

# 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  $\forall n : h_1(n) \geq h_2(n)$ .

## **Definice 1.12 (Forward checking)**

Testuje budoucí pozice, jestli tam vůbec může být cíl.

## **Definice 1.13 (Problém splňování)**

Problém splňování? sestává z konečného počtu proměnných (popisujících svět), oborů hodnot pro každou proměnnou a konečné množiny podmínek.

Řeší se Backtrackingem.

## **Definice 1.14 (Přípustné řešení)**

Přípustné řešení? je takové řešení, které je úplné (každá proměnná má přiřazenou hodnotu) a konzistentní (každá podmínka je splněna).

### **Definice 1.15** (Hranová konzistence)

Hrana (dvojice vrcholů) je konzistentní, pokud existují hodnoty z oborů hodnot proměnných tak, že když je dosadíme, tak jsou splněny všechny podmínky mezi těmito proměnnými.

Problém splňování je hranově konzistentní, pokud každá hrana je konzistentní.

### **Definice 1.16** (k-konzistence)

Podobně jako hranová konzistence, jen pro více vrcholů zároveň (a splnění všech, klidně binárních, podmínek mezi nimi).

### **Definice 1.17** (Fail-first princip (pořadí proměnných))

První zkoušíme dosadit do té proměnné, která nejpravděpodobněji selže.

Heuristiky na selhání jsou: dom heuristika = proměnná s nejmenším množstvím hodnot, deg heuristika = vybírá proměnnou, která je v nejvíce podmínkách.

### **Definice 1.18** (Succeed-first principe (pořadí hodnot))

Vybírá hodnotu, která má největší pravděpodobnost na úspěch.

### **Definice 1.19** (Programování s podmínkami)

Programování s podmínkami je deklarativní přístup řešení (kombinatorických) problémů.

### **Věta 1.4**

*Pokud je problém  $\forall i \in [n]$   $i$ -konzistentní, pak ho lze vyřešit bez backtracku.*

### **Definice 1.20** (CNF (Conjunctive normal form))

Literál je proměnná nebo její negace, klauzule je disjunkce literálů a formule v CNF je konjunkce klauzulí.

### **Tvrzení 1.5** (Z logiky)

*Každá sentence ve výrokové logice je logicky ekvivalentní formuli v CNF.*

### **Definice 1.21** (DPLL)

Jmenuje se podle autorů. Řeší SAT podobně jako jsme řešili CSP.

Nejdřív se ohodnocují jen ty proměnné, které se vyskytují pouze jednou (ty ohodnotíme BÚNO). Pak se vyřeší klauzule o jedné proměnné (ty jsou nutně určené). Nakonec se vyzkouší něco doplnit s backtrackingem.

V modernějších řešeních se ještě hledá, zda se SAT nedá rozdělit na disjunktní (co do proměnných) části. Pak se využívají heuristiky jako v CSP a dá se (stejně jako CSP)

používat náhodný restart.

### **Definice 1.22** (Znalostní agent)

Znalostní agent funguje tak, že odvozuje neznámé části světa podle známých.

## 2 Dynamický SAT

### **Definice 2.1** (Fluent)

V podstatě dvojice proměnná – čas. (Ohodnocení true znamená, že v čase platí proměnná.)

### **Definice 2.2** (Pozorovací model)

Vezme pozorované prostředí a převede je do modelu (např. přidá souřadnice, nebo čas).

### **Definice 2.3** (Přechodový model)

Popisuje, jak evoluuje svět.

### **Definice 2.4** (Frame problem)

V přechodovém modelu se musí logickými formulami zadat i proměnné, které se nemění. Pro jednoduchost se používá tzv. axiom následujícího stavu (pokud akce nemění proměnnou, tak se nemění zapíšeme jako proměnná je ekvivalentní sobě konjunkce akce měnící proměnnou).

### **Definice 2.5** (Precondition axioms)

Podmínky pro provedení akce se dají zapsat jako implikace.

### **Definice 2.6** (Situační kalkulus)

Výroková logika má nevýhodu, že nemá kvantifikátory, tedy musíme popis všeho většínou nagenarovat (například jedno pole je soused vedlejšího). S tím nám pomůže situační kalkulus, kde se používá logika 1. řádu.

### **Definice 2.7** (Klasické plánování)

Využívá popis situačního kalkulu, ale řeší problém pomocí např.  $A^*$ .

#### *Poznámka*

Negace se lze zbavit tak, že se předpoklad duplikuje, a duplikát představuje negaci.

TODO!!! (Bayesovské sítě.)

TODO? (Nebyl jsem ve škole)

TODO!!! (Rozhodovací sítě)

## 3 Teorie her

TODO!!! (MiniMax, Alfa-Beta ořezávání + pravděpodobnost (např. expected MiniMax)).

## 4 Umělá inteligence

TODO!!! (Učení se formulí, pár slov o neuronkách a další)

TODO!!! (Dodělání části strojového učení z minula, statistické metody, reinforcement learning)