Výroková logika	fakta	true/false/unknown
Logika prvního řádu	fakta, objekty, relace	true/false/unknown
Temporální logika	fakta, objekty, relace, časy	true/false/unknown
Pravděpodobnostní logika	fakta	stupeň víry z intervalu [0,1]
Fuzzy logika	Fakta se stupněm pravdy z [0,1]	známý interval hodnot

Predikátová logika 1.řádu

- **Rozšířený jazyk**:

$$
\begin{aligned}
\textit{Sentence} &\rightarrow \textit{AtomickáSentence} \mid \textit{KomplexníSentence} \\
\textit{AtomickáSentence} &\rightarrow \textit{Predicate} \mid \textit{Predicate}(\textit{Term}, \ldots) \mid \textit{Term} = \textit{Term} \\
\textit{KomplexníSentence} &\rightarrow (\textit{Sentence}) \\
&\mid \neg \textit{Sentence} \\
&\mid \textit{Sentence} \wedge \textit{Sentence} \\
&\mid \textit{Sentence} \vee \textit{Sentence} \\
&\mid \textit{Sentence} \Rightarrow \textit{Sentence} \\
&\mid \textit{Sentence} \Leftrightarrow \textit{Sentence} \\
&\mid \textit{Kvantifikátor Proměnná},\ldots \\
&\quad \textit{Sentence} \\
\textit{Term} &\rightarrow \textit{Funktor}(\textit{Term},\ldots) \\
&\mid \textit{Konstanta} \\
&\mid \textit{Proměnná}
\end{aligned}
$$

$$
\begin{aligned}
\textit{Kvantifikátor} &\rightarrow \forall \mid \exists \\
\textit{Konstanta} &\rightarrow A \mid X_1 \mid \textit{John} \mid \cdots \\
\textit{Proměnná} &\rightarrow a \mid x \mid s \mid \cdots \\
\textit{Predikát} &\rightarrow \textit{True} \mid \textit{False} \mid \textit{After} \mid \textit{Loves} \mid \textit{Raining} \mid \cdots \\
\textit{Funktor} &\rightarrow \textit{Mother} \mid \textit{LeftLeg} \mid \cdots
\end{aligned}
$$

PRECEDENCE OPERÁTORŮ : $\neg, =, \wedge, \vee, \Rightarrow, \Leftrightarrow$

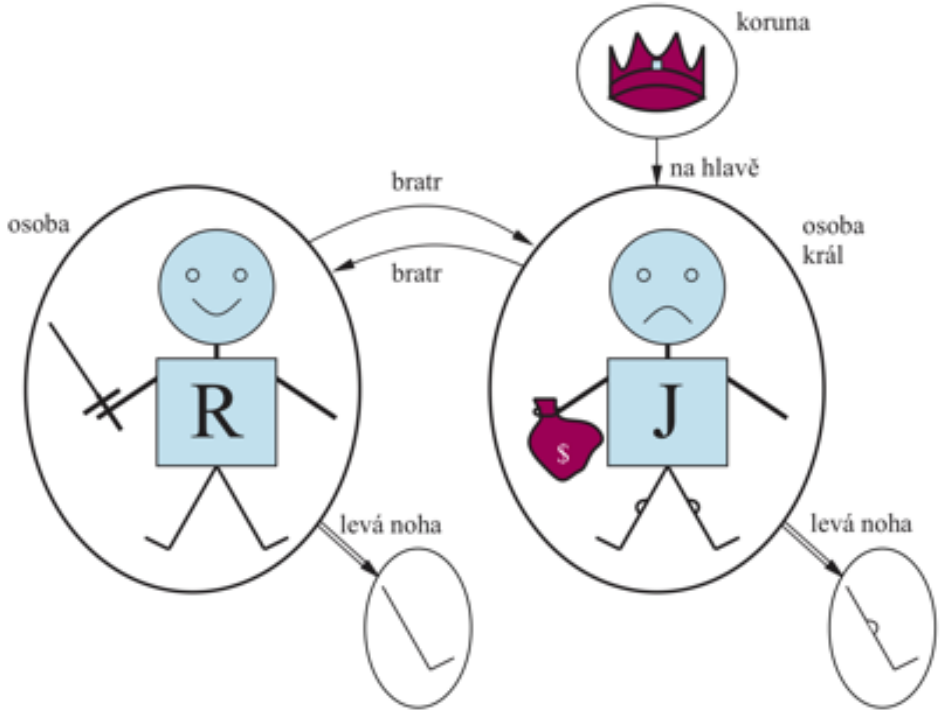
Gramatika pro logiku prvního řádu s rovností. Precence operátorů je specifikována od nejvyšší po nejnižší. Precedence of quantifikátorů je taková že se kvantifikátor vztahuje na vše vpravo od něj.

- Kvantifikátory
- Funkční syboly
- Predikáty

- Operace TELL dává do znalostní báze:
  - Axiómy (fakta)
  - Definice (tvrzení ve traru ekvivalencí)
  - Teorémy (lze odvodit z jiných, ale může zrychlovat vyvozování)

- Operace ASK:
  - Dotazy na obsah databáze
  - Odvozování dalších vlastností pro objekty dotazu
  - Instanciace objektů, pro něž platí nějaká vlastnost

Predikátová logika 1.řádu



Model vyjádřený prostředky predikátové logiky 1. řádu obsahující 5 objektů, 2 binární relace (bratr a na hlavě), 3 unární relace (osoba, král a koruna) a 1 unární funkci (levá noha).

- konstanty (jména objektů):
  - Richard, John, Koruna
  - Funkční symbol: LeváNoha
- Termy (jiný způsob pojmenování objektu):
  - Levánoha(John)
- Predikátové symboly:
  - Bratr, NaHlavě, Osoba, Král, Koruna
- Atomická tvrzení (popis relace mezi objekty):
  - Bratr(Richard,John)
- Složená tvrzení:
  - Král(Richard) ∨ Král(John)
  - ¬Král(Richard) ⇒ Král(John)
- Tvrzení o výběru objektů (kvantifikátory):
  - ∀x Král(x) ⇒ Osoba(x)
  - ∃x Koruna(x) ∧ NaHlavě(x,John)
  - ∀x,y Bratr(x,y) ⇒ Bratr(y,x)
  - ∃x,y Bratr(x,Richard) ∧ Bratr(y,Richard)
  - ∃x,y Bratr(x,Richard) ∧ Bratr(y,Richard) ∧ ¬(x=y)

Obrázky a schémata z Russell-Norwig: AI A Modern Approach, 4th ed., 2021

Predikátová logika 1.řádu

- Skolemizace pro práci s kvantifikátory (zachovává splnitelnost)
- Unifikace pro hledání instancí umožňujících rezoluční krok (instancí vedoucích k vygenerování kontradikce)

- Herbrandovy intepretace definující „symbolické“ interpretace bez ohledu na konkreétní problémovou doménu
- Specifikace vyjádřené logikou prvního řádu lze „uzemnit“ resp. „propozicionalizovat“ vygenerováním všech možných základních instancí formulí zahrnujících proměnné (resp. kvantifikátory) – znamená to ovšem obrovskou extenzi prohledávaného stavového prostoru: substitucí je typicky nekonečně mnoho

- Výroková i predikátová logika mají zásadní slabiny ve zvládání nejasně vymezených propozic

Predikátová logika 1.řádu

- Skolemizace pro práci s kvantifikátory (zachovává splnitelnost)

Skolemizace

V logice 1. řádu nemůžeme díky kvantifikátorům provést okamžitý převod formule do klauzulárního tvaru, o který se rezoluční metoda opírala v případě výrokového počtu. O katastrofu se však nejedná: obecných kvantifikátorů se zbavíme, budeme-li předpokládat, že všechny volné proměnné ve formulích jsou prostě obecně kvantifikovány, tj. např.

$$R(x, y, z)$$

znamená

$$(\forall x)(\forall y)(\forall z)R(x, y, z).$$

Existenčních kvantifikátorů se zbavíme tzv. skolemizací: Máme-li co do činění s formulí tvaru

$$(\forall x_1) \ldots (\forall x_n)(\exists y)P(x_1, \ldots, x_n, y),$$

podmínku na existenci y můžeme chápat prostě jako podmínku na existenci zobrazení f (tzv. výběrové funkce), která pro konkrétní hodnoty $x_1,\ldots,x_n$ konstruuje požadované $y = f(x_1, \ldots, x_n)$. Místo uvažované formule s existenčním kvantifikátorem tak můžeme uvažovat skolemizovanou formuli

$$(\forall x_1) \ldots (\forall x_n)P(x_1, \ldots, x_n, f(x_1, \ldots, x_n)).$$

Provedeme-li tuto úpravu s každým existenčním kvantifikátorem zevně dovnitř (vždy přitom volíme nový funkční symbol pro skolemovskou funkci), zbavíme se všech existenčních kvantifikátorů.

Predikátová logika 1.řádu

- Unifikace pro práci s funkcemi a proměnnými

Substitucí *rozumíme lib. funkci* θ *zobrazující výrazy do výrazů, pro které platí podmínky z obr. 2.*

Pro substituce používáme zpravidla postfixové notace, tj. píšeme $E\theta$ $\theta(E)$. *Výraz* $E\theta$ *se nazývá* instancí *výrazu* E *použitím substituce* θ.

Kompozice substitucí θ, ϑ *je kompozice funkcí* $\vartheta \circ \theta$; *místo* $\vartheta \circ \theta(E)$ *opět píšeme* $E\theta\vartheta$. *Operace kompozice je tedy zřejmě asociativní,* $E(\theta\vartheta)\lambda = E\theta(\vartheta\lambda)$, *a existuje jednotkový prvek* ϵ *(identická substituce), pro který* $E\epsilon = E$.

$\theta(E) = E$, pro lib. konstantu E;
$\theta(f(E_1, \ldots, E_n)) = f(\theta(E_1), \ldots, \theta(E_n))$,
pro lib. funkční symbol f;
$\theta(P(E_1, \ldots, E_n)) = P(\theta(E_1), \ldots, \theta(E_n))$,
pro lib. predikátový symbol P.

2: Substituce θ je definována jako homomorfismus výrazů.

Substituce je lib. homomorfismus výrazů (termů či formulí), který musí zachovávat vše kromě proměnných – ty může nahradit čímkoli (libovolným termem). Speciálně substituce, která proměnným přiřazuje pouze proměnné a je na množině proměnných jedno-jednoznačným zobrazením, se nazývá *přejmenování proměnných.*

Substituce jsou plně popsány svou hodnotou na proměnných. Většinou se zajímáme jen o tzv. *konečné substituce*, které skoro všechny proměnné(= všechny kromě konečného počtu) zobrazují na sebe samy. V takovém případě zpravidla ztotožňujeme tento popis substituce ve tvaru seznamu

$$[x_1/\xi_1, \ldots, x_n/\xi_n],$$

zadávající zobrazení proměnné x_i na ξ_i, $1 \leq i \leq n$, se samotnou substitucí. Např. zápis $P(x)[x/f(a)]$, vyjadřuje $P(f(a))$.

Predikátová logika 1.řádu

Rozdíl $\Delta(A,B)$ *nazýváme* zmenšitelným, *platí-li*

- $\Delta(A,B) \neq \emptyset$;
- *pro každý prvek* $\{U,V\} \in \Delta(A,B)$ *je alespoň jeden z výrazů* U, V *objektová proměnná a tato proměnná není obsažena v druhém z výrazů. Alespoň jedna z usp. dvojic* $[U,V]$ *či* $[V,U]$ *je tedy substitucí (podle toho, které z* U, V *je prostou proměnnou) – tato substituce se nazývá* redukcí *rozdílu* $\Delta(A,B)$.

```
unify(A,B):
     σ:=ϵ;     (*inicializace σ prázdnou substitucí*)
     while Δ(Aσ,Bσ) je zmenšitelný rozdíl do σ:=σμ,
                    kde μ je lib. redukce rozdílu
Δ(Aσ,Bσ);
     if Δ(Aσ,Bσ)      =      ∅ then return σ else return
''fail''.
```

Figure 4: Unifikační algoritmus.

$$\frac{\mathcal{P} \cup \{A_1, \ldots, A_m\} \qquad \{\neg B_1, \ldots, \neg B_n\} \cup \mathcal{Q}}{\mathcal{P}\rho\sigma \cup \mathcal{Q}\sigma},$$

kde ρ je takové přejmenování proměnných, že obě množiny nemají žádné společné proměnné, a σ je nejobecnější unifikátor množiny $\{A_1\rho, \ldots, A_m\rho, B_1, \ldots, B_n\}$.

Figure 5: Rezoluční princip pro klauzule logiky 1. řádu.

Predikátová logika 1.řádu

Příklad *Z předpokladu reflexivity*

$$(\forall x)P(x,x)$$

a vlastnosti

$$(\forall x)(\forall y)(\forall z)P(x,y) \wedge P(y,z) \rightarrow P(z,x)$$

predikátu P dokážeme, že tento predikát je rovněž symetrický, tj.

$$(\forall x)(\forall y)P(x,y) \rightarrow P(y,x).$$

Klauzule, se kterými budeme pracovat, jsou

$$\mathcal{C}_1 = \{P(x,x)\},$$
$$\mathcal{C}_2 = \{\neg P(x,y), \neg P(y,z), P(z,x)\}$$

a negované vlastnosti symetrie, tj.

$$(\exists x)(\exists y)\neg(P(x,y) \rightarrow P(y,x)),$$

čili po skolemizaci

$$\neg(P(f,g) \rightarrow P(g,f))$$

(kde f, g jsou nulární funkční konstanty – individuové konstanty), a tomu odpovídá dvojice klauzulí

$$\mathcal{C}_3 = \{P(f,g)\},$$
$$\mathcal{C}_4 = \{\neg P(g,f)\}.$$

Predikátová logika 1.řádu

Příklad *Z předpokladu reflexivity*

$$(\forall x)P(x,x)$$

a vlastnosti

$$(\forall x)(\forall y)(\forall z)P(x,y) \wedge P(y,z) \rightarrow P(z,x)$$

predikátu P dokážeme, že tento predikát je rovněž symetrický, tj.

$$(\forall x)(\forall y)P(x,y) \rightarrow P(y,x).$$

Klauzule, se kterými budeme pracovat, jsou

$$\mathcal{C}_1 = \{P(x,x)\},$$
$$\mathcal{C}_2 = \{\neg P(x,y), \neg P(y,z), P(z,x)\}$$

a negované vlastnosti symetrie, tj.

$$(\exists x)(\exists y)\neg(P(x,y) \rightarrow P(y,x)),$$

čili po skolemizaci

$$\neg(P(f,g) \rightarrow P(g,f))$$

(kde f, g jsou nulární funkční konstanty – individuové konstanty), a tomu odpovídá dvojice klauzulí

$$\mathcal{C}_3 = \{P(f,g)\},$$
$$\mathcal{C}_4 = \{\neg P(g,f)\}.$$

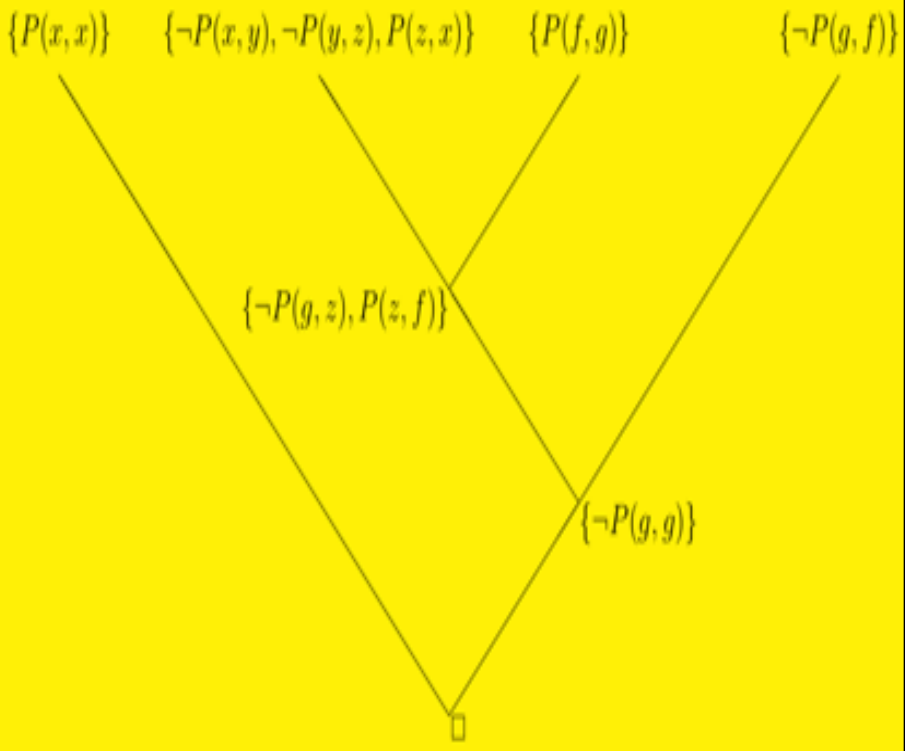


Figure 6: Rezoluční důkaz v logice 1. řádu.

Efektivní hledání rezolučních důkazů

- **Jednotková rezoluce**:
  - Snahou je získat prázdnou klauzuli, proto je dobré, pokud se nově odvozené klauzule zkracují
  - Preferujeme rezoluční krok, kde je jedna z klauzulí jednotková (obsahuje jediný literál)
  - Obecně omezení pouze na jednotkové rezoluce není úplné, ale pro Hornovy klauzule ano (tam je to dopředné řetězení)
- **Množina podpory**:
  - Vybrané klauzule, z nichž alespoň jedna se vždy účastní rezoluce (výsledek rezoluce se přidává do množiny podpory, která neobsahuje spor)
  - Počáteční množina podpory typicky obsahuje negovaný dotaz
- **Vstupní rezoluce**:
  - Každého rezolučního kroku se účastní alespoň jedna klauzule ze vstupu počáteční KB nebo dotaz
  - Nejde o úplnou metodu
- **Subsumce**:
  - Eliminujeme klauzule, které jsou zahrnuty jinými klauzulemi
  - Je-li v bázi znalostí P(x), je zbytečné přidávat P(A) ani P(A) v Q(B)

Rozšíření základní podoby rezolučních důkazů

- Zpracování rovnosti vyžaduje redukci prohledávaného stavového prostoru pomocí techniky *demodulace* a *paramodulace*
- Rezoluce na Hornových klauzulích se dá využít jako programovací jazyk (Prolog a další)
- Systémy logického programování založené na zpětném řetězení využívají důmyslných kompilačních technik a velmi efektivní odvozování; handicapem je citlivost zpětného řetězení na nekonečné cykly a redundantní odvozování
- Dopředné řetězení se využívá v deduktivních databázích a produkčních systémech, je to metoda, která je úplná pro jazyk Datalog, který tímto způsoberm pracuje v polynomiálním čase
- Automatizované dokazování je již prakticky použitelným nástrojem, který umožnil jak rekonstrukci starších výsledků (Gödelova věta o neúplnosti, 1986), tak výsledků nových (Např. Keplerova domněnka o minimálním objemu poskládaných koulí, 2005, resp. 2017