

# Organizační úvod

*Poznámka* (Organizační úvod)

Na zkoušku není nutné mít zápočet. Přednáška má stránku (se slidy, kvízem). Záznamy z loňského roku budou.

## 1 Úvod

### Definice 1.1 (Prostředí)

Prostředí může být s plnou informací nebo s částečnou informací (podle toho, zda agent dostává svými senzory vše, nebo ne), dále může být buď deterministické nebo stochastické (podle toho, zda je plně určené svým stavem a akcí), dále je buď epizodní nebo sekvenční (podle toho, zda se pořád opakuje to samé, např. návštěva u lékaře, nebo zda se neopakuje), dále statické nebo dynamické (podle toho, zda ho ovlivňuji jen já, nebo i něco jiného, semi-dynamické je, když přemýšlení ovlivňuje můj výkon, např. hry s hodinami), dále diskrétní nebo spojitě, dále jedno-agentová nebo více-agentové (kompetitivní/kooperativní).

### Definice 1.2 (Reflex agent)

Simple RA: Na základě pozorování světa vrátí akci. (V podstatě neprocedurální funkce beroucí svět a vracející akci.)

Model-based RA: Kromě vracení akce i mění svůj stav (pomocí stavového modelu).

### Definice 1.3 (Goal-based agent)

Funguje podobně jako Reflexní agent, ale má ještě navíc nějaký cíl (který lze měnit např. příkazem), který ovlivňuje akci.

### Definice 1.4 (Stav)

Stav může být reprezentován buď atomicky (nemá žádnou strukturu) nebo Factored? (stav je vektor hodnot) nebo strukturovaně (stav je množina objektů spojených různými relacemi).

### Definice 1.5 (Problem solving agent)

PSA je typ goal-based agenta, který používá atomickou reprezentaci stavů, cíl je jedním ze stavů a akce jsou popisy, jak se stavy mění.

Úkolem je najít sekvenci akcí, která dosáhne cílového stavu. Hledá se pomocí nějakého search algoritmu.

### **Definice 1.6** (Dobře definovaný problém)

Dobře definovaný problém má počáteční stav, přechodový model (který má rozumnou míru abstrakce, např. neovládá každý sval zvlášť), jím implikované stavy a test určující cílové stavy.

Tím je implicitně definovaný search tree. (Na něm je algoritmus tree search, který strom prochází tak, že do „množiny“ postupně přidává syny prvků, které v ní už jsou. Často je však problém s opakováním stavů.)

### **Definice 1.7** (Graph search)

Graph search je skoro totéž jako tree search, jen si u každého stavu pamatuje, zda již byl navštíven, nebo ne.

Search tree tohoto algoritmu má každý stav nanejvýš jednou.

### **Definice 1.8** (Kompletní algoritmus)

Algoritmus je kompletní, když správně najde řešení respektive dokáže, že neexistuje, pro všechny vstupy.

*Poznámka* (Neinformované prohledávání (prohledávání obecného stavového prostoru))

Následně se probíral breadth-first search, depth-first search a backtracking (na rozdíl od DFS nenačte hned všechny následníky vrcholu, ale jde jeden po druhém, což nemusí být vždy možné).

### **Definice 1.9** (Informované (heuristické) algoritmy, best-first search, $A^*$ )

Algoritmy, které využívají pro rozhodování používají navíc tzv. heuristiku.

Patří mezi ně např. best-first search, který prohledává stav, kde je nejmenší evaluační funkce  $f(n)$ , která kromě vzdálenosti od počátku ( $g(n)$ ) bere v potaz i heuristiku  $h(n)$ . Ten podle volby  $f(n)$  může být např. greedy best-first search:  $f(n) = h(n)$ , nebo  $A^*$ :  $f(n) = g(n) + h(n)$ .

### **Definice 1.10** (Přípustná a monotónní heuristika)

Přípustná heuristika je taková, která vrací hodnotu mezi nulou a nejlepší cestou.

Monotónní (nebo také konzistentní) heuristika je taková, která splňuje „trojúhelníkovou nerovnost“ (tedy rozdíl heuristik nemůže být větší než cesta mezi nimi).

### **Tvrzení 1.1**

*Je-li heuristika monotónní (a nezáporná), pak už je přípustná.*

┌  
Důkaz

$$h(\text{start}) - h(\text{cíl}) \leq |\text{nejkratší cesta}|.$$

└ □

### **Tvrzení 1.2**

---

*A\* v tree-search je optimální (první nalezená cesta je nejkratší).*

┌  
Důkaz

V otevřených vrcholech je vždy vrchol nejkratší cesty (jelikož počáteční stav je a vždy když vrchol uzavíráme, tak přidáme všechny sousedy).

Cíl musíme najít po nejkratší cestě, protože v cíli je  $f(n) = g(n)$  a my jsme ho poprvé potkali při nejmenším  $f(n)$ . □

### **Tvrzení 1.3**

---

*Je-li použitá heuristika monotónní, pak A\* v graph-search je optimální.*

┌  
Důkaz

Jednoduchý, podobně jako u tree-search, navíc se dokazuje jen, že do každého vrcholu přijde po nejkratší cestě. □

### **Definice 1.11 (Dominance)**

---

Heuristika  $h_1$  dominuje heuristice  $h_2$ , když  $\forall n : h_1(n) \geq h_2(n)$ .