Wumpusova jeskyně. Rezoluční metoda. Predikátová logika.

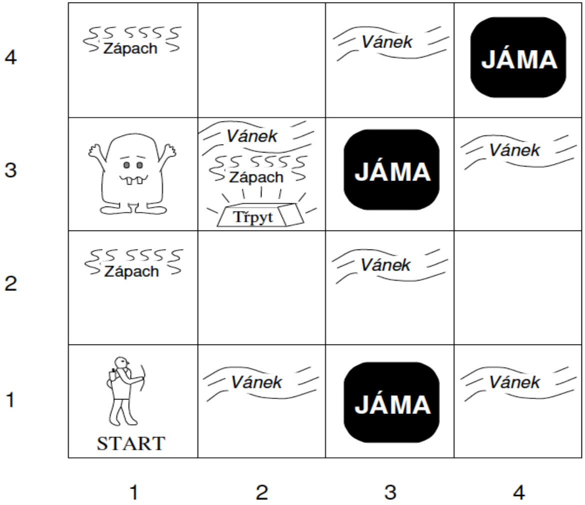
Luboš Popelínský

E-mail: popel@fi.muni.cz
http://nlp.fi.muni.cz/uui/

Obsah:

- Wumpusova jeskyně
- Rezoluční metoda
- Predikátová logika

Wumpusova jeskyně



PEAS pro Wumpusovu jeskyni

P(erformance measure) – míra výkonnosti, zlato +1000 smrt -1000, -1 za krok, -10 za užití šípu

E(nvironment) – prostředí. vedle = vlevo, vpravo, nahoře, dole Místnosti vedle Wumpuse zapáchají. V místnosti vedle jámy je vánek. V místnosti je zlato ⇔ je v ní třpyt. Výstřel zabije Wumpuse, pokud jsi obrácený k němu. Výstřel vyčerpá jediný šíp, který máš. Zvednutím vezmeš zlato ve stejné místnosti. Položení odloží zlato v aktuální místnosti.

A(ctuators) – akční prvky Otočení vlevo, Otočení vpravo, Krok dopředu, Zvednutí, Položení, Výstřel

S(ensors) – senzory Zápach, Vánek, Třpyt, Náraz do zdi, Chroptění Wumpuse

Wumpus: Hrajeme

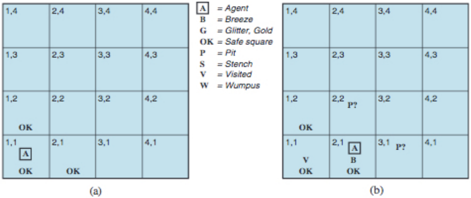
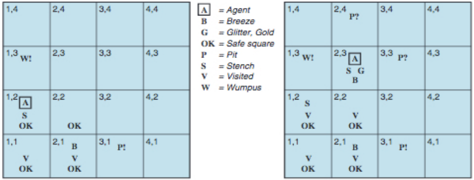
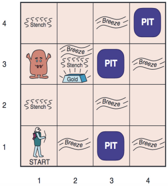
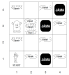


Figure 7.3 The first step taken by the agent in the wumpus world. (a) The initial situation, after percept [*None*, *None*, *None*, *None*, *None*]. (b) After moving to [2,1] and perceiving [*None*, *Breeze*, *None*, *None*, *None*].





Vlastnosti problému Wumpusovy jeskyně



pozorovatelné	ne	jen lokální vnímání
deterministické	ano	přesně dané výsledky
episodické	ne	sekvenční na úrovni akcí
statické	ano	Wumpus a jámy se nehýbou
diskrétní	ano	
více agentů	ne	Wumpus sám je spíš vlastnost prostředí

Elementární výroky pro Wumpusovu jeskyni

Definujeme výrokové symboly pro $i, j \in 1, 2, 3, 4$:

$J_{i,j}$ je pravda $\Leftrightarrow$ Na [i,j] je Jáma.

$W_{i,j}$ je pravda $\Leftrightarrow$ Wumpus je na [i,j], živý či mrtvý.

Co je modelem pro naši hru?

Model pro naši Wumpusovu jeskyni

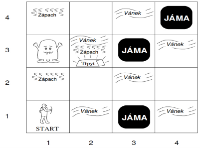
Definujeme výrokové symboly pro $i, j \in 1, 2, 3$:

$J_{i,j}$ je pravda $\Leftrightarrow$ Na [i,j] je Jáma.

$W_{i,j}$ je pravda $\Leftrightarrow$ Wumpus je na [i,j], živý či mrtvý.

Co je modelem pro naši hru?

$J_{1,3}, J_{3,3}, J_{4,4}, W_{3,1}$ mají hodnotu *True*



Báze znalostí KB

Elementární výroky, jejichž pravdivostní hodnota se odvodí z $J_{i,j}$ a $W_{i,j}$:

$V_{i,j}$ je pravda $\Leftrightarrow$ agent cítí na [i,j] Vánek.
$Z_{i,j}$ je pravda $\Leftrightarrow$ agent cítí na [i,j] Zápach.

- platí pro všechny jeskyně:
  - R1: $\neg J_{1,1}$
  - "V poli je Vánek $\Leftrightarrow$ ve vedlejším poli je Jáma." vedlejší = vlevo, vpravo, nahoře, dole. Pro pole sousední s [1,1]:
    R2: $V_{1,1} \Leftrightarrow (J_{1,2} \vee J_{2,1})$
    R3: $V_{2,1} \Leftrightarrow (J_{1,1} \vee J_{2,2} \vee J_{3,1})$
- pro tuto instanci hry:
  - R4: $\neg V_{1,1}$
  - R5: $V_{2,1}$

$$KB = R1 \wedge R2 \wedge R3 \wedge R4 \wedge R5$$

Inference

Stav: [Zápach, Vánek, Třpyt]

Sensory: [false, false, false]

KB = jak uvedeno = pravidla Wumpusovy jeskyně + pozorování

$KB \models$ "[1, 2] je bezpečné pole"

model checking (kontrola modelů) = jednoduchý způsob logické inference

Kontrola všech modelů je bezesporná a úplná (pro konečný počet výrokových symbolů)

jak víme, složitost $O(2^n)$ pro n výrokových symbolů, NP-úplný problém

Rezoluční metoda

Rezoluce: další formální systém

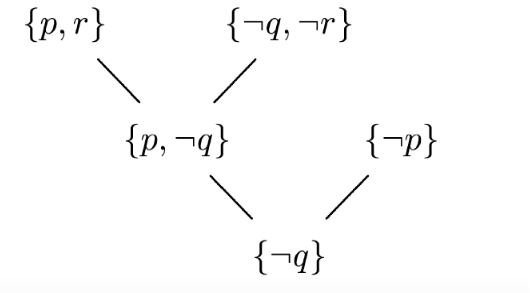
- vhodná pro strojové dokazování (Prolog)
- metoda založená na vyvracení: dokazuje se nesplnitelnost formulí
- pracujeme s formulemi v konjuktivní normální formě (též klauzulárním tvaru), ale používáme jinou notaci:
- klauzule je konečná množina literálů (chápaná jako jejich disjunkce); je pravdivá, pokud alespoň jeden prvek je pravdivý. Prázdná klauzule $\square$ je vždy nepravdivá – neobsahuje žádný pravdivý prvek.
  Příklad klauzule: $\{p, \neg q, r\}$ (tj. $p \vee \neg q \vee r$)
- formule je (ne nutně konečná) množina klauzulí (chápaná jako jejich konjunkce, tedy nkf); je pravdivá, je-li každý prvek pravdivý. Prázdná formule $\emptyset$ je vždy pravdivá – neobsahuje žádný nepravdivý prvek.
  Příklad formule: $\{\{\neg q\}, \{\neg p, q\}, \{p, q, r\}\}$ (tj. $\neg q \wedge (\neg p \vee q) \wedge (p \vee q \vee r)$)

Rezoluční pravidlo

- rezoluční pravidlo: nechť $C_1 = \{p\} \sqcup C_1'$, $C_2 = \{\neg p\} \sqcup C_2'$ jsou klauzule, jejich rezolventou je $C = C_1' \cup C_2'$ (rezolvovali jsme na literálu p)
- rezoluční pravidlo zachovává splnitelnost (lib. valuace splňující C_1 a C_2 splňuje i C; klauzule C_1, C_2 označujeme jako rodiče, C jako potomka)
- rezoluční důkaz klauzule C z formule S je konečná posloupnost klauzulí $C_1, C_2, \ldots, C_n$, kde $C_n = C$ a každé C_i je buď prvkem S, nebo rezolventou klauzulí C_j, C_k pro $j, k < i$. Existuje-li tento důkaz, je C rezolučně dokazatelná z S (píšeme $S \vdash_R C$). Odvození $\square$ z S je vyvrácením S.
- příklad: dokažte $C = \{\neg q\}$ z $S = \{\{p, r\}, \{\neg q, \neg r\}, \{\neg p\}\}$
  $C_1 = \{p, r\}$ (prvek S), $C_2 = \{\neg q, \neg r\}$ (prvek S),
  $C_3 = \{p, \neg q\}$ (rezolventa C_1, C_2), $C_4 = \{\neg p\}$ (prvek S),
  $C = C_5 = \{\neg q\}$ (rezolventa C_3, C_4)

Rezoluce – stromy

- přehlednější formou rezolučních důkazů jsou stromy
- strom rezolučního důkazu C z S je binární strom T s vlastnostmi:
  - kořenem T je C
  - listy T jsou prvky S
  - libovolný vnitřní uzel C_i s bezprostředními následníky C_j, C_k je rezolventou C_j, C_k
- příklad: strom důkazu $C = \{\neg q\}$ z $S = \{\{p, r\}, \{\neg q, \neg r\}, \{\neg p\}\}$

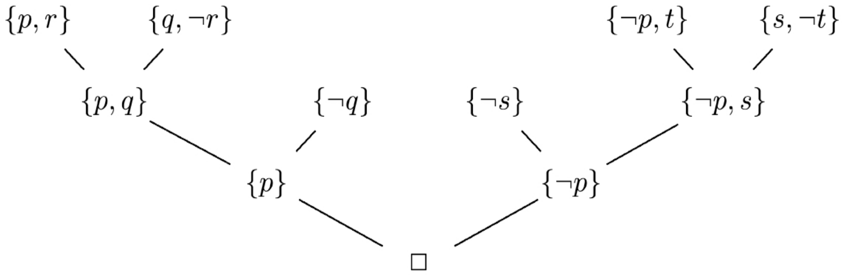


Příklad: Rezoluční vyvrácení

- příklad: vytvořte strom rezolučního vyvrácení formule $(p \vee r) \wedge (q \vee \neg r) \wedge \neg q \wedge (\neg p \vee t) \wedge \neg s \wedge (s \vee \neg t)$

Příklad: Rezoluční vyvrácení

- příklad: vytvořte strom rezolučního vyvrácení S (dokažte $S \vdash_R \square$), je-li $S = \{\{p, r\}, \{q, \neg r\}, \{\neg q\}, \{\neg p, t\}, \{\neg s\}, \{s, \neg t\}\}$



Rezoluce – vlastnosti

- Věta (korektnost a úplnost rezoluce): rezoluční vyvrácení formule S existuje právě tehdy, když je S nesplnitelná.
- důsledek: existuje-li rezoluční strom s listy z množiny S a kořenem $\square$, pak je S nesplnitelná
- obecné schéma důkazu "formule A je log. důsledkem množiny formulí **T**": vytvoříme konjunkci T' všech prvků z **T**, formuli $T' \wedge \neg A$ převedeme do nkf a ukážeme $\mathrm{nkf}(T' \wedge \neg A) \vdash_R \square$
- výhody pro strojové zpracování: systematické hledání důkazu, práce s jednoduchou datovou strukturou, jediné odvozovací pravidlo
- problém: strategie generování rezolvent – prohledávaný prostor může být příliš velký; př.: $\{\{p\}, \{p, \neg q\}, \{\neg p, q, r\}, \{\neg p, \neg r\}, \{r\}\}$ – postup, kdy rezolvujeme 2. a 3. klauzuli na p a výsledek poté se 4. na r, k důkazu nevede

Zjemnění rezoluce

- snaha omezit prohledávaný prostor
  - ukončením prohledávání neperspektivních cest
  - určením pořadí při procházení alternativních cest
- $=$ další podmínky na rodičovské klauzule nebo rezolventu v definici rezoluce
- každé omezení rezolučního pravidla je korektní, ne každé zachovává úplnost.

Příklady možných zjemnění

- vyloučení klauzulí s literálem, který je ve formuli S pouze v jedné paritě
- T-rezoluce: žádná z rodičovských klauzulí není tautologie.
  tautologie obsahuje týž literál v obou paritách např. $\{p, \neg q, \neg p, r\}$
- nechť $\mathcal{A}$ je libovolná interpretace, $\mathcal{A}$-rezoluce (sémantická rezoluce) je rezoluce, kde alespoň jedna z rodičovských klauzulí je v $\mathcal{A}$ nepravdivá.
  (Budou-li rodiče v dané valuaci pravdiví, bude v ní pravdivý i potomek – touto cestou k nesplnitelnosti nedojdeme. $\mathcal{A}$-rezoluce je korektní a úplná.)

Wumpus: rezoluce

Postup řešení:

Vyjdeme z definice: Zjišťujeme, zda cíl G je logickým důsledkem KB, $KB \models G$.

1. Určíme cíl G, např. "je pole [1,2] bezpečné".
2. Znalostní bázi spolu s negovaným cílem $KB \cup \neg G$ převedeme do konjunktivní normální formy
3. Přepíšeme tuto knf do množinové notace, máme množinu S.
4. Dokazujeme, zda S je nesplnitelná.
5. Pokud ano, G je logickým důsledkem KB

Nevýhoda? Pro každou změnu KB a každý cíl G musíme opakovat všechny kroky.

Wumpus: rezoluce

Lépe :

1. Iniciální znalostní bázi KB_0 - to, co platí pro všechny instance Wumpusovy jeskyně - převedeme do konjunktivní normální formy.
2. Přepíšeme tuto knf do množinové notace, máme množinu S.
3. Varianta 1:
   1. Určíme cíl G a převedeme $\neg G$ do konjunktivní normální formy, přepíšeme do množinové notace, množina S_G
   2. Dokazujeme, zda $S \cup S_G$ je nesplnitelná.
   3. Pokud ano, G je logickým důsledkem KB.
4. Varianta 2:
   1. Najdeme všechny logické důsledky, tj. kořeny všech rezolučních stromů;
   2. Vybereme jeden z nich, např. další z bezpečných polí;
   3. Všechny důsledky = klauzule, přidáme do KB
5. V obou variantách: Novou znalost KB_{new} (= množina klauzulí) sjednotíme s dosavadní S_t, tj. KB v čase t : $S_{t+1} = S_t \cup KB_{new}$

Predikátová logika

- plně přejímá výsledky výrokové logiky
- zabývá se navíc strukturou jednotlivých jednoduchých výroků – na základě této analýzy lze odvodit platnost některých výroků, které ve výrokové logice platné nejsou

  Příklad: mějme následující výroky (+ označení výrokovými symboly):

  Každý člověk je smrtelný. (p)
  Sokrates je člověk. (q)
  Sokrates je smrtelný. (r)

  Na základě výrokové logiky nevyplývá r z p a q; přesto je úsudek zřejmě platný (na jiné úrovni, než je výroková logika).

Základní pojmy

- Predikát je *n*-ární relace; vyjadřuje vlastnosti objektů a vztahy mezi objekty.
- konstanty reprezentují jména objektů (individuí); jedná se o prvky předem specifikované množiny hodnot – domény
- proměnné zastupují jména objektů, mohou nabývat libovolných hodnot z dané domény
- *n*-ární predikáty lze chápat jako množiny takových *n*-tic konstant, pro které je predikát splněn
- příklady:

  doména: přirozená čísla s nulou
  predikát $x < 4$ lze chápat jako $\{0, 1, 2, 3\}$

  doména: $\{0, 1, 2\}$
  predikát $x < y$ lze chápat jako $\{(0, 1), (1, 2), (0, 2)\}$

Funkce, termy. Kvantifikátory

- funkce reprezentují složená jména objektů
- příklad: nechť funkce $f(x, y)$ reprezentuje sčítání. Pak $f(1, 2)$ (stejně jako $f(2, 1), f(0, 3)$) jsou možná složená jména pro konstantu 3.
- poznámka: konstanty jsou nulární funkce
- výrazy složené pouze z funkčních symbolů, konstant a proměnných = termy. Příklad termu: $f(x, g(y, h(x, y), 1), z)$
- termy nabývají hodnot z dané domény
- např. pro doménu přirozených čísel s nulou
  term $2 + (2 * x)$ může nabývat hodnot z množiny $\{2, 4, 6, \ldots\}$

Složené predikáty lze vytvářet i pomocí kvantifikátorů

- univerzální (obecný) kvantifikátor $\forall$:
  $\forall x P(x)$ – pro každý prvek x domény platí $P(x)$
- existenční kvantifikátor $\exists$:
  $\exists x P(x)$ – existuje alespoň jeden prvek x domény, pro který platí $P(x)$

Poznámky k zavedenému formálnímu jazyku

- predikátová logika 1. řádu : jen objektové (individuální) proměnné lze vázat kvantifikátory. V logice druhého řádu jsou povoleny i predikátové proměnné.
- konkrétní volbou (konstant), funkčních a predikátových symbolů lze formulovat specifický jazyk, pro který budou jistě platit obecné logické principy. Navíc pro něj mohou platit v závislosti na vlastnostech zvolených prvků i jiné (mimologické) principy, které je ovšem třeba specifikovat pomocí axiomů nebo pravidel. Takový jazyk je pak označován jako jazyk prvního řádu.

  Příklad – jazyk elementární aritmetiky:

  zvolené symboly: konstanta 0, unární funkce následník s, binární $+, *$

  možné termy: $0, s(0), s(x), (x+y)*0, (s(s(0))+(x*y))*s(0)$

  možné formule: $s(0)=(0*x)+s(0), \exists x(y=x*z),$
  $\forall x((x \neq 0) \Rightarrow \exists y(x=s(y)))$

Vázaný a volný výskyt proměnných

- podformule formule A je libovolná spojitá podčást A, která je sama formulí
  Příklad: formule $A = \exists x((\forall y P(z)) \Rightarrow R(x, y))$ má kromě sebe samé následující podformule: $(\forall y P(z)) \Rightarrow R(x, y), \forall y P(z), R(x, y), P(z)$
- výskyt proměnné x ve formuli A je vázaný, pokud existuje podformule B formule A, která obsahuje tento výskyt x a začíná $\forall x$, resp. $\exists x$. Výskyt proměnné je volný, není-li vázaný.
  Příklad: výskyt proměnné x v předchozí formuli A je vázaný (hledanou podformulí je celá A), proměnné y a z jsou volné
- proměnná x se volně vyskytuje v A, má-li tam alespoň jeden volný výskyt
- sentence predikátové logiky je formule bez volných výskytů proměnných (všechny výskyty všech proměnných jsou vázané)
- otevřená formule je formule bez kvantifikátorů

Substituce proměnných

- ‚skutečnými proměnnými', za které lze dosadit (udělit jim hodnotu, provést substituci), jsou pouze volné proměnné
- term t je substituovatelný za proměnnou x ve formuli A, pokud pro každou proměnnou y obsaženou v t neobsahuje žádná podformule A tvaru $\forall yB$, $\exists yB$ volný výskyt proměnné x
- je-li t substituovatelný za x v A, označíme $A(x/t)$ výraz, který vznikne z A nahrazením každého volného výskytu x termem t
- příklad: ve formuli $A = \exists xP(x, y)$ je možné provést například následující substituce: $A(y/z) = \exists xP(x, z)$, $A(y/2) = \exists xP(x, 2)$, $A(y/f(z, z)) = \exists xP(x, f(z, z))$. Není však možné substituovat $A(y/f(x, x)) = \exists xP(x, f(x, x))$, protože by došlo k nežádoucí vazbě proměnných.

Sémantika predikátové logiky

- pro analýzu sémantiky potřebujeme nejprve specifikaci jazyka - doménu, konstanty, funkční a predikátové symboly
- příklad: formální jazyk s jediným binárním predikátovým symbolem $P(x, y)$ a jediným binárním funkčním symbolem $f(x, y)$ lze chápat mj. jako
  - přirozená čísla s $<$ a $+$
  - racionální čísla s $\geq$ a *max*
  - celá čísla s $>$ a $*$
- interpretace (realizace) jazyka predikátové logiky je struktura I složená z
  - libovolné neprázdné množiny **D** domény (oboru interpretace)
  - zobrazení $I(f) : \mathbf{D}^n \to \mathbf{D}$ pro každý n-ární funkční symbol $f, n \geq 0$
  - n-ární relace $I(P) \subseteq \mathbf{D}^n$ pro každý n-ární predikátový symbol P, $n \geq 1$

Interpretace jazyka: příklad

Příklad: mějme jazyk s binárním predikátovým symbolem $P(x, y)$, binárním funkčním symbolem $f(x, y)$ a symboly pro konstanty $a, a_1, a_2, \ldots, b_1, b_2, \ldots,$ který chceme interpretovat jako celá čísla s $>$ a $+$.

- **D** je množina celých čísel,
- $I(a) = 0, I(a_1) = 1, I(a_2) = 2, \ldots, I(b_1) = -1, I(b_2) = -2, \ldots$ (jedná se o nulární funkce),
- $I(f) = +$
  (funkce zadaná pomocí rovnosti funkcí: zobrazení $I(f)$ definujeme jako funkci sčítání celých čísel; pak např. $I(f)(4, -2) = 2$),
- $I(P) = \;>$
  (relace zadaná pomocí rovnosti relací: $I(P)$ definujeme jako relaci ‚větší než' pro celá čísla; např. $(2, -1) \in I(P)$)

Interpretace proměnných a termů

- interpretace volných proměnných spočívá v jejich ohodnocení, což je libovolné zobrazení V (valuace) z množiny všech proměnných do $\mathbf{D}$
- ohodnocení, které přiřazuje proměnné x prvek $d \in \mathbf{D}$ a na ostatních proměnných splývá s valuací V, označíme $V[x/d]$
- hodnotou termu t v interpretaci I a valuaci V je prvek $|t|_{I,V} \in \mathbf{D}$ takový, že
  - je-li t proměnná, $|t|_{I,V} = V(t)$
  - je-li $t = f(t_1, \ldots, t_n)$, pak $|t|_{I,V}$ je hodnotou funkce $I(f)$ pro argumenty $|t_1|_{I,V}, \ldots, |t_n|_{I,V}$
- příklad: mějme interpretaci I z předchozího příkladu (celá čísla s $>$ a $+$) a valuaci $V(x) = 2$. Pak
  $|f(b_1, f(b_2, b_2))|_{I,V} = +(-1, +(-2, -2)) = -5$
  $|f(f(a, b_1), f(x, a_1))|_{I,V} = +(+(0, -1), +(2, 1)) = 2$

Splnitelnost formulí

Formule A je splňována interpretací I a valuací V, pokud

- A je $P(t_1, \ldots, t_n)$ a $(|t_1|_{I,V}, \ldots, |t_n|_{I,V}) \in I(P)$
- A je $t_1 = t_2$ a $|t_1|_{I,V} = |t_2|_{I,V}$ (oba termy reprezentují týž prvek)
- A je $\neg B$ a I, V nesplňují B
- A je tvaru $B \wedge C$ a I, V splňují B i C
- A je $B \vee C$ a I, V splňují B nebo C
- A je tvaru $B \Rightarrow C$ a I, V nesplňují B nebo splňují C
- A je $B \Leftrightarrow C$ a I, V splňují B i C nebo nesplňují B i C
- A je $\forall x B$ a $I, V[x/d]$ splňují B pro libovolné $d \in \mathbf{D}$
- A je $\exists x B$ a $I, V[x/d]$ splňují B alespoň pro jedno $d \in \mathbf{D}$

Splnitelnost – příklad

Příklad: mějme dříve uvedenou interpretaci I (celá čísla s $>$ a $+$), mějme jinou interpretaci I' téhož formálního jazyka (celá čísla s $>$ a $*$, od I se liší pouze definicí $I'(f) = *$) a valuace definované na proměnné x takto: $V_1(x) = -2,\ V_2(x) = 2$

- formuli $\forall x P(f(x, a_1), x)$ interpretujeme v I jako $\forall x(x + 1 > x)$; formule je splňována I a libovolnou valuací. V I' interpretujeme formuli jako $\forall x(x * 1 > x)$ – není splněna pro libovolnou valuaci.
- $\forall x P(x, y)$ interpretujeme (v I i I') jako $\forall x(x > y)$, formule není splňována I (ani I') a libovolnou valuací
- formule $P(x, a)$ interpretovaná (v I i I') jako $x > 0$ je splňována I (i I') a V_2, není splňována I (ani I') a V_1
- formule $\forall x(P(x, a) \Rightarrow \exists y P(x, y))$ interpretovaná (v I i I') jako $\forall x((x > 0) \Rightarrow \exists y(x > y))$ je splňována I (i I') a libovolnou valuací

Pravdivost formulí a jejich klasifikace

- formuli A nazveme pravdivou v interpretaci I, je-li splňována I pro libovolnou valuaci, píšeme $\models_I A$
- pravdivost formule záleží pouze na valuaci volných proměnných, které se v ní vyskytují
- pravdivost sentence (uzavřené formule) nezávisí na valuaci vůbec

Formule A predikátové logiky se nazývá

- tautologie, je-li pravdivá pro každou interpretaci (tj. pro každou I platí $\models_I A$), značíme $\models A$
- splnitelná, pokud existuje alespoň jedna interpretace a valuace, které ji splňují (př.: $\forall x P(x, x)$)
- kontradikce, je-li $\neg A$ tautologie (tj. $\models \neg A$)

Shrnutí

Víme,

- jak popsat a řešit Wumpusovu jeskyni ve výrokové logice
- co je rezoluční metoda ve výrokové logice
- jak budovat rezoluční důkaz
- jak definovat syntax a sémantiku predikátové logiky