Racionální agenti a současné směřování umělé inteligence

Přednáška se zabývá pragmatickým zaměřením velké části současného výzkumu a aplikací umělé inteligentice na využití teoretických i empiricky získaných znalostí pro budování agentů.

Jan Žižka

Ústav informatiky
PEF, Mendelova universita v Brně
Zemědělská 1, 613 00 Brno

zizka@mendelu.cz

RACIONÁLNÍ AGENTI

- Umělá inteligence může být jako obor chápána různě, například:

  1. Systém *myslící jako člověk* – souvislost s psychologií a poznáváním
  2. Systém *chovající se jako člověk* – východisko je Turingův test
  3. Systém *myslící racionálně* – popis řešení logikou (sylogismus, aj.)
  4. Systém *chovající se racionálně* – poskytuje co nejlepší řešení

- Zde se budeme zabývat především agenty, kteří se chovají racionálně, neboli agenty, kteří se snaží vždy poskytnout *správné řešení*, tj. dělat *správné věci*.
- Racionálně se chovající (dále jen "racionální") agenti v současnosti představují preferovaný směr.
- Racionální agenti nezaručují dokonalou racionalitu jimi nalezeného řešení. To může být dáno např. složitostí prostředí, v němž působí (výpočetní složitost).
- Ideálem je vždy dosažení dokonalého racionálního agenta, ale je nutno mít na paměti, že v realitě mívá agent svou racionalitu omezenou, a to z nejrůznějších důvodů.

RACIONÁLNÍ AGENTI

- K základům umělé inteligence (UI) přispělo a přispívá mnoho různých oborů (filosofie, matematika, neurologie, psychologie, konstrukce počítačů, teorie řízení a kybernetika, lingvistika, aj.).
- Ekonomie od roku 1776 (Adam Smith) také slouží pro inspiraci rozvoje umělé inteligence (Smith tehdy publikoval ideu, že ekonomie se má považovat za vědu – vystupují zde agenti maximalizující svůj blahobyt):
  - Jak rozhodovat, aby byl maximalizován zisk?
  - Jak toho dosáhnout, když ostatní mohou postupovat jinak?
  - Jak toho dosáhnout, když ten zisk může být k dispozici až v daleké budoucnosti?
- Za zrod umělé inteligence se považuje rok 1956 (John McCarthy na universitě Princetown je údajně autorem termínu *artificial intelligence*, přestože vhodnějším názvem by snad bylo *computational rationality*).
- Umělá inteligence prošla obdobími jak nadneseného vyzvedávání, tak odmítání. Od roku 1987 je již považována za samostatný vědní obor.
- *Inteligentní agenti* se v UI objevují přibližně od roku 1995.

RACIONÁLNÍ AGENTI

- V současnosti oblast umělé inteligence zahrnuje mnoho nejrůznějších druhů aktivit, například:
  - Autonomní plánování a časové rozvržení činností
  - Hraní her (jako modelů vysoce složitých činností)
  - Samočinné řízení procesů
  - Diagnostika
  - Logistické plánování
  - Robotika
  - Porozumění přirozenému (lidskému) jazyku
  - A mnoho dalších
- Umělá inteligence se uplatňuje především tam, kde není možno využít exaktních matematických postupů vlivem složitosti problémů, a přesto je potřeba tyto problémy řešit.
- Je nutno tolerovat určitý stupeň nedokonalosti nalezených řešení.

RACIONÁLNÍ AGENTI

Inteligentní agenti

- Agentem zde rozumíme cokoliv, co vnímá své prostředí pomocí sensorů a působí na to prostředí pomocí efektorů.
- Člověk-agent z tohoto hlediska má oči, uši a další orgány jako sensory; dále má ruce, nohy, ústa a další části těla jako efektory.
- Robot-agent nahrazuje kamerami a infračervenými detektory sensory.
- Efektory jsou vlastně tvořeny různými motory.
- Softwarový agent pro vnímání a působení na prostředí disponuje bitovými řetězci.
- Cílem je návrh (umělých) agentů, kteří jsou inteligentně a výkonně schopni působit na své prostředí.

RACIONÁLNÍ AGENTI

Činnost agentů

- Za racionálního agenta budeme považovat takového, který provádí ***"správné věci"***.
- V prvním přiblížení je ***správná věc*** taková věc, která umožňuje agentovi nejlepší úspěch.
- Jak a kdy se měří ***výkonnost agenta***? Pro různé agenty neexistuje stejný způsob ***měření úspěšnosti***. Měření (a vyhodnocení) míry úspěšnosti není vhodné provádět subjektivně (tedy přímo agentem), např. agent to není schopen udělat, apod. Proto se používá objektivní míra úspěšnosti, kdy je agent pozorován např. námi z vnějšku v jeho prostředí.
- Např. (libovolný) agent, jehož úkolem je vysávat nečistotu z podlahy, může být jednoduše posuzován z hlediska množství odstraněné špíny během osmi hodin. Komplexnější posouzení může vzít do úvahy faktor spotřeby el. proudu, množství produkovaného hluku, apod.

RACIONÁLNÍ AGENTI

Činnost agentů

- Další otázkou je kdy agentovu výkonnost vyhodnocovat. Na začátku vysávání může některý agent vysávat usilovně a za hodinu polevit, jiný může pracovat rovnoměrně celou pracovní dobu, další např. celou dobu své existence. Tomu je nutno přizpůsobit měření.
- Je nutno také rozlišovat mezi ***racionalitou*** a ***vševědoucností*** (zde agent zná skutečný výsledek svých akcí a tomu může přizpůsobit svou činnost – to je však v realitě nemožné docílit). Racionalita je zaměřena na ***očekávaný*** úspěch na základě toho, co je agentem vnímáno (mnoho věcí nelze předvídat a vnímat).
- ***Racionalita*** závisí v libovolném čase na těchto čtyřech věcech:
  - Na měření výkonnosti, které definuje stupeň úspěchu.
  - Na všem, co agent vnímal do daného okamžiku (tzv. sekvence vnímání, což je kompletní historie agentova vnímání).
  - Na znalostech agenta o jeho prostředí.
  - Na akcích, které agent může provádět.

RACIONÁLNÍ AGENTI

Činnost agentů

- Definice ideálního racionálního agenta:

Pro jakoukoliv sekvenci vnímání, ideální racionální agent učiní vše, co se od něj očekává pro maximalizaci míry jeho výkonnosti, a to na základě evidence poskytnuté sekvencí vnímání a znalosti, kterou agent disponuje. ■

- Například před přechodem křižovatky by se měl robot rozhlédnout doleva a doprava; bez toho jeho (post mortem zkoumaná) sekvence vnímání neodhalí, že vkročil do silnice bez rozhlédnutí a srazilo ho auto – takový robot ovšem není racionální a jeho akce přejít křižovatku má nízkou míru výkonnosti.
- Důležitou součástí racionality (tj. “rozumnosti”) je získávání užitečné informace.
- Otázka: lze považovat normální hodinky za jednoduchého racionálního agenta? Proč ano či ne? (Vnímání, úprava času při přechodu do jiné časové zóny...)

RACIONÁLNÍ AGENTI

Ideální mapování ze sekvence vnímání do akcí

- Pokud tedy závisí chování agenta na jeho sekvenci vnímání do určitého okamžiku, lze každého agenta popsat pomocí tabulky akcí, které vykonává jako odezvu na každou (doposud) možnou sekvenci vnímání (pro mnoho agentů by byl seznam velice dlouhý).
- Tento seznam budeme nazývat *mapováním ze sekvencí vnímání do akcí*. Mapování popisuje agenta a ideální mapování ideálního agenta.
- Specifikace akcí, které má agent provádět jako odezvu na dané sekvence vnímání, dává možnost vzniku *ideálního agenta*.
- Není ovšem nutno vytvářet explicitní tabulku s buňkou pro každou možnou sekvenci vnímání – lze často definovat specifikaci bez vyčerpávajícího výčtu. (Např. výpočet druhé mocniny na kalkulačce nepotřebuje znát hodnotu pro každou možnou kombinaci stlačení tlačítek – mapování lze vytvořit např. metodou Newtona.)

RACIONÁLNÍ AGENTI

Autonomie

- Ideální racionální agent by měl obsahovat nějakou “vestavěnou” znalost.
- Pokud takovou znalost má a je založen ***výhradně*** na ní, pak ovšem postrádá autonomii (jeho akce pak ***nezávisí na vnímání***).
- ***Autonomie systému*** je dána vlastní zkušeností či znalostí agenta. Nelze ovšem např. požadovat kompletní autonomii agenta na slovo “jdi”, a je nutno se vyhnout jeho nahodilé činnosti.
- Agent by měl mít schopnost ***se učit*** (získávat znalost).
- Skutečně autonomní inteligentní agent musí být schopen úspěšné činnosti v širokém rozsahu ***různých prostředí***, a to za předpokladu, že má dost času k ***adaptaci***.

RACIONÁLNÍ AGENTI

Struktura inteligentního agenta

- Návrh programu pro umělého agenta je úkolem moderní umělé inteligence. ***Programem*** zde rozumíme funkci, která implementuje agentovo mapování z vnímání na akce. Program obvykle běží na určitém typu počítače, tj. vyžaduje se určitá architektura:

  agent = architektura + program.

- Před návrhem a implementací programu je nutno dobře znát možná vnímání a akce. Rovněž je nutno znát očekávanou výkonnost a cíle činnosti agenta, a také v jakém ***prostředí*** bude agent aktivní.

- Některá reálná prostředí mohou být ve skutečnosti velmi jednoduchá (robot pro třídění součástek). Oproti tomu někteří softwaroví agenti (softbot = software robot) existují ve složitých, neomezených doménách (např. softbot navržený pro létání na leteckém simulátoru pro Boening 747, softbot pro prohlížení on-line zpráv a výběr zajímavých pro zákazníky, a podobně).

RACIONÁLNÍ AGENTI

Příklady typů agentů

TYP AGENTA	VNÍMÁNÍ	AKCE	CÍLE	PROSTŘEDÍ
medicínský diagnostický systém	symptomy, nálezy, pacientovy odpovědi	otázky, testy, léčba	zdravý pacient, minimální náklady	pacient, nemocnice
systém analýzy satelitních snímků	pixely různé intensity, barva	tisk kategorie záběru	správná kategorizace	obrazy z obíhajícího satelitu
robot sbírající součástky	pixely různé intensity	zvednutí součástky a její uložení do přihrádky	umístění součástek do správných přihrádek	pás přepravující součástky
řízení čističky	teplota, tlak	otevření, zavření ventilů; přizpůsobení teploty	maximalizace čistoty, vydatnosti, bezpečnosti	čistička
interaktivní učitel angličtiny	napsaná slova	tisk cvičení, nápověda, opravy	maximalizace studentových výsledků v testech	soubor studentů

RACIONÁLNÍ AGENTI

Programy agentů

- Jednoduchá základní kostra vnímá prostředí a generuje akce, např.:

```
function Skeleton-Agent(percept) returns action
  static: memory // agentova paměť světa
  memory ← Update-Memory(memory, percept)
  action ← Choose-Best-Action(memory)
  memory ← Update-Memory(memory, action)
return action
```

- Po každém vyvolání funkce je agentova paměť aktualizována vzhledem k novému vjemu (vstup funkce).
- Je vybrána nejlepší akce a skutečnost o výběru akce je rovněž uložena do paměti.
- Paměť setrvává mezi jednotlivými vyvoláními funkce.

RACIONÁLNÍ AGENTI

Stačí pouze hledat odpovědi?

- K napsání jednoduchého agentova programu stačí např. vytvořit vyhledávací tabulku:

```
function Table-Driven-Agent(percept) returns action
  static: percepts // na počátku prázdná sekvence
          table    // tabulka indexovaná
                   sekvencemi
                   // vnímání, na počátku daná
  append percept to the end of percepts
  action ← Lookup(percepts, table)
return action
```

- Agent je založen na předem stanovené tabulce. Hlídá si sekvenci vnímání a pouze vyhledá nejlepší akci. V paměti se uchovává celková sekvence vnímání, která slouží jako index do tabulky, která obsahuje příslušné akce pro všechny možné sekvence.

RACIONÁLNÍ AGENTI

Stačí pouze hledat odpovědi?

- Metoda hledání odpovědí pro vjemy ***může selhat***:
  - Tabulka potřebná pro jednoduchého agenta, jehož úkolem je pouze hrát šachy, by potřebovala přibližně 35^{100} buněk.
  - Pro návrháře by to znamenalo velice dlouhý čas pro vytvoření tabulky.
  - Agent nemá žádnou autonomii, protože výpočet (výběr) nejlepší akce je zcela zabudován. Se změnou prostředí nějakým neočekávaným způsobem by mohl agent zcela zhavarovat.
  - I kdybychom agenta vybavili nějakým učícím mechanismem, aby měl nějaký stupeň autonomie, trvalo by nesmírně dlouho se naučit správné hodnoty pro všechny buňky tabulky.
- Přesto ukázaná funkce `Table-Driven-Agent` implementuje požadované mapování. Je ovšem nutno pochopit, proč usuzující agent – na rozdíl od pouze vyhledávajícího – může být lepší při vyhnutí se uvedeným čtyřem nedostatkům.

RACIONÁLNÍ AGENTI

Stačí pouze hledat odpovědi?

- Příklad:

  Předpokládejme agenta – umělého řidiče taxi. Úloha řízení není uzavřená: neexistuje limit pro možné nové kombinace okolností, které mohou nastat.

TYP AGENTA	VJEMY	AKCE	CÍLE	PROSTŘEDÍ
řidič taxíku	kamery, rychloměr, GPS, sonar, mikrofon	točení volantem, akcelerace, brzdění, mluvení k pasažérovi	bezpečná, rychlá, předpisová a pohodlná jízda, maximální zisk	silnice, další doprava, chodci, zákazníci

RACIONÁLNÍ AGENTI

Stačí pouze hledat odpovědi?

- Příklad (pokračování):

- Polohu, účast ostatních objektů na silnici, rychlost... potřebuje taxi znát.
- To lze získat z vjemů poskytovaných příslušnými zařízeními (kamery, rychloměr, atd.).
- Znalost projíždění zatáčkami vyžaduje akcelerometr.
- Stav vozidla je dán senzory v motoru, v elektrické výbavě...
- Akce jsou více-méně tytéž jako u řidiče-člověka:
  - kontrola motoru plynovým pedálem
  - kontrola řízení volantem a brzdami
- Navíc komunikace s dalšími vozidly, syntezátor hlasu pro mluvení na pasažéry.

RACIONÁLNÍ AGENTI

Stačí pouze hledat odpovědi?

- Příklad (pokračování):

  Míra výkonnosti:
  - dojetí na správné místo
  - minimalizace spotřeby a opotřebení
  - minimalizace času a ceny cesty
  - minimalizace narušení dopravních pravidel a omezování dalších řidičů
  - maximalizace bezpečnosti a pohodlí pasažérů
  - maximalizace zisku – je zřejmé, že některé požadavky jsou konfliktní, takže je nutno nalézt kompromisy

  Prostředí:
  - může být na lokálních silnicích, na dálnicích?
  - v oblasti se sněhem nebo bez?
  - jízda vždy vpravo nebo i vlevo (Británie, Japonsko)?

- Čím větší omezení, tím snadnější návrh.

RACIONÁLNÍ AGENTI

Stačí pouze hledat odpovědi?

- Nyní je nutno rozhodnout, jak vytvořit reálný program pro mapování vjemů na akce. Různá hlediska řízení mohou vyžadovat různé typy programů – zde budeme uvažovat čtyři typy:
  - Jednoduší reflexní agenti.
  - Agenti sledující svět.
  - Agenti zaměření na cíl.
  - Užitkově zaměření agenti.

RACIONÁLNÍ AGENTI

Stačí pouze hledat odpovědi?

Jednoduší reflexní agenti

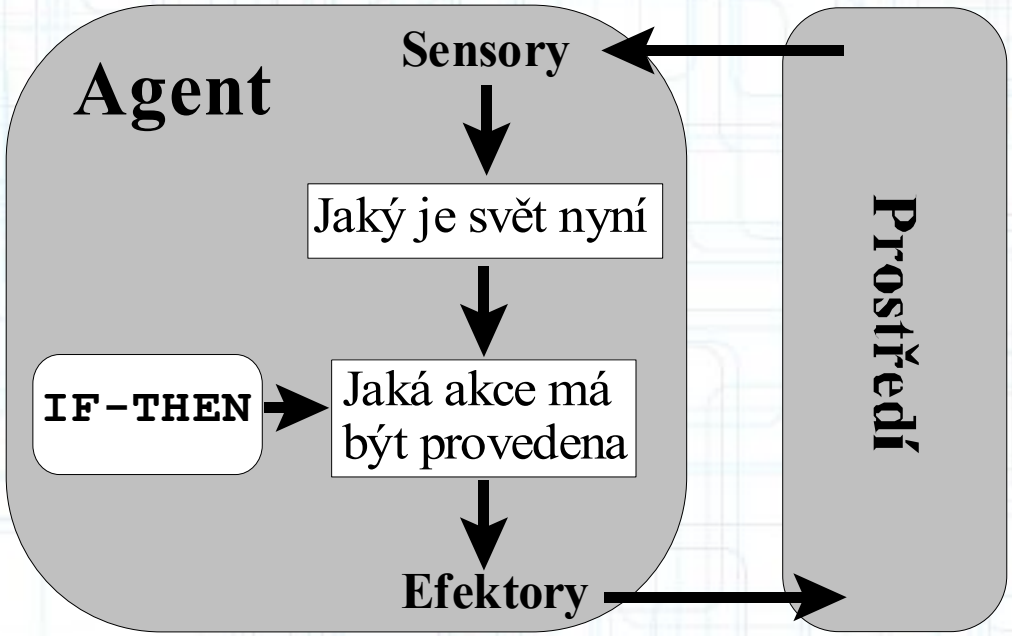
- Explicitní vyhledávací tabulky nepřicházejí do úvahy: visuální vstup z jednoduché kamery přichází v množství 50 MB/s (25 obrázků za vteřinu, 1000x1000 pixelů s 8 bity na barvy a 8 bity informace o intensitě).
- Tabulka by pak musela mít pro jednu hodinu $2^{60x60x50M}$ buněk, což je zjevně přílišný paměťový požadavek.Je ovšem možné sloučit části tabulky – existují některé obecně se vyskytující vstup/výstupní asociace, např. brzdí-li automobil před námi a jeho brzdová světla se rozsvítí, naše auto by mělo rovněž začít brzdit.
- To vede k možnosti použití pravidel typu IF-THEN (pravidlo typu podmínka-akce).

RACIONÁLNÍ AGENTI

Stačí pouze hledat odpovědi?

Jednoduší reflexní agenti (pokračování)

- Lidé takovýchto vstup/výstupních asociací rovněž používají mnoho (některá pravidla se naučí, jiná pocházejí z reflexů). Obrázek ukazuje strukturu jednoduchého reflexního agenta ve schematické formě, a ukazuje, jak agentovi pravidla umožňují propojit vjemy s akcemi:



RACIONÁLNÍ AGENTI

Stačí pouze hledat odpovědi?

Jednoduší reflexní agenti (pokračování)

- Jednoduchý reflexní agent hledá pravidlo, jehož podmínka odpovídá současné situaci (určené vjemem) a pak provede akci s tím pravidlem asociovanou:

```
function Simple-Reflex-Agent(percept) returns action
        static:  rules // soubor IF-THEN pravidel

        state ← Interpret-Input (percept)

        rule ← Rule-Match (state, rules)

        action ← Rule-Action[rule]

   return action
```

RACIONÁLNÍ AGENTI

Stačí pouze hledat odpovědi?

Agenti sledující svět

- Jednoduchý reflexní agent pracuje pouze tehdy, když správné rozhodnutí může být vytvořeno na základě okamžitého vjemu.
- Pokud auto vpředu je současný model, je možné z jednoduchého obrázku z kamery poznat, zda svítí brzdová světla či ne. Starší modely mohou mít jiná uspořádání a jejich brzdění nemusí být detekováno. Proto musí agent-řidič udržovat určitý druh vnitřního stavu pro správný výběr akce.
- Zde není vnitřní stav příliš rozsáhlý – potřebuje jen předchozí snímek kamery k určení stavů, kdy se co rozsvítí nebo nějak změní na autě vpředu při jeho brzdění, ukazování změny směru jízdy (blinkr, nebo mechanická ručka, ...), apod.

RACIONÁLNÍ AGENTI

Stačí pouze hledat odpovědi?

Agenti sledující svět (pokračování)

- Obdobně je nutno uvážit, že senzory nemusejí vždy poskytnout kompletní informaci o stavu na silnici, v jízdních pruzích apod.
- V takových případech musí agent udržovat nějakou vnitřní stavovou informaci k odlišení stavů světa, které generují stejný perceptuální vstup, ale jsou výrazně odlišné (tj. jsou zapotřebí výrazně odlišné akce).
- Např. (1) nejede-li ve vedlejším pruhu auto nebo (2) není-li vidět, generuje tentýž vstup, avšak ve druhém případě je správnou akcí pokračování v původním pruhu, v prvním případě možnost přejetí do vedlejšího.

RACIONÁLNÍ AGENTI

Stačí pouze hledat odpovědi?

Agenti sledující svět (pokračování)

- Aktualizace vnitřní stavové informace vyžaduje dva druhy znalosti zakódované do agentova programu:

  1. Je nutno znát, jak se svět vyvíjí nezávisle na agentovi – např. předjíždějící automobil bude obecně blíže za námi než před okamžikem.

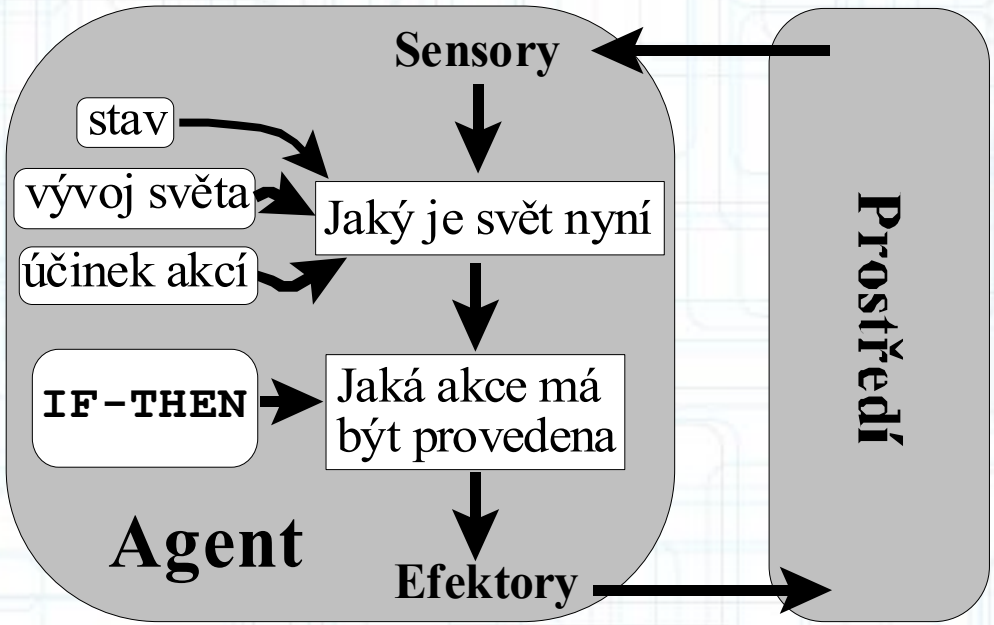
  2. Je nutno znát, jak vlastní agentovy akce ovlivňují svět – přejíždí-li agent do pravého pruhu, vzniká po něm přinejmenším dočasně mezera v původním pruhu, nebo že po cca pěti minutách jízdy směrem na sever se nachází cca pět kilometrů severně od místa, kde byl před pěti minutami (věci zdánlivě triviální pro člověka, ale pro stroj-agenta je nutno takto uvažovat).

RACIONÁLNÍ AGENTI

Stačí pouze hledat odpovědi?

Agenti sledující svět (pokračování)

- Schema reflexního agenta s interním stavem:



RACIONÁLNÍ AGENTI

Stačí pouze hledat odpovědi?

Agenti sledující svět (pokračování)

- Příslušná funkce v pseudokódu:

```
function Reflex-Agent-With-State(percept) returns
                                                action
static: rules // soubor IF-THEN pravidel
        state // popis stavu současného světa

state ← Update-State (state, percept)
rule ← Rule-Match (state, rules)
action ← Rule-Action[rule]
state ← Update-State (state, action)
return action
```

RACIONÁLNÍ AGENTI

Stačí pouze hledat odpovědi?

Agenti zaměření na cíl

- Znalost okamžitého stavu prostředí nemusí ve všech případech postačovat k rozhodnutí o činnosti. Na křižovatce může taxi jet doleva, doprava, nebo rovně. Správné rozhodnutí závisí na tom, jaký je ***cíl*** jízdy.
- Agent tedy potřebuje ještě informaci o cíli jízdy. Agentův program pak může zkombinovat tuto informaci s informací o výsledcích možných akcí (tatáž informace byla použita k aktualizaci vnitřního stavu u reflexního agenta) k rozhodnutí o ***výběru správné akce k dosažení cíle***.
- Někdy je to jednoduché (jedna akce), jindy složité (dlouhá posloupnost zatáčení apod. pro nalezení cesty k dosažení cíle).
- Umělá inteligence používá různé metody pro vyhledávání a plánování k tomuto účelu.

RACIONÁLNÍ AGENTI

Stačí pouze hledat odpovědi?

Agenti zaměření na cíl (pokračování)

- Je dobré si povšimnout, že tento typ rozhodování se fundamentálně liší od používání pravidel typu IF-THEN v tom, že zahrnuje ***úvahy o budoucnosti***: “Co se stane když udělám to-a-to?” a “Pomůže mi to nějak?”.
- U návrhů reflexních agentů se tato informace explicitně nepoužívá, protože návrhář předem stanovil správné akce pro různé případy.
- ***Reflexní agent*** brzdí, když před sebou uvidí svítit brzdová světla.
- ***Cílově zaměřený agent*** v principu může uvažovat o tom, že pokud má auto před ním rozsvícená brzdová světla, tak zpomalí. Z toho, jak se obvykle svět vyvíjí, plyne, že jediná akce k zamezení nárazu (dosažení cíle nesrazit se) je brzdit.

RACIONÁLNÍ AGENTI

Stačí pouze hledat odpovědi?

Agenti zaměření na cíl (pokračování)

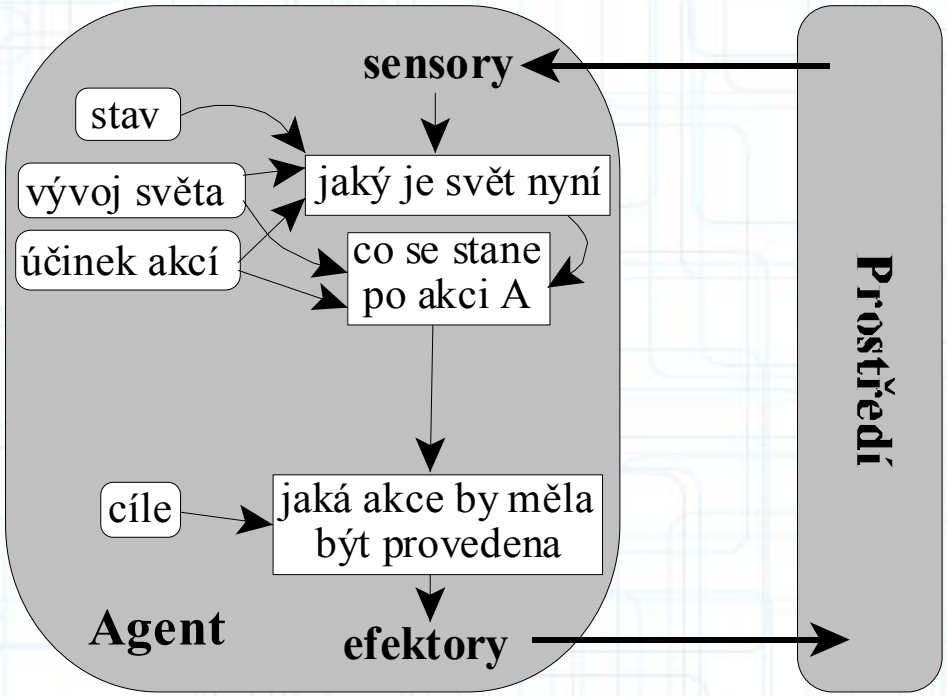
- Cílově zaměřený agent se může zdát méně efektivní, ale je daleko ***flexibilnější***.
- Začne-li např. pršet, ***cílový agent*** může aktualizovat svou znalost o efektivnosti jeho brzd – to může automaticky ovlivnit změny relevantních chování, aby se vyhovělo novým podmínkám.
- U ***reflexního agenta*** by bylo nutno přepsat velké množství pravidel typu podmínka-akce.
- Dále je ***cílově zaměřený agent*** také mnohem ***flexibilnější vzhledem k dosažení různých cílů***. Jednoduchou specifikací nového cíle dojezdu lze získat agenta s novým/aktualizovaným chováním. Pravidla reflexního agenta, kdy zatočit a kdy jet přímo, fungují pouze pro ***jeden*** cíl dojezdu. Pro dojezd jinam musí být všechna vyměněna.

RACIONÁLNÍ AGENTI

Stačí pouze hledat odpovědi?

Agenti zaměření na cíl (pokračování)

- Cílově zaměřený agent:



RACIONÁLNÍ AGENTI

Stačí pouze hledat odpovědi?

Užitkově zaměření agenti

- Cíle samy o sobě nepostačují ke generování vysoce kvalitního chování agenta. Např. existuje mnoho sekvencí akcí, které dostanou taxík do jeho určeného cíle – tím je tedy dosaženo cíle.
- Na druhé straně lze cíl ovšem dosáhnout za různých okolností: rychleji, bezpečněji, spolehlivěji, levněji, apod.
- Cíle pouze poskytují ***hrubé rozlišení*** mezi přínosnými a nepřínosnými stavy.
- ***Obecnější míra výkonnosti*** by měla poskytnout exaktní srovnání různých stavů světa (nebo sekvencí stavů) podle toho, do jaké míry jsou pro agenta přínosné, pokud jich dosáhne. (Pro ***přínosnost*** se používá termín ***agentův užitek***.)

RACIONÁLNÍ AGENTI

Stačí pouze hledat odpovědi?

Užitkově zaměření agenti (pokračování)

- ***Užitek*** je tedy funkce, která mapuje určitý stav na reálné číslo, které dále slouží pro určení asociovaného stupně přínosnosti. Kompletní specifikace funkce užitku umožňuje racionální rozhodování ve dvou případech, nastanou-li nějaké potíže s dosažením cílů:

  1. Pokud existují konfliktní cíle, pak pouze některé mohou být dosaženy (např. rychlost a bezpečnost). Funkce užitku zde stanoví vhodný ***kompromis***.

  2. Pokud existuje několik cílů, na jejichž splnění se agent může zaměřit a žádného z nich nelze dosáhnout s jistotou, pak užitečnost poskytuje způsob jak přiřadit váhu pravděpodobnosti úspěchu vzhledem k důležitosti dosažení cílů.

RACIONÁLNÍ AGENTI

Stačí pouze hledat odpovědi?

Užitkově zaměření agenti (pokračování)

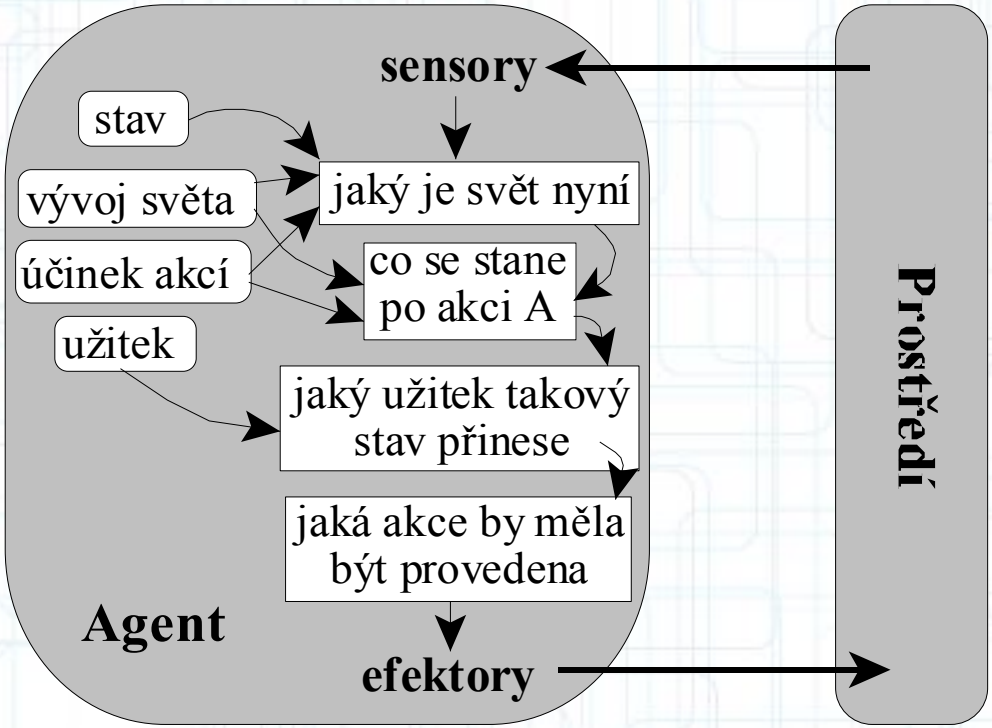
- Agent, který vlastní explicitní funkci užitku, může vytvářet racionální rozhodnutí, ale musí porovnávat potenciálně dosažitelné užitky, k nimž vedou různé posloupnosti akcí.
- Cíle, i když jsou z tohoto hlediska hrubější, umožňují agentovi výběr akce přímo, a to za předpokladu, že akce vyhovují pro dosažení cíle.
- Existují také případy, kdy lze převést funkci užitku na soubor cílů takovým způsobem, že cílově zaměřený agent při použití těchto cílů dosáhne stejného rozhodnutí o činnosti jako užitkově zaměřený agent používající danou funkci.
- Jiným příkladem užitkově zaměřeného agenta je např. agent hrající nějakou hru (šachy), který musí vytvářet značně “jemná” rozlišování mezi různými pozicemi vznikajícími na šachovnici.

RACIONÁLNÍ AGENTI

Stačí pouze hledat odpovědi?

Užitkově zaměření agenti (pokračování)

- Užitkově zaměřený agent:



RACIONÁLNÍ AGENTI

Prostředí

Vlastnosti prostředí

- Prostředí má několik charakteristik. Základní odlišnosti:

  - ***Přístupné a nepřístupné***: zda agentovy sensory umožňují poskytnout kompletní informaci o stavu prostředí. Pokud ano, je prostředí přístupné, jinak je nepřístupné. Sensory by měly umožnit získat údaje relevantní pro výběr akce. Přístupné prostředí má tu výhodu, že si agent nemusí udržovat vnitřní stavy sledovaného světa.

  - ***Deterministické a nedeterministické***: je-li následující stav prostředí zcela určen současným stavem a akcemi zvolenými agentem, pak se jedná o deterministické prostředí. V principu nemá agent starosti s nejistotou v přístupném a deterministickém prostředí. Zda je prostředí deterministické se obvykle musí určit přímo z hlediska agenta.

RACIONÁLNÍ AGENTI

Prostředí

Vlastnosti prostředí (pokračování)

- ***Episodické a neepisodické***: v episodickém prostředí je agentova znalost/zkušenost rozdělena mexi "episody". Každá episoda se skládá z agentových vnímání následovaných akcemi. Kvalita agentovy akce závisí pouze na dané episodě, protože následné episody na předcházejicích nezávisejí co do jejich vlastní kvality. Episodická prostředí mají výhodu v tom, že agent nemusí myslet dopředu.

- ***Statické a dynamické***: pokud se prostředí mění během agentova uvažování, pak je prostředí z jeho hlediska dynamické, jinak je statické. Statická prostředí mají výhodu v tom, že během svého uvažování nemusí agent sledovat svět, a také se nemusí zabývat časem.

- ***Semidynamické***: pokud se prostředí nemění v čase, ale agentova výkonnost na čase záleží, mluvíme o semidynamickém prostředí.

RACIONÁLNÍ AGENTI

Prostředí

Vlastnosti prostředí (pokračování)

- ***Diskrétní a kontinuální***: existuje-li omezené množství odlišných a jasně určených vnímání a akcí, pak se jedná o diskrétní prostředí (např. šachy jsou diskrétní prostředí – existuje pevné množství možných tahů v každé pozici). Řízení taxíku je kontinuální – rychlost a poloha taxíku včetně ostatních vozidel se mění v intervalu spojitých hodnot. *Pozn*.: Při dostatečně jemné úrovni granularity může dokonce i prostředí taxíku být diskrétní, protože obraz kamery je digitalizován na diskrétní hodnoty pixelů, ale smysluplný program agenta je normálně na abstraktnější úrovni, tj. na úrovni, kde je granularita považována za spojitou záležitost.

- Zjevně ***nejobtížnější případ*** nastane, když je prostředí nepřístupné, neepisodické a dynamické; obdobně je to i v případě prostředí dynamického. Rovněž se ukazuje, že většina reálných situací je tak složitá, že to, zda je prostředí doopravdy deterministické, je záležitost sporná a pro praktické účely se zachází se situací jako s nedeterministickou.

RACIONÁLNÍ AGENTI

Prostředí

Vlastnosti prostředí (pokračování)

Následující tabulka ukazuje některé známější typy prostředí:

PROSTŘEDÍ	přístupné	determinis-tické	episodické	statické	diskrétní
šachy s hodinami	ano	ano	ne	semi	ano
šachy bez hodin	ano	ano	ne	ano	ano
poker	ne	ne	ne	ano	ano
backgammon (vrhcáby)	ano	ne	ne	ano	ano
řízení taxi	ne	ne	ne	ne	ne
medicínská diagnostika	ne	ne	ne	ne	ne
analýza obrazů	ano	ano	ano	semi	ne

RACIONÁLNÍ AGENTI

Prostředí

Vlastnosti prostředí (pokračování)

Následující tabulka ukazuje některé známější typy prostředí (pokračování):

PROSTŘEDÍ	přístupné	determinis-tické	episodické	statické	diskrétní
robot třídící součástky	ne	ne	ano	ne	ne
řízení čističky	ne	ne	ne	ne	ne
interaktivní učitel angličtiny	ne	ne	ne	ne	ano

RACIONÁLNÍ AGENTI

Prostředí

Vlastnosti prostředí (pokračování)

- Hodnoty v tabulce mohou ovšem záviset na tom, jak je vytvořen pojem (tzv. konceptualizace) prostředí a agentů.
- Poker je deterministický, pokud si agent může udržovat údaje o sledování pořadí karet v balíčku, jinak bude poker nedeterministický.
- Obdobně může dojít k tomu, že prostředí je episodické na vyšší úrovni, než se odehrávají agentovy jednotlivé akce.
- Např. šachový turnaj se skládá z posloupnosti her (partií). Každá partie je episoda, protože přínos tahů, provedených agentem v jedné partii, pro celkovou úroveň agentovy výkonnosti není ovlivněn tahy z příští partie. Na druhé straně je nutno uvažovat fakt, že během jedné partie jsou jednotlivé tahy spolu určitým způsobem provázány, takže agent je musí promýšlet v nějakém počtu dopředu.

RACIONÁLNÍ AGENTI

Prostředí

Programy pro prostředí

- Obecný program pro prostředí ilustruje základní vztah mezi agenty a prostředími:

```
procedure Run-Environment(state, Update-Fn, agents,
                          termination)
   inputs: state        // počáteční stav prostředí
           Update-Fn    // funkce pro modifikaci prostředí
           agents       // soubor agentů
           termination // predikát k otestování na závěr
   repeat
     for each agent in agents do
       Percept[agent] ← Get-Percept(agent, state)
     end
     for each agent in agents do
       Action[agent] ← Program[agent]Percept([agent])
     end
     state ← Update-Fn (actions, agents, state)
   until termination(state)
```

RACIONÁLNÍ AGENTI

Prostředí

Programy pro prostředí (pokračování)

- Uvedený základní program pracuje v principu tak, že předává každému agentovi jeho vjemy (percepce), od každého agenta získává jeho akci, a pak aktualizuje prostředí.
- *Simulátory prostředí* mohou být založeny na uvedené programové kostře: simulátor dostane jako vstup jednoho (či více) agenta a zařídí, aby opakovaně každý agent obdržel správné vjemy; poté získá akce agentů. Následně simulátor aktualizuje prostředí na základě akcí, případně vlivu dalších dynamických procesů (které však nejsou agenty, např. déšť).
- Prostředí je tedy definováno počátečním stavem a aktualizační funkcí.
- (Pozn.: Každý agent pracující v simulovaném prostředí by samozřejmě měl fungovat i v reálném prostředí, které poskytuje tytéž druhy vjemů a přijímá tytéž druhy akcí.)

RACIONÁLNÍ AGENTI

Prostředí

Programy pro prostředí (pokračování)

- Procedura *Run-Environment* musí korektním způsobem testovat agenty v prostředí. Pro některé druhy agentů (např. takové, kteří se účastní dialogu v přirozeném jazyce) může postačovat pouhé sledování jejich chování.
- Pro získání detailnějších informací o agentově výkonnosti se začlenuje nějaký kód pro měření výkonnosti.
- Může to být např. funkce *Run-Eval-Environment*: měří výkonnost každého agenta a vrací seznam výsledných skóre.
- Proměnná *score* obsahuje sledování skóre pro každého agenta (viz následující funkci *Run-Eval-Environment*):

RACIONÁLNÍ AGENTI

Prostředí

Programy pro prostředí (pokračování)

```
function Run-Eval-Environment(state, Update-Fn, agents,
         termination, Performance-Fn) returns scores
    local variables: scores // vektor daný počtem agentů
                            // na počátku vše 0

    repeat
      for each agent in agents do
          Percept[agent] ← Get-Percept(agent, state)
      end
      for each agent in agents do
          Action[agent] ← Program[agent]Percept([agent])
      end
      state ← Update-Fn(actions, agents, state)
      scores ← Performance-Fn(scores, agents, state)
    until termination(state)
return scores
```

RACIONÁLNÍ AGENTI

Prostředí

Programy pro prostředí (pokračování)

- Měření výkonnosti obecně může záviset na celkové posloupnosti stavů prostředí generovaných během činnosti programu.
- Obvykle se však používá jednoduchá akumulace (součet, výpočet průměru, nebo použití maxima). Např. uvážíme-li měření výkonnosti agenta vysávajícího podlahy a použijeme-li jako míru celkové množství odstraněné špíny, pak *scores* jednoduše udržuje hodnotu o tom, kolik špíny již bylo doposud vysáto.
- *Run-Eval-Environment* vrací míru výkonnosti pro jedno prostředí, definované jedním počátečním stavem a konkrétní aktualizační funkcí.

RACIONÁLNÍ AGENTI

Prostředí

Programy pro prostředí (pokračování)

- Agent je však obvykle navržen pro práci *ve třídě* prostředí, čímž se rozumí celý soubor různých prostředí.
- Například se může jednat o šachový program, který hraje proti širokému souboru lidských a strojových protihráčů. Pokud bychom takový program navrhli proti jedinému protihráči, pak by mohlo dojít k tomu, že se sice využijí konkrétní slabosti daného protivníka, ale nemuseli bychom získat program hrající obecně dobře (viz *Deep Blue* pro Garry Kasparova).
- Z toho plyne zřejmý požadavek: pro měření výkonnosti agenta je zapotřebí *generátor prostředí*, který (s určitou pravděpodobností) vybírá konkrétní prostředí, v němž je agent testován. Nás pak zajímá *průměrná agentova výkonnost přes celou třídu prostředí*.

RACIONÁLNÍ AGENTI

Prostředí

Programy pro prostředí (pokračování)

- Je nutno ovšem při realizaci simulátoru prostředí dbát na rozdíl mezi stavovou proměnnou simulátoru prostředí a stavovou proměnnou v agentovi:
- Agent nesmí vidět stavovou proměnnou *simulátoru* prostředí.
- *Agentova stavová proměnná* může pouze nabývat hodnot na základě *agentových* vjemů, bez kompletního přístupu k informacím.

RACIONÁLNÍ AGENTI

Prostředí

Příklad prostředí a agentů ve světě vysávání špíny

Uvedený svět lze popsat následovně:

- ***Vjemy***: každý vysavačový agent dostává tříprvkový vektor vjemů pro každé kolo vysávání. První element (z dotykového sensoru) dává 1 pokud stroj do něčeho vrazí, jinak 0. Druhý element (fotosensor na spodku stroje) dává 1 pokud je zjištěno smetí, jinak 0. Třetí element dává 1 pokud je agent ve výchozím místě ("doma"), jinak 0.
- ***Akce***: pět akcí je k dispozici – jdi vpřed, otoč se doprava o 90°, otoč se doleva o 90°, vysaj smetí, vypni se.
- ***Cíle***: Cílem každého agenta je odstranit smetí a vrátit se domů. Exaktnější zadání: 100 bodů za každý vysátý kus smetí, -1 bod za každou akci, -1000 bodů pokud se agent vypne jinde než doma.
- ***Prostředí***: je dáno mřížkou se čtverci. Některé čtverce obsahují překážky (zdi, nábytek, apod.), jiné obsahují volný prostor. Některé volné čtverce obsahují smetí. Každá akce "jdi vpřed" je přesun na další čtverec, pokud tam není překážka – v tom případě agent zůstane stát v původním čtverci, ale dotykový senzor je nadále aktivní. "Vysaj smetí" vždy odstraní detekovaný kus smetí. "Vypni se" vždy zcela ukončí simulaci.

RACIONÁLNÍ AGENTI

Prostředí

Příklad prostředí a agentů ve světě vysávání špíny (pokračování)

Složitost prostředí lze měnit ve třech směrech:

- ***Tvar místnosti***: v nejjednodušším případě má místnost *n* × *n* čtverců (pro nějaké pevné *n*). Obtížnější případ nastane změnou na tvar obdélníka, tvar písmene L, nebo nepravidelný tvar, případně série místností spojených chodbami.
- ***Nábytek***: umístění nábytku vytváří složitější problém. Z hlediska agenta-vysavače nemusí vjem rozlišovat mezi zdí a kusem nábytku, obojí aktivuje sensor tak, že vydá hodnotu 1.
- ***Rozmístění smetí***: nejjednodušší je smetí rozmístit rovnoměrně v místnosti. Realističtější je více smetí umístit v určitých lokacích (např. dlouhá a velmi používaná cesta do další místnosti, nebo kolem židle u jídelního stolu, apod.).

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

- ***Řešení úloh*** (problémů) – z hlediska inteligentních agentů – znamená, jak může agent dosáhnout cílů a jaké by měly být sekvence akcí, které mohou k daným cílům vést.
- Řešený problém zde představuje cíl a soubor prostředků pro dosažení cíle.
- Proces prozkoumávání prostředků (tj. co mohou prostředky, které jsou k dispozici, dosáhnout), se nazývá vyhledávání (či prohledávání, pátrání).
- Agent řešící problémy – typ agenta zaměřeného na cíle. Například jednoduchý reflexní agent je příliš omezen (kvůli svým vlastnostem nemůže uvažovat směrem do budoucna).

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

- Agenti, kteří řeší problémy, rozhodují o své činnosti tím, že se snaží nalézt sekvence akcí, které vedou k požadovaným stavům.
- Obecně se o inteligentních agentech předpokládá, že provádějí akce tak, aby prostředí procházelo posloupností stavů z hlediska *maximalizace* měřené výkonnosti (méně obecně: z hlediska účinnosti agenta splnit vybraný cíl).
- Prvním krokem k řešení problému je *formulace cíle*, založená na okamžité situaci.
- Za cíl lze např. považovat soubor stavů světa – a to takové stavy, v nichž je cíl splněn.
- Na akce pak lze hledět jako na činnosti, které způsobí přechody mezi jednotlivými stavy světa, takže úkolem agenta je nalézt vhodné akce k dosažení cílového stavu.

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

- Obecně nelze uvažovat o akcích na určité detailní úrovni. Například při požadavku *vyjet z parkoviště* není pro agenta vhodné usuzovat o akcích typu "*natoč volant 6 stupňů doleva*" apod., protože generace řešení na takovéto úrovni detailů by mohla být ve většině případů nezvládnutelným problémem (člověk také neusuzuje tímto způsobem).
- *Formulace problému* je proces rozhodnutí, jaké akce a stavy uvažovat; tato formulace následuje po *formulaci cíle*.
- Dostatečná znalost: lze dojet z jednoho města do druhého, výběr z více cest, apod. Pokud není k dispozici dostatek informací a k cíli se lze dostat více možnostmi, pak agent nemá nic lepšího než náhodně zvolit.
- Obecně platí, že pokud má agent několik možností neznámé hodnoty, pak k rozhodnutí o své činnosti potřebuje napřed prozkoumat různé možné posloupnosti akcí vedoucích do stavů o známé hodnotě, a pak prostě vybrat tu nejlepší sekvenci. Tento proces se nazývá ***vyhledávání***.

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

- ***Algoritmus vyhledávání*** má jako *vstup* řešený problém.
- Na *výstupu* poskytuje řešení ve formě sekvence akcí. Jakmile je řešení nalezeno, je doporučeno (nabídnuto) k provedení – tzv. výkonná fáze.
- Z toho plyne jedna z možností, jak navrhnout agenta na základě trojice pojmů:

  1) formuluj,
  2) vyhledej,
  3) proveď.

- Po provedení zvolené sekvence akcí pak agent zvolí nový cíl a vše se pro aktuální situaci opakuje znovu.

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Jednoduchý agent řešící problém:

```
function Simple-Problem-Solving-Agent(p) returns action
   inputs: p        // vjem (percepce)
   static: s        // sekvence akcí, na počátku prázdná
           state    // popis stavu současného světa
           g        // cíl (goal), na počátku prázdný
           problem // formulace problému

   state ← Update-State(state, p) // aktualizace stavu

   if s is empty then
      g ← Formulate-Goal(state) // formulace cíle
      problem ← Formulate-Problem(state, g)
      s ← Search(problem) // hledání sekvence akcí

   action ← Recommendation(s, state) // 1. akce v sekvenci
   s ← Remainder(s, state) // ostatní akce v sekvenci

return action
```

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Formulace problémů

- Formulace problémů závisí na množství znalostí, které má agent k dispozici, aby bylo možno uvažovat jeho akce a stavy, v nichž se ocitne. To závisí na propojení agenta s prostředím pomocí jeho vjemů a akcí.

- Existují čtyři *základní* různé typy problémů:

  1) problémy pro jeden stav,
  2) problémy pro více stavů,
  3) problémy nahodilé,
  4) problémy k prozkoumání.

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Formulace problémů (pokračování)

Typy problémů

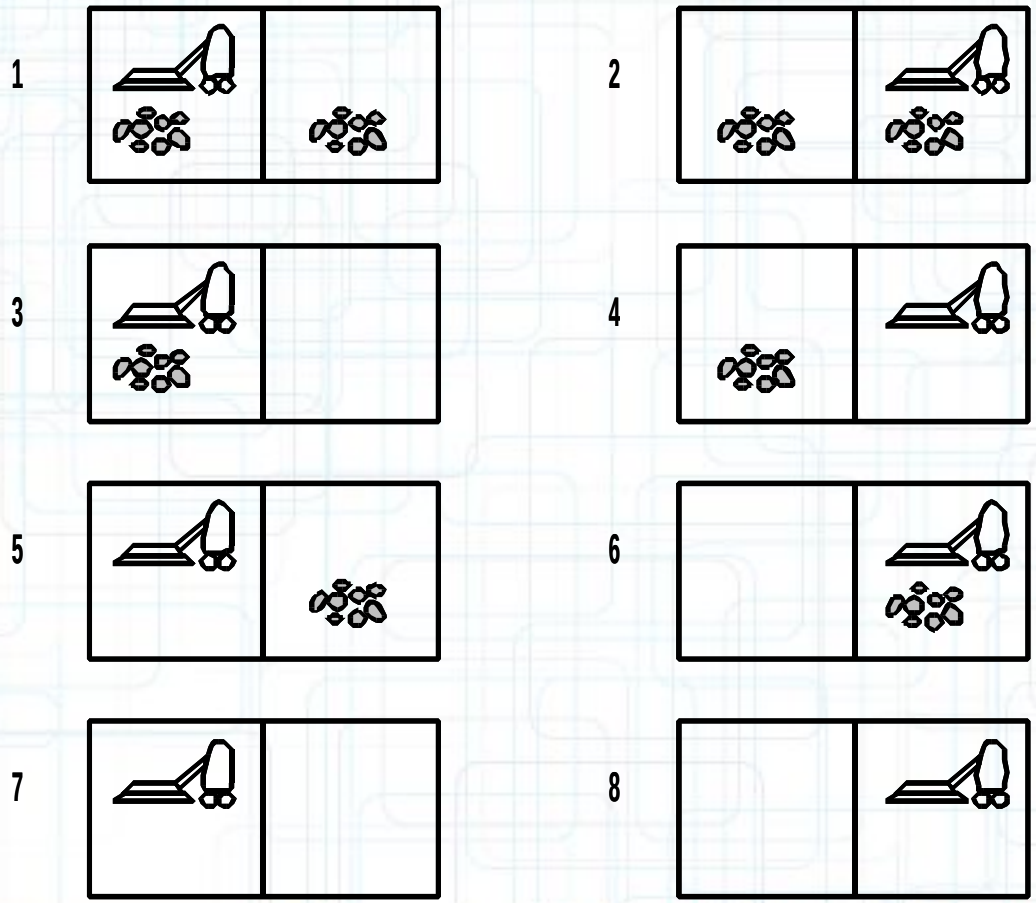
- Uvažme opět svět vysávání nečistoty:
- Nechť pro jednoduchost obsahuje pouze dvě místa.
- Každé z míst *může nebo nemusí* obsahovat nečistotu.
- Agent se může zpočátku nacházet v libovolném z obou míst.
- Existuje celkem 8 možných stavů světa, {1, 2, 3, 4, 5, 6, 7, 8}, viz následující obrázek. Agent má k dispozici tři možné akce (v této verzi světa): *Doleva, Doprava, Vysávat*. Dále předpokládejme, že vysávání je účinné na 100%. Cílem je zlikvidovat veškerou nečistotu, tj. cíl je ekvivalentní s dosažením množiny stavů {7, 8}.

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Formulace problémů (pokračování)

Agent-vysavač a jeho svět s osmi možnými stavy:



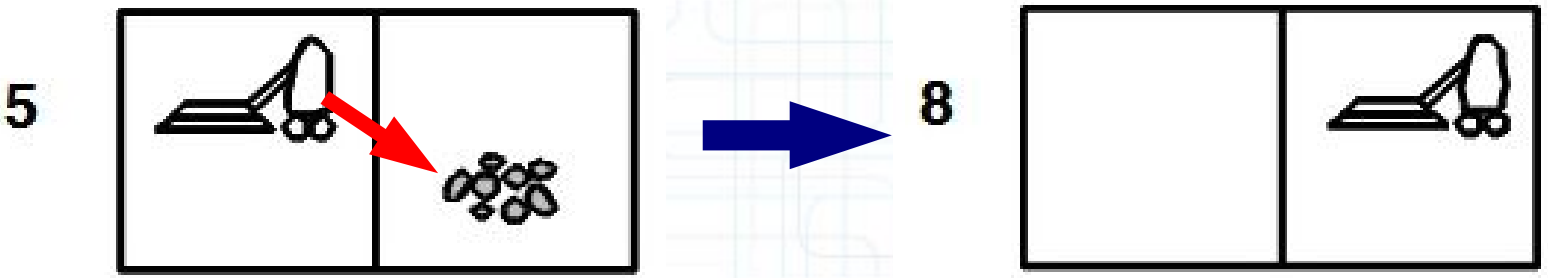
RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Formulace problémů (pokračování)

- Předpokládejme dále, že agentovy sensory mu poskytují dostatek informace k tomu, aby přesně věděl, v jakém stavu se nachází (svět je přístupný) a dále, že agent ví, co každá jeho akce udělá. Pak je možné exaktně spočítat, v jakém stavu bude po jakékoliv sekvenci akcí.

- Pokud je např. počáteční stav 5, pak lze určit, že po sekvenci akcí [*Doprava, Vysávat*] bude dosaženo cílového stavu. To je nejjednodušší případ, zvaný ***problém jednoho stavu***.



RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Formulace problémů (pokračování)

- Uvažujme dále agenta, který *zná všechny důsledky svých akcí*, ale má *omezený přístup ke stavům světa* (např. v extrémním případě nemusí mít žádné sensory). V takovém případě jediné, co agent ví, je, že jeho počátečním stavem je jeden z množiny stavů {1, 2, 3, 4, 5, 6, 7, 8}.
- Kupodivu agentova kritická situace není beznadějná – protože ví, k čemu po jeho akcích dochází, pak může určit, že např. po akci *Doprava* se ocitne v jednom ze stavů {2, 4, 6, 8}. Ve skutečnosti může agent odhalit, že po akční sekvenci [*Doprava, Vysávat, Doleva, Vysávat*] je *zaručeno dosažení cíle* bez ohledu na počáteční stav.
- Shrnutí: Pokud není svět plně přístupný, pak agent musí uvažovat o *množině stavů*, do nichž se může dostat, nikoliv o jednotlivých stavech. Uvedená situace se nazývá ***vícestavový problém***.

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Formulace problémů (pokračování)

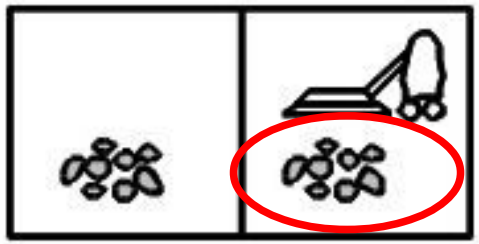
- Případ *neznalosti důsledku* akcí lze řešit obdobným způsobem. Uvažujme situaci, kdy prostředí se jeví *nedeterministické* (a splňuje *Murphyho pravidlo*):

  Akce zvaná Vysávání *někdy umístí nečistotu na koberec, ale pouze v tom případě, že tam ještě není žádná nečistota.*

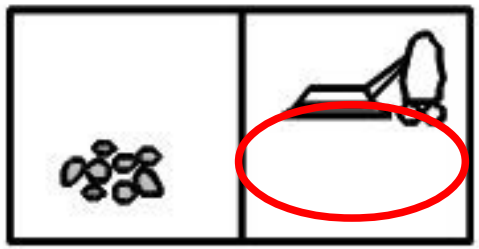
- Pokud agent tedy ví, že se nachází např. ve stavu 4, pak také ví, že bude-li vysávat, dosáhne jednoho ze stavů {2, 4}.

- Pro jakýkoliv známý počáteční stav existuje posloupnost akcí, která zaručuje dosažení cílového stavu.

2



4

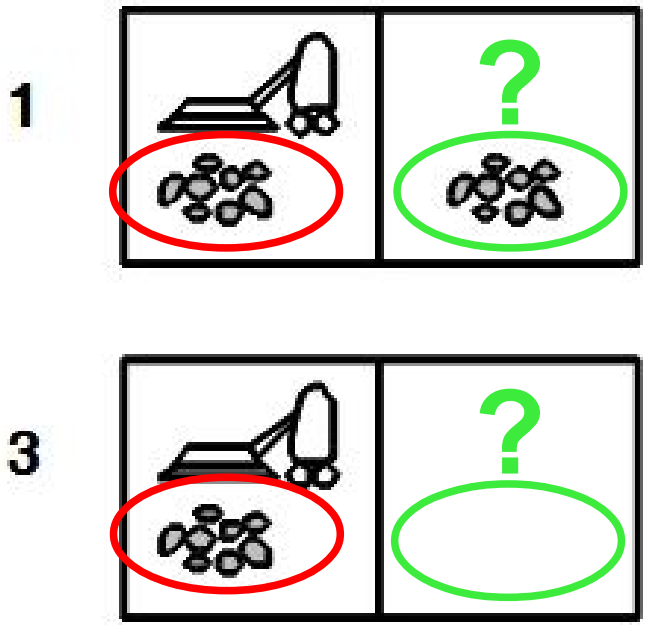


RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Formulace problémů (pokračování)

- Někdy ovšem neznalost zabraňuje agentovi nalézt sekvenci zaručující dosažení cíle. Dejme tomu, že agent je ve světě Murphyho pravidla, má poziční sensor a sensor pro lokaci nečistoty, ale nemá sensor schopný detekovat nečistotu v jiných místech (čtvercích). Řekněme, že sensory poskytují informaci, že agent je v jednom ze stavů {1, 3}.

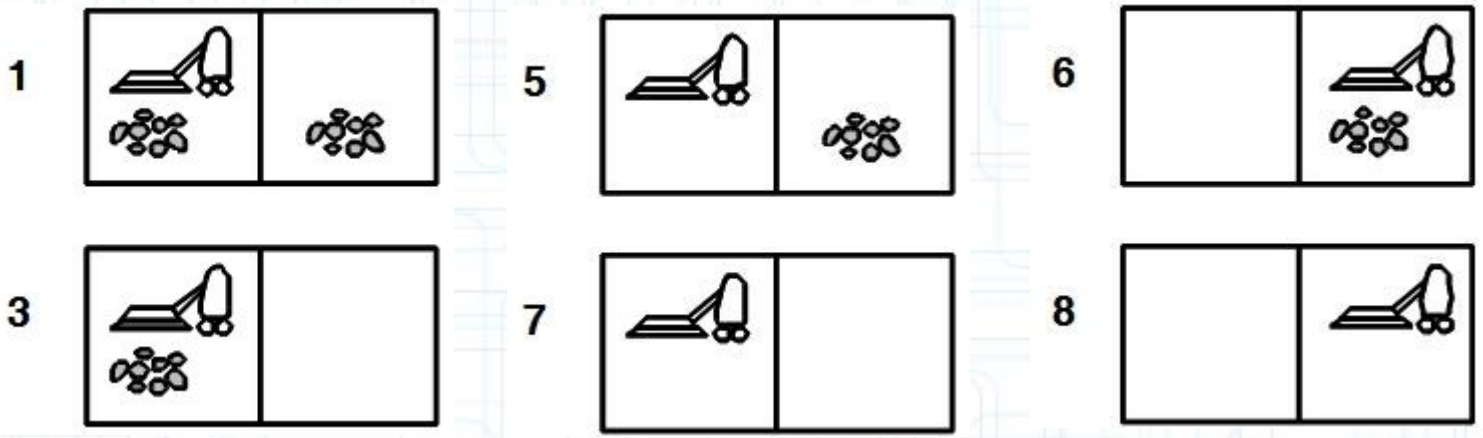


RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Formulace problémů (pokračování)

- Agent může formulovat sekvenci akcí [Vysát, Doprava, Vysát]. Vysátí změní stav na jeden z {5, 7}, pohyb doprava na jeden z {6, 8}. Je-li skutečným stavem 6, pak sekvence akcí je úspěšná; pokud 8, pak plán selhal.



- Pokud by agent vybral jednodušší sekvenci akcí [Vysát], pak to někdy může uspět, ale ne vždy. Ukazuje se, že neexistuje žádná fixní posloupnost akcí, která by zaručila řešení problému.

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Formulace problémů (pokračování)

- V uvedeném případě má agent cestu k řešení problému z jednoho stavu {1, 3}: *vysát, zatočit doprava, a pak vysát* ***pouze pokud*** *tam je nečistota*.
- Tento postup ovšem vyžaduje *možnost vnímání během exekuční fáze*. V tomto případě musí agent spočítat *celý* strom akcí (nikoliv jen sekvenci pro jednu akci).
- Obecně se každá z větví stromu zabývá možnou nahodilostí, která se může objevit. Proto se tato situace nazývá ***problém nahodilosti***.
- V realitě k těmto problémům patří velké množství situací, protože přesná predikce je nemožná – z tohoto důvodu např. lidé obvykle důkladně sledují situaci při přecházení silnice nebo při řízení.

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Formulace problémů (pokračování)

- Problémy jednoho stavu a více stavů mohou být zpracovávány podobnými vyhledávacími technikami.
- Problémy s *nahodilostí* však vyžadují mnohem složitější algoritmy a často vedou ke změně návrhu agenta, kdy agent může provést akci *před* nalezením zaručeného plánu.
- To je užitečné proto, že může být snazší nebo lepší prostě něco začít dělat a teprve pak zjišťovat, jaké nahodilosti se objevily, ve srovnání s postupem, kdy se předem uvažuje *každá* možná nahodilost, která by se mohla vyskytnout během exekuční fáze.
- Agent pak může pokračovat v řešení problému za předpokladu získání přídavné informace. Tento postup se nazývá ***prokládání činnosti vyhledávání a exekuce***.

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Dobře definované problémy a řešení

- Pod pojmem ***problém*** zde budeme rozumět souhrn informací, které agent použije k rozhodnutí o své činnosti.
- Základními prvky definice problému jsou *stavy* a *akce*:
  - Počáteční stav – stav, o němž agent ví, že v něm je.
  - Operátor – soubor agentových možných akcí. Termín operátor se používá k popisu akce, a to pomocí pojmů, jakých stavů lze docílit provedením akce z konkrétního stavu. Někdy se používá formulace ***následnická funkce S***: je-li dán konkrétní stav *x*, pak *S(x)* vrací soubor stavů dosažitelných z *x* libovolnou jednou akcí.
  - Stavový prostor problému – soubor všech stavů dosažitelných z počátečního stavu libovolnou sekvencí akcí.

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Dobře definované problémy a řešení (pokračování)

- Cesta – ve stavovém prostoru jde prostě o jakoukoliv sekvenci akcí, která vede z jednoho stavu do druhého.
- Test cíle – test aplikovaný agentem na popis jednoho z cílů, zda jde o cílový stav. Pokud je cílových stavů více, test cíle rozezná, zda se o jeden z nich jedná či nikoliv.

  Pozn.: Někdy je cíl specifikován nějakou abstraktní vlastností spíše než explicitně vyjmenovaným seznamem stavů. Např. v šachu jde o dosažení cíle mat *(protivníkův král je napaden figurou a nemá kam ustoupit, a ani to napadení nelze zlikvidovat, tj. dalším tahem lze krále vzít bez ohledu na to, co oponent vlastnící daného krále udělá).*

- Funkce ceny cesty *g* – funkce *g* přiřazující dané cestě nějaké náklady (nebo cenu). Nadále budeme uvažovat o ceně cesty jako o sumě cen za jednotlivé akce podél dané cesty.

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Dobře definované problémy a řešení (pokračování)

- Počáteční stav, soubor operátorů, test cíle a cena cesty dohromady ***definují problém***. Typ dat representující problém:

```
datatype Problem
  components: Initial-State,
              Operators,
              Goal-Test,
              Path-Cost-Function
```

- Instance uvedeného typu dat slouží jako *vstup* vyhledávacího algoritmu; *výstupem* je řešení, tj. cesta z počátečního stavu do stavu vyhovujícímu cílovému testu. Pro zpracování ***vícestavových*** problémů se provede jednoduchá modifikace: problém je složen z počátečního *souboru* stavů a dále ze *souboru* operátorů. Test cíle a funkce ceny cesty zůstávají jako předtím. Cesta nyní propojuje *soubor stavů* a řešením se nyní rozumí cesta vedoucí do souboru těch stavů, které vyhovují cílovým stavům. Stavový prostor je nahrazen prostorem souboru stavů.

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Měření výkonnosti řešení problémů

- Tři možnosti:

  1. Bylo vůbec nalezeno řešení?

  2. Je řešení dobré (tj. cesta s nízkou cenou)?

  3. Jaká byla cena za nalezené řešení vzhledem k požadované paměti a času?

- Celková cena vyhledávání je tvořena *součtem ceny cesty a ceny vyhledávání* (v robotice se cena vyhledávání, tj. části prováděné před interakcí s prostředím, nazývá offline cena a cena cesty online cena).

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Měření výkonnosti řešení problémů (pokračování)

- Např. cesta z města A do města B: cena cesty může být úměrná počtu kilometrů plus náklady za opotřebení na různých typech silnic.
- Cena vyhledávání závisí na okolnostech:
  - Ve statickém prostředí je nulová (nezávislost na čase).
  - Je-li nějaká urgentní potřeba dojet do B, pak je prostředí semidynamické, protože delší uvažování o cestě cenu zvýší (můžeme uvažovat o přibližně lineární závislosti na době výběru, přinejmenším pro kratší časové úseky). Zde je tedy nutno dát dohromady kilometry a sekundy (cenu cesty a cenu vyhledávání, což není obecně snadný problém. Navíc pro velmi malé stavové prostory je snadné najít řešení s minimální cenou cesty.
    Pro rozsáhlé a komplikované problémy je nutno nalézt kompromis.

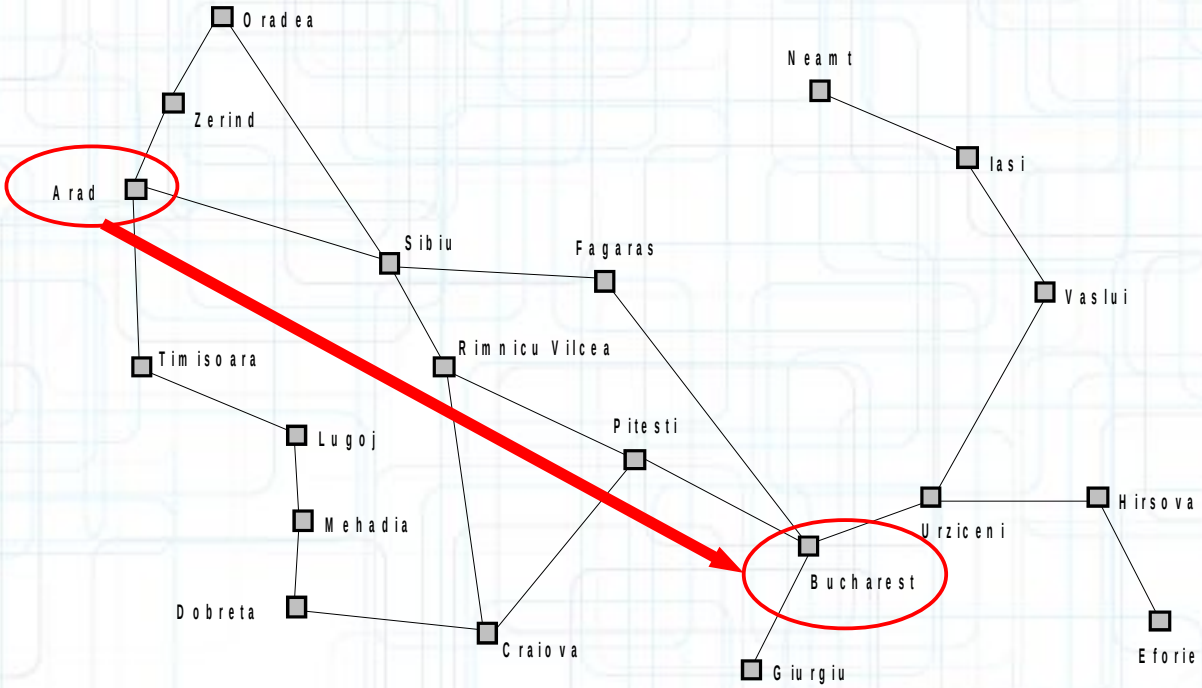
RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Měření výkonnosti řešení problémů (pokračování)

- Výběr stavů a akcí

Příklad: dojet z města **A**rad do města **B**ucharest, přičemž k dispozici je mapa cest:



RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Měření výkonnosti řešení problémů (pokračování)

- Odpovídající stavový prostor má 20 stavů (měst), každý stav je definován pouze polohou specifikovanou jako město. Počáteční stav je tedy "*v Aradu*", test cíle je "*je to Bucharest?*".
- Operátory odpovídají jízdě po silnicích mezi městy.

- Jedno možné řešení: Arad→Sibiu→Rimnicu Vilcea→Pitesti→Bucharest.

- (Je mnoho dalších řešení.) K rozhodnutí o tom, které řešení je lepší, je nutno znát, *co měří funkce ceny*: celkovou *délku* cesty nebo celkový *čas* cesty?

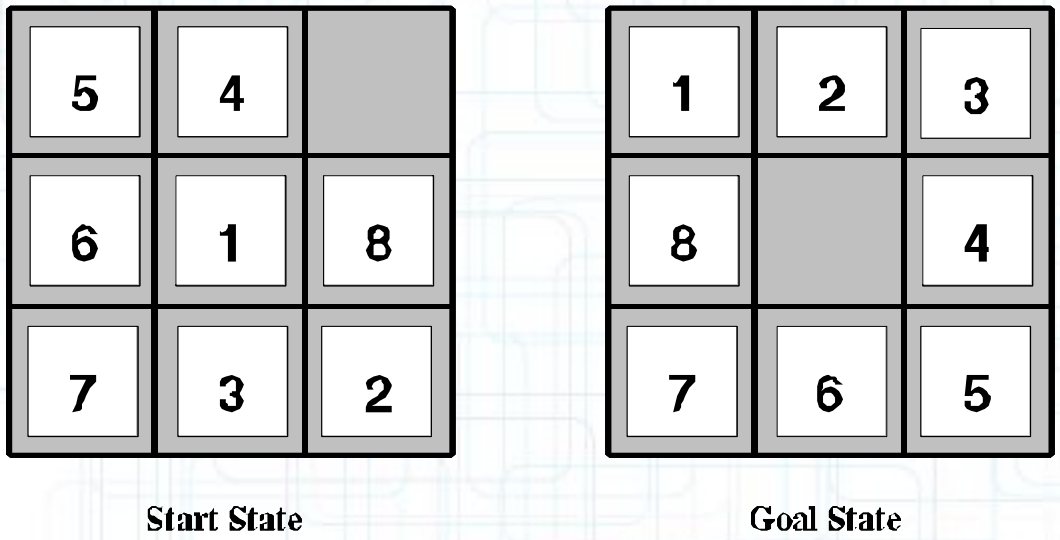
- Daná mapa neobsahuje ani jedno, takže se použije počet kroků (průjezdních silnic mezi městy): to je 3 přes Sibiu a Fagaras, tedy nejlepší řešení.

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Příklady problémů

- Obvykle je vhodné uvažovat o problémech her a o problémech reálných (které jsou opravdové, avšak obvykle mnohem obtížnější pro hledání řešení). U problémů her je ale snadnější vytvořit jasný a exaktní popis, takže mohou být snadněji použity pro srovnání různých metod apod.
- Problémy her
- 8-hlavolam:



RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Příklady problémů (pokračování)

- Při řešení problému seřazení posuvných kosteček je vhodné použít určité “triky”, které řešení usnadní.
- Příklad: místo použití operátorů typu *přesun kostky s číslem 4 na volnou pozici* je mnohem rozumnější použít operátory typu *volná pozice mění místo s kostkou nalevo*, apod. (Vede to k menšímu počtu operátorů.)
- Lze vytvořit následující formulaci:
  - *Stavy*: popis stavu specifikuje umístění každé z osmi kosteček na jednom z devíti polí (je užitečné zahrnout i pozici volného místa).
  - *Operátory*: volné místo se pohybuje doleva, doprava, nahoru, dolů.
  - *Test cíle*: stav se shoduje s cílovou konfigurací na obrázku.
  - *Cena cesty*: každý krok stojí 1 (cesta stojí totéž, co je její délka).

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Příklady problémů (pokračování)

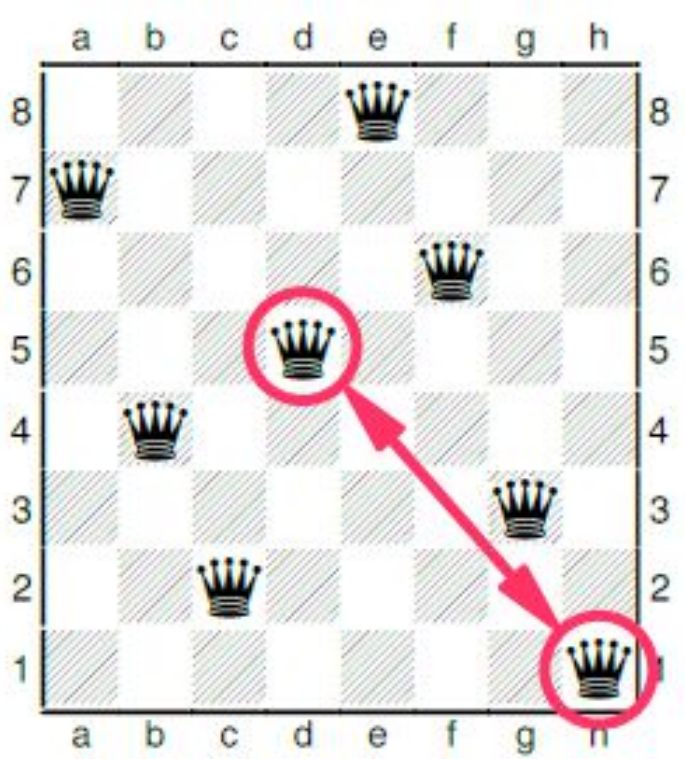
- 8-hlavolam patří ke skupině úloh přemísťování bloků, což obecně jsou *NP-kompletní problémy*, kde nelze očekávat, že bude nalezeno řešení lepší než pomocí vyhledávacího algoritmu.
- Principiálně stejná, avšak značně složitější úloha 15-hlavolam se spolu s 8-hlavolamem používá často k testování nových vyhledávacích algoritmů v umělé inteligenci.

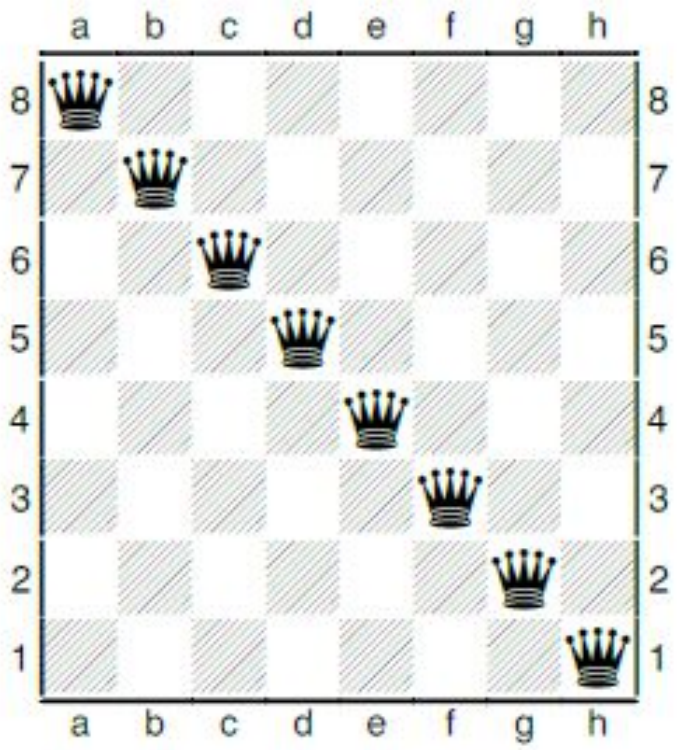
RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Příklady problémů (pokračování)

- Osm dam na šachovnici 8x8 (resp. N dam na šachovnici NxN) patří do skupiny tzv. *dobře definovaných problémů*.





RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Algoritmy iterativního vylepšování

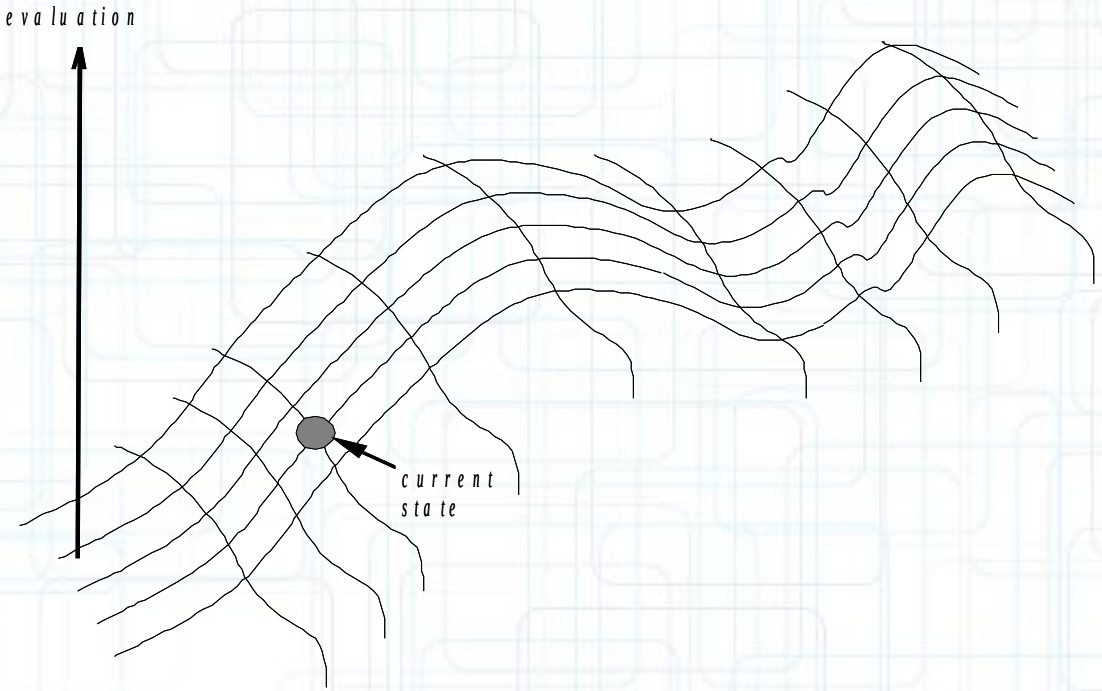
- Pro řadu dobře definovaných problémů (např. 8 dam, VLSI) platí, že *popis stavu sám o sobě obsahuje veškerou informaci potřebnou pro řešení*. Konkrétní cesta k řešení (zda je řešení dosaženo tak či jinak) nemusí být (a často není) relevantní. V těchto případech poskytují algoritmy s iterativním vylepšováním nejpraktičtější přístup.
- Např. lze začít se všemi 8 dámami na šachovnici nebo všemi vodiči v konkrétních spojovacích kanálech řešeného obvodu VLSI. Pak můžeme pohybovat dámami ve snaze redukovat počet vzájemného napadání, nebo přemísťovat vodič z jednoho kanálu do druhého ve snaze redukovat zahlcení.
- Ideou je začít s kompletní konfigurací a modifikacemi zlepšit kvalitu.

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Algoritmy iterativního vylepšování (pokračování)

- Pro představu lze uvažovat o stavech umístěných na povrchu nějaké krajiny. Výška povrchu v nějakém místě odpovídá vyhodnocovací funkci (evaluation) pro stav v daném bodě (current state):



RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Algoritmy iterativního vylepšování (pokračování)

- Podle myšlenky iterativního vylepšování je vhodné se v krajině pohybovat a hledat nejvyšší vrcholek, tj. optimální řešení. Iterativní vylepšování obvykle udržuje údaje pouze o cestě aktuálního uzlu a nehledí dále než k bezprostředním sousedům daného stavu (hledání vrcholku hory v mlze zároveň s postižením zapomnětlivostí). Přes zmíněné nedostaky je tento postup v řadě velmi obtížných případů praktický – typickou ukázkou použití zmíněného postupu jsou umělé neuronové sítě.

- Jednou ze skupin zmíněných algoritmů je tzv. **slézání kopce** (v originále *hill-climbing*, někdy pojmenované také jako *gradient descent*, tj. gradientní sestup). Další skupinou jsou algoritmy tzv. simulovaného žíhání (simmulated annealing). Zde se zmíníme o lezení po kopcích.

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Algoritmy iterativního vylepšování (pokračování)

- Slézání kopce (gradientní sestup)

Algoritmus je symbolicky popsán následovně:

```
function Hill-Climbing(problem) returns stav řešení

    inputs: problem // problém
    local  variables: current // aktuální stav-uzel
           next // další stav-uzel

    current ← Make-Node(Initial-State[problem])
    loop do
        next ← následný uzel s nejvyšší hodnotou
        if Value[next] < Value[current] then return current
        current ← next
    end
```

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Algoritmy iterativního vylepšování (pokračování)

- Ve smyčce se hledající pohybuje směrem ke vzrůstající hodnotě (do kopce). Algoritmus nepracuje s vyhledávacím stromem, takže datová struktura pro uzel musí pouze zaznamenávat stav a jeho hodnotu *Value*. Uvedený jednoduchý postup má tři známé nedostatky:

  – **Lokální maxima**: po dosažení lokálního maxima se algoritmus zastaví, přestože globální maximum je mnohem lepší.

  – **Roviny**: ve všech směrech poskytuje vyhodnocovací funkce stejnou hodnotu, takže postup krajinou vede k náhodné cestě.

  – **Hřebeny**: hřeben má strmě klesající strany, takže se na něj lze dostat snadno, ale další stoupání hřebenu k vrcholku je velmi mírné a dochází k oscilaci hledání ze strany na stranu se zanedbatelným pokrokem v přibližování se k maximu.

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Algoritmy iterativního vylepšování (pokračování)

- V každém případě se algoritmus dostane do bodu, odkud se nelze dostat výš. V takových případech se lze pokusit znovu z jiného startovacího bodu – k tomu se používá metoda zvaná *slézání kopce s náhodným počátkem*, takže dochází k sérii hledání z náhodně vybraných různých míst, a nejlepší dosažené řešení se uchovává. Ukončení vyhledávání pak závisí na tom, zda byl stanoven maximální předem stanovený počet těchto hledání nebo zda doposud dosažený nejlepší výsledek nebyl po nějaký počet iterací zlepšen.
- Pro dostatečný počet iterací může (ovšem i nemusí) hledání s náhodným restartem nakonec najít optimální řešení.
- Úspěch zmíněného typu hledání velmi silně závisí na tvaru povrchu, který je dán prostorem uvažovaných stavů. S malým počtem lokálních maxim je řešení nalezeno rychle. Realistické problémy však představují spíše podobu povrchu dikobraza.

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Algoritmy iterativního vylepšování (pokračování)

- Pro NP-kompletní problémy nelze dosáhnout lepší než exponenciální časové závislosti.
- Přesto jsou takto založené metody velmi často úspěšné proto, že poskytují velmi dobré řešení, i když to nemusí být zrovna optimum (a často není možné ani zjistit, jaké to optimum vlastně je).
- Proto se použije nějaký přijatelný výsledek dosažený po nepříliš velkém počtu iterací – praktické hledisko hraje často výraznou roli (např. časové omezení na vyřešení úlohy).

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

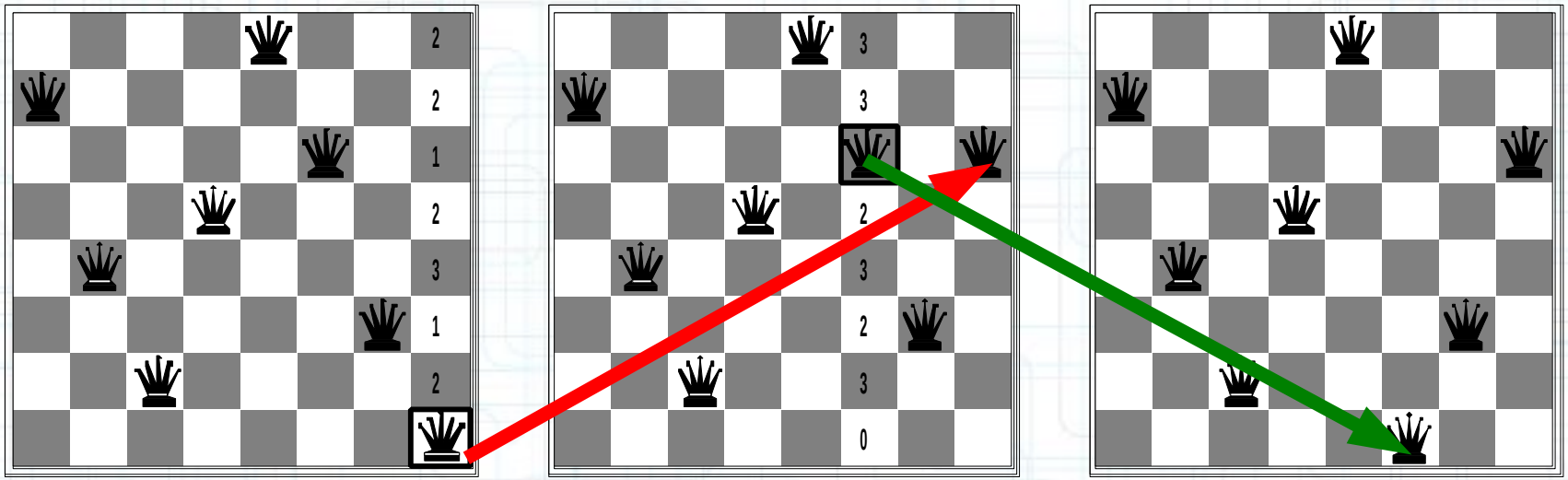
Aplikace pro problémy vyhovující omezením (CSP)

- Problémy vyhovující omezením (Constraint Satisfaction Problems) mohou být řešeny metodami iterativního zlepšování tak, že napřed jsou všem proměnným dosazeny nějaké hodnoty a pak za použití modifikačních operátorů se konfigurace pohybuje směrem k řešení.
- Modifikační operátory jednoduše přiřazují jiné hodnoty proměnným.
- Např. pro problém 8 dam je počátečním stavem 8 dam nějak rozestavených na šachovnici a modifikační operátor může dámou pohnout o políčko vedle.
- Tyto algoritmy se nazývají **heuristické opravování**, neboť napravují nekonsistence aktuální konfigurace. Při výběru nové hodnoty proměnné se (heuristicky) vybírají takové hodnoty, které jako výsledek dávají minimální konflikty s ostatními proměnnými – tzv. heuristika min-konflikt:

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Aplikace pro problémy vyhovující omezením (CSP) (pokračování)



- Problém je zde vyřešen ve dvou krocích. Čísla ve sloupci s dámou, jejíž postavení se vylepšuje (Dh8 v pravém dolním rohu), udávají, s kolika jinými dámami je **v konfliktu**. Dáma je přesunuta na pole h6 (prostřední diagram), kde má konflikt s Df6. Dáma z pole f6 našla nekonfliktní místo na poli f1 a protože další konflikty nejsou, úloha je vyřešena.

RACIONÁLNÍ AGENTI

Řešení úloh pomocí agentů

Aplikace pro problémy vyhovující omezením (CSP) (pokračování)

- Překvapivě je metoda min-konflikt úspěšná pro mnoho problémů s omezením (CPS) , např. problém s milionem dam má průměrné řešení v méně než 50 krocích.
- V nedávné době byla min-konflikt použita pro plánování pozorování vesmírného teleskopu Hubble (HST, Hubble Space Telescope), kde redukovala čas nutný pro naplánování týdenního pozorování ze tří týdnů na přibližně 10 minut (astronomové si mohou podávat žádosti o pozorování pomocí HST a velké množství velmi různých žádostí vyžaduje velmi dobrý rozvrh).