Umělá inteligence I

Roman Barták, KTIML

roman.bartak@mff.cuni.cz
http://ktiml.mff.cuni.cz/~bartak

9

Dnes

- Už umíme používat výrokovou logiku pro reprezentaci znalostí a odvozování důsledků.
- Dnes zopakujeme stejný postup jako minule, ale použijeme obecnější nástroj – **predikátovou logiku.**
  - **reprezentace znalostí**
  - **odvozování v predikátové logice**
    - **převodem na výrokovou logiku**

Reprezentace znalostí

Hledáme formální jazyk, který nám umožní

- **reprezentovat znalosti**
- **pracovat se znalostmi**

Co třeba **programovací jazyky** (C++, Java,…)?

- asi nejrozšířenější třída formálních jazyků
- fakta lze popisovat **datovými strukturami**
  - array svět [4,4]
- **programy** popisují výpočtové procesy (změny datových struktur)
  - svět[2,2] ← díra
- Jak ale obecně odvozovat další fakta z faktů známých?
  - změny datových struktur jsou dělány ad-hoc procedurami → **procedurální přístup**
  - **deklarativní přístup** naproti tomu odděluje znalost a mechanismus pro práci se znalostí (ten je navíc obecný, nezávislý na problému)
- Jak reprezentovat znalosti typu „na pozicích [2,2] nebo [3,1] je díra"?
  - proměnné v programech nabývají jediné hodnoty

Přirozené jazyky

- Nemohli bychom pro reprezentaci znalostí použít **jazyk přirozený**, kde můžeme vyjádřit „vše"?
  - Bylo by to krásné, ale zatím se nepodařilo najít přesnou formální sémantiku!
  - Současný pohled na přirozený jazyk je, že slouží spíše jako **médium pro komunikaci** než médium pro čistou reprezentaci.
    - Věta sama nekóduje informaci, záleží také na **kontextu**, ve kterém je vyslovena.
      - „Podívejte!"
    - Jazyk **není kompozicionální** – význam jedné věty může záviset na významu vět okolních.
      - "Pak jsem to uviděl."
    - Problémem přirozeného jazyka je jeho **nejednoznačnost**.
      - koleje, stopky, …

Zpět k logice

- **Výroková logika** je deklarativní, má kompoziciální sémantiku, která nezáleží na kontextu a je jednoznačná.
- Některé vlastnosti světa ale vyjadřuje těžkopádně (ne moc kompaktně).
  - Wumpus: v okolí každá buňky s dírou je vánek
    - $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
    - $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
    - ...
- Poučme se v přirozeném jazyce:
  - máme zde podstatná jména, která představují **objekty** (díra, buňka,...)
  - slovesa vyjadřují **vztahy** mezi objekty (sousedí s, ...)
  - některé relace jsou vlastně **funkce** (je otcem od)
- Místo prostých faktů (výroková logika) začneme pracovat s objekty, relacemi a funkcemi a navíc budeme vyjadřovat fakta o některých nebo o všech objektech světa (**predikátová logika**).

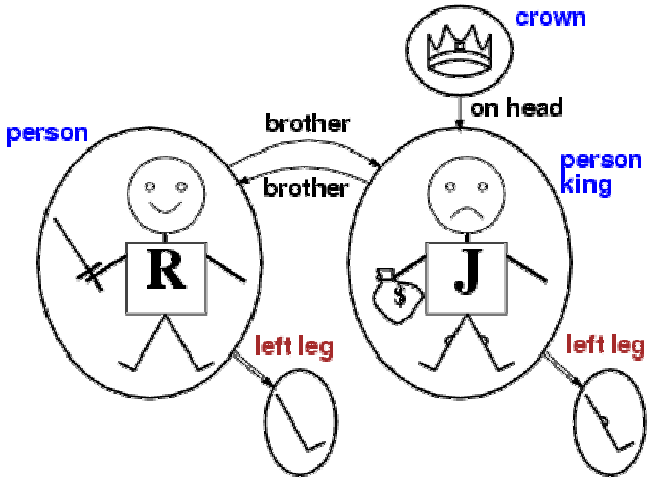
Logiky všeobecně

Výroková logika	fakta, která platí nebo ne
Predikátová logika	objekty a relace mezi nimi platí nebo ne
Temporální logika	fakta a časy, kdy daná fakta platí nebo ne
Logika vyšších řádů	relace mezi objekty jsou také objekty (lze dělat tvrzení o všech relacích)

Predikátová logika
syntax

- konstanty — John, 2, Crown,...
- predikáty — Brother, >,...
- funkce — Sqrt, LeftLeg,...
- proměnné — x, y, a, b,...
- spojky — $\neg, \Rightarrow, \wedge, \vee, \Leftrightarrow$
- rovnost — $=$
- kvantifikátory — $\forall, \exists$

Příklad



- **konstanty** (jména objektů):
  - Richard, John, TheCrown
- **funkční symbol**:
  - LeftLeg
- **termy** (jiný způsob pojmenování objektu)
  - LeftLeg(John)
- **predikátové symboly**:
  - Brother, OnHead, Person, King, Crown
- **atomická tvrzení** (popis relace mezi objekty):
  - Brother(Richard,John)
- **složená tvrzení**:
  - King(Richard) $\vee$ King(John)
  - $\neg$King(Richard) $\Rightarrow$ King(John)
- **tvrzení o výběru objektů** (kvantifikátory):
  - $\forall$x King(x) $\Rightarrow$ Person(x)
  - Pozor: $\forall$x King(x) $\wedge$ Person(x) !!!
  - $\exists$x Crown(x) $\wedge$ OnHead(x,John)
  - Pozor: $\exists$x Crown(x) $\Rightarrow$ OnHead(x,John) !!!

- $\forall$x,y Brother(x,y) $\Rightarrow$ Brother(y,x)
- $\exists$x,y Brother(x,Richard) $\wedge$ Brother(y,Richard)
- $\exists$x,y Brother(x,Richard) $\wedge$ Brother(y,Richard) $\wedge \neg$(x=y)
  Rovnost říká, že dva termy odkazují na stejný objekt (Father(John) = Henry).

Kvantifikátory

- **Univerzální kvantifikátor ∀x P**
  - P je pravda pro každý možný objekt x
  - zhruba odpovídá konjunkci všech instaciovaných P
    - P(John) ∧ P(Richard) ∧ P(TheCrown) ∧ P(LeftLeg(John)) ∧ ...
  - typicky se váže s implikací (výběr objektů, pro které je něco pravda)
    - ∀x King(x) ⇒ Person(x)
- **Existenční kvantifikátor ∃x P**
  - existuje objekt x, pro kterou je P pravda
  - zhruba odpovídá disjunkci všech instaciovaných P
    - P(John) ∨ P(Richard) ∨ P(TheCrown) ∨ P(LeftLeg(John)) ∨ ...
- **Vazby mezi kvantifikátory**
  - ∀x ∀y je to samé jako ∀y ∀x
    ∃x ∃y je to samé jako ∃y ∃x
  - ∃x ∀y není to samé jako ∀y ∃x (∃x ∀y Loves(x,y) vs.∀y ∃x Loves(x,y))
  - ∀x P je to samé jako ¬∃x ¬P
    ∃x P je to samé jako ¬∀x ¬P

PL a báze znalostí

- Podobně jako u výrokové logiky použijeme **operace TELL** pro přidávání tvrzení do báze znalostí:
  - TELL(KB, King(John))
  - TELL(KB,∀x King(x) ⇒ Person(x))
  - Přidáváme **axiomy** (**fakta** jako atomická tvrzení, **definice** s použitím ⇔ a jiná složená tvrzení) a někdy i **teorémy** (lze je logicky odvodit z axiomů, ale „zrychlují" odvozování)
- a **operace ASK** pro dotazování se na logické důsledky KB:
  - ASK(KB, King(John)) — jako databázový dotaz
  - ASK(KB, Person(John)) — tady už je potřeba odvození
  - ASK(KB, ∃x Person(x)) — zde budeme chtít nejen odpověď true, ale také hodnotu x, pro kterou to platí – substituce {x/John}

Odvozování v PL

- Již umíme odvozovat ve výrokové logice a teď se podíváme na odvozování v logice predikátové.
- Základní odlišnosti:
  - kvantifikátory → **skolemizace**
  - funkce a proměnné → **unifikace**
- Zbytek už je nám známý:
  - **dopředné řetězení** (dedukční databáze, produkční systémy)
  - **zpětné řetězení** (logické programování)
  - **rezoluce** (dokazování vět)

Redukce PL na VL

- **Odvozování v predikátové logice může udělat převedením do logiky výrokové a použitím jejích odvozovacích nástrojů.**
  - „uzemnění" (grounding)
    - za všechny proměnné dosadíme všechny možné termy
    - atomická tvrzení potom odpovídají výrokovým proměnným
  - A co kvantifikátory?
    - univerzální kvantifikátor: proměnná nahrazena termem
    - existenciální kvantifikátor: skolemizace (proměnné nahrazena novou konstantou)

Redukce PL na VL

odstranění kvantifikátorů

Univerzální instanciace

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

Pro proměnnou a instanciovaný term g aplikujeme substituci g za x.

Aplikujeme opakovaně pro různé instanciace g.

- Příklad: $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$ vede na:
  - $King(John) \wedge Greedy(John) \Rightarrow Evil(John)$
  - $King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$
  - $King(LeftLeg(John)) \wedge Greedy(LeftLeg(John)) \Rightarrow Evil(LeftLeg(John))$
  - …

Existenciální instanciace

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

Pro proměnnou a novou konstantu k aplikujeme substituci k za x.

Aplikujeme jedenkrát s novou konstantou, která se nevyskytuje v bázi znalostí (Skolemova konstanta)

- Příklad: $\exists x\ Crown(x) \wedge OnHead(x,John)$ vede na:
  - $Crown(C_1) \wedge OnHead(C_1,John)$

Redukce PL na VL

příklad

- Ze znalostní báze v predikátové logice (zatím bez funkcí):
  - $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
  - $King(John)$
  - $Greedy(John)$
  - $Brother(Richard,John)$
- vytvoříme znalostní bázi ve výrokové logice všemi možnými instanciacemi:
  - $King(John) \wedge Greedy(John) \Rightarrow Evil(John)$
  - $King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$
  - $King(John)$
  - $Greedy(John)$
  - $Brother(Richard,John)$
- Odvození provedeme ve výrokové logice.

Problém: funkční symboly vedou k nekonečně mnoha termům $LeftLeg(John)$, $LeftLeg(LeftLeg(John))$,…

- Herbrand: odvození v PL z dané KB existuje, pokud existuje odvození ve VL v konečné podmnožině instanciované KB
- postupně přidáváme větší a větší termy do KB, dokud nenajdeme odvození
- když ale odvození neexistuje, neskončíme ☹