Umělá inteligence I

11

Roman Barták, KTIML

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

Dnes

Umíme **reprezentovat znalosti** v logice a **odvozovat** nové znalosti.

Jak ale reprezentace znalostí vypadá v praxi?

- **Znalostní inženýrství**
  - postup tvorby báze znalostí
- **Reprezentace znalostí**
  - kategorie a taxonomie
  - akce a plány
    - situační kalkulus

Znalostní inženýrství

- **Znalostní inženýrství** se zabývá procesem, jak obecně budovat znalostní báze.
- **Znalostní inženýr** musí:
  - **pochopit** příslušnou problémovou **oblast** (doménu)
    - Jak příslušná oblast funguje?
    - typicky ve spolupráci s expertem v dané oblasti
  - **určit** jaké **koncepty** jsou v ní důležité pro řešení problémů
    - Jaké otázky budeme klást a co potřebujeme znát pro nalezení odpovědi?
  - **navrhnout** formální **reprezentaci** objektů a relací
    - Jak vše formálně zakódovat, aby si s tím počítač poradil?

Znalostní inženýrství

v krocích

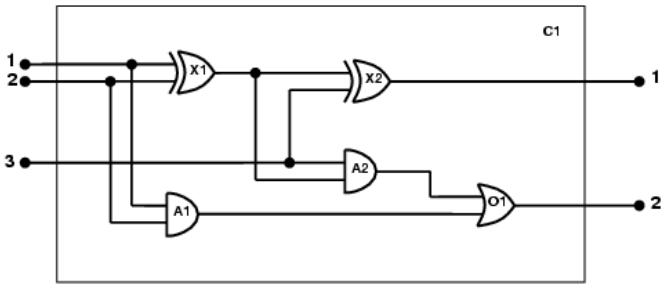
1. **identifikace úlohy**
   - Jaké otázky budeme klást do znalostní báze?
   - Wumpus: vybíráme akce nebo se jen ptáme na vlastnosti prostředí?
2. **sestavení (získání) relevantních znalostí (knowledge acquisition)**
   - Jak daná doména skutečně funguje?
   - Wumpus: co znamená vítr a zápach?
3. **rozhodnutí o slovníku predikátů, funkcí a konstant**
   - Jak přeložit koncepty světa na logické termíny?
   - Wumpus: je díra elementární fakt nebo funkce buňky?
   - Výsledkem je **ontologie** dané domény (slovník používaných pojmů).
4. **zakódování obecných informací o doméně**
   - Jaké axiomy v doméně platí?
   - Wumpus: vítr znamená díru v okolní buňce
5. **zakódování specifické problémové instance**
   - Co aktuálně víme o stavu domény?
   - Wumpus: stojíme na buňce (1,1) s pohledem doprava.
6. **kladení otázek inferenčnímu mechanismu a získání odpovědí**
   - Jak funguje obecný inferenční mechanismus s naší bází znalostí?
   - Wumpus: je buňka (2,2) opravdu bezpečná?
7. **ladění znalostní báze**
   - Co (jaké axiomy) jsme zapomněli uvést v bázi znalostí?
   - Wumpus: ve světě žije jediný Wumpus

Elektronické obvody

identifikace úlohy (1/7)

Bitová sčítačka

- 1 a 2 jsou vstupní bity, 3 je vstupní přenosový bit
- 1 je výstupní bit pro součet, 2 je výstupní přenosový bit



Co nás bude zajímat?

- Sčítá správně?
- Jak vypadá výstup pro daný vstup?
- Jak vypadá vstup pro požadovaný výstup?

- **Jiný typ otázek může vyžadovat jiný typ znalostí.**
  - Kolik takový čip stojí?
  - Kolik plochy zabere?
  - Jakou má spotřebu?

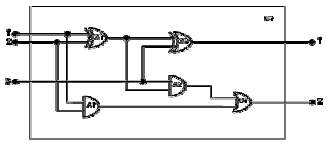
knowledge acquisition

Elektronické obvody

získání znalostí (2/7)

Co víme o elektronických obvodech?

- obvody se skládají z drátů a hradel
- mezi hradly putují signály 0 a 1 přes dráty
- dráty přivádí signál na vstup(y) hradla
- každé hradlo má jeden výstup, jehož hodnota je dána vstupy a typem hradla
- máme čtyři typy hradel: AND, OR, XOR, NOT
- obvod má také vstupy a výstupy
- dráty nás zde zajímají pouze jako spojnice vstupů a výstupů
- neuvažujeme zpoždění signálu, spotřebu a tvar hradel

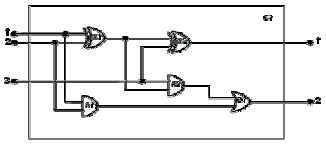


Elektronické obvody

slovník pojmů (3/7)

Jaké budou konstanty, funkce, a predikáty?

- potřebujeme popisovat obvody, hradla, vstupy, výstupy, signály a spojnice
  - **hradla** můžeme označit **konstantami X_1, X_2, A_1, ...**
  - není potřeba popisovat chování každého hradla zvlášť, chování záleží jen na **typu hradla**
    - zavedeme konstanty AND, OR, XOR, NOT
    - typ hradla určíme **funkcí Type(X_1) = XOR**
    - můžeme použít i predikáty typu Type(X_1,XOR) nebo XOR(X_1)
      - Pozor! Budeme potřebovat axiomy, že typ hradla je jedinečný.
  - **vstupy a výstupy** hradel můžeme také pojmenovat konstantami (X_1In$_1$, ...), ale stejně je musím nějak navázat na konkrétní hradlo
    - asi lepší bude opět použít **funkci In(1, X_1), ...**
  - **spojnice** můžeme popisovat **predikáty**
    - **Connected(Out(1, X_1),In(1, X_2)), ...**
    - Pozor! Spojujeme vstupy a výstupy (ne hradla).
  - **signály** na vstupech a výstupech určíme **funkcí**
    - **Signal(g) = 1**



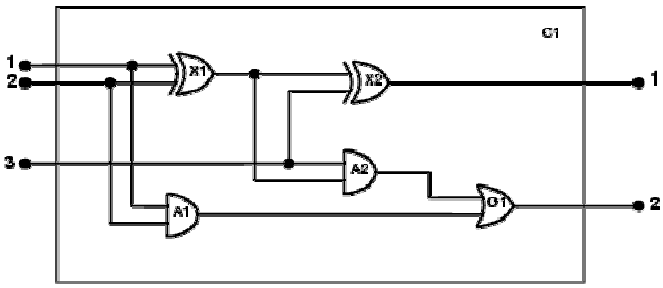
Elektronické obvody

kódování obecných znalostí (4/7)

- **Signály na koncích spojnice jsou totožné**
  - $\forall t_1,t_2$ Connected(t_1, t_2) $\Rightarrow$ Signal(t_1) = Signal(t_2)
- **Signál je pouze tvaru 0 nebo 1, ale ne obojí**
  - $\forall t$ Signal(t) = 1 $\vee$ Signal(t) = 0
  - $1 \neq 0$
- **Spojnice je komutativní**
  - $\forall t_1,t_2$ Connected(t_1, t_2) $\Rightarrow$ Connected(t_2, t_1)
- **Chování hradla je určeno jeho typem**
  - $\forall g$ Type(g) = OR $\Rightarrow$ Signal(Out(1,g)) = 1 $\Leftrightarrow$ $\exists n$ Signal(In(n,g)) = 1
  - $\forall g$ Type(g) = AND $\Rightarrow$ Signal(Out(1,g)) = 0 $\Leftrightarrow$ $\exists n$ Signal(In(n,g)) = 0
  - $\forall g$ Type(g) = XOR $\Rightarrow$ Signal(Out(1,g)) = 1 $\Leftrightarrow$ Signal(In(1,g)) $\neq$ Signal(In(2,g))
  - $\forall g$ Type(g) = NOT $\Rightarrow$ Signal(Out(1,g)) $\neq$ Signal(In(1,g))

Elektronické obvody

kódování instance problému (5/7)



Type(X_1) = XOR
Type(X_2) = XOR
Type(A_1) = AND
Type(A_2) = AND
Type(O_1) = OR

Connected(Out(1,X_1),In(1,X_2))
Connected(Out(1,X_1),In(2,A_2))
Connected(Out(1,A_2),In(1,O_1))
Connected(Out(1,A_1),In(2,O_1))
Connected(Out(1,X_2),Out(1,C_1))
Connected(Out(1,O_1),Out(2,C_1))

Connected(In(1,C_1),In(1,X_1))
Connected(In(1,C_1),In(1,A_1))
Connected(In(2,C_1),In(2,X_1))
Connected(In(2,C_1),In(2,A_1))
Connected(In(3,C_1),In(2,X_2))
Connected(In(3,C_1),In(1,A_2))

Elektronické obvody

dotazy a ladění (6,7/7)

Dotazy klademe v podobě **logické formule**.

- Co musí být na vstupu, abychom dostali výstup 0 s přenosem 1?
  - $\exists i_1,i_2,i_3$ Signal(In(1,C_1)) = $i_1 \wedge$ Signal(In(2,C_1)) = $i_2 \wedge$ Signal(In(3,C_1)) = $i_3 \wedge$ Signal(Out(1,C_1)) = 0 $\wedge$ Signal(Out(2,C_1)) = 1

Odpověď je v podobě **substituce proměnných** i_1,i_2,i_3.

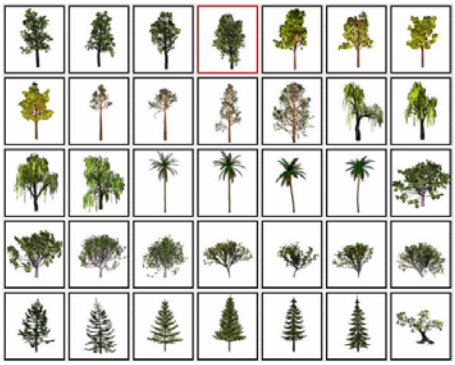
- {i_1/1, i_2/1, i_3/0}, {i_1/1, i_2/0, i_3/1}, {i_1/0, i_2/1, i_3/1}

Ladění báze znalostí

- Dotazy, které dávají nečekanou (špatnou) odpověď, indikují nějaký problém v bázi znalosti (špatný axiom).
  - Typickým příkladem je chybějící axiom říkající, že dvě různé konstanty označují různé objekty.
    - 1 ≠ 0

Objekty a kategorie

- Všimněme si, že
  - agent **pracuje** s reálnými **objekty**
  - ale **uvažování** provádí na úrovni **kategorií** objektů
  - Agent z pozorování světa odvodí (na základě vnímaných vlastností) pro daný objekt jeho příslušnost do dané kategorie a potom používá informaci o této kategorii k dělání předpovědí o objektu.
- **Kategorie**
  = množina svých členů
  = komplexní objekt s relacemi
  - být členem (MemberOf)
  - být podmnožinou (SubsetOf)



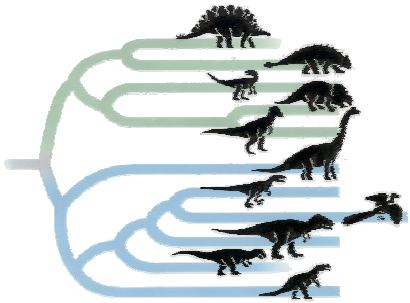
Kategorie a logika

Jak reprezentovat kategorie logickým způsobem?

- objekt je **členem** kategorie
  - MemberOf(BB_{12},Basketballs)
- kategorie je **podtřídou** jiné kategorie
  - SubsetOf(Basketballs,Balls)
- všichni členové kategorie mají nějakou **vlastnost**
  - $\forall$ x (MemberOf(x,Basketballs) $\Rightarrow$ Round(x))
- všichni členové kategorie jsou **rozpoznatelní** na základě společných vlastností
  - $\forall$ x (Orange(x) $\wedge$ Round(x) $\wedge$ Diameter(x)=9.5in $\wedge$ MemberOf(x,Balls) $\Rightarrow$ MemberOf(x,BasketBalls))
- kategorie jako celek může mít nějakou vlastnost
  - MemberOf(Dogs,DomesticatedSpecies)

Taxonomie

- Kategorie organizují a zjednodušují bázi znalostí prostřednictvím **dědění vlastností**.
  - vlastnost definujeme pro kategorii, ale dědí ji všichni členové kategorie
  - potrava je jedlá, ovoce ke potrava, jablka jsou ovoce, tudíž všechna jablka jsou jedlá
- Podtřídy organizují kategorie do **taxonomie**
  - hierarchická struktura sloužící pro kategorizaci objektů
  - původně kategorizace všech živých organizmů (alfa taxonomie)
  - kategorizace veškerého vědění
    - klasifikace v knihovnictví
    - Dewey Decimal Classification
    - 330.94 European economy

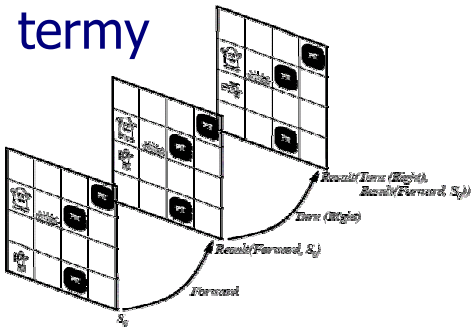


Akce a situace

- Zatím jsme se soustředili na popis znalostí o statickém světě.
- **Jak ale uvažovat o akcích a jejich důsledcích?**
- Ve výrokové logice potřebujeme kopii každé akce pro každý čas (situaci):
  - $L^t_{x,y} \wedge FacingRight^t \wedge Forward^t \Rightarrow L^{t+1}_{x+1,y}$
  - potřebujeme horní limit na počet časových kroků a i tak dostaneme obrovské množství formulí
- Můžeme akce reprezentoval lépe v logice predikátové?
  - kopiím axiomů popisujícím akce se můžeme vyhnout univerzální kvantifikací přes čas (situace)
  - $\forall t$ P je výsledkem v čase t+1 provedení akce A

Situační kalkulus

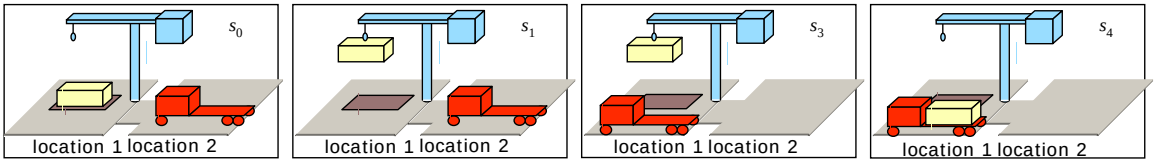
- **akce** reprezentujeme logickými termy
  - Go(x,y)
  - Grab(g)
  - Release(g)
- **situace** jsou logické termy
  - počáteční situace: S_0
  - situace po aplikaci akce a na situaci s: Result(a,s)



- **flexibilní predikáty a funkce** (fluents), které se mění s časem
  - situace je v posledním argumentu
  - Holding(G, S_0)
- **neměnné** (rigid, eternal) **predikáty a funkce**
  - Gold(G)
  - Adjacent(x,y)

Plány

- Je užitečné uvažovat také o posloupnostech (seznamech) akcí – **plánech.**
  - Result([],s) = s
  - Result([a|seq],s) = Result(seq, Result(a,s))
- Jaké úlohy s plány bude agent řešit?
  - **projekční úloha** – jaká je výsledná situace po aplikování dané posloupnosti akcí?
    - At(Agent, [1,1] , S_0) ∧ At(G, [1,1], S_0) ∧ ¬Holding(o, S_0)
    - At(G, [1,1], Result([Go([1,1],[1,2]),Grab(G),Go([1,2],[1,1])], S_0))
  - **plánovací úloha** – jaká posloupnost akcí vede k dané situaci?
    - ∃seq At(G, [1,1], Result(seq, S_0))



Reprezentace akce

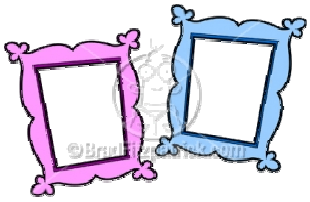
- Akci můžeme popsat dvěma axiomy:
  - **axiom použitelnosti** Preconditions $\Rightarrow$ Poss(a,s)
    - At(Agent,x,s) $\wedge$ Adjacent(x,y) $\Rightarrow$ Poss(Go(x,y),s)
    - Gold(g) $\wedge$ At(Agent,x,s) $\wedge$ At(g,x,s) $\Rightarrow$ Poss(Grab(g),s)
    - Holding(g,s) $\Rightarrow$ Poss(Release(g),s)
  - **axiom efektu** Poss(a,s) $\Rightarrow$ Changes
    - Poss(Go(x,y),s) $\Rightarrow$ At(Agent,y,Result(Go(x,y),s))
    - Poss(Grab(g),s) $\Rightarrow$ Holding(g,Result(Grab(g),s))
    - Poss(Release(g),s) $\Rightarrow \neg$Holding(g,Result(Release(g),s))
- Pozor! To nám ještě nestačí, abychom mohli odvodit, že plán vede k cíli.
  - Axiomy efektu popisují, co se ve světě mění, ale neříkají, že vše ostatní se nemění!
  - odvodíme At(Agent, [1,2], Result(Go([1,1],[1,2]), S_0))
  - ale neodvodíme At(G, [1,2], Result(Go([1,1],[1,2]), S_0))
  - **problém rámce** (frame problem)

frame problem

Problém rámce

- Reprezentace všeho, co se po provedení akce nemění.
- Jednoduchý **axiom rámce**, který říká, co se nemění:

  At(o,x,s) $\wedge$ o$\neq$Agent $\wedge \neg$Holding(o,s) $\Rightarrow$ At(o,x,Result(Go(y,z),s))
  - pro F flexibilních predikátů a A akcí potřebujeme O(FA) axiomů rámce
  - To je hodně, zvlášť když si uvědomíme, že akce většinu predikátů nemění.



Problém rámce

efektivní reprezentace

Lze řešit problém rámce efektivněji (menší počet axiomů)?

- **Axiom následujícího stavu**

  $Poss(a,s) \Rightarrow$
  $(\text{fluent platí v } Result(a,s) \Leftrightarrow$
  $\text{fluent je efektem a} \vee (\text{fluent platí v s} \wedge \text{a fluent nemění}))$

  - dostaneme F axiomů (F je počet flexibilních predikátů) s celkovým počtem literálů O(AE) (A je počet akcí, E je počet efektů na akci)

  Příklady:

  $Poss(a,s) \Rightarrow$
  $(At(Agent,y,Result(a,s)) \Leftrightarrow a=Go(x,y) \vee (At(Agent,y,s) \wedge a \neq Go(y,z)))$

  $Poss(a,s) \Rightarrow$
  $(Holding(g,Result(a,s)) \Leftrightarrow a=Grab(g) \vee (Holding(g,s) \wedge a \neq Release(g)))$

  - **Pozor na implicitní efekty!**
    - Pokud agent něco drží a přesune se jinam, potom se tam přesune také dotyčný objekt.
    - **problém důsledku** (ramification problem)

    $Poss(a,s) \Rightarrow$
    $(At(o,y,Result(a,s)) \Leftrightarrow$
    $(a=Go(x,y) \wedge (o=Agent \vee Holding(o,s))) \vee$
    $(At(o,y,s) \wedge \neg\exists z\ (y \neq z \wedge a=Go(y,z) \wedge (o=Agent \vee Holding(o,s)))))$

Problém rámce

efektivní projekce

- Axiom následující stavu je pořád veliký, v průměru má O(AE/F) literálů.
  - Čas řešení projekční úlohy délky t (kam se dostaneme danou posloupností akcí) tedy záleží nejen na t, ale i na počtu akcí – O(AEt).
  - Pokud v každém kroku známe danou akci, nešlo by to rychleji, třeba O(Et)?
- **Klasický axiom následujícího stavu:**

  $Poss(a,s) \Rightarrow$
  $(F_i(Result(a,s)) \Leftrightarrow (a=A_1 \vee a=A_2 \vee \ldots) \vee (F_i(s) \wedge a \neq A_3 \wedge a \neq A_4 \ldots))$

  akce, které mají F_i jako svůj efekt

  akce, které mají $\neg F_i$ jako svůj efekt

- Můžeme zavést **pozitivní a negativní efekty** akcí
  - **PosEffect(a, F_i)** akce a způsobí, že F_i bude pravda
  - **NegEffect(a, F_i)** akce a způsobí, že F_i nebude pravda
- **Upravený axiom následujícího stavu:**

  $Poss(a,s) \Rightarrow (F_i(Result(a,s)) \Leftrightarrow PossEffect(a, F_i) \vee (F_i(s) \wedge \neg NegEffect(a,F_i)))$
  $PosEffect(A_1, F_i)$
  $PosEffect(A_2, F_i)$
  $NegEffect(A_3, F_i)$
  $NegEffect(A_4, F_i)$

Skryté předpoklady

Příklad:

- Předpokládejme, že máme k dispozici následující tvrzení:
  - *„V létě budou vyučovány kurzy CS101, CS102, CS106 a EE101“*
  - v řeči predikátové logiky máme fakta
    - Course(CS,101), Course(CS, 102), Course(CS,106), Course(EE,101)
- Kolik bude v létě vyučováno kurzů?
  - někde mezi jedním a nekonečnem!!

Proč?

- obecně předpokládáme, že poskytnutá informace je úplná, tj. že atomická tvrzení, která nejsou uvedena, nejsou pravdivá – **předpoklad uzavřeného světa** (CWA – Closed World Assumption)
- predikátová logika ale takový předpoklad nemá, bázi znalostí je potřeba zúplnit:

  $\text{Course}(d,n) \Leftrightarrow$
  $[d,n] = [CS,101] \vee [d,n] = [CS,102] \vee [d,n] = [CS,206] \vee [d,n] = [EE,101]$
- také jsme předpokládali, že různá jména reprezentují různé objekty – **předpoklad jednoznačnosti jmen** (UNA – Unique Name Assumption)
- opět je potřeba explicitně uvést, že se jedná o různé objekty
  - $[CS,101] \neq [CS,102]$, …