Umělá inteligence I

Roman Barták, KTIML

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

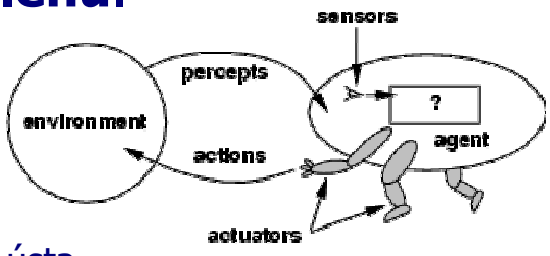
2

Inteligentní agenti

- Připomeňme, že z různých pohledů na UI bude tato přednáška pokrývat **racionální jednání**, tj.
  - zajímá nás výsledek řešení (ne postup řešení),
  - který by měl být co nejlepší (ne nutně lidský)
- Budeme konstruovat **racionální agenty**
  - Co je to **agent**?
  - A co je **racionální agent**?
  - V jakém **prostředí** se agent pohybuje?
  - Jaké **vlastnosti prostředí** nás zajímají?
  - Jaké **vnitřní struktury** agentů máme?

Agent

- **Agent** je cokoliv, co vnímá okolní **prostředí** prostřednictvím **senzorů** a ovlivňuje ho prostřednictvím **akčních členů.**

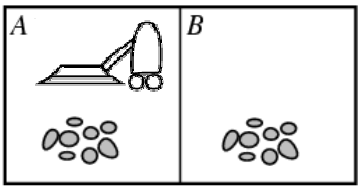


- **Příklady:**
  - člověk
    - oči, uši, nos, … → ruce, nohy, ústa,…
  - robot
    - kamera, IR detektor, … → paže, kola, …
  - software
    - klávesnice, pakety ze sítě … → obrazovka, pakety do sítě,…

Agent přesněji

- Agent přijímá **vjemy** a jeho chování je plně určeno posloupností všech vjemů, které kdy přijal.
- Formálně tedy můžeme každého agenta popsat **agentovou funkcí** (tabulkou):
  - V* → A, kde V: množina vjemů, A: množina akcí
  - Na základě chování agenta, můžeme sestavit jeho agentovou funkci (vnější charakteristika).
    - tedy pokud můžeme agenta „restartovat" a máme nekonečný prostor pro zápis tabulky a také hodně času
  - Tabulka je **abstraktní matematický popis** agenta.
- Interně ale bude chování agenta popsáno spíše **agentovým programem.**
  - Program je **konkrétní implementace** agentovy funkce.

Příklad agenta



Vysavač

- vjemy: místo (A,B)
  obsah (čisto,špína)
- akce: vysaj, doleva, doprava, nic

Agentova funkce:

seznam vjemů	akce
(A,čisto)	doprava
(A,špína)	vysaj
(B,čisto)	doleva
(B,špína)	vysaj
(A,čisto), (A,čisto)	doprava
…	

Agentův program:

if obsah=špína **then**
  vysaj
else if místo=A **then**
  doprava
else if místo=B **then**
  doleva

Míra výkonu

- **Jak správně vyplnit agentovu tabulku?** aneb **Co odlišujete dobrého agenta od špatného?**
- Potřebujeme měřit **míru výkonu** agenta, tj. jak úspěšné je jeho chování a to pokud možno **objektivně**.
- **Kdo určuje míru výkonu?**
  - tvůrce agenta
- **Jak určit míru výkonu?**
  - Podle toho, jak chceme aby vypadalo prostředí agenta, spíše než podle toho, jak si myslíme, že se má agent chovat.
- **Příklad** (vysavač)
  - míra výkonu: sesbírej co nejvíce špíny
  - možný výsledek: vysaj, vysyp, vysaj, vysyp, …
  - míra výkonu (lépe): co největší čistá plocha

Racionální agent

- **Racionální agent** je takový agent, který pro každou možnou posloupnost vjemů zvolí akci maximalizující očekávanou míru výkonu a to na základě údajů daných tou posloupností vjemů a vnitřních znalostí agenta.
- Pozor, neplést se vševědoucností!
  - agent maximalizuje **očekávanou** míru výstupu
  - vševědoucí maximalizuje **skutečnou** míru výstupu
- Racionální agent by měl být **autonomní** – nespoléhá jen na znalosti dané tvůrcem, ale měl by se učit, aby kompenzoval předchozí částečné nebo špatné znalosti.

Prostředí agenta

- Kromě **senzorů**, **akčních** členů a míry **výkonu** potřebujeme ještě **prostředí**, ve kterém se agent pohybuje.

Příklad: automatický řidič taxi

Agent	Výkon	Prostředí	Akční členy	Senzory
řidič taxi	bezpečnost rychlost legálnost pohodlí zisk	silnice doprava chodci zákazník	brzda volant plyn směrovka houkačka	kamera sonar rychloměr GPS

Vlastnosti prostředí

- **Plná (částečná) pozorovatelnost**
  - senzory poskytují úplný obraz prostředí
- **Deterministické (stochastické)**
  - další stav prostředí je plně určen současným stavem a akcí agenta
  - strategické = stav prostředí mění pouze další agenti
- **Episodické (sekvenční)**
  - chování agenta lze rozdělit do epizod obsahujících po jedné akci, které jsou na sobě ale nezávislé (akce závisí pouze na dané epizodě)
- **Statické (dynamické)**
  - statické prostředí se nemění zatímco agent přemýšlí
  - semi-dynamické = prostředí se s časem nemění, ale míra výkonu ano
- **Diskrétní (spojité)**
  - rozlišení dle stavu prostředí, měření času, schopností snímačů a akčních prvků
- **Jeden agent (multi-agentní)**
  - Kdo musí být agent?
    Objekt maximalizující míru kvality, které závisí na daném agentovi.
  - kompetitivní vs. kooperativní multi-agentní prostředí

Příklady prostředí

Prostředí	Pozorovatelnost	Determinismus	Epizodické	Statické	Diskrétní	Agenti
Křížovka	úplná	deterministické	sekvenční	statické	diskrétní	jeden
Šachy s hodinami	úplná	strategické	sekvenční	semi	diskrétní	multi
Taxi	částečná	stochastické	sekvenční	dynamické	spojité	multi
Analýza obrazů	úplná	deterministické	epizodické	semi	spojité	jeden

- **Nejjednodušší prostředí**
  - plně pozorovatelné, deterministické, episodické, statické, diskrétní a jeden agent
- **Nejsložitější prostředí (běžné reálné)**
  - částečně pozorovatelné, stochastické, sekvenční, dynamické, spojité a multi-agentní

Struktura agenta

agent = architektura + program

- **architektura** = výpočtové zařízení s fyzickými senzory a akčními členy
- **program** = implementace agentovy funkce
  - mapuje vjemy na akce
  - přesněji řečeno, vstupem je jeden vjem (jediné, co je přímo dostupné z prostředí prostřednictvím čidel)
  - pokud akce záleží na posloupnosti vjemů, musí si ji agent sám pamatovat
  - samozřejmě program musí být vhodný pro danou architekturu!

Table-driven

Tabulkový agent

- **Triviální agent, který udržuje seznam vjemů a používá ho jako index do tabulky s akcemi.**

function TABLE-DRIVEN_AGENT(*percept*) **returns** an action

static: *percepts*, a sequence initially empty
table, a table of actions, indexed by percept sequence

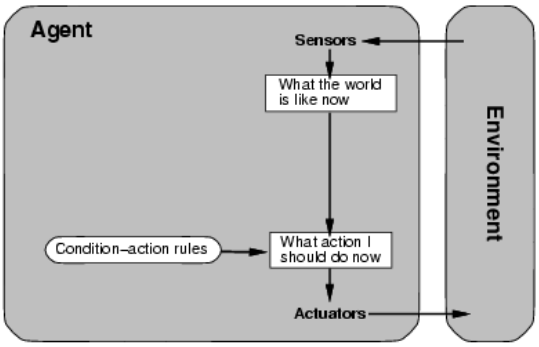
append *percept* to the end of *percepts*
action ← LOOKUP(*percepts*, *table*)
return *action*

Problémy:

- tabulka neúměrně velká (i pro agenta fungujícího omezený čas)
- tvůrce agenta nemá čas ji vyplnit
- agent se nemůže sám naučit celou správnou tabulku
- tvůrce agenta také neví jak tabulku vyplnit

Takto to asi nepůjde!

Agent s reflexy



- Akce vybrána pouze na základě současného vjemu.
- Implementováno formou **pravidel** podmínka-akce (**if** obsah=špína **then** vysaj)
- Velká **redukce** možných situací vjem-akce (max. na počet vjemů)

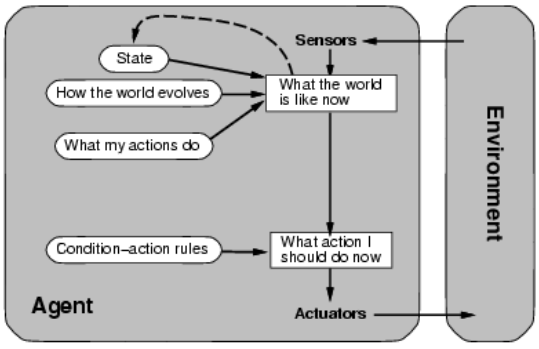
function SIMPLE-REFLEX-AGENT(*percept*) **returns** an action

static: *rules*, a set of condition-action rules

state ← INTERPRET-INPUT(*percept*)
rule ← RULE-MATCH(*state*, *rule*)
action ← RULE-ACTION[*rule*]
return *action*

- Plně funguje pouze pro plně pozorovatelné prostředí, jinak může cyklit.
- Řešením může být randomizace výběru akcí.

Agent s modelem



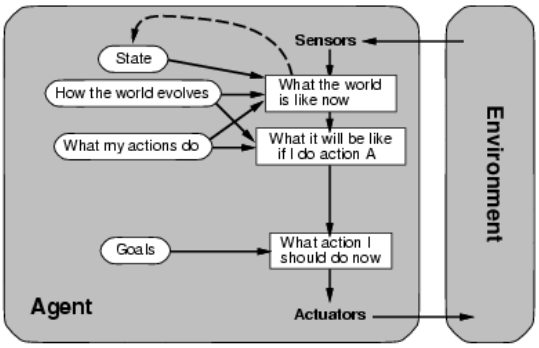
- Problémy s částečně pozorovatelným světem lze řešit udržováním vnitřního stavu agenta.
- Udržujeme dva typy informací
  - jak se svět vyvíjí (nezávisle na agentovi)
  - jak agentovy akce ovlivňují svět
- **Model světa**

function REFLEX-AGENT-WITH-STATE(*percept*) **returns** an action

static: *rules*, a set of condition-action rules
state, a description of the current world state
action, the most recent action.

state ← UPDATE-STATE(*state*, *action*, *percept*)
rule ← RULE-MATCH(*state*, *rule*)
action ← RULE-ACTION[*rule*]
return *action*

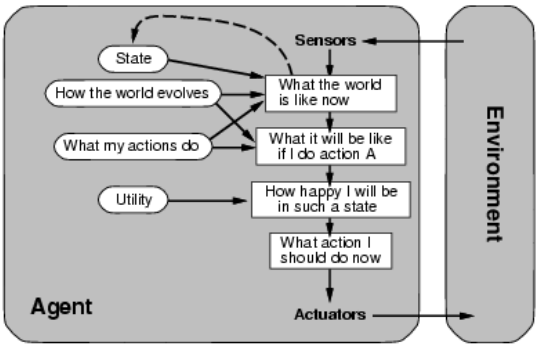
Agent s cílem



- Zatím jsme se starali pouze o výběr další akce, ale ten také záleží na tom, co chce agent dosáhnout.
- Agent potřebuje **cíl** říkající, jaké situace jsou žádoucí.

- Základní změna je zahrnutí **uvažování o budoucnosti.**
- Hledáním (delší) posloupnosti akcí pro dosažení cíle se zabývá **plánování.**
- Řízení cílem je (zpravidla) **méně efektivní** než řízení reflexem, je ale **více flexibilní.**

Agent se žádostmi

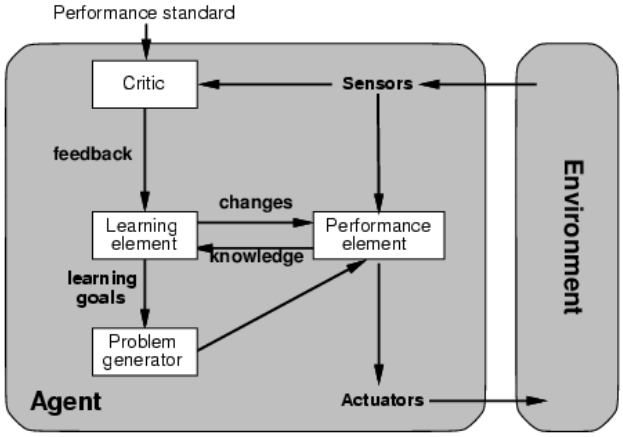


- Cíle sami o sobě nestačí pro „kvalitní" chování (cíl splněn/nesplněn).
- Často je potřeba vyjádřit míru splnění či dosažení cíle nebo vybrat mezi různými i konfliktními cíli.

- Je možné mapovat stavy (nebo jejich posloupnosti) na reálná čísla určující **míru „žádoucnosti"** dosažení daného stavu.
- Agent se potom při výběru akce řídí i tím, jak je případně dosažitelný stav žádoucí.

Agent učící se

- Zatím jsme se zabývali výběrem akcí, ale **odkud se berou programy pro výběr akcí**?
- Jedna z možností je vytvořit **stroje schopné se učit** a potom je učit místo podávání instrukcí.
- Agent potom může pracovat i v původně neznámém prostředí.
- Všechny dosud zmíněné struktury agentů lze rozšířit na učící se agenty přidáním komponent pro učení:



- **výkonný prvek**
  - původní agent, který vybírá akce
- **učící se prvek**
  - vylepšuje program původního agenta
- **kritika**
  - dává zpětnou vazbu, jak dobře se agent chová (senzory na to nestačí)
- **generátor problémů**
  - navrhuje akce, které by mohly vést k nové zkušenosti (trochu experimentování)