MULTIAGENTNÍ SYSTÉMY NAIL106

Roman Neruda, Martin Pilát
MFF UK Praha, 2014

Literatura:

- M. Wooldridge: An Introduction to Multiagent Systems, (2nd ed), 2009.
- S. Russell, P. Norvig: Artificial Intelligence: A Modern Approach, (3rd ed), 2009.
- G. Weiss (ed): Multiagent Systems (2nd ed), 2013.
- Y. Shoham, K. Leyton-Brown: Multiagent systems: Algorithmic, Game-Theoretic, and Logical Foundations, 2009. [online]
- David Ensley, Jon Kleinberg: Networks, Crowds, and Markets: Reasoning about a highly connected word [online]

Témata

- Agenti, prostředí, abstraktní architektury.
- Reaktivní a plánovití agenti, hybridní architektury.
- Agenti založení na logice a zdůvodňování, BDI.
- Komunikace, řečové akty, ACL.
- Ontologie, OWL, KIF.
- Distribuované řešení problémů, kooperace.
- Multiagentní interakce, Nashova ekvilibria, Paretova efektivita.
- Alokace zdrojů, aukce, smlouvání.
- Metodologie návrhu multiagentních systémů, Gaia, role.
- Jazyky a prostředí multiagentních systémů, JADE.

O čem se (moc) mluvit nebude

- Co to je přesně ten agent
- Učení agentů a učení v MAS
- Modální, temporální logiky a další zajímavé logické věci
- Robotika
- Plánování
- Umělý život
- ABM - Agent Based Model

Co se cvičí

- BattleCode
  - The 6.370 Battlecode programming competition is a unique challenge that combines battle strategy, software engineering and artificial intelligence.
- JADE
  - JADE (Java Agent DEvelopment Framework) is a software Framework fully implemented in Java language. It simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications and through a set of graphical tools that supports the debugging and deployment phases.
- AgentSpeak a Jason
  - AgentSpeak is an agent-oriented programming language. It is based on logic programming and the BDI architecture for (cognitive) autonomous agents.

1. Úvod

Vize, vymezení a pomluvy okolních oblastí, základní pojmy

Propaganda neboli trendy informatiky

- Všudypřítomnost (ubiquity)
  - Ze sálů na stoly, ze stolů do kapes a ledniček
- Propojenost a distribuovanost
  - Internet, cloudy
  - Pohled na výpočet jako interakci
- Inteligence
  - Složitost problémů se stále zvětšuje
- Delegace
  - Hledání na webu, fly-by-wire
- Příjemnost pro člověka
  - GUI, personalizace

Multiagentní systémy

- Vznik nové disciplíny informatiky
- Agent je (počítačový) systém schopný nezávisle pracovat v zastoupení svého uživatele.
- Multiagentní systém sestává z agentů, kteří spolu komunikují, nejčastěji výměnou zpráv prostřednictvím počítačové sítě.
- Jak takové agenty nezávislé agenty schopné provádět delegované úlohy vytvořit?
- Jak vytvořit systém agentů schopných kooperace, koordinace, vyjednávání?

Softwarové inženýrství

- Agenti jako jedno z možných paradigmat programování:
  - strojový kód
  - Strukturované programy
  - Objekty
  - CORBA
  - …
- Zvládnutí složitých distribuovaných systémů komunikujících komponent
- Objects do it for free, agents because they want to.

Distribuované, konkurentní systémy

- V distribuovaných systémech se po desetiletí zabývají teorií, programovacími jazyky a metodami pro popis a vývoj systémů s mnoha komponentami.
  - Mutex, deadlock, …

- Agenti jsou autonomní, mechnismy koordinace nejsou předem určeny, vše se děje a musí řešit v čase běhu.
- Agenti mají své zájmy, nemusejí sdílet žádný společný cíl. Je třeba studovat mechanismy dohadování a dynamické koordinace.

Umělá inteligence

- Tradiční pohled:
  - MAS jsou součástí AI.
- Russell, Norvig (AIMA):
  - Cílem AI je vývoj inteligentních agentů.
- Odvážný pohled:
  - AI je součástí MAS.
- Hardcore (Etzioni):
  - MAS is 1% AI, 99% computer science.
- MAS využívá techniky AI (reprezentace, plánování)
- MAS zdůrazňuje sociální aspekty (kooperace, vyjednávání), které donedávna AI opomíjela.

Teorie her, ekonomie

- Již von Neumann a Turing ve 40.letech …
- Teorie her je dnes jedním z důležitých teoretických nástrojů pro studium MAS
  - Nashovo ekvilibrium, …

- MAS zpochybňuje (jednou zpochybní?) pojem racionálních agenta, který je klíčový pro teorii her.
- Matematické základy teorie her se zabývají hlavně existenčními otázkami, MAS klade důraz na výpočetní aspekty (složitost, praktické využití).

Sociologie

- Klíčovým předmětem studia MAS jsou společenství agentů (agent societies).
- Sociologie se zabývá studiem lidských společenství.
- Stejně jako u AI a lidské inteligence se můžeme v MAS inspirovat u sociologie.
- Ale nemusíme (viz např. mnoho oblastí AI, teorie her nebo letadla).
- Sociologie (i ekologie a další vědy) rády používají agenty pro své simulace (Agent Based Model).

2. Inteligentí agenti

Lepší definice, teorie a metodologie

Jak začít MAS?

- Umět vytvořit alespoň jednoho agenta
- Klíčovým problémem návrhu agenta je **výběr akce**:
  - Co má agent v danou chvíli udělat na základě informací o vnějším prostředí.
- **Architektura agenta** je softwarová architektura, která umožní proces rozhodování – výběru akce.
  - … a particular methodology for building [agents]. It specifies how…the agent can be decomposed into the construction of a set of component modules and how these modules should be made to interact. The total set of modules and their interactions has to provide an answer to the question of how the sensor data and the current internal state of the agent determine the actions…and future internal state of the agent. An architecture encompasses techniques and algorithms that support this methodology. (Maes, 1991)

Co to je agent?

(Franklin, Graesser, 1996 – Is it an agent or just a program?)

- MuBot (mobilní agent Crystaliz, Inc)
  - *"The term agent is used to represent two orthogonal concepts. The first is the agent's ability for autonomous execution. The second is the agent's ability to perform domain oriented reasoning."*
- AIMA (Russell and Norvig)
  - *"An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors."*
- Pattie Maes, MIT Media Lab
  - *"Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed."*

... anebo?

- KidSim, Apple
  - *"Let us define an agent as a persistent software entity dedicated to a specific purpose. 'Persistent' distinguishes agents from subroutines; agents have their own ideas about how to accomplish tasks, their own agendas. 'Special purpose' distinguishes them from entire multifunction applications; agents are typically much smaller."*
- Barbara Hayes-Roth, Stanford Knowledge system lab
  - *"Intelligent agents continuously perform three functions: perception of dynamic conditions in the environment; action to affect conditions in the environment; and reasoning to interpret perceptions, solve problems, draw inferences, and determine actions."*

… a ještě …

- IBM, Intelligent Agent strategy white paper
  - *"Intelligent agents are software entities that carry out some set of operations on behalf of a user or another program with some degree of independence or autonomy, and in so doing, employ some knowledge or representation of the user's goals or desires."*
- SodaBot , MIT AI Lab
  - *"Software agents are programs that engage in dialogs [and] negotiate and coordinate transfer of information."*
- Brustoloni
  - *"Autonomous agents are systems capable of autonomous, purposeful action in the real world."*
- Software agents mailing list FAQ
  - *"This FAQ will not attempt to provide an authoritative definition …"*

Shrňme to

- Franklin:
  - *An **autonomous agent** is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future.*
- Wooldridge, Jennings:
  - *An **agent** is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its delegated objectives.*

Shrňme to česky

- Franklin:
  - *Autonomní agent je systém nacházející se v nějakém prostředí, jehož je součástí, vnímá toto prostředí a vykonává v něm akce, to vše v čase, za účelem plnění svých cílů a to tak, že ovlivňuje, co bude vnímat v budoucnu.*
- Wooldridge, Jennings:
  - *Agent je počítačový systém, který se nachází v nějakém prostředí, a který je schopen autonomní akce v tomto prostředí za účelem splnění svých delegovaných cílů.*


Prostředí poskytuje:
Vjemy, Feedback
Agent: Vnímání,
Rozhodování,
Akce
Senzory,
Efektory
imgflip.com

Prostředí

- *Plně pozorovatelné / Částečně pozorovatelné*
  - Agent svými senzory může pozorovat úplný stav prostředí.
  - Nemůže.
- *Statické / Dynamické*
  - Mění se jen jako důsledek akcí agenta.
  - I jinak.
- *Deterministické / Nedeterministické*
  - Každá akce má právě jeden zaručený výsledek.
  - Nemá.
- *Diskrétní / Spojité*
  - Má pevný konečný počet vejmů a akcí.
  - Má jich víc.
- Bohužel, většina zajímavých prostředí, jako reálný svět nebo internet, jsou spojitá, nedeterministická, dynamická a částečně pozorovatelná.

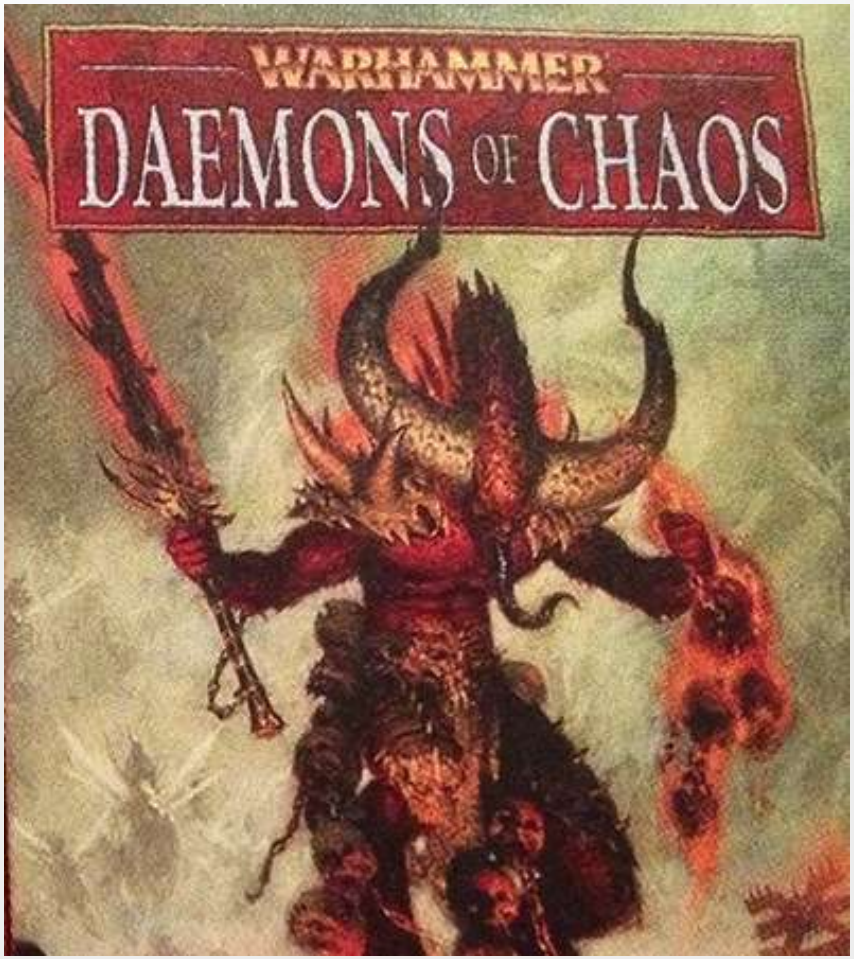
Poznámky k agentovi

- Autonomie:
  - Člověk vs. metoda v Javě
  - Agenti někde mezi - autonomně zvolit způsob řešení delegovaného problému, volba podcílů, ne cílů
- Rozhodování:
  - Agent má k dispozici repertoár akcí, ne všechny lze provést vždy
  - Klíčovým problémem agenta je vybrat nejlepší akci ke splnění svých cílů.
- Architektura agenta je tedy architekturou softwarového systému zapouzdřeného v okolí, která slouží k výběru akcí (embeded decision-making system)

Příklad: Termostat

- ne-inteligentní agent
- v prostředí (místnosti)
- 2 vjemy: zima, OK
- 2 akce: vyp, zap
- Rozhodovací jednotka:
  - zima => zap
  - OK => vyp
- nedeterministické prostředí (otevřené dveře)

Příklad: softwarový démon



- např unixový xbiff
- nachází se v prostředí operačního systému
- vnímá ho pomocí spouštění příkazů, signálů, …
- akce má softwarové – spusť poštovního klienta, změň ikonu
- rozhodovací algoritmus na úrovni termostatu

Inteligentní agent

- Reaktivní
  - Vnímá prostředí a je schopen reagovat v rozumném čase na jeho změny.
- Proaktivní
  - Má své cíle a je schopen aktivně je plnit.
- Sociální
  - Je schopen komunikovat s dalšími agenty, případně lidmi.

- Čistě reaktivní i čistě cílené/plánovité chování je jednoduché, složité je naopak vyvážit reaktivitu a proaktivitu.

Intencionální systém

- Často, když mluvíme o agentech, připisujeme jim *mentální stavy* (něčemu věří, chtějí, touží, doufají, …)
- Daniel Dennet definuje *intencionální systém* jako entitu, jejíž chování lze predikovat tím, že jí přiřadíme vlastnosti jako přesvědčení, touha a racionální důvtip.
- Hierarchie: I.S. prvního řádu, druhého (přesvědčení a touhy o přesvědčeních a touhách), třetího …
- Zajímavé je, že I.S. používáme (např. potkáme se na smluvené schůzce)

Příklad: Vypínač jako intencionální systém

- Shoham: *Vypínač světla v místnosti je (velmi) kooperující agent, který dle své vůle dokáže vést elektrický proud, ale dělá to jen tehdy, pokud věří, že my to chceme. Přepnutím dáváme vypínači najevo naše přání.*
- Konzistentní, elegantní, stručné, odpovídající chování vypínače.
- A přesto mnoha lidem to připadá absurdní, dětinské.
- Protože (asi) máme stejně věrné ale jednodušší vysvětlení.

Intencionální postoj

- Fyzikální postoj:
  - pro popis a predikci fungování stačí fyzikální zákony
  - (hodím kámen, nemusím mluvit o jeho touhách, abych zjistil, kdy a kam dopadne, stačí mi hmotnost ,rychlost, gravitační zákon).
- Projektový postoj (design stance):
  - U složitějších systému nemusím znát přesně fyzikální příčiny, stačí mi vědět účel za jakým vznikly. (teleologické vysvětlení)
  - (vím jak pracuje budík, nemusím znát fyzikální zákony, dokážu ho nařídit, predikovat, kdy zazvoní)
- Intencionální postoj:
  - k predikci použiji pohled intencionálního systému.

Intencionální postoj

- Někdy nemám fyzikální/projektový popis k dispozici,
- Někdy ano, ale jsou nepraktické pro popis/predikci chování systému
  - Např. I když mám schéma počítače a chápu elektroniku, těžko z toho odvodím, proč se po kliknutí v editoru na menu otevře nějaké okno.
- V mnoha situacích je intencionální postoj jednodušší než alternativy.
- Z hlediska informatiky je to abstrakce za účelem zvládnutí složitosti problému.
- Pro mnoho informatiků je programování prostřednictvím intencionálního postoje nejdůležitějším rysem MAS.

A co na to Daniel Dennet?

- Problém, zda existuje intencionalita entit sama o sobě, nebo zda se intencionalita realizuje až v procesu interpretace, řeší Dennett nemožností existence intencionality bez interpretace.

3. Abstraktní architektury agentů

Stavy, prostředí, agenti

Prostředí

- Snažíme se formalizovat abstraktní pohled na agenta a interakce s prostředím.
- Prostředí může být v jednom z konečného množství diskrétních stavů
- E = {e, e’, … }
- I kdyby prostředí nebylo doopravdy diskrétní, tak ho diskretizujeme
- To je standardní předpoklad u kdejakého modelování
- Každé spojité prostředí můžeme modelovat diskrétním prostředím s libovolnou přesností

Agent

- Agent má k dispozici konečné množství akcí
- A = {a, a’, …}
- Jak agent interaguje s prostředím:
  - Prostředí je v nějakém počátečním stavu
  - Agent si zvolí akci
  - Prostředí může odpovědět přechodem do několika stavů
  - Prostředí odpoví přechodem do jednoho konkrétního stavu
  - Není předem jasné do kterého
  - Na základě stavu agent odpoví další akcí
  - …

Běh

- **Běh** agenta v prostředí je posloupnost střídajících se stavů prostředí a akcí
- $r = e_0, a_0, e_1, a_1, e_2, a_2, \ldots a_{n-1}, e_n$
- Nechť:
- R je množina všech konečných posloupností nad E a A
- R_A je podmožina těch posloupností z R, které končí akcí z A
- R_E je podmnožina těch posloupností z R, které končí stavem prostředí

Funkce transformace stavu prostředí

- Vliv akcí agenta na prostředí modelujeme funkcí transformace stavu prostředí
- T: R_A->2^E
  - T zobrazuje běh agenta (končící akcí) na množinu stavů prostředí (těch, které mohou být výsledkem té akce)
- Prostředí mají historii
  - Následující stav prostředí nezávisí jen na předchozím stavu a na poslední akci agenta, ale i na dřívějších akcích/stavech.
- Prostředí jsou nedeterministická
  - Není jasné, který stav prostředí po vykonání akce nastane.

Prostředí ještě jednou

- Když je pro r z R_A T(r) = O, řekneme, že systém skončil běh.
- Tedy nejsou žádné další možné stavy po r
- Odteď si budeme myslet, že všechny běhy jednou skončí.
- Prostředí je trojice:
- Env = (E, e_0, T)
  - E je množina stavů prostředí
  - e_0 je počáteční stav
  - T je funkce transformace stavu prostředí

Agent ještě jednou

- Agent je funkce, která zobrazuje běhy (končící stavy prostředí) na akce:
- Ag: R_E -> A
  - Takže agent se rozhoduje, kterou akci provést na základě dosavadní historie celého systému.
  - Agent je deterministický
- AG je množina všech agentů
- Systém je dvojice obsahující agenta a prostředí
- Každý systém má množinu možných běhů
- R(Ag,Env) je množina běhů agenta Ag v prostředí Env
  - Uvažujeme jen takové běhy r, které končí, tj. T(r)=O

Agent běží prostředím

- Řekneme, že posloupnost $(e_0,a_0,e_1,a_1,e_2, \ldots)$ je **během agenta Ag v prostředí Env=(E,e_0,T),** když:
  1. e_0 je počáteční stav Env,
  2. $a_0 = Ag(e_0)$,
  3. pro každé u>0:
  - $e_u \in T((e_0,a_0,\ldots,a_{u-1}))$ a
  - $a_u = Ag((e_0,a_0,\ldots,e_u))$
- Ag_1 a Ag_2 jsou **funkčně ekvivalentní** vzhledem k Env, právě tehdy když platí $R(Ag_1,Env) = R(Ag_2,Env)$
- Ag_1 a Ag_2 **jsou jednoduše funkčně ekvivalentní**, jsou-li funkčně ekvivalentní vzhledem ke všem Env.

Čistě reaktivní agent

- Čistě reaktivní (neboli tropistický) agent:
- Ag: E -> A
- Reaguje bez ohledu na historii, jen na základě aktuálního stavu prostředí.
- Ke každému čistě reaktivnímu agentovi existuje ekvivalentní standardní agent (nemusí platit obráceně).
- Termostat:

```
Ag(e) = vyp, if e=teplota OK
Ag(e) = zap, jinak
```

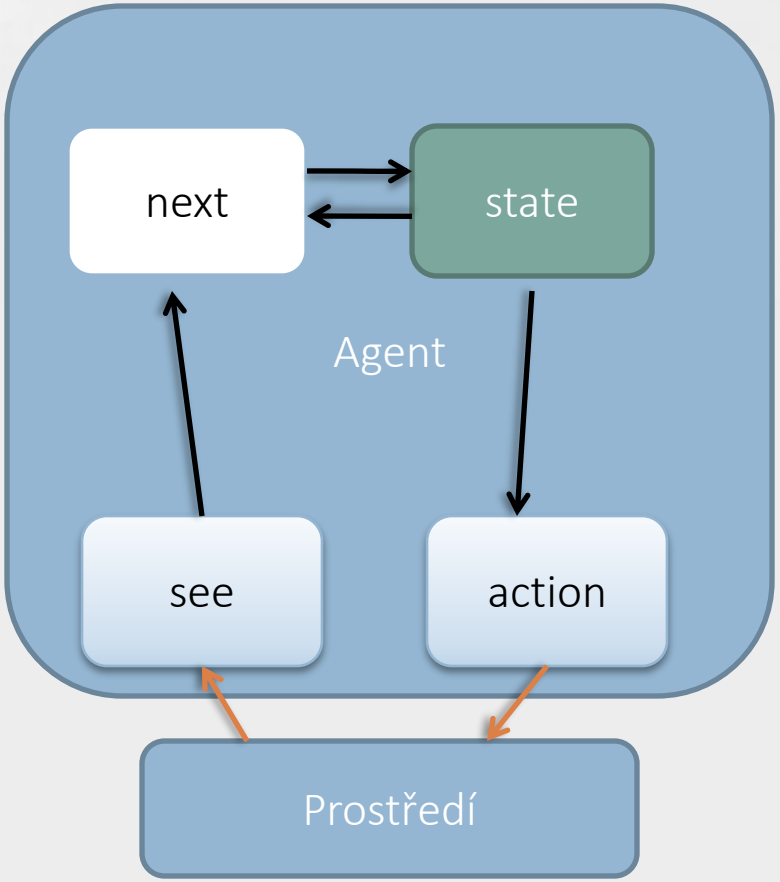
Agent se stavem

  - Stav zatím nebudeme definovat, je to prostě informace, typicky o stavech a historii prostředí, uchovávaná v interní datové struktuře, která pomáhá agentovi v rozhodování

- I množina všech interních stavů agenta
- Per je množina všech vjemů agenta
- see je funkce vnímání agenta see: E -> Per
- action: I -> A
- next: I x Per -> I
- Každého agenta se stavem lze převést na funkčně ekvivalentního standardního agenta.

Agent se stavem pracuje

- Agent začne v interním stavu i_0
- Vnímá stav prostředí e, a generuje vjem see(e)
- Aktualizuje si vnitřní stav na next(i_0,see(e))
- Vybere akci
  - action(next(i_0,see(e)))
- Tu udělá, a začne znovu další cyklus



Co má agent dělat a jak mu to říct

- Nechceme agenta zcela a pevně naprogramovaného
- Chceme mu říct co má dělat, ale ne jak to má dělat
- Oblíbenou cestou je definovat problém/cíl nepřímo pomocí nějaké míry úspěšnosti agenta
- Utility/Zisk – účelová funkce, ohodnocení stavů prostředí
- u: E -> $\mathscr{R}$
- Agent má pak za úkol dostat prostředí do stavů s vysokou hodnotou zisku

Zisk běhu

- Celková úspěšnost agenta v prostředí:
  - zisku nejhoršího stavu, kterým agent prošel
  - průměrného zisku stavů, kterými agent prošel
    - Není jasné, co je lepší, záleží na úkolu
- Oceňovat jednotlivé stavy je možná příliš krátkozraké, jak ale ocenit dlouhodobější pohled na úspěšnost agenta
- Zisk běhu r:
- u: R -> $\mathscr{R}$
- Lepší pro agenty běžící nezávisle delší dobu

Př: Tileword

  - Agenti na šachovnici, pohybuji se do 4 směrů
  - Jsou tam díry, překážky a kostičky
  - Agent má strkat kostky do děr
  - Dynamické prostředí - díry, překážky i kostky náhodně vznikají a zanikají
- $u(r) = N_f/N_a$,
  - N_f – počet děr zacpaných kostkami v běhu r
  - N_a – počet všech děr v běhu er
- A to celé několikrát
- Agent musí reagovat na změny prostředí
  - (kostka, kterou tlačím, zmizí)
- Agent musí být schopen využít příležitost, když nastane
  - (nová kostka se objeví vedle mě)

Maximalizace očekávaného zisku

- Má-li funce u(r) horní mez, můžeme mluvit o maximalizaci:
- Optimální agent je ten, který maximalizuje očekávaný zisk.
- P(r|Ag,Env) – pravděpodobnost běhu r agenta Ag v prostředí Env
- $\sum P(r|Ag,Env) = 1$, pro všechna $r \in R(Ag,Env)$
- Ag_{opt} optimální agent:
  - $Ag_{opt} = \arg\max \{Ag \in AG\} \sum u(r)\, P(r|Ag,Env)$
- To nám ale neříká nic o tom, jak takového agenta sestrojit
- A někdy je komplikované sestrojit i funkci zisku

Predikátová specifikace úlohy

- Utility je funkce do {0,1}
  - Běh je úspěšný, když u(r)=1
- F je predikátová specifikace (zatím se nezabýváme tím, jak vlastně vypadá):
  - F(r) je splněn, právě tehdy, když u(r)=1
- Prostředí úlohy (Env,F):
  - Env je prostředí
  - F: R ->{0,1}
- TE je množina všech prostředí úloh
- Prostředí úlohy specifikuje:
  - Vlastnosti systému (prostřednictvím Env)
  - Kritéria, zda agent úlohu splnil (skrze F)

Prostředí úlohy

- RF(Ag,Env) je množina běhů agenta Ag v prostředí Env, které splňují F
  - RF(Ag,Env) = {r|r ϵ R(Ag,Env) & F(r)}
- Kdy agent Ag úlohu (Env,F) vyřeší:
  - Pesimista: RF(Ag,Env) = R(Ag,Env)
    - Neboli každý běh splní F
  - Optimista: ∃r ϵ R(Ag,Env) tak, že F(r)
    - Neboli, aspoň jeden běh agenta splní F
  - Realista:
    - Rozšíříme T tak, aby zahrnul rozložení pravděpodobnosti přes možné výsledky (a tedy i přes všechny běhy)
    - Úspěch agenta je pak pravědpodobnost splnění F:
    - P(F|Ag,Env) = ∑ P(r|Ag,Env) pro r ϵ RF(Ag,Env)

Typy úloh (ze života)

- Něco najít (Achievment tasks)
  - Dosáhnout nějakého z množiny cílových stavů G
  - G je množina stavů z E takových, že F(r) platí pokud alespoň jeden ze stavů z G se objeví v r
  - Agent je úspěšný, když každý jeho běh skončí stavem z G
  - Př: Kdejaká úloha z AI (hledám řešení)
- Něco zachovat (Maintenance tasks)
  - Agent naopak musí zabránit některým stavům prostředí
  - B je množina stavů taková, že F(r) je nepravda pokud se některý ze stavů z B objeví v r
  - Př: hry, B jsou prohrávající stavy, prostředí je oponent
- Kombinace: např. dosáhni stavu z G a vyvaruj se stavů z B

4. Deduktivně usuzující agent

Indukce, dedukce, agent jako dokazovač vět

Klasický přístup v AI

- Tradiční přístup jak AI tvoří „inteligentní systém“:
  - Symbolická reprezentace prostředí a chování
    - Zaměříme se na reprezentaci logickými formulemi
  - Syntaktická manipulace s touto reprezentací
    - Ta pak odpovídá logické dedukci či dokazování vět
  - Takže teorie o tom, jak se agent chová (viz např. minulá kap.) je rovnou spustitelnou specifikací, dává nám konkrétní akce agenta.
- Problém transdukce: Jak ten svět reprezetovat
  - *"Grau, teurer Freund, ist alle Theorie, und Grün des Lebens goldner Baum."*
  - Vidění, zpracování přirozeného jazyka, učení, …
- Problém reprezentace a usuzování
  - Způsoby reprezentace znalostí, dokazovače, plánování, …

Mimochodem

- **Dedukce:** odvození nutně pravdivých závěrů za předpokladu, že předpoklady platí
  - obecné -> specifické
    - Sylogismy: „Všichni lidé jsou smrtelní, Sokrates je člověk, takže Sokrates je smrtelný“
- **Indukce:** jsou-li splněny předpoklady, pak závěr spíš platí než neplatí.
  - zvláštní případ -> obecné
    - Policie říká, že S. je vrah (/viděli ho dva svědci/nechal tam otisky/přiznal se), tak je S vrah.
- Sherlock Holmes indukuje, i když říká, že dedukuje.
- Matematická indukce je dedukce.

Agent dokazovač

- L je množina formulí v logice prvního řádu, $D=2^L$ je množina databází formulí L, vnitřní stav DB agenta je pak DB ϵ D.
- Rozhodování se děje prostřednictvím odvozovacích pravidel P v dané logice
  - DB $\vdash_P$ f – formuli f odvodíme z DB (jen dedukčními) pravidly P
- see: S -> Per
- next: D x Per -> Per
- action: D -> A podle pravidel P

Výběr akce jako dokazování

```
Function action(DB:D) returns action A
begin
   for each a ϵ A do
      If DB ⊢P Do(a) then return a
   for each a ϵ A do
      If DB !⊢P ¬Do(a) return a
return null
end
```

- vrací akci, pro kterou jde dokázat Do(a)
- anebo zkusí najít akci konzistentní (ne v rozporu s DB)

Příklad: Vacuum world 3x3

- In(x,y) – agent je na (x,y)
- Dirt(x,y) – je tam špína
- Facing(d) – agent se dívá směrem d
- Odvozování akcí:
- Vysaj:
  - In(x,y)& Dirt(x,y) => Do(suck)
- A projeď ten svět, např 00-01-02-12-11-10-…

- In(0,0) & Facing(north) & not Dirt(0,0) => Do(forward)
- In(0,1) & Facing(north) & not Dirt(0,1) => Do(forward)
- In(0,2) & Facing(north) & not Dirt(0,2) => Do(turn)
- In(0,2) & Facing(east) => Do(forward)
- …

Klady a zápory

- Je to elegantní a má to krásnou sémantiku
- Je to dost nepraktické
- Dlouho to trvá, nebo vůbec neskončí
  - Spočitatelná racionalita: agent na základě stavu prostředí něco odvodí, ale mezitím se prostředí změní a akce už není optimální. To je špatné pro rychle se měnící prostředí.
- Někdy se těžko hledá reprezentace pomocí see()
  - Jak z obrázku dostat formule
  - Jak reprezentovat časová data
- Uvidíme prvky tohoto přístupu v dalších architekturách – praktické uvažování

5. Prakticky usuzující agent

Beliefs-Desires-Intentions

Praktické usuzování

- Inspirace lidským rozhodováním
- Teoretické usuzování (viz Sokrates) má vliv jen na to, co se domníváme o světě
- Praktické usuzování vede k akcím
- Dvě fáze:
  - Rozhodování/Deliberation
    - Co chceme dosáhnout
      - Chci vystudovat
  - Koncové usuzování/Means-Ends reasoning
    - Jak toho chceme dosáhnout
      - Musím přijít s plánem, jak to provést
- A to vše nesmí trvat dlouho

Záměry

- Záměr (intention) je takový stav světa, kterého chce agent dosáhnout
- Záměry u agenta vedou k akcím (za účelem dosažení toho stavu), následuje po nich usuzování, které vyústí v plán
- Záměry přetrvávají
  - Dokud se jich nedosáhne,
  - Dokud si agent nezačne myslet, že je nemůže splnit
  - Dokud nezmizí důvody k tomuto závěru
- Závěry omezují další deliberaci
- Závěry ovlivňují, co si bude agent myslet v budoucnu

Touhy

- Touha (desire) je jeden z možných záměrů.
  - *My desire to play basketball this afternoon is merely a potential influencer of my conduct this afternoon. It must vie with my other relevant desires … before it is settled what I will do. In contrast, once I intend to play basketball this afternoon, the matter is settled: I normally need not continue to weigh the pros and cons. When the afternoon arrives, I will normally just proceed to execute my intentions. (Bratman, 1990)*
- Touhy reprezentují motivační stav agenta

Domněnky

- Domněnky (Beliefs) shrnují znalosti agenta, jsou vlastně jeho informačním stavem.
- Neříkáme jim znalosti (knowledge), abychom zdůraznili, že:
  - jsou subjektivní z hlediska agenta,
  - nemusí být pravdivé,
  - a mohou se v budoucnu změnit.
- Mohou obsahovat i inferenční pravidla umožňující dopředné řetězení

B ... D ... I

- Uvažujme nějakou explicitní reprezentaci domněnek, tužeb a záměrů, např. symbolickou, ale konkrétní implementace není podstatná.
- B
  - proměnná, která obsahuje agentovy aktuální domněnky,
  - Bel je mna všech takových domněnek
- D
  - proměnná pro touhy,
  - Des množina všech tužeb
- I
  - proměnná pro záměry,
  - Int množina všech záměrů

Rozhodování

- Funkce generování možností
  - options: $2^{Bel} \times 2^{Int} \rightarrow 2^{Des}$
- Funkce filtru = výběru z možností
  - filter: $2^{Bel} \times 2^{Des} \times 2^{Int} \rightarrow 2^{Int}$
- Funcke aktualizace domněnek
  - brf: $2^{Bel} \times Per \rightarrow 2^{Bel}$

Means-Ends Reasoning neboli Plánování

- Proces rozhodování, jak dosáhnout cíle (záměr) pomocí dostupných prostředků (akcí).
- Vstup:
  - Cíl = záměr
  - Aktuální stav prostředí = domněnky agenta
  - Akce, které má agent k dispozici
- Výstup:
  - Plán = posloupnost akcí,
  - Když je agent provede, cíl bude splněn

STRIPS

- Nilsson, Fikes, 1971
- Model světa = množina formulí v logice prvního řádu
- Množina schémat akcí:
  - Podmínky/Preconditions
  - Následky/Effects:
    - Add – fakta, která budou nově platit
    - Delete – fakta, která už nebudou platit
- Plánovací algoritmus:
  - Najde rozdíly mezi koncovým a aktuálním stavem světa
  - Zmenší rozdíly aplikací vhodné akce
  - To je sice rozumné, ale celkem nepraktické, algoritmus se často ztratí v low-level detailech

Svět kostiček

- Predikáty:
  - On(x,y) = x je na y
  - OnTable(x,y) = x je na stole
  - Clear(x) = na x nic není
  - Holding(x) = robot drží x
  - ArmEmpty = robot nic nedrží
- Počáteční stav:
  - {Clear(A), On(A,B), OnTable(B), OnTable(C), Clear(C)}
- Cíl:
  - {OnTable(A), OnTable(B), OnTable(C)}

Svět kostiček - akce

- Stack(x,y)
  - Pre {Clear(y), Holding(x)}
  - Del {Clear(y), Holding(x)}
  - Add {ArmEmpty, On(x,y)}
- UnStack(x,y)
  - Pre {On(x,y), Clear(x), ArmEmpty)}
  - Del {On(x,y), ArmEmpty)}
  - Add {Holding(x), Clear(y)}
- Pickup(x)
  - Pre {OnTable(x), Clear(x), ArmEmpty)}
  - Del {OnTable(x), ArmEmpty)}
  - Add {Holding(x)}
- PutDown(x)
  - Pre {Holding(x)}
  - Del {Holding(x)}
  - Add {ArmEmpty, OnTable(x)}

Definice plánů

- Množina akcí Ac = $\{a_1, \ldots, a_n\}$
- Deskriptor akce a je $[P_a, D_a, A_a]$:
  - Pa je množina formulí v FOL určující podmínku akce a
  - Da je mna formulí v FOL, které přestanou platit po akci a
  - Aa je mna formulí v FOL, které začnou platit po akci a
  - Pro jednoduchost obsahují jen uzemněné atomy – žádné spojky či proměnné
- Plánovací problém je $[B_0, O, G]$:
  - B_0 – domněnky agenta o počátečním stavu světa
  - O = $\{[P_a, D_a, A_a]: a \in Ac\}$ deskriptor pro každou akci
  - G = mna flí v FOL reprezentující cíl/záměr

Definice plánování

- Plán p je posloupnost $(a_1,\ldots,a_n)$, $a_i \in Ac$
  - Plán p pro plánovací problém $[B_0,O,G]$ určuje posloupnost databází domněnek $B_0, B_1, \ldots, B_n$:
    - $B_i = (B_{i-1} \setminus D_{ai}) \cup A_{ai}$; pro i = 1, …, n
  - Plán p je přípustný pro $[B_0,O,G]$ ⇔
    - $D_{i-1} \models P_{ai}$; pro i = 1, …, n
  - Plán p je korektní pro $[B_0,O,G]$ ⇔
    - p je přípustný pro $[B_0,O,G]$ a
    - $D_{i-1} \models G$.
- Plánování: pro $[B_0,O,G]$ najdi korektní plán, nebo řekni, že neexistuje.

- Plan = množina plánů (nad Ac)
- pre(p) = podmínka plánu p
- body(p) = tělo plánu p
- empty(p) = booleovská fce určující, že je plán prázdný
- execute(p) = procedura plnící plán (postupně všechy jeho akce)
- hd(p) = první akce v těle plánu
- tail(p) = druhá až poslední akce v těle plánu
- sound(p,I,B) = plán p je korektní plán pro množiny cílů I a domněnek B

Agentova plánovací funkce

- plan: $2^{Bel} \times 2^{Int} \times 2^{Ac}$ -> Plan
- Agent nemusí plány konstruovat na místě
- Často je plan() implementována pomocí knihovny plánů
- Pak stačí jednou projít knihovnu a otestovat zda:
- podmínky plánu odpovídají agentovým domněnkám
- následky plánu odpovídají cíli

Implementace

- B:= B_0; I:= I_0
- while true do
  - v:= see(); B:= brf(B,v); D:= options(B,I); I:= filter(B,D,I);
  - p = plan(B,I,Ac);
  - while not (empty(p) or succeed(I,B or impossible (I,B)) do
    - a:= hd(p); execute(a); p:= tail(p);
    - v:= see(); B = brf(B,v);
    - if reconsider(I,B) then D:= options(B,I); I:= filter(B,D,I) endif
    - if not sound(p,I,B) then p = plan(B,I,Ac);
  - endwhile
- endwhile

Jak urputný je agent

- Mechanismus, kdy opustit cíl = strategie oddanosti:
- Slepá oddanost
  - Agent pokračuje v práci na cíli, dokud nebude splněn cíl, přesněji, domnívá se, že splnil cíl.
- Myslí jen na to jedno
  - Agent pracuje, dokud nevěří, že cíl je splněn, anebo že cíl už splnit nejde.
- Mysl otevřená
  - Cíl zůstává dokud se o něm věří, že to je možné.

Jak urputný je agent vzhledem k plánům

- Agent myslí jen na ten jeden plán, neboli
  - skončí, když
    - Věří, že cíl je splněn
    - Věří že cíl je možný
    - Plán je prázdný
- succeeded(I,B) = I platí za předpokladu B
- impossible(I,B) = I nemůže platit z B

Jak urputný je agent vzhledem k cílům

- Kdy se má agent zastavit a přehodnotit své záměry?
- Klasické dilema – deliberace trvá dost času, prostředí se mění pod rukama (za zády)
  - Agent, který nepřehodnocuje záměry dost často, může plnit záměr, který už nemá smysl plnit.
  - Agent, který přehodnocuje záměry pořád, nemá dost času na jejich plnění a tak třeba nic nesplní.
- meta-řízení (Wooldridge, Parsons, 1999)
  - funkce reconsider(), která je výpočetně jednodušší než rozhodování
  - Pozorování: reconsider() se chová dobře, když doporučí-li rozhodování, agent skutečně změní záměr

„To boldly go where no man has gone before.“

- Klasická extrémní řešení dilematu:
  - Odvážný (bold) agent – rozmýšlí se pouze ve chvíli, kdy skončí provádění aktuálního plánu
  - Opatrný agent – rozmýšlí se po každé akci
- Míra odvahy – kolik akcí mám provést před rozmýšlením
- Rychlost změny prostředí – kolikrát se mi změní prostředí za 1 cyklus agenta
- efektivita agenta = splněné záměry / všechny záměry
  - Když se okolní svět mění pomalu, odvážný agent je efektivní
  - Když se prostředí mění rychle, opatrný agent je lepší

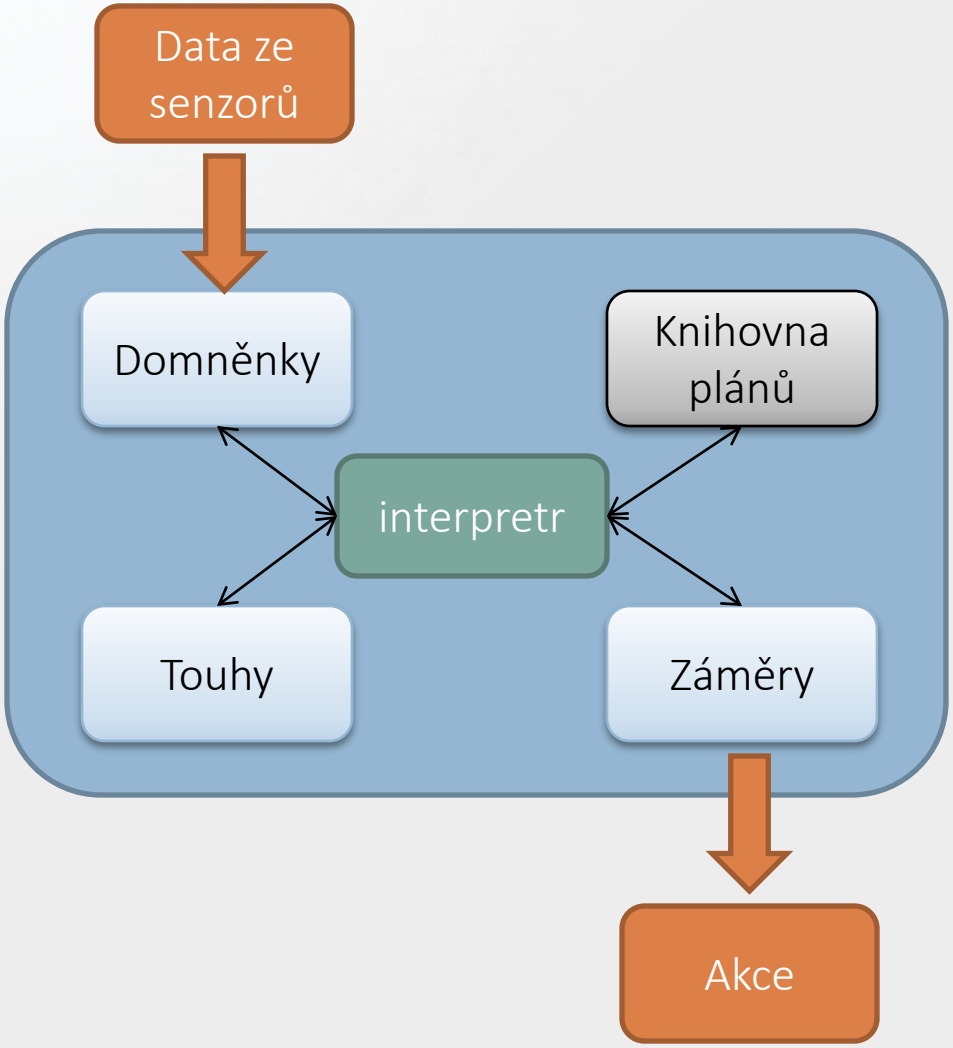
6. Procedural Reasoning System

BDI architektura v praxi

PRS

- (Georgeff et al, 80s), Stanford
- První pokus o implementaci BDI
- Asi nejúspěšnější agentní architektura, mnohokrát reimplementována
  - AgentSpeak/Jason
  - Jam
  - Jack
  - JADEX
- A použita v praxi
  - OASIS řízení letového provozu v Sydney
  - SPOC (single point of contact) organizace byznys procesů
  - SWARMM – vojenský letecký simulátor

PRS agent


Data ze senzorů
Domněnky
Knihovna plánů
interpretr
Touhy
Záměry
Akce

Plán v PRS

- Agent má k dispozici knihovnu plánů
- Takže sám neplánuje, jen vybírá
- Plán
  - Cíl – podmínka platná po provedení
  - Kontext – podmínka nutná ke spuštění
  - Tělo –akce, které se mají provést
- Tělo plánu:
  - Nejen lineární posloupnost akcí (jako dříve)
  - Plán může obsahovat cíle
    - Achive f,
    - achieve f or g,
    - keep achieving f until g

Plánování v PRS

- Na počátku:
  - domněnky B0 (prologovské atomy FOL)
  - top-level cíl
- Zásobník záměrů
  - Zásobník obsahuje aktuálně rozpracované cíle
  - Interpretr vezme cíl na vrcholu a hledá v plánech, zda se jejich cíl shoduje
  - Z nich jen některé mají splněn kontext vzhledem k aktuálním domněnkám
  - To jsou všechny aktuální touhy/options/desires

Rozhodování v PRS

- Rozhodování – výběr záměru z tužeb
  - Původní PRS měl meta-plány
    - Plány o plánech, modifikovaly záměry agenta
    - Ale to je těžkopádné
  - Utility
    - Každý plán je ohodnocen číselnou hodnotou očekávaného zisku
    - Vybere se ten s největší
- Vybraný plán se spustí, což může vyústit v přidávání dalších záměrů na zásobník …
- Když plán selže, agent si z options vybere jiný

Příklad: JAM

```
GOALS:
  ACHIEVE blocks_stacked;
FACTS:
  FACT ON "Block5" "Block4";      FACT ON "Block4" "Block3";
  FACT ON "Block1" "Block2";      FACT ON "Block2" "Table";
  FACT ON "Block3" "Table";       FACT CLEAR "Block1";
  FACT CLEAR "Block5";            FACT CLEAR "Table";

Plan: {
  NAME: "Top-level plan"
  GOAL: ACHIEVE blocks_stacked;
  CONTEXT:
  BODY:     ACHIEVE ON "Block3" "Table";
            ACHIEVE ON "Block2" "Block3";
            ACHIEVE ON "Block1" "Block2";
}

Plan: {
  NAME: "Stack blocks that are already clear"
  GOAL: ACHIEVE ON $OBJ1 $OBJ2;
  CONTEXT:
  BODY:     ACHIEVE CLEAR $OBJ1;
            ACHIEVE CLEAR $OBJ2;
            PERFORM move $OBJ1 $OBJ2;
  UTILITY: 10;
  FAILURE: EXECUTE print "\n\nStack blocks failed!\n\n";
}

Plan: {
  NAME: "Clear a block"
  GOAL: ACHIEVE CLEAR $OBJ;
  CONTEXT: FACT ON $OBJ2 $OBJ;
  BODY:     ACHIEVE ON $OBJ2 "Table";
  EFFECTS: RETRACT ON $OBJ1 $OBJ;
  FAILURE: EXECUTE print "\n\nClearing block failed!\n\n";
}
```

Příklad: Jason

```
/* Initial beliefs */
// initially, I believe that there are some beers in the fridge
available(beer,fridge).
// my owner should not consume more than 10 beers a day
limit(beer,10).

/* Rules */
too_much(B) :-
.date(YY,MM,DD) &
.count(consumed(YY,MM,DD,_,_,_,B),QtdB) &
limit(B,Limit) &
QtdB > Limit.

/* Plans */
@h1
+!has(owner,beer)
: available(beer,fridge) & not too_much(beer)
<- !at(robot,fridge);
open(fridge);
get(beer);
close(fridge);
!at(robot,owner);
hand_in(beer);
// remember that another beer will be consumed
.date(YY,MM,DD); .time(HH,NN,SS);
+consumed(YY,MM,DD,HH,NN,SS,beer).

@h2
+!has(owner,beer)
: not available(beer,fridge)
<- .send(supermarket, achieve, order(beer,5));
!at(robot,fridge). // go to fridge and wait there.

@h3
+!has(owner,beer)
: too_much(beer) & limit(beer,L)
<- .concat("The Department of Health does not allow me ",
"to give you more than ", L,
" beers a day! I am very sorry about that!",M);
.send(owner,tell,msg(M)).

@m1
+!at(robot,P) : at(robot,P) <- true.3.4. EXAMPLE: A COMPLETE
        AGENT PROGRAM 63

@m2
+!at(robot,P) : not at(robot,P)
<- move_towards(P);
!at(robot,P).
// when the supermarket finishes the order, try the 'has'
// goal again

@a1
+delivered(beer,Qtd,OrderId)[source(supermarket)] : true
<- +available(beer,fridge);
!has(owner,beer).
// when the fridge is openned, the beer stock is perceived
// and thus the available belief is updated
@a2
+stock(beer,0)
: available(beer,fridge)
<- -available(beer,fridge).
@a3
+stock(beer,N)
: N > 0 & not available(beer,fridge)
<- +available(beer,fridge).
```

Příklad: ARTS

GOALS:
ACHIEVE PrepareLecture agents101 :PRIORITY 9 :DEADLINE 50;
ACHIEVE HaveLunch :PRIORITY 7 :DEADLINE 40;
ACHIEVE BorrowBook R&N :PRIORITY 2 :DEADLINE 30;
CONCLUDE LectureNotes agents101 myNotes;
PLAN: {NAME: ”Plan 1”; DOCUMENTATION: ”Prepare for lecture”;
CUE: ACHIEVE PrepareLecture $x, y;
PRECONDITION: TEST LectureNotes $x, y;
BODY:
EXECUTE revise-lecture $y :TIMEOUT 35;}

PLAN: {NAME: ”Plan 2”; DOCUMENTATION: ”Pickup a book from the library”;
CUE: ACHIEVE BorrowBook $x;
BODY:
EXECUTE goto library :TIMEOUT 10;
ACHIEVE Pickup $x;}

PLAN: {NAME: ”Plan 3”; DOCUMENTATION: ”Pick up something”;
CUE: ACHIEVE Pickup $x;
BODY:
EXECUTE pickup $x :TIMEOUT 2;}

PLAN: {NAME: ”Plan 4”;DOCUMENTATION: ”Have lunch”;
CUE: ACHIEVE HaveLunch;
BODY:
EXECUTE eat-sandwich :TIMEOUT 20;}

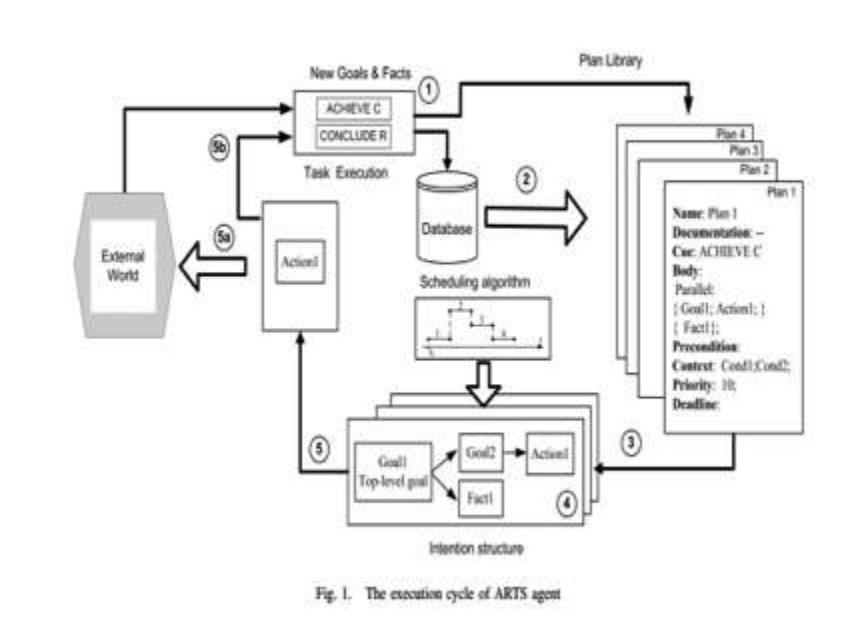


Fig. 1. The execution cycle of ARTS agent

ACHIEVE C	achieve condition C
CONCLUDE F	add fact F to the database
TEST C	test for the condition C
RETRACT F	retract fact F from database
WAIT C	wait until condition C is true

7. Reaktivní a hybridní architektury

Brooksova subsumpční architektura, horizontální a vertikální vrstevnaté architektury, Stanley

Reaktivní přístup

- Problémy se symbolickou reprezentací a zdůvodňováním
- 80.-90.léta – malé změny nestačí
- Alternativní paradigmata AI
- Odmítnutí symbolické reprezentace a dedukce založené na syntaktické manipulaci
- Interakce - Inteligentní chování je odvislé od prostředí, ve kterém se agent nachází
- Embodiment - Inteligentní chování není jen logika, ale je produktem vtěleného agenta
- Emergence – inteligentní chování vzniká interakcí jednoduchých chování

Reaktivní agenti

- Behaviorální – důraz na vývoj a kombinace jednotlivých chování
- Situované – agent v prostředí, embodiment
- Reaktivní
  - Agent hlavně (pouze) reaguje na prostředí
  - Agent neprovádí uvažování
- Subsymbolická reprezentace
  - Konekcinoismus
  - Konečné automaty
  - Jednoduchá reaktivní pravidla

Subsumpční architektura agenta

- Asi nejúspěšnější y reaktivních přístupů
- Rodney Brooks, 1991
- Inteligentní chování lze vytvořit bez symbolické reprezentace
- Inteligentní chování lze vytvořit bez explicitního abstraktního uvažování-zdůvodňování
- Inteligence je emergentní vlastností dostatečně komplexních systémů
  - Sitouvanost a vtělenost
  - Inteligence jako produkt interakce a emergence

Architektura

- Agent se rozmýšlí prostřednictvím jednoduchých cílených chování
  - Každé chování je mechanismem výběru akce
  - Každé chování dostává vjemy a transformuje je na akce
  - Každé chování má na starost určitý úkol
- Konečný automat
  - Neexistuje složitá symbolická reprezentace
  - Neexistuje zdůvodňovací mechanismus
- Pravidla situace -> akce
- Chování pracují paralelně
- Chování jsou v subsumpční hierarchii určující jejich priority

Výběr akce

- Na vstupy reaguje subsumpční mechanismus:
  - Výběr pravidel, která odpovídají situaci
  - Pro každé chování, které je aplikovatelné (fires) se kontroluje, zda existuje chování s vyšší prioritou
  - Pokud ne, vybere se toto chování a jeho akce
  - Pokud se nic nevybere, nic se neprovede

- Je to jednoduché (ale není to úplně snadné naprogramovat)
- Je to rychlé (hw implementace, konstantní odezva)

Příklad: Steelsův průzkumník Marsu

- Cílem je sbírat cenný materiál na jiné planetě pomocí hejna robotických průzkumníků

- Prostředky:
- stanice vysílá navigační signál, není třeba komunikovat, stačí mít gradient signálu
- Každý robot má k dispozici radioaktivní drobečky pro nepřímou komunikaci mezi sebou

První přiblížení – náhodná procházka a návrat

- R1: if detect obstacle then change direction
- R2: if carying sample and at the base then drop sample
- R3: if carying sample and not at the base then travel up gradient
- R4: if detect a sample then pick sample up
- R5: if true the move randomly

- Priority!

Druhé přibížení: lepší prohledávání

- Co dělat když nesu vzorek:
- R6: if carrying a sample and at the base drop sample
- R7: if carrying a sample and not the base then drop 2 crumbs and travel up gradient
- R8: if sense crumb then pickup 1 crumb and travel up gradient

- Priority: R1 < R6 < R7 < R4 < R8 < R5

Architektura sítě agentů

- Maes, 1991, Agent network architecture
- Každý agent je množina kompetenčních modulů (něco jako Brooksova chování)
- Každý modul má
  - Preconditions
  - Postconditions
  - Aktivační práh (udává, jak relevantní je k dané situaci, působí jako priorita při výběru akce)
- Moduly jsou propojeny do sítě podle svých podmínek (odpovídající pre a post) – následnický spoj
- A ještě další spoje (předcházení, konflikt)
- Moduly se v síti aktivují a nejvíc aktivovaný určí akci

Omezení reaktivních architektur

- Nedělají si model, musejí vše odvodit z prostředí (takže si ho třeba musí změnit radioaktivními drobky)
- Krátkodobý pohled – jednají na základě svého aktuálního stavu a lokální informace, těžko pracovat s globálními podmínkami a dlouhodobými plány
- Emergence chování není dobrý inženýrský přístup k programování
- Mnoho vrstev se špatně navrhuje

Hybridní architektury

- Horizontální
- Vrstvy jsou propojeny se senzory a efektory paralelně
- Velmi jednoduché, ale vrstvy se mohou ovlivňovat – mediátorová funkce – řeší konflikty (bottleneck)

- Vertikální
- Vrstvy jsou propojeny se senzory a efektory sériově
- Jednoprůchodové
  - Přirozené uspořádání a hierarchie
- Dvouprůchodové
  - Nahoru vjemy, dolů akce
  - Připomíná to firmy

Touring Machines

- Horizontální třívrstvá architektura
- Modelovací vrstva
  - Modeluje agenta a prostředí, určuje konflikty, cíle, ty dává plánovací vrstvě
- Plánovací vrstva
  - Proaktivní chování, výběr y předprogramovaných plánů ala PRS
- Reaktivní vrstva
  - Klasická reaktivní pravidla (vyhýbání překážkám)

InteRRaP

- Vertikální dvouprůchodová architektura se třemi vrstvami
- Kooperativní plánovací vrstva
  - Sociální interakce
- Lokální plánovací vrstva
  - Day-to-day planning
- Behaviorální vrstva
  - Reaktivní
- Každá vrstva má svou bazi znalostí na jiné úrovni abstrakce

Stanley

- Volkswagen Touareg R5
- Vyhrál DARPA Grand Challenge 2005 – ujel 132 mil v Mohavské poušti
- Vrstva senzorů
- Vrstva vjemů
- Plánovací a řídící vrstva (plán cesty a řízení)
- Vrstva rozhraní vozidla
- Vrstva uživatelského rozhraní (panel, start)
- Vrstva globálních služeb (filesystém, komunikace, hodiny)

8. Ontologie

Jak si navzájem porozumět, od slovníků k is-a, od Aristotela k programování

Od agentů k MASům

- Už víme, jak udělat agenta několika způsoby
- Jak mají agenti komunikovat/spolupracovat v MAS
  - Reprezentace znalostí
  - Společné “slovníky”
  - “Standardní” zprávy
  - Předvídatelné chování při komunikaci – komunikační protokoly
  - Sémantika komunikace
  - Technické problémy spojené s komunikací
    - Distribuovanost
    - Mobilita
    - Asynchronnost
    - Nespolehlivé kanály

Ontologie

- Ontologie (z řeckého to óv jsoucí + λόγος, logos slovo, řeč) je filosofická disciplína, která se zabývá jsoucnem, bytím jako takovým a základními pojmy. Aristotelés pro ni používá označení první filosofie, která je součástí metafyziky a zabývá se nejobecnějšími otázkami.
- Ontologie je v informatice výslovný (explicitní) a formalizovaný popis určité problematiky. Je to formální a deklarativní reprezentace, která obsahuje glosář (definici pojmů) a tezaurus (definici vztahů mezi jednotlivými pojmy). Ontologie je slovníkem, který slouží k uchovávání a předávání znalosti týkající se určité problematiky.

Historie ontologií v AI

- "Toward Principles for the Design of Ontologies Used for Knowledge Sharing”, T. Gruber, 1995
  - An ontology is a description (like a formal specification of a program) of the concepts and relationships that can formally exist for an agent or a community of agents. This definition is consistent with the usage of ontology as set of concept definitions, but more general. And it is a different sense of the word than its use in philosophy.
- "The Semantic Web”, T. Berners-Lee, Scientific American Magazine, 2001.
  - “… remains largely unrealized”, 2006

Ontologie, příklad ze života

- Anna: Už jsi slyšel 7777?
- Boris: Ne, co to je?
- Anna: Nové CD kapely SRPR, je to takovej alternativní rock, vlastně I elektronika, a má to moc dobrý texty.
- Boris: Aha.

- 7777 je hudební album
- 7777 spadá do alt.rocku, a zároveň do elektronické hudby
- 7777 má písně, interprety (SRPR), autory hudby, textu
- SRPR je kapela, má členy, ty asi hrají na nástroje
- …

Ontologie obecně

- Třídy – věci, které mají něco společného
- Instance (objekty) – konkrétní příklady tříd
- Vlastnosti
- Vztahy mezi třídami
- Podtřída
  - Tranzitivní relace
- Další vlastnosti a vztahy (background knowledge?)
- Strukturální část – té se obvykle říká ontologie
- Fakta o konkrétních členech
- Dohromady často jako báze znalostí

Ontologie ontologií

- Od slabších neformálních systémů k silným formálním
  - Slovník (controlled vocabulary) – vybrané termíny
  - Glosář – definice významů pomocí vybraných termínů, často v přirozeném jazyce
  - Tezaurus – definice synonym
  - Neformální hierarchie – naznačeny podtřídy, Amazon, wiki
  - Formální is-a hierarchie – subsumpce tříd formálně
  - Třídy s vlastnostmi
  - Omezení hodnot (value restrictions) – každý člověk má 1 matku
  - Libovlná logická omezení (constrains) – typicky složité zdůvodňovací algoritmy

Jiná ontologie ontologií

- Aplikační
  - Nejspecifičtější, nejméně lze využít i jinde
- Doménová
  - Častá oblast výzkumu i praktického využití
- Horní (upper)
  - Věc, živá věc, důkaz, vegetarián …
  - Slouží jako základ pro doménové ontologie
  - BFO, GFO, UFO
  - Wordnet, IDEAS – není určena k formálnímu zdůvodňování
  - Metodologické námitky – Wittgenstein, Tractatus logico-philosphicus

Ontologické jazyky

- Formální deklarativní jazyky pro reprezentaci znalostí
- Obsahují fakta a pravidla zdůvodňování
- Často založeny na FOL nebo DL
- Rámce
  - Historický předchůdce (Minsky)
  - Vizualizace lidského uvažování a zpracování jazyka

Frame Terminology	OO Terminology
Frame	ObjectClass
Slot	Object property or attribute
TriggerAccessor, Mutator methods	Method

XML

- XML
  - Není k tomu určen, často se ale používá
  - Pochází z webu
  - Hlavní výhoda – definice nových tagů
  - Tagy XML pak přirozeně reprezentují slovník (toť vše)

```
<catalog>
<product typ="CD">
    <title>7777</title>
    <artist>SRPR</artist>
    <price currency="CZK">250</price>
</product>
<product typ="CD">
    <title>Dlask</title>
    <artist>SRPR</artist>
    <price currency="CZK">200</price>
</product>
</catalog>
```

RDF (resource definition framework)

- Standard reprezentace znalostí pro web
- Je jednoduchý
  - Malá vyjadřovací síla
  - Jednoduché algoritmy „uvažování“
- Reprezentace trojic subjekt-predikát-objekt

MarriedTo(Karel,Jája), FatherOf(Karel,Péťa), FatherOf(Karel, Jíťa)

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ns="http://www.example.org/#"> <ns:Person
rdf:about="http://www.example.org/#john"> <ns:hasMother
rdf:resource="http://www.example.org/#susan" /> <ns:hasFather>
<rdf:Description rdf:about="http://www.example.org/#richard">
<ns:hasBrother rdf:resource="http://www.example.org/#luke" />
</rdf:Description> </ns:hasFather> </ns:Person>
</rdf:RDF>

OWL - Web Ontology Language

- Taky pochází z (sémantického) webu
- Ver 1 vs ver 2
- Je to sbírka několika formalismů na popis ontologií
- Syntax v XML (ale i v RDF, funkcionálním tvaru …)
- Snaha o formální a prakticky zvládnutelný tvar
- Deskripční logika – rozhodnutelný fragment FOL
  - In the beginning, IS-A was quite simple. Today, however, there are almost as many meanings for this inheritance link as there are knowledge-representation systems. (Ronald J. Brachman, What ISA is and isn't)
- Předpoklad otevřeného světa
  - [The closed] world assumption implies that everything we don’t know is false, while the open world assumption states that everything we don’t know is undefined. (Stefano Mazzocchi, Closed World vs. Open World: the First Semantic Web Battle)

Deskripční logiky

- Koncepty (třídy), role (properties, predikáty) a individua (objekty)
- Axiom – logický výraz o rolích a konceptech
  - To je jiné než v rámcích/OO (tam máme třídu kompletně definovanou)
- Jde o rodinu logik dle přípustných pravidel-axiomů
- Např:
  - ACL – negace a průnik tříd, omezený existenční kvatifikátor, omezující podmínky
  - Další rozšíření: sjednocení tříd, hiearchie rolí, tranzitivita rolí, …
- Terminological T-Box, Axiom A-box

OWL 1

- OWL-Lite (SHIF)
  - Nejjednuduší
  - Mnoho omezení na axiomy, za účelem srozumitelnosti a strojového zpracování
- OWL-DL (SHOIN)
  - Odpovídá Deskripční logice
  - Umožňuje vyjádřit, že třídy jsou disjunktní, např.
  - Úplnost, rozhodnutelnost
- OWL-Full
  - Nejsilnější výrazový prostředek
  - Mnoho problémů v něm je nerozhodnutelných

OWL 2 (SROIQ)

- OWL-EL
  - Polynomiální složitost zdůvodňování
- OWL-QL
  - Specializace na dotazování v databázích fakt
- OWL-RL
  - Speciální tvar - pravidla

KIF - Knowledge interface format

- Reprezentace znalostí v logice prvního řádu
  - (salary 015-46-3946 widgets 72000)
  - (> (* (width chip1) (length chip1)) (* (width chip2) (length chip2)))
  - (interested joe `(salary ,?x ,?y ,?z))
  - (progn (fresh-line t) (print "Hello!") (fresh-line t))

  - (forall (?F ?T)
    (=>
    (and
    (instance ?F Farmer)
    (instance ?T Tractor))
    (likes ?F ?T)))
  - (exists (?F ?T)
    (and
    (instance ?F Farmer)
    (instance ?T Tractor)
    (likes ?F ?T)))

DAML+OIL

- DAML – Darpa agent markup language
- OIL – Ontology Interchange language
- Předchůdce OWL
- Ukončen v r. 2006

9. Komunikace agentů

Řečové akty, KQML, ACL, KIF, protokoly

Komunikace agentů

- V OOP je „komunikace“ volání metod s parametry
- Agenti nemohou přímo způsobit, že jiný agent něco udělá/změní si vnitřní proměnné
- Agenti musejí komunikovat – provádět komunikační akt:
  - aby předali informace,
  - aby ovlivnili jiné agenty v tom, co mají udělat
- Ostatní agenti mají svou agendu, cíle, a je na nich, jak s informacemi, žádostmi, dotazy naloží
- Dnes je venku hezky

Řečové akty

- Austin, 1962 – některé části jazyka mají charakter akcí, protože mění stav světa stejně jako naše fyzické akce – to jsou řečové akty
  - Prohlašuji vás mužem a ženou
  - Deklaruji válku Rusku
- Slovesa jako request, inform, promise
- Lokuční akty – jakékoliv pronášení slov
  - Udělej mi čaj
- Ilokuční akty – lokuce+perfomativní význam (otázka, přání…)
  - Požádal mě o čaj
- Perlokuční akty – vyvolání účinku
  - Donutil mě, abych mu udělal čaj

Searle a jeho rozvinutí řečových aktů

- Příklad SPEAKER request HEARER action
- Normální I/O podmínky
  - H slyší, není to ve filmu, …
- Přípravné podmínky
  - Co musí platit, aby si S mohl akt zvolit
    - H musí být schopen akce, S věří, že H je schopen akce, není zřejmé, že by H provedl akci tak jako tak
- Podmínky upřímnosti
  - S opravdu chce, aby se akce provedla

Searlovy kategorie aktů

- Starší
  - Žádost,
  - Rada,
  - Tvrzení,
  - Slib
- Novější
  - Asertivy (Reprezentativy) - informace
  - Direktivy – žádost o akci - request
  - Komisivy - slib akce mluvčím
  - Expresivy – vyjádření emoce, psych. stavu - poděkování
  - Deklarativy - změna stavu věcí – válka, sňatek

Plánová teorie řešových aktů

- Cohen, Perrault, 1979
  - *„… modelling [speech acts] in a planning system as operators defined … in terms of speakers and hearers beliefs and goals. Thus speech acts are treated in the same way as physical actions.“*
- STRIPS
  - preconditions,
  - postconditions
- Modální operátory
  - beliefs,
  - abilities,
  - wants

Příklad: Request a Inform

- Request (S,H,A)
  - Preconditions
    - Cando:
      - (S believe (H cando A))&(S believe(H believe (H cando A)))
    - Want:
      - (S believe (S want requestinstance))
  - Effect:
      - (H believe (S believe (S want A)))
- Inform(S,H,F)
  - Preconditions
    - Cando:
      - (S believe F)
    - Want:
      - (S believe (S want informinstance))
  - Effect:
    - (H believe (S believe F))

Agentní jazyky

- 1990s: DARPA – Knowledge sharing effort (KSE)
  - KQML – knowledge query and manipulation language
    - Vnější jazyk agentní komunikace (obálka)
    - Ilokuční obsah zprávy
  - KIF – knowlewdge interchange format
    - Vnitřní jazyk, obsah zprávy
    - Reprezentace znalostí

- FIPA ACL (foundation of physical agents, agent communication language)
  - Zjednodušení, systematika v performativech, sémantika, praktická implementace
  - JADE

KQML

- Perfomativ
- Obsah
- Adresát
- Jazyk
- Ontologie

```
(ask-one
  :content (PRICE IBM ?price)
  :receiver stock-server
  :language LPROLOG
  :ontology NYSE-TICKS
)
```

Parametry a performativy KQML

- Content
- Force
- Reply-with
- In-reply-to
- Sender
- Receiver

- Achieve
- Advertise
- Ask-about, ask-one, ask-all, ask-if
- Break, sorry, error
- Broadcast
- Forward
- Recruit-all, -one
- Reply
- Subscribe

ACL

```
(inform
   :sender agent1
   :receiver agent2
   :content (price good2 150)
   :language sl
   ontology: hpl-auction
)
```

ACL performativy

- Request, request-when
- Inform, inform-if, inform-ref
- Subscribe
- Cfp
- Propose
- Proxy
- Refuse
- Reject-proposal
- Confirm, disconfirm
- Agree, cancel

10. Kooperace agentů

Distribuované koordinované řešení problémů, Contract net

Když mají agenti spolupracovat ...

- Agenti mají různé cíle, nutnost synchronizace
- Agenti pracují v čase, nejsou předprogramovaní, nutnost dynamického domlouvání, synchronizace

- Sdílení cílů
- Sdílení informace
- Dynamická (run-time) koordinace

- Koherence – jak dobře se chová systém jako celek
- Koordinace – jak dobře agenti minimalizují režijní aktivity jako synchronizace, ...

CDPS

- Kooperativní distribuované řešení problému – CDPS, Lesser et al, 80s
  - Kooperace individuálních „agentů“ při řešení problému přesahujícím schopnosti (informace, zdroje) jednotlivce
  - Agenti implicitně sdílejí společný cíl, nedochází ke konfliktům – benevolence
  - Měřítkem úspěchu je celková úspěšnost systému
  - Agent pomůže celku, i když to je pro něj nevýhodné
  - Benevolence značně zjednodušuje návrh systému

CDPS vs. PPS vs. MAS

- Neplést CDPS s PPS (paralel problem solving, Bond, Gasser, 80s)
  - důraz na paralelismus,
  - homogenní a jednodušší procesory
- Obecný výzkum v MAS situaci ještě komplikuje:
  - MAS je společnost agentů s vlastními cíly
  - Agenti nesdílejí společný cíl
  - Stejně musejí kooperovat
    - Proč a jak kooperovat
    - Jak agenti identifikují a řeší konflikty
    - Jak agenti vyjednávají a smlouvají

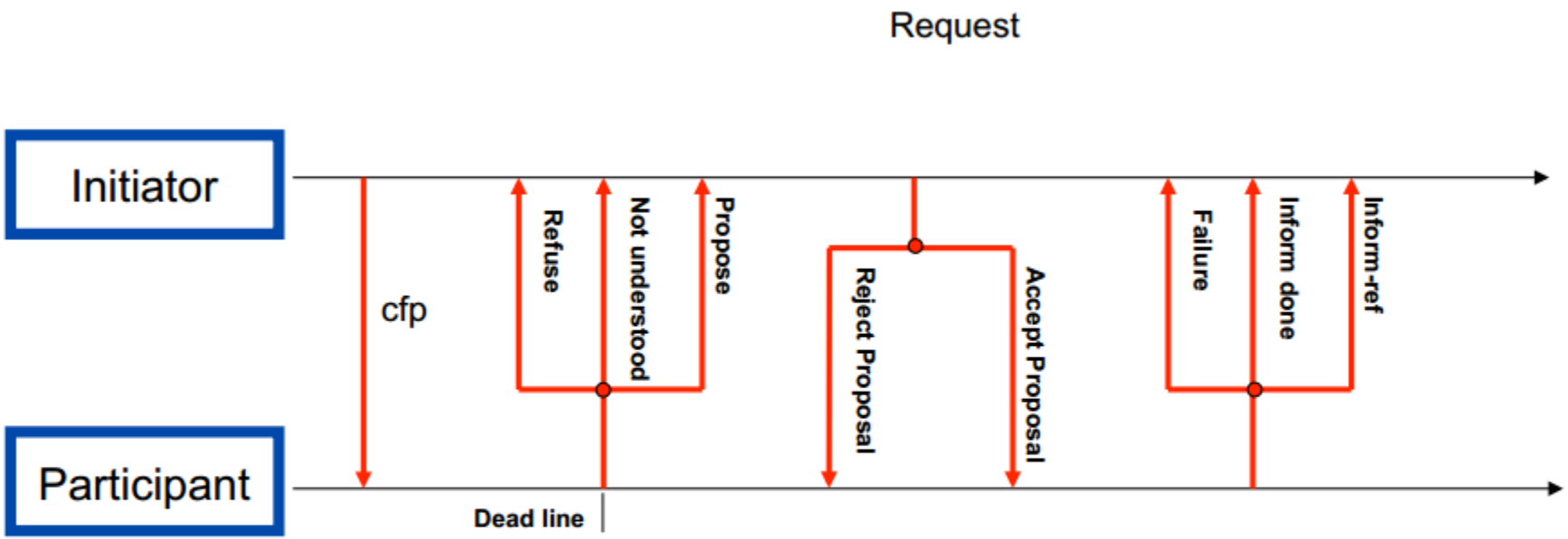
Sdílení úkolů a výsledků

- Dle CDPS:
  - Dekompozice problému
    - Typicky hierarchické
    - Kde skončit (ACTOR – pro každý podproblém nový agent)
    - Nutnost (trochu) znát problém i agenty
  - Řešení subproblémů
    - Nutnost sdílet informace mezi agenty
  - Syntéza řešení
    - Opět hierarchická
- Sdílení úkolů – dohoda agentů
- Sdílení výsledků – proaktivní, reaktivní

CNET

- Contract Net protokol, Smith and Davis, 1977
- Metafora pro sdílení úkolů pomocí „smlouvy“
  - Oznámení úkolu – manažer udělá broadcast
  - Nabídky na řešení – pracanti sdělují, zda jsou schopni řešit a případně jak „dobře“
  - Přidělení úkolu – manažer vybere kontraktora
  - (moc se nestaráme o provedení úkolu)
- Jednoduché,
- nejstudovanější,
- nejimplementovanější

FIPA ACL CNET protokol



10. Přednáška

- Literatura:
  - Woldridge: Introduction to MAS, kap 8
  - BBS (“Blackboard Architectures,” AI Game Programming Wisdom, Volume 1, pp. 333 – 344)

- Software
  - Zase JADE

- Obrazová příloha:
  - surrealismus

Blackboard – tabule

- BBS - data-driven nebo i plánovitý přístup, event-based
  - Několik agentů expertů sedí u tabule
  - Na tabuli jsou napsané (a dopisují se) problémy
  - Když expert vidí, že může něčím přispět, napíše to na tabuli
  - To se děje dokud se na tabuli neobjeví řešení problému
- Použití hlavně při distribuovaném řešení problému (rozpoznávání řeči, …)
- Vyžaduje společný jazyk pro interakce
- Typicky obsahuje různé úrovně abstrakce

BBS pokračování

- Arbitr
  - Vybírá experty, kteří budou řešit daný problém – „půjdou k tabuli“
  - Buď reaktivní, nebo uvažuje plány maximalizující užitek
  - Zodpovědný za vyšší úroveň řešení problému (motivace, emoce, …)
- Experti
  - Kooperující agenti
  - Reagují na cíle na tabuli, provádějí akce
- Tabule
  - Sdílená paměť
  - Důležité je, jak jsou informace na tabuli reprezentovány
  - Typicky hierarchické uspořádání cílů, akcí

BBS příklad – hra BBWar

- Tabule
  - je hashovací tabulka – mapuje požadované schopnosti na úkoly
  - Zveřejňují se na ní otevřené mise-úkoly
- Experti
  - Řešiči jednotlivých úkolů, hierarchická struktura
  - Seznam schopností a efektivity
- Př:
  - Velitel hledá cíle typu ATTACK-CITY, vytváří mise ATTACK-LOCATION
  - Vojáci hledají mise ATTACK-LOCATION

BBS – (ne)výhody

- Jednoduchý mechanismus pro koordinaci a kooperaci agentů
- Experti nemusejí vědět o ostatních expertech, se kterými kooperují
- Zprávy na tabuli lze přepisovat – delegovat úkoly, vyměňovat experty, …
- Někdy se používá i pro samotnou komunikaci mezi agenty
- Agenti musejí typicky sdílet stejnou architekturu a u tabule je nával (distribuované hash-tables)

Sdílení výsledků

- Vcelku přímočaré
- Kromě triviálních důvodů lze sdílení využít pro:
  - Konfidence
    - Nezávislá řešení stejných problémů lze porovnávat
  - Úplnost
    - Agenti sdílí lokální pohledy pro lepší globální pohled
  - Přesnost
    - Sdílení při řešení za účelem zlepšení přesnosti celkového řešení
  - Včasnost
    - Zřejmá výhoda distribuce (i když by to 1 agent třeba spočítal celé sám)

Příklad: FELINE

- Pravěk (před KQML i ACL), Wooldridge, Jennings, 1990
- Distribuovaný expertní systém
- Sdílení znalostí, distribuce podúloh
- Každý agent je pravidlový systém
  - Skills – pravdivost tohoto umím dokázat/vyvrátit
  - Interests – pravdivost tohoto mě zajímá
- Komunikace
  - Sender, receiver, content (hypotéza + řečový akt)
  - Request, response, inform

Interakce agentů

- Agenti si mohou pomáhat, ale i překážet
  - Roboti pohnou cihlou
  - Roboti najednou neotevřou dveře
- Agenti ovlivňují různé části prostředí
- Agenti mohou mít společenství, podřízené, nepřátele
- Je důležité vědět, jaký typ interakcí existuje mezi agenty v konkrétním systému,
- Jinak nelze navrhnout efektivní řídící mechanismy
- Nejjednodušší – interakce dvou sobeckých agentů v prostředí připomínajícím hry

Agenti se svými zájmy kooperují

- Proč by měl agent říkat pravdu o svých schopnostech
- Proč by měl agent dokončit přidělený úkol
- Pokud je systém homogenní (uděláme si ho celý sami), je benevolence dobrá strategie
- Ale často se systém skládá z agentů s různými zájmy
  - Konflikt mezi společným cílem a cíli jednotlivých agentů
  - Např. řízení leteckého provozu
- Někdy je MAS tak komplikovaný, že ani není jasné, co je společný zájem
  - Pak je lepší uvažovat návrh sobeckých agentů

Co chce sobecký agent?

- Agent se svými zájmy se snaží maximalizovat svůj očekávaný užitek
- Na to existuje spousta metod AI
- Ale jsou tu ostatní agenti, ti chtějí to samé
- Každý agent typicky ví užitek u možných výsledků akcí
- Strategické uvažování
  - Maximalizuj svůj užitek
  - Za předpokladu, že všichni jednají racionálně
  - Nemaximalizuje to společný užitek
  - Ale je to robustní strategie

Agenti se svými zájmy

- Self-interested agent, neplatí podmínka benevolence
- Má množinu možných výsledků O={o1, o2, …},
  - Ta je společná pro všechny (oba) agenty
- A nad nimi preference – užitkovou funkci u: O->R
  - Ta je pro každého agenta jiná
  - Užitková funkce každému agentovi utřídí výsledky
- K zamyšlení:
  - Užitek nejsou jen peníze
  - Peníze mají jiný účinek pro bohatého a jiný pro chudšího
  - Zvláštní, že extrémy jsou symetrické

11. Interakce agentů

Nashovo ekvilibrium, Paretova fronta, Vězňovo dilema

Rozhodování jako hra

- Oba agenti ovlivňují výsledek
- – změnu stavu prostředí
  - e: Ai x Aj -> O
- Agent má strategii si (sj)
- Si je dominantní pro agenta i, pokud dá lepší nebo stejný výsledek než jakékoliv jiná strategie agenta i pro všechny strategie agenta j

Nashovo ekvilibrium

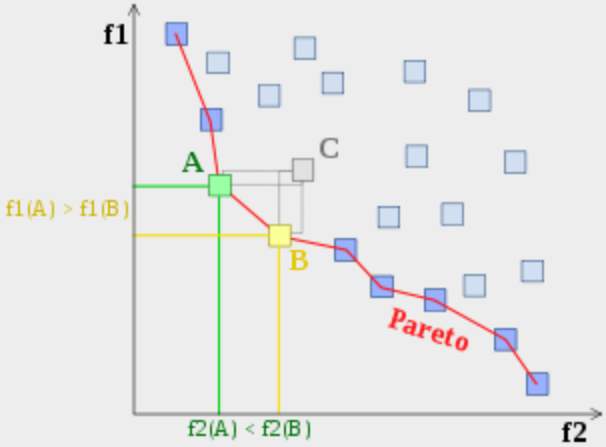
- s1 a s2 jsou v Nashově ekvilibriu, když:
  - Hraje-li agent i s1, agent j dosáhne nejlepšího výsledku s s2
  - Hraje-li j s2, pro i je nejlepší hrát s1
- Neboli, s1 a s2 jsou na sebe vzájemně tou nejlepší odpovědí.
- Najít ekvilibria pro n agentů a m strategií trvá m^n
- Tohle je Nashovo ekvilibrium ryzích strategií
  - Ale ne každá hra má Nashovo ekvilibrium v ryzích strategiích
  - A některé hry mají víc Nashových ekvilibrií

Nashova věta

- Smíšené strategie – náhodný výběr mezi ryzími strategiemi
- Nashova věta: Každá hra s konečným počtem strategií má Nashovo ekvilibrium ve smíšených strategiích.
- Jak dlouho to trvá nalézt – total search problem, od 2006 víme, že to je PPAD úplné
-

Paretova efektivita

- Řešení je Pareto-optimální/efektivní, když neexistuje jiná strategie, která by zlepšila agentův výnos bez zhoršení výnosu jiného agenta.
- Řešení je neefektivní podle Pareta: - lze zlepšit výnos jednoho, aniž by se zhoršil výnos druhého



Společenské dobro

- Agenti by mohli maximalizovat společný užitek – social welfare
  - suma užitků/výnosů všech agentů
- Ale to je výhodné jen při kooperaci
- Typicky, když jsou agenti součástí jednoho týmu, mjí jednoho vlastníka.

Proč je vězňovo dilema vlastně dilema?

- DD je Nashovo ekvilibrium
- DD je jediné řešení, které není Pareto-optimální
- CC je řešení, které maximalizuje společný užitek
  - Tragedy of the commons
  - Co to je racionální a jsou lidé racionální?
  - Stín budoucnosti – iterované verze – Axelrod, TFT

i/j	D	C
D	2 / 2	0 / 5
C	5 / 0	3 / 3

Rozhodování jako volby

- Společenský užitek – **social welfare –** agregace individuálních užitků
- Společenská volba – **social choice** - výběr nejlepšího kandidáta/užitku
- **Př**:
  - o2>o1>o3, o3>o2>o1, o2>o3>o1
  - Social welfare: o2>o3>01
  - Social choice: o2

Pluralitní volby

- každý vybere nejlepšího kandidáta
- Jednoduché při dvou kandidátech
- Při více kandidátech často dochází k paradoxu, že je zvolen kandidát, který byl pro většinu špatný – Condorcetův paradox – v demokratickém hlasování se může stát, že výsledek nebude optimální pro většinu kandidátů
- Taktické hlasování (nehlasuju podle svých preferencí ale „proti někomu“)

Příklad: Condorcetův paradox

- Volič1: A > B > C
- Volič2: B > C > A
- Volič3: C > A > B

- Neexistuje Condorcetův vítěz

- Neboli, Kolektivní preference (social welfare) je cyklická, I když jednotlivé preference nejsou

Jiné volby

- Sekvenční pluralitní
  - „turnaj“ po dvojicích, vítěz postupuje do dalšího kola
  - Záleží nejen na preferencích ale i na pořadí turnajů
- Bordův systém
  - Každý volič dá pořadí, ta se agregují, nejlepší se vybere správně
  - Používá se i v životě: Slovinsko, Nauru, Island
- Slaterův systém
  - Snaha o optimalizaci agregace preferencí
  - NP-úplné

Vlastnosti voleb

- Paretova vlastnost
  - Když každý agent i X > i Y, tak platí X >sw Y
  - Platí pro pluralitní systém, Bordův systém
  - Neplatí pro sekvenční pluralitní systém
- Condorcetův vítěz
  - Condorcetův vítěz je ten, který porazí všechny kandidáty v párových porovnáních
  - To je hodně silná vlastnost
  - Podmínka Condorcetova vítěze: Condorcetův vítěz zvítězí ve volbách
  - To zní rozumně, ale jen sekvenční pluralitní systém to splňuje

Vlastnosti voleb

- Nezávislost irelevantních alternativ
  - Společenské ohodnocení určující pořadí dvou kandiátů, např. X >sw Y by mělo záviset jen na jednotlivých ohodnoceních určujících pořadí X a Y u agentů.
  - Takže pokud všechny preference u agentů týkající se pořadí X a Y zůstanou stejné, neměla by se změnit ani relace X >sw Y.
  - I když se třeba změní u agenta preference např. preference mezi X a Z nebo Z a W.
  - Neplatí pro pluralitní, sekvenční plularitní, Bordův
  - Je to možná moc silný předpoklad, viz preference typu kámen-nůžky-papír
  - Říká se, že člověk realisticky porušuje NIA při skalárním uspořádávání vektorů, protože používá vážení jednotlivých složek, kde váhy jsou na sobě závislé

Vlastnosti voleb

- Neomezenost domény sociálního užitku
  - Všechny možné individuální užitky všech agentů se berou v úvahu.
  - To zní triviálně.
  - Ale např. když se vlastnost oslabí na lineární uspořádání užitků, ze kterých si agent vybere (např. jak hlasitě má hrát hudba), pak už Arrowova věta neplatí
- Diktátorství
  - Existuje jedinec, jehož preference jsou vždy brány jako společné preference.
  - Ne-diktátorství: No voter in the society is a dictator in the sense that, there does not exist a single voter *i* in the society such that for every set of orderings in the domain and every pair of distinct social states *x* and *y*, if voter *i* strictly prefers *x* over *y*, *x* is socially selected over *y*.

Arrowova věta

- (Pro 3 a více kandidátů) Žádný volební systém založený na pořadí nedokáže vytvořit z individuálních preferencí (úplné a tranzitivní) uspořádání za zachování podmínek:
  - neomezená doména,
  - ne-diktátorství,
  - Pareto-optimalita,
  - nezávislost irelevantních alternativ.
- Pesimistická (a pomýlená) intepretace: jen diktátorství je dobrý volební systém

12. Aukce

Anglická, holandská, obálková, Vickryova

Aukce/Dražba

- Mechnismy, jak alokovat zdroje agentům
- V životě:
  - eBay, Sothesby, …
  - Nerostné bohatství, frekvence mobilních operátorů, …
  - Licitovaný mariáš
- V informatice
  - Procesor, …
- Efektivita aukce:
  - Alokovat zdroje těm agentům, kteří je chtějí nejvíc

Aukce

- Licitátor
  - Chce maximalizovat cenu
- Účastník – vydražitel
  - Chce minimalizovat cenu
  - Má svůj užitek
  - Užitek se může lišit od obecného užitku (dolarová bankovka z peněženky Michaela Jacksona, krev Jana Pavla II)
- Dražební protokoly
  - Určení vítěze – first price, second price
  - Otevřenost nabídek – zalepené obálky, vyvolávání
  - Mechanismus – jednorázové, kolové

Obálková metoda

- Jednokolová, uzavřená, zapečtěné obálky
- Vyhrává nejvzšší nabídka, platí se tato nabídka
- First-price sealed bid auction
- Pyrrhovo vítězství
  - nebezpečí, že vítěz zaplatí víc, než je jeho užitek
  - *„Ještě jedno takové vítězství a jsme zničeni.“*
- Dominantní strategií není nabídnout cenu odpovídající mému užitku, ale nižší.
- Tržní cenu nelze odhadnout, není interakce mezi účastníky

Vickryova aukce

- Aukce druhé nejlepší nabídky
- Vítězí nejvyšší nabídka, platí se druhá nejvyšší cena
- Dominantní strategie je nabízet skutečnou hodnotu zboží.
- Google AdWords, dražby známek
- Prodávající může využít volavku ke zvýšení ceny
- Vickrey ukázal, že užitek pro licitátora může být stejný jako v obálkové metodě

Anglická aukce

- Otevřená aukce se zvyšující se cenou
- Nejběžnější – starožitnosti, umění, aukro
- Dominantní strategie – po malých částech přihazovat až do skutečného užitku agenta
- V praxi nejčastější případ aukce, kde dochází k přeplacení účastníkem

Holandská aukce

- Otevřená aukce s klesající cenou
- Tulipány, ryby, zelenina, někdy i ceny letenek apod
- Prodejci ji mají rádi, často dosáhnou vyšších výnosů
- Není možné odhadnout tržní cenu
- Jako u obálkové metody není možné odhadnout preference ostatních účastníků
- Nabízet cenu odpovídající užitku není dominantní strategie

Příklad

- Máme-li kupříkladu zboží, o nějž mají zájem tři dražitelé, přičemž jejich individuální ohodnocení ceny zboží se rovná následujícímu:

- dražitel A – 80 Kč
- dražitel B – 60 Kč
- dražitel C – 30 Kč
- Bude vydražená cena pro různé typy aukcí přibližně následující:

- Anglická aukce – 61 Kč (tedy druhá nejvyšší cena plus možný příplatek pro jeho přeplacení)
- Holandská aukce – 80 Kč (tedy nejvyšší subjektivní cena, maximálně ponížená do míry rizika, které je dražitel A ochoten nést)
- Obálková metoda – 80 Kč (jako v případě holandské aukce)
- Vickreyova aukce – 60 Kč (vyhraje dražitel A, ale zaplatí jen druhou nejvyšší nabídku dražitele B)
- Tyto hodnoty ovšem platí v případě, že znalost preferencí ostatních dražitelů je nulová, v reálném světě je ovlivněna známými informacemi, jako například vítězná cena nedávno prodaného obdobného zboží.

Další aukce

- Kombinatorická aukce:
  - prodává se více položek, účastníci sázejí na balíčky – podmonožiny
  - Oblíbená teorie, NP-úplné problémy
- Aukce se zaplacením všech nabídek:
  - Oblíbená u ekonomů jako nástroj pro výzkum lobbingu a úplatků
  - Oblíbená u sportovců – účastnický poplatek
- Tichá aukce
  - Varianta anglické na papíře, cfp?