Pokyny pro projekt do předmětu AGS

Ak. r. 2012/2013

Ing. Jiří Král a Ing. Jan Horáček
E-mail ikral@fit.vutbr.cz
Ustav inteligentních systémů, FIT, VUT v Brně

1 Programovací prostředí

Vaším úkolem bude napsat kód agentů s využitím prostředí *Jason*. Agenti ze společného týmu budou mít za úkol sesbírat co nejvíce surovin (zlata a dřeva) z prostředí a donést ho do cílového místa. Nejprve je potřeba rozchodit tento nástroj. *Jason* stáhněte ze stránky:

`http://Jason.sourceforge.net/`

Prostředí je možné provozovat jak na Windows, tak i na Linuxu. V podadresáři bin spustíme "*Jason*.bat"případně "*Jason*.sh". Následně musíme správně nastavit cestu k Javě. Položku najdeme v:

$$Plugins \rightarrow Plugins \rightarrow Options \rightarrow Jason \rightarrow JavaHome$$

Nyní již můžeme zkusit některé z příkladů v podadresáři `examples`. K doporučení je především `examples/gold-miners`.

2 Úkol

Jako první bod přejmenujte soubory "`teamxSlow.asl`", "`teamxMiddle.asl`"a "`teamxFast.asl`", kde podle přiděleného písmena skupiny nahrad'te symbol x. Například pro skupinu C budou názvy souborů vypadat "`teamCSlow.asl`", "`teamCMiddle.asl`"a "`teamCFast.asl`". Následně modifkujte soubor "`JasonTeam.mas2j`"v části obsahující cesty k těmto souborům. Opět příklad pro skupinu C:

```
MAS miners {
        infrastructure: Centralised

        environment: mining.MiningPlanet(1,50,yes)
                // parameters: 1. environment configuration
                   id (from 1 to 4)
                // 2. sleep time (in ms) after each action
                // 3. whether display the gui
        agents:
                aSlow teamCSlow;
                aMiddle teamCMiddle;
                aFast teamCFast;
                bSlow teamySlow;
                bMiddle teamyMiddle;
                bFast teamyFast;
}
```

Tato definice říká, že agent pojmenovaný jako aSlow je definován v souboru "`teamCSlow.asl`". Vašim úkolem je tedy doplnit těla těchto tří agentů.

Typ	Pohybových bodů/kolo	Kapacita surovin
Slow	1	5
Middle	2	3
Fast	3	1

Tabulka 1: **Vlastnosti agentů**

3 Agentní prostředí a dostupné akce

Agenti budou umístěni do defnovaného prostředí, k dispozici jsou celkem 4 rozložení. Rozložení prostředí lze změnit v souboru "`JasonTeam.mas2j`"pomocí řádku:

```
environment: mining.MiningPlanet(1,50,yes).
```

Celkem máme tři typy agentů v týmu. Každý typ agenta se proto hodí na odlišnou úlohu. Je zde jak agent, který je schopný se rychle pohybovat, ale unese málo surovin, dále jeho pravý opak, který se pohybuje pomalu, ale unese více a agent, který je vyváženou kombinací obou vlastností. Vlastnosti agentů můžeme vidět v tabulce 1. Každý agent může v daný moment nést pouze jeden typ surovin. Suroviny nelze uvnitř jednoho agenta navzájem kombinovat (lze ale např. předat zlato kolegovi, který již zlato nese, případně je prázdný).

Prostředí v každém kole informuje agenta o jeho okolí. Informace jsou vloženy do báze znalostí na začátku každého kola (či po provedení akce). Příklad báze znalostí agenta:

```
carrying_capacity(1)[source(percept)].
carrying_gold(0)[source(percept)].
carrying_wood(0)[source(percept)].
obstacle(7,28)[source(percept)].
gold(7,27)[source(percept)].
wood(7,27)[source(percept)].
enemy(7,26)[source(percept)].
ally(6,27)[source(percept)].
depot(16,16)[source(percept)].
friend(bMiddle)[source(percept)].
friend(bSlow)[source(percept)].
grid_size(35,35)[source(percept)].
moves_left(3)[source(percept)].
moves_per_round(3)[source(percept)].
pos(6,27)[source(percept)].
step(0)[source(percept)].
```

Asi nejdůležitější informací je informace o aktuálním kole step(n). V kódu každého agenta musí být reakce na přidání takovéto informace do báze znalostí a následné provedení akcí v tomto kroku. Význam ostatních informací je opět v tabulce 2.

Název	Parametry	Význam
carrying capacity	int x	Agent má možnost nést maximálně x surovin.
carrying gold	int x	Agent právě nese x zlata.
carrying wood	int x	Agent právě nese x dřeva.
obstacle	int x, int y	Na pozici $[x, y]$ je překážka.
gold	int x, int y	Na pozici $[x, y]$ je zlato.
wood	int x, int y	Na pozici $[x, y]$ je dřevo.
enemy	int x, int y	Na pozici $[x, y]$ je nepřítel.
ally	int x, int y	Na pozici $[x, y]$ je přítel.
depot	int x, int y	Zlato se nosí na pozici $[x, y]$.
friend	string A	Mým přítelem je agent A.
grid size	int x, int y	Prostředí má rozměry x na y.
moves left	int x	Agentu zbývá x pohybů v tomto kole.
moves per round	int x	Tento typ agenta má x pohybů na kolo.
pos	int x, int y	Pozice agenta je $[x, y]$.
step	int x	Nacházíme se v x-tém kole.

Tabulka 2: **Význam informací v bázi znalostí.**

3.1 Akce

Každý agent má v průběhu kola možnost vykonat určitý počet akcí. Jedná se například o pohyb agenta v různých směrech, sbírání zlata na aktuální pozici apod. Pohybové akce odeberou vždy jeden pohybový bod (PB). Oproti tomu akce jako sebrání zlata, položení zlata, či převzetí zlata od jiného agenta potřebují maximální počet PB, který má agent k dispozici na kolo. U agenta typu slow tedy stojí nepohybové akce 1 PB, u agenta typu fast 3 PB. Seznam akcí, pomocí kterých agent komunikuje s prostředím je uveden v tabulce 3.

Agent musí v každém kole vypotřebovat všechny svoje pohybové body! Pokud agent nechce provést žádnou akci (je pro něj výhodné zůstat na pozici), lze využít akce `do(skip)`. Další poznámka se týká akce `do(pick)`. Pro úspěšné sebrání suroviny z dané pozice musí být na stejné pozici alespoň jeden spřátelený agent. Aby se zamezilo ukradnutí zlata jiným (především nepřátelským) agentem těsně před provedením akce `do(pick)`, je toto zlato odebráno, ale pokud se jiný agent pokusí provést akci, pak proběhne úspěšně a zlato bude přičteno i jemu. Tato vlastnost funguje v rámci jednoho kola. V dalším kole již není možné z této pozice zlato vzít. Pokud tedy na danou pozici dorazí celý tým agentů a všichni provedou akci `do(pick)` dojde k naklonování suroviny pro každého z nich.

Název	Parametry	Cena(PB)	Význam
do(up)	-	1	Přesune agenta nahoru.
do(down)	-	1	Přesune agenta dolů.
do(left)	-	1	Přesune agenta doleva.
do(right)	-	1	Přesune agenta doprava.
do(skip)	-	1	Prázdná akce.
do(pick)	-	max (1j3j5)	Z aktuální pozice vezme surovinu (pokud zde je surovina a alespoň jeden spřátelený agent).
do(drop)	-	max (1j3j5)	Na aktuální pozici (musí být depot) položí suroviny.
do(transfer, A, N)	string A, int N	max (1j3j5)	Vezme N počet stejného typu surovin od agenta A k sobě.

Tabulka 3: **Akce agenta.**

4 Základy programování v *Jason*

Tento systém vychází ze specifikace *AgentSpeak(L)*. Každý agent má svoji bázi znalostí a své plány. Dále bude popsána základní práce s těmito strukturami.

4.1 Báze znalostí

Fakt může být do báze znalostí uložen jak v pozitivním tvaru, že něco platí, tak i v negativním (agent si je jist nepravdou daného tvrzení). Příklad pozitivního a negativního (silná negace) faktu:

```
matka(eva).
~matka(pavel).
```

Tímto způsobem definujeme počáteční obsah báze znalostí. Z těchto znalostí lze pomocí odvozovacích pravidel získávat další znalosti. Princip i syntax je podobný jako v jazyce Prolog. Jako anonymní proměnná zde pro účely unifkace slouží opět symbol ' '. Musíme si dát pozor na velikost písmen. Slova začínající velkým písmenem jsou proměnnými!

Uvedeme také příklad pravidla počítajícího faktoriál. Výstupem pravidla musí být logická hodnota pravda/nepravda. Pro naše účely můžeme využít spojky '&':

```
factorial(0,1).
factorial(N,F) :- factorial(N-1,F1) & F = N*F1.
```

4.2 Cíle

Cíle jsou základním kamenem pro definování funkčnosti agenta. Posloupnost cílů a akcí totiž tvoří tělo plánu. Rozlišujeme tyto typy cílů:

- ? - dotaz zda je možné dosáhnout cíle.
- ! - dosáhni cíle (čekej dokud nebude dosažen).
- !! - dosáhni cíle, ale bez čekání na dosažení cíle.

Dotaz do báze znalostí se nejčastěji provádí uvnitř těla plánu pomocí symbolu ' ?'. V příkladu se dotazujeme zda je Eva matkou. Dostáváme tedy logickou hodnotu pravda/nepravda. Pokud bylo možné cíle dosáhnout (hodnota pravda), pak plán pokračuje prováděním dalšího cíle, jinak končí. Slíbený příklad:

```
?matka(eva)
```

V následujícím příkladu se ptáme zda zde existuje nějaká osoba, která je matkou. Tuto osobu budeme chtít namapovat do proměnné:

```
?matka(Osoba)
```

Výsledkem je namapování proměnné Osoba na hodnotou eva a jelikož se podařilo najít toto mapování, je výsledkem logická hodnota pravda. Pokud nepotřebujeme s proměnnou dále pracovat, lze využít anonymní proměnné:

```
?matka(_)
```

4.3 Plány

Obecná struktura plánu vypadá následovně:

- `@nepovinné_návěští událost : nepovinná_podmínka ← tělo plánu.`

Návěští identifikuje plán. Navíc zde můžeme uvést další informace jako například někdy užitečnou skutečnost, aby se plán vykonal atomicky. Dva nejužitečnější typy událostí:

- +fakt - reakce na přidání faktu do báze znalostí.
- +!cíl - reakce na zavolání dosažení cíle.

Dále zde máme možnost uvést spouštěcí podmínku plánu. Můžeme uvést spojek '&' a '|'.

Tělo plánu je tvořeno sekvencí cílů a akcí. Globální akce pro práci s prostředím již byly představeny v tabulce 3. Dále máme k dispozici vestavěné akce které vždy začínají tečkou. Seznam vybraných akcí je v tabulce 4.

Dále zde jsou k dispozici akce pro práci s bází znalostí. Uvnitř těla plánu pak můžeme přidávat znalosti pomocí symbolu '+', ale také je odebírat pomocí symbolu '-'.

Následující příklad demonstruje vypsání hodnoty faktoriálu pěti s využitím plánu:

.concat(”a”,”b”,X)	X=”ab”
.length(”abc”, X)	X=3
.max([c,a,b],X)	X=c
.member(c,[a,b,c])	true
.min([c,a,b],X)	X=a
.nth(1,[a,b,c,d],X)	X=b
.sort([c,a,b],X)	X=[a,b,c]
.substring(”b”,”aaa”)	false
.print(”Hello, world”)	Výpis textu (bez diakritiky!)
.random(X)	X=0.31
.my name(X)	X=jméno agenta

Tabulka 4: **Vestavěné akce.**

```
!start.
@p1[atomic] +!start : .my_name(agentA) <- ?factorial(5,F);
  +hodnota_faktorialu(5,F); .print("Factorial(5)␣=␣", F).
@p2[atomic] +!start <-
   .print("Nejsem␣agentA,␣nereknu␣ti␣nic").
+hodnota_faktorialu(X,F) <-
   .print("Ulozeno␣fact(", X, ")=",F).
```

Prvním řádkem vyžadujeme dosažení cíle start. Takovéto požadavky spolu s inicializační podobou báze znalostí a pravidly musejí předcházet defnicím plánů! Náš plán je označen návěštím p1, které navíc říká, že plán bude prove- den atomicky. Následuje spouštěcí podmínka a tělo plánu, kde jednotlivé části jsou odděleny středníkem. Nesmíme zapomenout na ukončovací tečku.

Opět zde funguje vlastnost známá z Prologu, kdy spustitelnost plánů je testována řádek po řádku (odzhora dolů). Jakmile je možné nějaký z plánů spustit, provede se jeho tělo.

Na posledním řádku ještě můžeme vidět reakci na přidání informace do báze znalostí.

4.4 Komunikace

Pro komunikaci využijte vestavěných funkcí:

```
.send("jmeno_agenta",achieve,cil(parametry));
.send("jmeno_agenta",tell,fakt(hodnoty));
```

První akce zašle druhému agentovi požadavek na dosažení cíle, druhá pak pouze přidá informaci do báze znalostí. Nepoužívejte broadcast! Zpráva by pak byla zaslána i agentům nepřátelské strany. Stejně tak je zakázána jakákoli komunikace s nepřátelskou stranou, či zjišt'ování informací z báze znalostí nepřátelského agenta.

5 Hodnocení

Za projekt dostanete jako základ 10 bodů. To v případě, že agenti vždy sesbírají a donesou všechny suroviny. V tomto testu bude jako nepřátelská strana skupina statických (nic nedělajících) agentů. Test bude proveden pro všechny 4 rozložení prostředí. Projekty nefunkční, budou hodnoceny za 0 bodů. Projekty částečně funkční (myšleno pracujícící pouze na určitém typu prostředí, často se "zasekávající") budou posuzovány individuálně, vždy ale již v bodovém rozmezí 0-10 bodů (jelikož zadání nebylo splněno a nemohou se účastnit soubojů s ostatními týmy).

Další testování již bude provedeno jako soutěž každý s každým. Sestaví se pořadí jednotlivých týmů a první 4 týmy získávají plný počet bodů. Za každé další umístění se získává o bod méně. Je proto vaším cílem být lepší než jiný tým.

V případě zájmu o otestování výkonnosti vaší agentní skupiny oproti jiné skupině je nabídnuta možnost zveřejnit průběžné výsledky. V tomto případě odevzdejte prosím projekt do WISu a informujte mě pomocí e-mailu (pro jistotu) že došlo ke změne. Na konci týdne bude sestvena tabulka průběžného hodnocení.

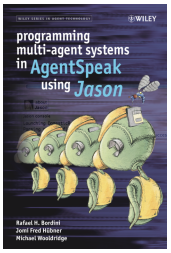
Odevzdávají se pouze soubory "`teamxSlow.asl`", "`teamxMiddle.asl`"a "`teamxFast.asl`"(dle příslušné skupiny). Veškerá funkčnost tedy musí být defnována uvnitř těchto tří souborů.

6 Tipy

Lze využít různých strategií jak dosáhnout lepšího výsledku. Například lze využít specifika akce `do(pick)`, která ve své podstatě umožňuje klonování surovin. V jednom kole tak lze jednu pozici zlata přeměnit na tři zlata. Pokud by agent nemohl z kapacitních důvodů sebrat surovinu a v jeho dosahu je spřátelený agent, lze rovněž použít akce `do(transfer)`. Fantazii se meze nekladou.

Vyzkoušejte si běh vaší skupiny na obou stranách (pojmenování agenta aSlow oproti bSlow).

Jako pomocný text při programování v *Jason* lze doporučit:



Obrázek 1: Bordini, Hübner, Wooldridge, *Programming multi-agent systems in AgentSpeak using JASON.*