Sémantická reprezentácia pragmatických znalostí

Diplomová práca

Peter Jankovič **2007**

Sémantická reprezentácia pragmatických znalostí

DIPLOMOVÁ PRÁCA

Peter Jankovič

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
KATEDRA APLIKOVANEJ INFORMATIKY

Informatika

Školiteľ diplomovej práce:
RNDr. Martin Takáč

BRATISLAVA, 2007

Čestne vyhlasujem, že som diplomovú prácu vypracoval samostatne, len s použitím uvedenej literatúry a pod odborným dohľadom školiteľa.

Bratislava, máj 2007

….....

Peter Jankovič

Ďakujem svojmu školiteľovi Martinovi Takáčovi za cenné rady a pripomienky, ktoré mi poskytol pri písaní diplomovej práce. Zároveň sa chcem poďakovať rodine a priateľom, že mi boli oporou počas celého štúdia.

Abstrakt

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
KATEDRA APLIKOVANEJ INFORMATIKY

Autor: Peter Jankovič

Názov diplomovej práce: Sémantická reprezentácia pragmatických znalostí

Školiteľ diplomovej práce: RNDr. Martin Takáč

Rozsah práce: 65 strán

Bratislava, máj 2007

V práci navrhujeme a implementujeme kauzálny a plánovací modul pre agenta v modeloch osvojovania a emergencie jazyka. Sémantika jazyka je založená na rozlišovacích kritériách – funkciách, ktoré umožňujú rozlíšiť, či daný vstup spadá do danej kategórie. V práci využívame existujúcu množinu elementárnych kritérií. Zavádzame rozlišovacie kritériá akcií a plánov, nad ktorými robíme inferenciu vrátane revízií. Agentova vnútorná reprezentácia je teda jednotná, agentove znalosti o kauzalite, ktoré využíva v plánovacom procese, sú založené na koncepte rozlišovacích kritérií. Model implementujeme v programovacom jazyku JAVA.

Kľúčové slová: kauzalita, plánovanie, rozlišovacie kritériá, BDI

Predhovor

Diplomová práca sa zaoberá problémom sémantickej reprezentácie pragmatických znalostí. Snažíme sa modelovať sémantiku prirodzeného jazyka, pričom významy zakladáme na tzv. rozlišovacích kritériách. Vychádzame z teoretického návrhu sémantiky od Jána Šefránka a konkrétnej implementácie od Martina Takáča. V existujúcom modeli boli zatiaľ implementované len elementárne kritériá, reprezentujúce statické objekty, vzťahy dvoch objektov a elementárnu zmenu objektu v čase. Pre vývoj umelých entít (tzv. agentov) s vnútornou reprezentáciou sveta založenou na rozlišovacích kritériách ale tieto znalosti nepostačujú. V práci sa zaoberáme možnosťami reprezentácie kauzality v modeli. Ďalej chceme obohatiť agenta o plánovací modul, aby dokázal v prostredí cielene konať akcie.

V práci najskôr uvedieme čitateľa do problematiky spomenutím prác a modelov v danej oblasti. Potom stručne popíšeme existujúci model, na ktorom budeme stavať. V jadre práce sa najprv zaoberáme teoretickým návrhom modelu. Popíšeme reprezentáciu kauzality a plánovač. Načrtneme tiež možnosti rozšírenia práce. Nakoniec uvedieme simulačné experimenty a zhrnieme výsledky.

Pri tvorbe práce som si najskôr naštudoval relevantné články, aby som získal prehľad o problematike. Potom som vytvoril teoretický návrh riešenia problému. V súlade s teoretickým návrhom som moju časť doprogramoval do existujúceho softvéru. V tejto fáze som sa snažil odstrániť niektoré nedostatky teoretického modelu, na ktoré som prišiel počas implementácie. V prostredí som vykonal experimenty na potvrdenie praktickej funkčnosti modelu a zistenie jeho nedostatkov.

Počas prípravy, ako aj pri písaní konečnej podoby diplomovej práce mi veľmi pomohol môj školiteľ Martin Takáč – jednak poskytnutím relevantných informačných zdrojov, ako aj cennými pripomienkami k návrhu riešenia a konečnému obsahu práce.

Obsah

1 Úvod

Medziľudská komunikácia je jeden z hlavných predpokladov pokračujúceho vývoja našej spoločnosti a práve schopnosť používať komplexný jazyk odlišuje ľudí od ostatných živočíšnych druhov. Okrem empirických poznatkov získaných pri pozorovaní a analyzovaní medziľudskej komunikácie sa na skúmanie jazyka v ostatnom čase stále viac používajú počítačové modely. Dajú sa aplikovať na skúmanie rôznych aspektov komunikácie, ale aj na sledovanie vývoja jazyka ako globálneho systému či osvojovanie jazyka u jednotlivcov. S jazykovými výrazmi, ktoré sú externe pozorovateľné pri komunikačných aktoch, sú asociované významy, ktoré má jedinec (nazvime ho agent) interne uložené vo svojom kognitívnom orgáne. V diplomovej práci sa zaoberám práve počítačovým modelovaním jazyka. Sémantika jazyka je vybudovaná na rozlišovacích kritériách, ktoré sú založené na rozlišovacej schopnosti agenta a vytvárajú sa počas jeho interakcie s prostredím. Významy teda nie sú umelé, ale sú ukotvené vo svete, v ktorom sa agent pohybuje.

Moja práca stavia na teoretickom modeli kognitívnej sémantiky rozlišovacích kritérií Jána Šefránka ([1]) a praktickej implementácii modelu od Martina Takáča ([2]). Komunita používateľov jazyka je modelovaná softvérovými agentami, ktoré medzi sebou komunikujú a vytvárajú si významy kategorizáciou percepčných vstupov. V modeli je možné reprezentovať statické objekty, vzťahy medzi dvoma objektami a zmenu objektu v čase. Agenty môžu o svojich významoch komunikovať, model sa dá teda použiť napríklad na skúmanie medzigeneračného kultúrneho prenosu jazyka.

Mojou úlohou je obohatiť model o kauzálny a plánovací modul agenta. Kauzálny modul má prispieť k agentovmu chápaniu dôsledkov vykonaných akcií. Agent vykonáva akcie, aby nebol len nečinnou súčasťou prostredia, aby dokázal prostredie aktívne ovplyvňovať. Ak agent dokáže asociovať akcie s predpokladmi a dôsledkami, bude pripravený reprezentovať sémantiku niektorých elementárnych slovies. Plánovací modul má zabezpečiť, aby výber akcií nebol náhodný, ale aby agent vykonával akcie za účelom splnenia svojich cieľov. Agent si musí dokázať vybudovať reprezentáciu sveta, ktorá zodpovedá požadovanej (imaginárnej) situácii.

V modeli sa mi podarilo implementovať elementárne chápanie kauzality agentom. Zvolil som zjednodušenie, ktoré predpokladá, že každá akcia ovplyvňuje iba jeden objekt prítomný na scéne. Teoreticky som navrhol možné rozšírenie na ľubovolný počet objektov. Taktiež som implementoval plánovač, pomocou ktorého dokáže agent zostaviť sekvenciu akcií, ktoré ho majú doviesť k požadovanému cieľu. Agent je schopný plán počas jeho vykonávania tiež revidovať.

Do budúcna ostáva doriešiť kauzálne prepojenie akcií s viacerými objektami, čo predpokladá zásahy do kauzálneho, ale aj plánovacieho modulu agenta. Taktiež samotné naučenie dôsledkov akcií može byť problematické, ak by sa agent pohyboval v príliš dynamickom prostredí, v ktorom by akcie naraz vykonávalo veľa agentov. Zaujímavé je aj prepojenie kauzálneho modelu s jazykovým, čo umožní agentovi pri komunikácii okrem podstatných a prídavných mien využívať aj slovesá, ktoré hrajú v jazyku veľmi dôležitú úlohu.

2 Prehľad

2.1 Multi-agentové systémy

Multi-agentové systémy (MAS) sú moderný prístup k riešeniu mnohých komplexných problémov. Slúžia na modelovanie distribuovaného systému tzv. agentov. Základnou vlastnosťou agenta je jeho autonómnosť. Riadenie celého systému je decentralizované a celkové globálne správanie emerguje z lokálnych interakcií.

Dôležitou zložkou MAS je prostredie. Agent ho vníma pomocou receptorov a ovplyvňuje ho prostredníctvom efektorov (ktoré zabezpečujú vykonávanie akcií). Agenty využívajú prostredie aj na prenos informácií medzi sebou. Okrem receptorov a efektorov agent obsahuje modul definujúci jeho správanie. Správanie agentov môže byť ovplyvnené iba vstupmi (dátami), vtedy hovoríme o reaktívnych agentoch (*reactive*, *data-directed*). Správanie niektorých agentov je naopak orientované na cieľ (*goal-directed*). Agenty, ktoré sú schopné sa rozhodovať a uvažovať o možných následkoch akcií, sa nazývajú deliberatívne (*deliberative*). Potrebujú mať vytvorený vnútorný model sveta a pri výbere najlepšej akcie na vykonanie používajú plánovací mechanizmus. Reaktívny agent model sveta nepotrebuje, pri výbere akcie sa riadi len momentálnymi vstupmi. Existujú aj hybridné agenty, ktoré obvykle reaktívne riešia akútne problémy a deliberatívne sa správajú, ak majú dosť času. Racionálne agenty v sebe môžu obsahovať zložitý kognitívny orgán, ktorý používajú na riadenie svojich

činností.

MAS našli aplikáciu v mnohých oblastiach, kde neboli konvenčné programovacie techniky úspešné. Ako príklad poslúži riadenie dopravy alebo riadenie autonómneho robota. Okrem toho sa dajú použiť na modelovanie akejkoľvek komunity viacerých koexistujúcich jedincov.

2.2 Vývoj jazyka kultúrnym prenosom

Multi-agentové systémy sa dajú použiť aj na modelovanie spoločenstva komunikujúcich jedincov. My sa zameriavame hlavne na procesy osvojovania (akvizície) a emergencie jazyka. V otázke vývoja jazyka v ľudskej populácii existujú dva názorové tábory odborníkov – jedni uprednostňujú genetický a druhí negenetický prenos, väčšinou ale pripúšťajú obidva vplyvy. V našom modeli sa nezaoberáme mierou jedného či druhého vplyvu, predpokladáme obidva faktory[1].

Šírenie jazyka v populácii simulujeme kultúrnym prenosom. Je to evolučný proces s nasledovnými vlastnosťami (podľa [3]):

- jazykové štruktúry sú uchovávané v pamäti jednotlivcov a nie v génoch
- prenos je zabepečený učením a nie dedením
- variabilitu spôsobujú nepresnosti v komunikácii, zašumené prostredie, prípadne vedomé inovácie hovorcov
- výsledná podoba jazyka záleží od rôznych selekčných tlakov (najmä maximalizovanie komunikačného úspechu a minimalizovanie kognitívneho spracovania)

Existuje viacero modelov vývoja jazyka kultúrnym prenosom, spomedzi ktorých uvedieme dva asi najpodstatnejšie – iterované učenie a modely založené na samoorganizácii. Prehľad týchto dvoch prístupov je tiež možné nájsť v [3].

2.2.1 Iterované učenie

V modeli iterovaného učenia existuje populácia agentov, ktoré sú rozdelené do dvoch generácií, na rodičov a deti. Jazyk tu existuje v dvoch formách. Jednak ako

1 Jazykové výrazy sa prenášajú negeneticky, tzv. kultúrnym prenosom v rámci populácie. Významy výrazov sa ale vytvárajú algoritmami, ktoré majú agenty zabudované (analógia genetického prenosu).

vnútorná reprezentácia, ktorú sa agent (dieťa) musí naučiť. Okrem toho sú tu externe pozorovateľné jazykové výpovede, ktoré sa prenášajú komunikáciou. Striedajú sa fázy akvizície (osvojovania) a produkcie jazyka. Deti si osvojujú jazyk prostredníctvom učenia od rodičov a vytvárajú si vlastnú vnútornú reprezentáciu. Po čase sa populácia detí zmení na populáciu rodičov, starí rodičia odídu a prídu nové agenty, ktoré predstavujú populáciu detí v ďalšej iterácii. Kultúrny prenos je v tomto prípade medzigeneračný, vertikálny, agenty vrámci jednej generácie medzi sebou nekomunikujú.

Publikované modely však majú niekoľko nedostatkov. Sémantika komunikovaných výrazov je zanedbaná (výrazy nie sú ukotvené[2]) – významy sú umelé, vopred dané a rovnaké pre všetky agenty. V procese komunikácie sa tiež využíva telepatia – učiaci sa dostane od učiteľa okrem externej výpovede aj učiteľovu vlastnú internú reprezentáciu, čo sa pri bežnej medziľudskej komunikácii nedeje.

2.2.2 Samoorganizácia

V modeloch vývoja jazyka založených na samoorganizácii je jazyk považovaný za dynamický systém, ktorý postupne emerguje prostredníctvom lokálnych interakcií medzi jednotlivými používateľmi jazyka. Jazyk sa vyvinie vďaka pozitívnej spätnej väzbe medzi výberom jazykového prostriedku a jeho úspešnosťou. Ak má napríklad agent pomenovať nejaký objekt na scéne, vyberie si taký výraz zo svojho slovníka, ktorý už predtým viedol k porozumeniu zo strany poslucháča. Ak poslucháč výraz opäť pochopí, väzba medzi výrazom a významom sa posilní, ak nie, väzba slabne. Počas interakcií sa tu komunikujú iba výrazy, významy má každý agent interne uložené v podobe tzv. diskriminačného stromu. V tomto modeli sú významy ukotvené, pretože je vytváranie významov priamo previazané s percepciou. Kultúrny prenos v týchto experimentoch je horizontálny – komunikujú rovnocenné agenty (t.j. nedelia sa na rodičov a deti).

Aj ku týmto modelom máme isté výhrady. Komunikujú sa iba objekty aktuálne prítomné na scéne – komunikácia je situačne viazaná. Ďalší nedostatok je, že agenty si budujú len reprezentácie zodpovedajúce podstatným alebo prídavným menám – slovesá a komplexnejšie kategórie chýbajú.

2 Problém ukotvenosti symbolov je popísaný v časti venovanej subsymbolovému prístupu alebo v ([4]).

2.3 Modelovanie sémantiky prirodzeného jazyka

Prirodzený jazyk sa dá skúmať z dvoch základných pohľadov, máme na mysli syntax a sémantiku jazyka. Syntax sa zaoberá označením jednotlivých pojmov v jazyku a do hry vstupujú aj gramatické pravidlá. Sémantika sa zameriava na skúmanie významov pojmov. Tieto dva fenomény sú samozrejme úzko prepojené a ovplyvňujú sa navzájom. Pred sémantickým rozborom jazyka treba obvykle previesť syntaktickú analýzu. Platí ale tiež, že pri syntaktickej analýze zložitejších jazykových konštruktov sa bez sémantických znalostí nezaobídeme.

Modelovanie syntaxe sa vo všeobecnosti považuje za jednoduchší problém. Významnou osobnosťou, ktorá sa zaoberala okrem iného syntaxou jazyka, je psycholingvista Noam Chomsky. V druhej polovici 20. storočia bol jednou z hlavných postáv novovznikajúcej vednej disciplíny – kognitívnej vedy. Svojou prácou sa mu dokonca podarilo prispieť aj do oblasti informatiky, pretože vytvoril hierarchiu formálnych gramatík, dnes známu ako Chomského hierarchia (napr. v [5]). Gramatiky sú v nej usporiadané podľa zložitosti nasledovne (od najjednoduchšej po najzložitejšiu): regulárne, bezkontextové, kontextové, frázové. Regulárne a bezkontextové sa dajú relatívne efektívne automaticky spracovať. Žiaľ, prirodzený jazyk obsahuje aj štruktúry, ktoré sú za hranicami bezkontextových gramatík, čiže ešte stále nemáme algoritmus, ktorý by dokázal efektívne generovať a analyzovať vety prirodzeného jazyka.

Modelovanie sémantiky je ešte o poznanie zložitejšie. Nielenže nemáme potrebné algoritmy, nepoznáme ani dostatočne robustnú formu, v ktorej by sme dokázali významy zachytiť. V ďalších odstavcoch sa budeme zaoberať konkrétnymi prístupmi k modelovaniu sémantiky prirodzeného jazyka.

2.3.1 Konekcionizmus

Konekcionistický (nazývaný tiež subsymbolový) prístup je inšpirovaný ľudským mozgom. Základným stavebným prvkom takéhoto systému je umelý neurón, pripomínajúci neuróny v mozgu. Umelé neuróny sa dokážu poprepájať a vytvoriť tak umelú neurónovú sieť (NS). Umelé NS dokážu vykonávať niekoľko činností, ktoré sú využiteľné pri modelovaní sémantiky prirodzeného jazyka. Sú to najmä zovšeobecňovanie, abstrakcia príznakov, klasterizácia alebo redundantné uchovávanie

informácie. Výhodou umelých NS je tiež, že aj v ľudskom mozgu je jazyk zakódovaný v podobe NS, čiže takáto reprezentácia je aj kognitívne plauzibilná.

Nevýhodou NS (nielen v oblasti spracovania jazyka) je, že sieť funguje ako čierna skrinka, nedokážeme nahliadnuť do vnútornej reprezentácie. Z čisto funkčného hľadiska nám to zas nemusí až tak vadiť, ak NS robí to, k čomu bola navrhnutá.

Spomeniem dva konekcionistické modely sémantiky jazyka. Prvým je model, v ktorom organizmus disponujúci robotickým ramenom reaguje na zrakové a sluchové (jazykové) vstupy. Druhým je samoorganizujúca sa mapa, ktorá si dokázala vytvoriť reprezentáciu významov slov.

Cangelosi a Parisi (prehľad napr. v [6]) popisujú umelý organizmus s robotickým ramenom, ktorý vníma prostredie pomocou zrakových a sluchových senzorov. Jeho kognitívny orgán je implementovaný neurónovou sieťou, ktorá v sebe zahŕňa všetky aspekty správania robota (senzomotorické schopnosti, percepciu, kategorizáciu a jazyk). To umožňuje skúmať interakciu medzi jazykom a ostatnými kognitívnymi a senzomotorickými schopnosťami.

Autori sa snažili naučiť sieť diferencovať medzi slovesami a podstatnými menami, ktoré sú zo sémantického pohľadu asi najvzdialenejšie slovné druhy (slovesá sa týkajú akcií a procesov, podstatné mená popisujú statické entity). Ak sieť dokáže rozlišovať medzi týmito dvoma slovnými druhmi, môžeme ju skúsiť natrénovať aj na rozlišovanie ďalších (prídavné mená, atď.).

Vstupy siete (senzorické receptory) sú troch typov – zrakové, sluchové a proprioceptívne (informácie o polohe ramena). Jazykový vstup je teda vo zvukovej forme. Výstupy siete kódujú pohyby svalov vedúce k pohybu ramena.

Ak by sme neuvažovali jazykový modul agenta, tak by sa rozhodol, akú akciu má vykonať, len na základe vizuálneho vstupu. Keď však pridáme jazyk, môže byť správanie ovplyvnené nielen zrakovými, ale aj sluchovými senzormi. Potom sa môže agent pri tom istom vizuálnom vstupe správať odlišne na základe iného verbálneho podnetu (napr. vo forme počutého príkazu).

V experimentoch sa podarilo rozlíšiť slovesá od podstatných viet. Bola nájdená kovariancia medzi slovesami a akciami (výstupné aktivácie), pričom tie boli relatívne nezávislé od vizuálnych vstupov. Naopak, pri podstatných menách sa sieť zameriava (najmä) na vizuálne vstupy.

Druhý konekcionistický model zachytenia sémantiky jazyka, ktorému sa chcem venovať, je implementovaný pomocou **samoorganizujúcej sa mapy** (SOM). Autorom modelu SOM je Teuvo Kohonen[3]. Ako už názov napovedá, pôjde v tomto prípade o samoorganizáciu (t.j. učenie bez učiteľa). SOM sa využíva na zobrazenie mnohorozmerného priestoru do 2D (prípadne 3D), dokážeme teda oveľa lahšie vizuálne vnímať závislosti medzi objektami. Dobrou vlastnosťou SOM je, že objekty blízke v pôvodnom priestore sú si blízke aj vo výslednom zobrazení – podobnosti medzi objektami teda ostávajú zachované. Uvediem dva experimenty, v ktorých sa sieť naučila významy slov (podľa [7]).

V prvom experimente dostávala sieť na vstupe vektory reprezentujúce zvieratá. Každá zložka vektora znamenala nejakú vlastnosť zvierata – veľkosť, popis tela (počet nôh, srsť, krídla, atď.), obľúbená činnosť (plávanie, behanie, atď.). Dokopy každé zviera reprezentovalo 13 atribútov. Sieti sa počas učenia prezentovali jednotlivé druhy a nakoniec dokázala zvieratá rozmiestniť v dvojrozmernom priestore, pričom zvieratá s podobnými charakteristikami boli umiestnené blízko seba. Inak povedané, sieť si vytvorila vnútornú reprezentáciu zvierat podľa vstupných atribútov.

Druhý experiment pracoval s jednoduchými trojslovnými vetami (napr. „Robo pomaly beží.“). Sieť sa mala naučiť reprezentovať význam slova na základe dvojslovného kontextu. Opäť sa jej podarilo rozumne usporiadať slová na dvojrozmernej mriežke. Oddelila slovné druhy a aj vrámci nich zoskupila sémanticky blízke slová.

2.3.2 Symbolový prístup

Protipólom ku konekcionizmu je symbolový prístup, ktorý je založený na priamej manipulácii so symbolmi. Systém nie je ako čierna skrinka, ktorej funkčnosť sa dá analyzovať len podľa zvonku pozorovaného správania. Dôležitá je vlastnosť kompozicionality – zložitejšie štruktúry sa dajú komponovať z jednoduchších a tvorca môže do vnútornej reprezentácie nahliadnuť a porozumieť jej. Takáto reprezentácia je samozrejme želaná, prináša však so sebou nové úskalia. Najväčším z nich je asi **problém ukotvenosti symbolov** (*symbol grounding problem*). Symboly vo svojej podstate nenesú žiaden význam – pracujeme s nimi iba ako s tokenmi pomocou

3 Preto je sieť niekedy označovaná aj ako Kohonenova sieť, resp. mapa.

syntaktických pravidiel. Keď ale symboly majú poslúžiť na reprezentáciu významov, vzniká problém, ako tie významy ukotviť v realite. Riešenie, ktoré navrhuje Harnad v [4], je ukotvenie symbolovej reprezentácie pomocou nesymbolovej reprezentácie. Reprezentácia by teda mala byť viacvrstvová. Spodné dve vrstvy by boli nesymbolové, tvoriac obrazovú (*iconic*) a kategorickú (*categorical*) reprezentáciu. Obrazová reprezentácia je iba projekcia objektov a dejov zachytených senzormi, kategorická zahŕňa kategórie vytvorené detekciou príznakov. Až nad nimi sa nachádza vrstva symbolovej reprezentácie, ktorá operuje so symbolmi, tie sú už ale ukotvené v nesymbolovej podvrstve. Dobrá sémantická reprezentácia by mala byť modulárna, a okrem symbolovej by mala obsahovať aj nejakú nesymbolovú (napr. konekcionistickú) vrstvu.

V ďalšej časti sa zameriam na dva konkrétne prístupy, konceptuálny priestor Petra Gänderforsa a sémantiku rozlišovacích kritérií Jána Šefránka[4].

Kognitívna sémantika Petra Gänderforsa ([8]) je založená na **konceptuálnom priestore**. Koncepty (pojmy) sú reprezentované v mnohorozmernom prietore, pričom každá dimenzia zodpovedá nejakej vlastnosti konceptu. Indivíduá predstavujú body v konceptuálnom priestore, triedy objektov sú množiny bodov. Prirodzený pojem je ľubovolná konvexná oblasť v konceptuálnom priestore. Pojmy blízke v konceptuálnom priestore zodpovedajú podobným pojmom aj v skutočnosti. Vieme tu reprezentovať prototypy, čiže najlepších reprezentantov kategórií. Intuitívne, prototypy získame spriemernením hodnôt všetkých vlastností všetkých členov kategórie.

Nedostatky Gänderforsovej sémantiky sú zhrnuté v [1]:

- chýba presné kritérium prirodzených pojmov (nie každý konvexný podpriestor konceptuálneho priestoru je prirodzený pojem)
- konštrukcia konceptuálneho priestoru reprezentujúceho reálny svet sa zdá byť nemožná
- mnohé dimenzie majú kvalitatívny a nie kvantitatívny charakter (ťažko definovať metriku a podobnosť)

4 Hoci tieto modely uvádzam v časti venovanej symbolizmu, oba sa dajú zaradiť skôr na pomedzie symbolového a nesymbolového prístupu. Sám Gänderfors nazýva svoj prístup konceptuálnym. Ako neskôr uvidíme, aj naša implementácia Šefránkovho modelu zahŕňa kategorickú aj symbolovú vrstvu, ako ich navrhol Harnad v [4].

- nemá vhodné prostriedky na reprezentáciu slovies (teda ani viet)

Pre moju prácu je dôležitá práve posledná výhrada, keďže sa zaoberám kauzalitou a plánovaním, ktoré úzko súvisia so sémantikou slovies. Gänderforsov konceptuálny priestor teda zrejme nie je vhodný model, s ktorým by som mohol pracovať.

Ján Šefránek v už spomínanom článku [1] zakladá významy na abstrakcii schopnosti rozlišovať. Zavádza pojem **rozlišovacieho kritéria**. Význam jazykového výrazu charakterizuje ako kritérium, ktoré dovoľuje rozlíšiť, či nejaký objekt zodpovedá tomuto jazykovému výrazu, prípadne či bol výraz adekvátne použitý v nejakej situácii. Formálne si môžeme rozlišovacie kritérium predstaviť ako funkciu, ktorá objektu priradí hodnotu zodpovedajúcu jeho príslušnosti do danej kategórie. Konkrétnu implementáciu modelu aj rôzne typy kritérií uvádzame v ďalších častiach práce.

2.4 Plánovanie

Jadro práce sa zaoberá plánovaním, preto na tomto mieste uvedieme plánovacie techniky, ktoré neskôr využijeme. Plánovací mechanizmus záleží do veľkej miery od externých a interných faktorov modelu sveta. Za externé faktory považujeme vlastnosti prostredia, v ktorom sa agent pohybuje. Interné faktory sú dané znalosťami a schopnosťami agenta. Okrem toho pri plánovaní určité vlastnosti sveta ignorujeme, aby sme náš model zjednodušili (napr. pracujeme s diskrétnym časom, zanedbávame dobu vykonania akcií, atď.).

Prostredie v našom modeli je dynamické – objekty môžu vznikať a zanikať, prípadne meniť niektoré svoje vlastnosti (napr. pozíciu). Zmena môže byť spôsobená prostredím samotným alebo ako dôsledok agentovej akcie. Agent teda nie je schopný v každom okamihu poznať presný stav prostredia, a už vôbec nedokáže s určitosťou povedať, ako bude prostredie vyzerať v budúcnosti. Musí si teda vytvoriť vnútornú reprezentáciu, ktorá by čo najpresnejšie zodpovedala skutočnosti. V ďalšom texte sa zameriame na existujúce prístupy k plánovaniu vhodné pre náš model, najskôr ale načrtneme plánovanie vo všeobecnosti.

2.4.1 Všeobecne o plánovaní

V [9] je problém plánovania prirovnaný k hľadaniu jazyka, ktorý spĺňa nasledovné

dve podmienky:

- jazyk je dostatočne expresívny pre problémy, ktoré chceme riešiť
- jazyk ešte stále ponúka dostatočne efektívny algoritmus

V prvej podmienke požadujeme istú expresivitu jazyka (čo zároveň predpokladá istý stupeň jeho zložitosti). Druhá podmienka má zaručiť, aby zvolený jazyk bol stále dostatočne jednoduchý na implementáciu výkonného plánovača. Pri pokuse o vytvorenie fungujúceho plánovača by sme teda nemali vziať ľubovolný reprezentačný jazyk a snažiť sa vytvoriť plánovač inferujúci nad týmto jazykom, ale reprezentáciu vhodne prispôsobiť, aby vyhovovala našim požiadavkám.

Pri voľbe vhodného jazyka uvažujeme štyri základné reprezentačné zložky:

- reprezentácia akcií
- reprezentácia stavov
- reprezentácia cieľov
- reprezentácia plánov

V skratke uvediem implementáciu týchto štyroch elementov v jazyku STRIPS[5], ktorý bol použitý v rovnomennom priekopníckom plánovacom programe. Stavy aj ciele sú v ňom reprezentované ako konjunkcia literálov (predikáty aplikované na konštanty, ktoré môžu, ale nemusia, byť negované).

Akcie sa v STRIPS-e nazývajú operátory a majú tri komponenty:

- popis akcie (*action description*)
- predpoklad (*precondition*)
- dôsledok (*effect*)

Popis akcie je jednoducho meno (alebo iný identifikátor) akcie spolu s jej parametrami. Predpoklad v STRIPS-e tvorí konjunkcia pozitívnych literálov. Pred (prípadným) použitím operátora musia byť jeho predpoklady splnené. Dôsledok je konjunkcia literálov (pozitívnych alebo negatívnych) popisujúca zmenenú situáciu po aplikácii operátora. Tento trojzložkový prístup k reprezentácii akcií je použitý snáď vo všetkých plánovačoch a my sa ho budeme tiež pridržiavať (reprezentácia samotných zložiek bude samozrejme odlišná).

5 Z angl. *STanford Research Institute Problem Solver*. Program sa síce nazýva riešič problémov (*problem solver*) a nie plánovač, v čase vzniku programu (1970) sa však medzi riešením problémov a plánovaním nerozlišovalo.

V STRIPS-e je použitý plánovač s čiastočným usporiadaním (*partial order planner*). To znamená, že kroky, z ktorých plán pozostáva, môžu, ale nemusia, byť usporiadané. V prípade, že sú kroky usporiadané, musí ich agent vykonať v presne definovanom poradí. Ak nie sú usporiadané, poradie vykonávania je ľubovolné. To umožňuje vytvárať istým spôsobom univerzálne plány, vyžaduje si to ale dodatočné štruktúry v rámci plánovača. Alternatíva je plánovač s úplným usporiadaním (*total order planner*), kde výsledný plán tvorí presná postupnosť krokov. Podrobnejší popis programu STRIPS spolu s príkladmi nájde čitateľ v [9], v kapitole o plánovaní.

Mnoho plánovačov (o.i. STRIPS) požaduje, aby boli poznatky agenta o svete korektné. Spoliehajú sa na bezchybné senzory, ktoré im poskytnú vždy správnu informáciu o stave prostredia a možných dôsledkoch akcií. Takáto presnosť reprezentácie je však príliš obmedzujúca podmienka, preto uvedieme model, ktorý takúto reštrikciu nepredpokladá.

2.4.2 BDI architektúra agenta

BDI agenty sú špeciálnym typom racionálnych agentov, ktoré sú vhodné do dynamického prostredia. Agenty koexistujú v prostredí a musia reagovať na zmeny v reálnom čase. V [10] sú zhrnuté základné charakteristiky takéhoto prostredia:

1. V každom okamihu existuje veľa spôsobov, ako sa môže prostredie vyvíjať (formálne, prostredie je nedeterministické).
2. V každom okamihu existuje veľa akcií, ktoré môže agent vykonať.
3. V každom okamihu môže mať agent veľa cieľov, ktoré sa snaží naplniť.
4. Akcie, pomocou ktorých dokážeme (najlepšie) naplniť ciele, sú závislé od stavu prostredia a nezávislé od vnútorného stavu agenta.
5. Agent môže vnímať prostredie iba lokálne (vníma iba časť prostredia v určitom čase).
6. Čas výpočtu a vykonania akcie by mal zodpovedať rýchlosti zmien prostredia.

Agenty musia byť schopné reprezentovať neurčitú a nejasnú informáciu, plánovať využijúc tieto poznatky a počas vykonávania plánu reagovať na prípadné zmeny (refaktorovať, meniť plán). Takéto správanie im umožňuje trojica mentálnych postojov – *Beliefs*, *Desires*, *Intentions* (BDI). Tieto tri zložky predstavujú informačný, motivačný

a deliberatívny stav agenta.

BDI model praktického uvažovania u ľudí navrhol Michael Bratman, svoje uplatnenie však našiel aj v oblasti umelých (softvérových) agentov.

Presvedčenia

Presvedčenia (*beliefs*[6]) používa agent na reprezentáciu súčasného stavu prostredia, ako aj na modelovanie dôsledkov vykonania akcií – sú informačnou zložkou agenta. Presvedčenia sa dajú dať do kontrastu s konceptom znalostí, známym z klasickej logiky. Znalosti sú vždy pravdivé, platia v danom prostredí a čase. Naopak, presvedčenia predstavujú niečo, čomu agent verí, čo považuje v danom momente za pravdivé. Tento prístup je lepší na reprezentáciu prostredia, o ktorom nemá agent dostatok (pravdivých) informácií.

Okrem statických informácií obsahuje množina presvedčení už spomenuté (pravdepodobné) dôsledky vykonania akcií. Tie vie agent využiť na inferenciu, čo mu umožňuje okrem súčasnosti modelovať aj možné budúce stavy sveta.

Želania

Motivačný stav agenta predstavujú želania (*desires*[7]). Sú to v podstate ciele, ktoré sa snaží naplniť. Treba však opäť rozlíšiť medzi klasicky ponímanými cieľmi a želaniami. Aktuálnych želaní, narozdiel od cieľov, môže mať agent viac a môžu byť navzájom nekompatibilné. S každým želaním je spojená priorita alebo úžitok, ktorý jeho splnenie prinesie.

Pokiaľ má agent nejaké želanie, neznamená to ešte, že práve vykonáva akcie na jeho uspokojenie. Na prepojenie želaní a aktuálne vykonávaných akcií slúžia zámery (ktorých sémantika sa už viac podobá klasickej sémantike cieľov).

Zámery

Zámery (intentions[8]) popisujú deliberatívny stav agenta. Je to sekvencia akcií

6 Anglický výraz *belief* znamená *presvedčenie, viera* alebo *domnienka* – v danom kontexte sa *presvedčenie* javí byť najlepší preklad.

7 *Desire* tiež vieme preložiť viacerými spôsobmi – *želanie, túžba, prianie* – budeme používať slovo *želanie*.

8 *Intention* sa prekladá ako *zámer, účel*, prípadne *intencia* – *zámer* bude asi najvhodnejší preklad na

vybraná na vykonávanie. Na zámer sa dá pozerať aj ako na aktuálne želanie spolu s plánom na jeho dosiahnutie. Aktuálnych zámerov (podobne ako želaní) môže byť viac, v množine zámerov musíme definovať aspoň čiastočné usporiadanie, pretože niektoré zámery musia byť realizované pred inými.

2.4.3 General Problem Solver

Podobne ako BDI architektúra, aj *General Problem Solver* (GPS) bol inšpirovaný ľudským myslením. Tento ambiciózny systém navrhli v roku 1959 A. Newell, J. C. Shaw a H. A. Simon. Mal slúžiť na simulovanie ľudského konania pri hľadaní riešení širokej škály problémov ([11]). Ambicióznosť projektu je skrytá práve v jeho predpokladanej univerzálnosti, ktorú autori vyjadrili aj priamo v názve (*G – general*). Dnes už je zrejmé, že šlo o prehnaný optimizmus, systém sa však stále dá použiť v niektorých oblastiach riešenia problémov a plánovania.

Práca GPS je založená na analýze prostriedkov a cieľov. Keď sa pokúšame dosiahnuť cieľový stav problému, najprv určíme rozdiel (diferenciu) medzi súčasným a požadovaným stavom. V prípade, že diferencia neexistuje, už sme v cieľovom stave. Ak diferencia existuje, hľadáme operátor (alebo iný prostriedok) na jej redukciu. Ak sa nájdený operátor nedá použiť na aktuálny stav, vznikne podproblém dostať sa do stavu, v ktorom sa operátor dá aplikovať. Keď operátor použijeme, stav, do ktorého sa dostaneme, ešte nemusí byť koncový. Z toho vyplynie ďalší podproblém, dostať sa do koncového stavu. Technika analýzy prostriedkov a cieľov sa použije rekurzívne na vzniknuté podproblémy. Ak sú diferencie a operátory na ich redukciu definované správne, malo by byť riešenie podproblémov jednoduchšie ako riešenie pôvodného problému.

Systém GPS pripomína klasickú paradigmu známu z návrhu algoritmov, rozdeľuj a panuj (lat. *divide et impera*), ktorej hlavná myšlienka je rozdeliť veľký problém na menšie podproblémy. Postupujeme rekurzívne, až kým problém nevieme priamo vyriešiť (v našom prípade aplikovaním akcie).

Najdôležitejšie pojmy pri špecifikácii GPS sú diferencie a operátory. Diferenciu môžeme chápať intuitívne ako rozdiel medzi dvoma stavmi. Jej presná definícia záleží od použitého reprezentačného jazyka. Operátor nápadne pripomína akcie

naše účely ;-)

reprezentované v jazyku STRIPS. Obsahuje nasledovné tri komponenty:

- transformačná funkcia
- podmienky aplikovateľnosti
- zoznam diferencií

Transformačná funkcia (analógia popisu akcie v STRIPS-e) určuje prechod zo stavu do stavu. Podmienky aplikovateľnosti (predpoklad v STRIPS-e) definujú, v akom stave je možné operátor aplikovať (napr. operátor možno aplikovať vtedy, ak aktuálny stav patrí do definičného obora operátora). Zoznam diferencií (dôsledok v STRIPS-e) obsahuje diferencie, ktoré daný operátor redukuje.

Systém GPS sa považuje za príklad kvalitatívnej heuristiky. V pôvodnej verzii *užívateľ* určuje, ktoré stavy považuje za rozdielne, ktorý operátor sa dá použiť na redukciu ktorej diferencie a tiež poradie dôležitosti diferencií[9] (poradie môže závisieť od obtiažnosti alebo prínosu odstránenia danej diferencie). Tieto informácie majú nekvantitatívny charakter, preto hovoríme o kvalitatívnej heuristike.

9 V našom modeli si agent sám nazbiera spomínané informácie a následne postupuje heuristicky pri plánovaní.

3 Použité multi-agentové prostredie

V tejto kapitole popíšem prostredie, ktoré vytvoril môj školiteľ Martin Takáč ([2]) a ktoré ja ďalej rozširujem. Najskôr sa venujem popisu prostredia, v druhej časti uvediem spôsoby reprezentácie prostredia agentami. Zameriam sa hlavne na tie aspekty prostredia, ktoré sú podstatné pre moju prácu.

3.1 Multi-agentové prostredie

V prehľadovej časti som uviedol hlavné vlastnosti multi-agentového prostredia, tu sa zameriam na našu konkrétnu implementáciu. Každá implementácia multi-agentového prostredia predpokladá isté zjednodušenia, abstrahujúc tak od zložitostí reálneho sveta, ktoré nie sú pre model podstatné. V našom modeli neuvažujeme zložité vizuálne vnímanie prostredia, čas beží diskrétne a akcie sú vykonateľné v jednom časovom kroku. Priblížim, čo tieto zjednodušenia znamenajú.

Agenty sa pohybujú po dvojrozmernej mriežke. Abstrahuje sa od vizuálneho vnímania scény – predmety vnímajú v podobe rámcov. Rámec je zoznam čŕt vnímaného objektu. Každá črta je identifikovateľná menom, ku ktorému je priradená jej hodnota. Príkladom črty je veľkosť, pozícia, farba, atď. Aj každému agentu je priradený rámec obsahujúci jeho vlastnosti. Okrem vlastností typických pre všetky objekty môže mať agent aj abstraktnejšie charakteristiky, napr. sila, veľkosť hladu, atď.

Ďalším zjednodušením je, že čas sa mení v diskrétnych krokoch. V každom

časovom kroku sa zmenia niektoré vlastnosti (napr. veľkosť, pozícia) vybraných objektov a každý agent vykoná svoju činnosť.

Vykonávanie akcií je tiež zjednodušené. Každá akcia je vykonateľná v jednom kroku bez ohľadu na jej komplikovanosť. Agent teda pri plánovaní nemusí brať do úvahy časovú zložitosť akcie.

Už som uviedol, že agent v každom kroku vykoná svoju činnosť. Používame klasický *sense/select/act* cyklus agenta. V prvej fáze vníma prostredie, v druhej vyberá najlepšiu akciu na vykonanie a v tretej akciu vykoná. Vykonanie akcie môže byť buď automatické – prostredie hneď akciu vykoná zohľadňujúc schopnosti agenta a vlastnosti prostredia. Druhá možnosť je zapísanie akcie na tabuľu (*blackboard*) a vykonanie akcie až po zozbieraní a vyhodnotení navrhovaných akcií od ostatných agentov – tento prístup slúži na riešenie možných konfliktov. V prípade, že simulujeme prostredie s jediným agentom, ten môže akciu bez otáľania vykonať.

3.2 Úrovne popisu prostredia agentom

Prostredie sa dá popísať mnohými spôsobmi – v našom modeli existujú štyri úrovne reprezentácie. Jednak je to čisto percepčná úroveň popisu, ktorá. je závislá od senzorického aparátu agenta. Ďalej si každý agent vytvára svoju vlastnú vnútornú reprezentáciu sveta. Keďže simulujeme aj komunikáciu medzi agentami, máme jazykovú úroveň – je to externe prístupná reprezentácia prostredia, agent ju priamo poskytuje ostatným, ak chce popísať scénu. Nakoniec existuje pragmatická úroveň, ktorej hlavnými prvkami sú plány a ciele agenta.

3.2.1 Percepčná úroveň

Táto úroveň tvorí rozhranie medzi vonkajším prostredím a vnútornou reprezentáciou prostredia agentom. V našom modeli agent na tejto úrovni operuje s rámcami, ktoré slúžia na popis jednotlivých objektov a agentov (viac o rámcoch je v kapitole venovanej popisu prostredia). Rámce predstavujú vstupy pre ďalšiu úroveň reprezentácie. Dala by sa pridať ešte jedna úroveň, ktorá by prevádzala vizuálny vnem na rámce, od toho, ako som už spomínal, v našom modeli abstrahujeme.

Treba upozorniť, že rámec ešte nie je reprezentácia. V prípade, že dva agenty

pozorujú tú istú scénu, vnímané rámce budú rovnaké, vnútorná reprezentácia bude (veľmi pravdepodobne) odlišná.

3.2.2 Reprezentačná úroveň

Každý agent si scénu interne reprezentuje. Vytvára si kategórie pre statické objekty aj dynamické deje v prostredí. Pre každý koncept má agent zodpovedajúce **rozlišovacie kritérium** (RK), ktoré predstavuje význam daného konceptu. RK je implementované ako funkcia, ktorá priradí percepčnému vstupu hodnotu, ktorá znamená mieru alebo pravdepodobnosť, s akou je konkrétny vstup inštanciou konceptu. Existuje viac typov RK, jednak sú to elementárne kritériá (objektov, vlastností, vzťahov, zmien), a potom zložitejšie kritéria, operujúce nad elementárnymi (RK situácií, udalostí, akcií – tieto navrhnem v mojej práci).

Kritériá objektov a vlastností sú z implementačného hľadiska totožné. Operujú nad individuálnymi rámcami. Uvediem zopár príkladov:

- *JankoHraško(f)* – indivíduum
- *ovocie(f)*, *študent(f)* – trieda objektov
- *ženatý(f)*, *veľký(f)* – vlastnosť

V stručnosti uvediem jeden zo spôsobov, ako môže agent postupovať pri učení týchto kritérií (podrobnosti nájde čitateľ v [2]). V prípade, že agent nemá pre práve spracovávaný rámec žiadne kritérium (t.j. žiadne kritérium nevracia dostatočne vysokú hodnotu), vytvorí si nové s týmto rámcom ako prvým príkladom. Naopak, ak je nejaké kritérium aktívne, aktualizuje ho týmto rámcom. Aktívnych môže byť naraz viac kritérií, agent môže teda aktualizovať viac kategórií súčasne.

Kritériá vzťahov (relácií) popisujú vzťahy dvoch objektov, čiže pracujú s dvoma rámcami. Príkladom môže byť *väčší(f_1,f_2)* alebo *blízko(f_1,f_2)*. Niektoré vzťahy sú kvantitatívne, popísujúce veľkosť podobnosti (napr. *blízko(f_1,f_2)* vracia väčšiu hodnotu pre bližšie ako pre vzdialenejšie objekty). Iné relácie sú definované na kvalitatívnej báze. Príkladom je spomínané *väčší(f_1,f_2)*, ktoré vracia 1 (pravda) ak vzťah platí, inak vracia 0 (nepravda).

Posledným typom elementárnych kritérií sú **kritériá zmien**. Podobne ako kritériá objektov, aj kritériá zmien operujú nad jediným rámcom. Berú však do úvahy predchádzajúcu a aktuálnu hodnotu rámca a reprezentujú vzťah medzi týmito

hodnotami. Vzťah môže byť rovnako ako pri RK vzťahov kvantitatívny alebo kvalitatívny. Príkladom môže byť *narástol(f)* alebo *zostal(f)*. RK zmien sú v modeli nevyhnutné pre zachytenie dynamiky prostredia. Učenie RK vzťahov a zmien je tiež popísané v článku [2].

3.2.3 Jazyková úroveň

Každý agent má v sebe zabudovaný jazykový modul, ktorý slúži na asociovanie naučených významov a slov. Slová používa pri komunikácii s inými agentami. V každom kole komunikácie vystupujú dva agenty – poslucháč a hovorca. Hovorca povie slovo a neverbálnymi technikami označí referovaný objekt na scéne (napr. na neho ukáže). Poslucháč sa snaží vo svojom reprezentačnom module nájsť význam počutého slova. Ako vidieť, v našom modeli pri komunikácii nepredpokladáme telepatiu, neprenášajú sa slová spolu s významami – významy má každý agent uložené interne. V nasledujúcom kole sa vyberú ďalšie dva agenty ako poslucháč a hovorca, a jazyková hra sa opakuje.

3.2.4 Pragmatická úroveň

Na tejto úrovni agent plánuje a dosahuje ciele v prostredí. Využíva svoje poznatky (presnejšie presvedčenia) o kauzálnych vzťahoch platných v prostredí. Vytvára si asociácie medzi rôznymi kategóriami – akcie spája s predpokladmi (vo forme RK objektov a vzťahov) a dôsledkami (RK zmien). Nadobudnuté presvedčenia využíva na prechod z aktuálnej do želanej situácie. Pragmatickej úrovni sa podrobne venujem v mojej práci.

4 Reprezentácia kauzality

Vnímanie zmien je pre agenta pohybujúceho sa v dynamickom prostredí esenciálne. Samotné zachytenie zmeny však nie je postačujúce, dôležité je tiež pochopenie, čo danú zmenu spôsobilo. So zmenou by mal mať agent asociovanú akciu, ktorá túto zmenu vykonala, ako aj situáciu, ktorá platí pred vykonaním akcie (predpoklady akcie). V ďalších podkapitolách bližšie popíšem štruktúry, ktoré zabezpečia reprezentáciu kauzality.

4.1 Akcie

Agent pohybujúci sa v našom dynamickom prostredí je schopný vykonávať akcie. Aby bol schopný vybrať z repertoára akcií tú správnu a priblížiť sa tak k požadovanému cieľu, musí si pamätať dôsledky akcií, ktoré už niekedy v minulosti vykonal. Tieto kauzálne znalosti (presnejšie presvedčenia) si uchováva podľa nasledovnej schémy:

$$(predpoklady, akcia \rightarrow dôsledky)$$

Predpoklady a dôsledky sú externé faktory, závisiace od nastavenia prostredia. Agent môže dôsledky ovplyvňovať zvolením vhodnej akcie, konečný efekt ale nakoniec aj tak záleží od mechanizmu, ktorým akcie spracúva prostredie (agent napríklad môže chcieť zdvihnúť nejaký objekt, prostredie ale pred samotným vykonaním akcie skontroluje, či je agent na zdvihnutie daného objektu dostatočne silný).

V našom modeli je táto ternárna asociácia uchovávaná v podobe **rozlišovacieho**

kritéria akcií. RK akcií je zložené z jednoduchších kritérií. Aby sme sa vyhli prílišným komplikáciám, uvažujeme iba jednoduchý model kauzality – vykonanie každej akcie sa vzťahuje iba na jeden objekt zo scény. Neskôr spomenieme aj možné zovšeobecnenie pre ľubovolný počet objektov.

Keďže každá akcia ovplyvňuje len jeden objekt, predpoklady sú vyjadrené vo forme elementárneho RK objektu. Agent môže vykonať akciu na inom objekte aktuálne prítomnom na scéne (napr. *zdvihni(X, výška)* – zdvihnutie objektu *X* do požadovanej výšky), prípadne sám na sebe (napr. *presuň(posX, posY)* – spôsobí presun na dané pozície). Každý agent je implementovaný ako objekt v prostredí, aj akcia *presuň* teda operuje s práve jedným objektom. V prípade, že akcia neobsahuje žiadny objekt ako argument, implicitne sa predpokladá, že agent vykoná akciu sám na sebe.

Akcia v sebe zahŕňa svoj názov a parametre, a je reprezentovaná **rozlišovacím kritériom popisu akcie**. Toto nové RK je takmer totožné s existujúcim RK objektu, obsahuje ale navyše názov akcie. Parametre sú akcii predávané ako rámce, kategorizácia parametrov funguje rovnako ako pri elementárnom RK objektu ([2]). Ak chceme zistiť, či konkrétna akcia vyhovuje danému RK akcie, porovnáme najprv typy akcií. Ak sa nezhodujú, návratová hodnota celého kritéria je 0. Ak sa zhodujú, porovnávame parametre akcie s hodnotami uloženými v RK, presne v súlade s algoritmom pre RK objektov.

Dôsledky akcie sú reprezentované elementárnym RK zmeny. Toto kritérium popisuje, ako sa objekt zmenil v dôsledku vykonania akcie. RK zmien sú najdôležetejšie pre výber správnej akcie počas plánovania. Spolu s RK objektov a popisov akcií sa ich agent učí nižšie popísaným spôsobom.

4.1.1 Učenie akcií

Na učenie RK akcií používame tzv. **asociačný algoritmus**, ktorý navrhol Martin Takáč v [12]. Slúži na vytváranie ternárnych asociácií medzi objektami, zmenami a samotnými akciami (typ akcie a parametre). Vzniknuté RK akcií sú hlavnou zložkou pre chápanie kauzality agentom.

V algoritme sa budujú všetky reprezentácie vo vzťahu k akciám, slúži teda nielen na naučenie RK akcií, ale aj na nastavenie elementárnych kritérií. Dalo by sa postupovať aj odlišne, dvojfázovo: agent by zvlášť kategorizoval objekty a zmeny, a

následne by si vytváral kritéria akcií, využijúc už existujúce elementráne kritériá. Prepojenie učenia akcií a ostatných kritérií do jedného celku ale zdôrazňuje dôležitosť vykonávania akcií v živote agenta.

Prejdime k bližšiemu popisu asociačného algoritmu. V prípade, že agent vykonal v predchádzajúcom časovom kroku nejakú akciu, snaží sa odhaliť zmeny, ktoré táto akcia spôsobila. Keďže v našom zjednodušenom modeli každá akcia ovplyvňuje práve jeden objekt, postupuje pre každý zmenený objekt zvlášť. Najprv si nájde, ako objekt vyzeral v predchádzajúcom čase, čo je vlastne predpoklad akcie. Potom si vypočíta zmenu medzi súčasným a predchádzajúcim stavom objektu. Z množiny všetkých RK akcií vyberie tú, ktorá najviac zodpovedá ostatnej akcii, predpokladom a zmene.

V prípade, že nie je nájdené žiadne RK akcie, vytvorí sa nové, pričom sa snažíme využiť už existujúce elementárne kritériá. Hľadajú sa teda najlepšie elementárne kritériá zodpovedajúce objektom (predpokladom), akciám a zmenám (dôsledkom). Aktivita vybraných kritérií musí byť ale nadprahová, aby vzniknuté asociácie neboli príliš všeobecné. Ak nie je nájdené dostatočne dobré elementárne kritérium, vytvorí sa nové – tu je vidieť spomínané prepojenie učenia akcií a jednoduchších kategórií.

Predpokladajme teraz, že bolo nájdené aktívne RK akcie. Najskôr zisťujeme, či aktuálna zmena zodpovedá predpokladanej (t.j. či RK zmeny vracia nadprahovú hodnotu). Ak áno, prenastavíme všetky vnorené kritériá s použitím aktuálnych pozorovaní, čím vlastne prispôsobujeme aj celé RK akcie momentálne vnímanej realite. Ak zmena nezodpovedá kritériu, vypočítame podobnosti objektu (ku kritériu objektu) a popisu akcie (ku kritériu popisu akcie), aby sme vedeli, ktoré elementárne kritérium treba zmeniť, aby kritérium akcie správne reprezentovalo kauzalitu. Ak zistíme, že predpoklady akcie sú relatívne správne, tak vytvoríme nové kritérium popisu akcie a pomocou neho vytvoríme aj nové RK akcie. Naopak, ak je správny popis akcie, ale nesprávne predpoklady, musíme vytvoriť nové RK objektu a to použiť v novom RK akcie.

Ako vidíme, asociačný algoritmus je pomerne zložitý a na viacerých miestach v ňom používame prahové hodnoty, ktoré slúžia na zistenie, či vnímaný prípad spadá do danej kategórie. Tieto hodnoty sú väčšinou dynamické, menia sa počas života agenta. Obyčajne sa postupom času zvyšujú až po vopred stanovenú prahovú hodnotu, aby agent spočiatku medzi jednotlivými prípadmi príliš nerozlišoval a snažil sa tak nájsť

spoločné, všeobecné črty. Až neskôr, keď sa prahové hodnoty zvyšujú, sa agent orientuje na špecifické črty vnemov. Je vhodné vytvoriť si najprv všeobecné pojmy, aby reprezentácia neobsahovala príliš veľa špecifických konceptov, ktoré by boli v ranej fáze vývoja agenta aj tak zbytočné. Až neskôr, keď už agent disponuje všeobecnými pojmami, sa môže zamerať na presnejšie rozlišovanie.

Veľkým problémom pre chápanie kauzality je nájdenie skutočného pôvodcu pozorovanej zmeny v prostredí. V našom modeli agent pri kategorizácii predpokladá, že spôsobuje všetky zmeny na scéne, snaží sa teda asociovať vykonanú akciu s každou zmenou[10]. Tento prístup je vhodný, ak sa agent osamote pohybuje v prostredí a to sa mení len následkom jeho konania. Avšak v prípade, ak je prostredie príliš dynamické, musí agent nejakým spôsobom rozpoznať, ktoré zmeny spôsobil on a ktoré iné agenty. To sa dá čiastočne dosiahnuť napríklad tým, že bude kategorizovať len zmeny vo svojom bezprostrednom okolí.

4.1.2 Zlučovanie kategórií

Každá kategória je v dobe vytvorenia nezávislá od ostatných, postupom času sa ale môže stať, že sa niektoré kategórie stanú príliš podobné, t.j. že vyjadrujú tie isté koncepty. Je vhodné, aby takéto kategórie boli zlúčené do jednej. Okrem elementárnych kritérií sa takto dajú zlučovať aj celé RK akcií. Prebieha to tak, že ak existujú RK akcií s rovnakými predpokladmi a popismi akcie, prislúchajúce dôsledky (RK zmien) sa zlúčia a zlúčia sa aj samotné kritériá akcií. Zlučovanie kategórií je tiež popísané v [12].

4.2 Situácie

V našom modeli sú predpoklady a dôsledky akcií viazané len na jeden objekt, čo je isté obmedzenie. Stručne navrhneme spôsob, ako by sa dali akcie previazať s viacerými objektami.

Významy sú v našom modeli reprezentované rozlišovacími kritériami. RK objektov a vlastností majú ako argument jeden objekt nachádzajúci sa na scéne a vracajú mieru príslušnosti daného objektu do kategórie reprezentovanej RK. RK

10 Podobný prístup k ranému chápaniu kauzality bol pozorovaný aj u detí, nazýva sa *magická kauzalita* ([13]).

vzťahov slúžia na popis vzťahu dvoch objektov a RK zmien popisujú zmenu jedného objektu v dvoch po sebe nasledujúcich časových krokoch. Pomocou elementárnych RK môžeme zachytiť významy operujúce nad jedným alebo dvoma objektami, to ale nie je vždy postačujúce. Človek dáva často do vzťahu viacero objektov, pričom ich počet nemusí byť vopred známy. Vytvorenie nového typu RK, operujúceho nad presne definovaným počtom objektov (napr. ternárne kritérium – vzťah troch objektov), teda zrejme nie je to pravé riešenie.

Riešením je popísať vzťahy potenciálne neobmedzeného počtu objektov, využijúc pritom existujúce elementárne kritériá. Predstavíme **rozlišovacie kritériá situácií**, ktoré tvorí kompozícia viacerých elementárnych RK – konkrétne RK objektov, vlastností a vzťahov. RK situácií slúžia na popis statickej scény, preto v sebe neobsahujú RK zmien (RK udalostí, popísané nižšie, majú aj tento dynamický komponent).

RK situácií sa dajú implementovať viacerými spôsobmi, uvediem dva z nich, ktoré sa odlišujú vo výpočte svojej návratovej hodnoty[11].

V prvom variante si RK situácií pamätá zoznam elementárnych kritérií, ktoré ho tvoria. Ku každému kritériu má pridruženú aj referenciu, na ktorú časť scény sa RK vzťahuje. Referencia je zostavená z relatívnych identifikátorov (ID) objektov. Pri RK objektov a vlastností je to jeden ID, pretože tieto RK operujú nad individuálnym objektom. Pre RK vzťahov si pamätá dva ID určujúce argumenty (referenciu) kritéria. Pri výpočte návratovej hodnoty sa nájde najvhodnejšie mapovanie relatívnych ID na absolútne (ID objektov aktuálne prítomných na scéne). Najvhodnejšie mapovanie je také, ktoré maximalizuje návratovú hodnotu RK. Do výsledku pritom väčšou mierou prispievajú RK objektov ako RK vzťahov, vlastnosti individuálneho objektu sú obyčajne podstatnejšie ako vzájomný vzťah dvoch objektov. Návratová hodnota sa následne určí tak, že sa sčítajú návratové hodnoty elementárnych kritérií a výsledok sa znormalizuje.

RK situácií môže obsahovať viacero rovnakých elementárnych kritérií, každé z nich ale musí referovať na iný objekt (vieme si napr. predstaviť RK zodpovedajúce významu *dva_kruhy*, pozostávajúce z dvoch elementárnych RK pre *kruh*, pričom každé z nich má inú referenciu). Každý komponent RK situácií je teda jednoznačne určený rozlišovacím kritériom a príslušnou referenciou.

11 Návratová hodnota RK situácií je z intervalu <0;1>, aby boli konzistentné s elementárnymi kritériami.

Druhý spôsob je založený na koncepte elementárnych RK. Elementárne kritériá operujú nad rámcami. Ak uvažujeme implementáciu kvantitatívnych RK založenú na varianciách, tak pre každý atribút (jednoznačne identifikovaný svojim menom) si kritérium pamätá jeho hodnotu, varianciu a počet výskytov. Podobne môžeme postupovať aj pri RK situácií. V tomto prípade je atribút jednoznačne identifikovaný rozlišovacím kritériom s pridruženou referenciou a hodnota atribútu je návratová hodnota kritéria pre konkrétny objekt (resp. množinu objektov). Variancie a počty výskytov sa počítajú rovnako ako pri elementárnych kritériách. Výpočet návratovej hodnoty celého RK situácie je potom analogický k výpočtu návratovej hodnoty elementárneho kritéria založeného na varianciách (pozri [2]).

Zatiaľ sme uvažovali len RK situácií zložené z elementárnych kritérií objektov, vlastností a vzťahov. Rovnako dobre si vieme predstaviť aj RK situácií ako komponent zložitejšieho RK situácií. Takto by sme vedeli zostaviť zložito štrukturovanú hierarchiu kritérií.

Načrtli sme už, ako by sa dala vypočítať podobnosť konkrétnej situácie (množiny objektov) k zapamätanému významu (vo forme RK situácie). Nevieme ale, ako by sa agent RK situácií učil. V asociačnom algoritme sa agent učí predpoklady (RK objektov) aj dôsledky (RK zmien) spolu s kategorizáciou akcií. Namiesto RK objektov by sme teda použili RK situácií. Ako má ale agent určiť, ktorá časť scény je pre danú akciu podstatná, inak povedané, ktorá časť scény reprezentuje predpoklad akcie? Existujú dva intuitívne spôsoby. Na vytvorenie nového RK situácie môže použiť buď všetky statické kritériá (RK objektov, vlastností a zmien) platné v predchádzajúcom časovom kroku (akcia sa totiž analyzuje až po jej vykonaní). Alebo agent použije len tie kritériá, ktoré sa týkajú zmenených objektov. Ani jeden spôsob sa nám nezdá úplne správny. Pri použití všetkých objektov si agent k akcii pridruží aj nepodstatné predpoklady. Ak si všíma len zmenené objekty, môže naopak nejaké predpoklady prehliadnuť. Tento problém vzniknutý pri inicializácii sa dá ešte napraviť pri pozorovaní ďalších príkladov, môže sa ale stať, že kvôli nepresnosti v predpokladoch bude neskôr konkrétna akcia príliš odlišná od kritéria, ktoré by ju malo zastrešovať.

Otvoreným problémom teda ostáva nájsť algoritmus, pomocou ktorého by sa agent mohol učiť RK situácií a ktorý by bol schopný rozlíšiť medzi požadovanými a nepodstatnými aspektami situácie.

4.3 Udalosti

RK situácií by sa dali použiť na reprezentáciu predpokladov akcií operujúcich s viacerými objektami, nepopisujú však zmeny viacerých objektov. Na popísanie zmien celej scény (alebo jej významnej časti) v dvoch odlišných časoch môžu poslúžiť **rozlišovacie kritériá udalostí**. Z formálneho hľadiska sú RK udalostí takmer totožné s RK situácií, môžu ale ako zložku obsahovať aj elementárne RK zmeny. Práve tento komponent je podstatný pre zachytenie dynamiky prostredia. RK (elementárnej) zmeny je postačujúce, ak sa zmena týka jediného objektu. Často sa však naraz zmení celá množina objektov. Pomocou RK udalostí vieme dať do vzťahu situáciu platnú pred vykonaním akcie a dôsledky jej vykonania.

Predpoklady a dôsledky akcie by sme si potom ukladali v jednom kritériu udalosti. Dokázali by sme presne určiť, čo musí objekt spĺňať pred akciou a ako ho akcia ovplyvní. Problémy uvedené pri RK situácií žiaľ platia aj pre RK udalostí.

4.4 Verbálna reprezentácia

Hoci sa v mojej práci nezaoberám jazykovým modulom agenta, načrtnem, aké slovné druhy zodpovedajú definovaným kritériám a ako by ich agent mohol využiť pri komunikácii.

RK situácií sa dajú použiť na zachytenie významu komplexných pojmov, ktoré zahŕňajú viac individuálnych objektov, prípadne popisujú stav celej scény. Takýmito zložitými pojmami sú napr. *pokoj* alebo *neporiadok*. Oveľa častejšie ale popisujú situáciu, ktorá sa (obyčajne) nedá v prirodzenom jazyku vyjadriť jedným slovom, napr. výraz *mačkaNaRozpálenejPlechovejStreche*[12] sa vzťahuje na dva objekty na scéne, označme ich *X* a *Y*. V súlade s našim návrhom by kritérium situácie pozostávalo z nasledovných elementárnych kritérií:

mačka(X), rozpálená(Y), plechová(Y), strecha (Y), na(X,Y)

Prvé štyri kritériá sú RK objektov, posledné je RK vzťahov. V prípade, že sú dostatočne aktívne elementárne kritériá, je aktívne aj výsledné RK situácie a teda výraz

12 Celý výraz sme napísali spolu, aby bolo jasné, že jeho významom je jedno RK situácie a nie súbor elementárnych kritérií (hoci v našej implementácii RK situácie zaobaľuje elementárne kritériá).

mačkaNaRozpálenejPlechovejStreche môžeme adekvátne použiť.

Ako už názov RK udalostí naznačuje, tieto kritériá popisujú nejakú udalosť odohrávajúcu sa na scéne. Takáto udalosť je častokrát verbálne popísateľná slovesným podstatným menom (gerundiom). Príkladom môže byť *zväčšovanie sa* alebo *pohybovanie*. V niektorých prípadoch si vystačíme aj s elementárnym kritériom zmeny, RK udalostí pomôžu, ak sa naraz menia viaceré objekty.

V našom modeli sme boli zatiaľ schopní sémanticky podchytiť iba podstatné a prídavné mená. Komunikácia medzi ľuďmi sa však do značnej miery opiera o slovesá. Práve RK akcií implementované v našom modeli môžu poslúžiť na ukotvenie významu (minimálne jednoduchých) slovies. Agent ich môže použiť v komunikácii na popis vlastnej akcie alebo akcie iného agenta. Taktiež by agent mohol pochopiť význam elementárnych viet.

Vieme si tiež predstaviť, že ich agenty použijú ako pomôcku pri plánovaní. V prípade, že sa agent *A* spýta agenta *B*, čo má urobiť na splnenie požadovaného cieľa, agent *B* mu môže odpovedať slovesom. Následne si agent *A* vo svojom lexikóne nájde RK akcie asociované s počutým slovom a vykoná danú akciu. V prípade, že sú slovníky oboch agentov dostatočne podobné, mal by agent *A* vykonať presne to, čo mu agent *B* poradil.

4.5 Životný cyklus agenta

Prostredie sa vyvíja v diskrétnych časových krokoch. V každom časovom kroku sa vykoná jeden obeh životným cyklom[13] agenta. Pozostáva z troch základných fáz: ***sense***, ***select*** a ***act***. Vo fáze *sense* agent vníma prostredie, fáza *select* slúži na výber akcie a fáza *act* predstavuje samotné vykonanie akcie. V ďalšej časti detailnejšie popíšem jednotlivé fázy.

Uvediem ešte krátku poznámku k životnému cyklu agenta. Ak uvažujeme viacero agentov súčasne existujúcich v prostredí, môže byť niekedy výhodnejšie, aby boli všetky agenty naraz v tej istej fáze. Najskôr by vnímali prostredie (každý agent sa nachádza vo fáze *sense*), potom by každý vyberal akciu (fáza *select*) a nakoniec by ju

13 Životný cyklus je v modeli simulovaný volaním metódy `live()` pre každého agenta v každom časovom kroku.

vykonal (fáza *act*). Ak ale jeden agent vykoná akciu, ostatné agenty budú mať pravdepodobne nesprávnu informáciu o aktuálnom stave prostredia. Takáto implementácia teda vyžaduje sofistikovanejšiu správu vykonávania akcií, napríklad zápis všetkých zamýšľaných akcií na tabuľu (*blackboard*) a následné vykonanie všetkých akcií až po odstránení potenciálnych konfliktov[14].

4.5.1 Fáza sense

Základnou činnosťou vykonávanou v tejto fáze je vnímanie prostredia. Agent si uloží aktuálne vnímanú scénu, pričom si zapamätá aj vnem scény z predchádzajúceho časového kroku.

Pre chápanie kauzality dejov prebiehajúcich v prostredí a najmä pochopenie dôsledkov akcií vykonaných agentom je dôležité vyhodnotenie predchádzajúcej akcie, ktoré sa tiež koná vo fáze *sense*. Túto činnosť má na starosti už spomínaný asociačný algoritmus.

Nakoniec sa ešte zlučujú kategórie, ktoré sú si dostatočne podobné. Keďže zlučovanie kategórií je dosť časovo náročná operácia, nevykonáva sa v každom kroku, ale vo vopred definovaných periódach.

4.5.2 Fáza select

Ako názov tejto fázy napovedá, vyberá sa tu najvhodnejšia akcia na následné vykonanie. Zatiaľ sa uspokojíme s výberom ľubovolnej akcie z repertoára akcií s náhodnými parametrami. Túto fázu rozšírime v kapitole venovanej plánovaniu.

4.5.3 Fáza act

Vo fáze *act* pošle agent prostrediu požiadavku na vykonanie akcie s vybranými parametrami. Prostredie túto akciu vykoná v prípade, že spĺňa simulované fyzikálne (poprípade iné) zákony. Ak nebola vybraná žiadna akcia, agent v tejto fáze nevykoná nič. Vykonanú akciu si agent zapamätá, aby ju mohol využiť pri aktualizácii RK akcií v ďalšom časovom kroku.

14 V našom modeli sa tomuto problému vyhýbame práve volaním metódy `live()` obsahujúcej všetky tri fázy, agent má teda zaručenú aktuálnosť vnemu.

4.6 Zhrnutie

Predstavili sme, ako si agent dokáže interne reprezentovať kauzalitu. Za týmto účelom sme definovali rozlišovacie kritériá akcií. Načrtli sme RK situácií a udalostí, ktoré by sa dali využiť pri zložitejšom modelovaní kauzality. Uviedli sme tiež navrhované prepojenie kauzálneho modulu s jazykovým a stručný popis, ako prebieha životný cyklus agenta.

Otvoreným problémom, ktorým sa ďalej nezaoberáme, ostáva učiaci algoritmus pre RK situácií a udalostí, ako aj rozpoznanie presného pôvodcu zmeny v prostredí.

5 BDI architektúra agenta

Agent, majúci doposiaľ popísané vlastnosti, dokáže vnímať statické či dynamické prostredie a vykonávať v ňom náhodné akcie. Nedisponuje ale žiadnymi mechanizmami, pomocou ktorých by dokázal účelovo (teda nie náhodne ako doteraz) vyberať akcie, ktoré by ho priblížili k požadovanému cieľu. Chceme vytvoriť agenta, ktorého správanie by bolo orientované na cieľ. Musíme pritom pamätať na isté obmedzenia nášho modelu, najmä to, že agentove znalosti o prostredí sú neúplné a nepresné.

Jedným z možných typov racionálneho agenta je BDI agent, ktorého správanie sa opiera o jeho presvedčenia (*beliefs*), želania (*desires*) a zámery (*intentions*). V ďalšom texte popíšeme jednotlivé zložky vo vzťahu k nášmu modelu.

Stojí ešte za zmienku, že väčšina, ak nie všetky, zatiaľ existujúce implementácie BDI agentov sú založené na jazyku logiky. Náš prístup je odlišný, definujúc významy ako rozlišovacie kritériá. Teoretické pozadie BDI architektúry je však natoľko všeobecné, že nie je problém namapovať ju na náš model.

5.1 Presvedčenia (beliefs)

Ako už bolo spomenuté, presvedčenia, na rozdiel od znalostí, predstavujú niečo, čo agent považuje v danom momente za pravdivé. Tento koncept výborne vyhovuje nášmu modelu, pretože rozlišovacie kritériá priraďujú každému objektu (resp. množine

objektov) mieru, do akej daný objekt (množina objektov) spĺňa kategóriu reprezentovanú RK. Množina presvedčení sa skladá z elementárnych RK (objektov, vlastností, vzťahov a zmien), ktoré sú aktívne pre aktuálnu scénu. Patrili by sem aj navrhované zložitejšie RK situácií a udalostí, ak by vykazovali dostatočnú aktivitu. Okrem toho agent disponuje aj presvedčeniami o akciách a ich dôsledkoch (RK akcií) a zatiaľ nespomenutými RK plánov. Pomocou nich dokáže agent inferovať a využíva ich v plánovacom procese.

Presvedčenia teda slúžia agentovi na popis aktuálnej situácie a popis možných (imaginárnych) situácií, do ktorých sa z aktuálnej vie dostať (vykonaním vhodných akcií).

5.2 Želania (desires)

Želania môžeme chápať ako požadované situácie. V našom zjednodušenom modeli sú želania reprezentované rozlišovacími kritériami objektov spolu s identifikátorom objektu, na ktorý sa želanie vzťahuje. Želania by sa dali vernejšie popísať navrhovanými kritériami situácií, ktoré by zachytávali podstatné aspekty požadovanej situácie.

Želania stoja na začiatku plánovacieho procesu. S každým želaním je asociovaná jeho priorita, ktorá reflektuje, do akej miery je želanie pre agenta dôležité (zrejme želanie *prežiť* bude mať dosť vysokú prioritu a agent sa ho bude snažiť naplniť za každú cenu, menej podstatné želanie *presunu na inú pozíciu* bude mať oveľa nižšiu prioritu). Agent si udržiava množinu aktuálnych želaní v prioritnej fronte. Postupne z nej vyberá jednotlivé želania a snaží sa ich naplniť. Bližší popis výberu želaní nájde čitateľ v kapitole venovanej plánovaniu.

Množina želaní sa samozrejme počas života agenta mení. Existujú dva základné typy želaní – ***achievement*** a ***maintenance*** želania ([14]). *Achievement* sú také, ktoré agent raz splní a následne ich vymaže zo zoznamu. Príkladom je *presun na inú pozíciu*. *Maintenance* želania, na druhej strane, ostávajú v zozname želaní aj po splnení, ale ich priorita sa zníži, simulujúc tak mieru naplnenia/nenaplnenia potreby. Napríklad želanie *byť nasýtený* ostane vo fronte aj tesne po najedení, ale so zníženou prioritou.

Zaujímavá je otázka, ako želania do zoznamu pribúdajú. Existuje viacero spôsobov.

Niektoré želania má agent počas celého života – väčšinou sú to práve želania typu *maintenance* (napr. už spomínané *byť nasýtený*). Iné dostáva postupne vo forme príkazov (splnenie nejakého cieľa – *achievement* želania). Želania si môže agent dokonca aj sám generovať.

5.3 Zámery (intentions)

V množine zámerov sú tie želania, ktoré sa agent rozhodol naplniť, spolu s plánmi vybranými na ich realizáciu. Z tohto hľadiska možno na zámery pozerať ako na krátkodobé požadované situácie a na želania ako na ich dlhodobé náprotivky. Rozdiel je práve v tom, že pri zámeroch sme podstatne bližšie k cieľovej situácii, keďže už uvažujeme aj plány na jej dosiahnutie.

Zoznam zámerov je implementovaný, podobne ako zoznam želaní, vo forme prioritnej fronty. Existujú dva základné typy zámerov – externé a interné. Externé sú tie, ktoré sa do fronty dostali zo zoznamu želaní. Interné sú naopak tie, ktoré si agent vygeneroval sám a vložil ich priamo medzi ostatné zámery. Týmto spôsobom dokáže agent splniť podcieľ a pokračovať vo vykonávaní nadradeného plánu. Zámery teda pripomínajú procesy známe z konvenčných operačných systémov a môžu nadobúdať rôzne stavy – bežiaci, pozastavený, ukončený, atď. ([15]). Detaily o spôsobe výberu zámerov na realizáciu sú opäť popísané v kapitole o plánovaní.

6 Plánovač

Plánovač má na starosti tri hlavné činnosti, z ktorých sa skladá plánovací proces. Najskôr je to samotné plánovanie, čiže výber aktuálneho zámeru z množiny intencií a nájdenie plánu na splnenie tohto zámeru. Druhá fáza je vykonávanie plánu, ktoré zahŕňa aj sledovanie, či sa plán vyvíja správnym smerom, spolu s prípadnými revíziami. Poslednou činnosťou plánovacieho mechanizmu je zisťovanie, či už agent dosiahol požadovaný cieľ – agent si overuje splnenie cieľových podmienok po vykonaní každej elementárnej akcie.

Na vytvorenie plánov agent používa RK akcií. Tie sa učí pomocou asociačného algoritmu popísaného v kapitole o kauzalite. Agent najprv vykonáva iba náhodné akcie a tie kategorizuje. Až neskôr, keď disponuje základnými poznatkami o kauzalite môže začať plánovať. Takýto postup sa nazýva *starting small* a prvýkrát ho využil Elman pri učení umelých neurónových sietí ([16]).

6.1 Plánovanie

Agent sa pri plánovaní opiera o RK akcií a nové **rozlišovanie kritériá plánov**. RK plánov sú zoznam RK akcií tvorený tými akciami, ktoré niekedy v minulosti priviedli agenta k požadovanému cieľu. Okrem toho v sebe RK plánu obsahuje aj predpoklady plánu (v podobe RK objektu) a cieľ (takisto RK objektu). RK plánov sú veľmi podobné kritériám akcií, plán je zreťazenie viacerých akcií. Pri použití RK plánu si ale agent

môže byť o niečo istejší, že sa mu podarí splniť cieľ, pretože tento plán už bol niekedy použitý na splnenie toho istého cieľa.

Predstavme si teraz, že agent má vybraný zámer s najvyššou prioritou. Pri hľadaní plánu na jeho realizáciu sa najskôr pozrie do zoznamu RK plánov, či sa mu už podobný cieľ nepodarilo niekedy splniť. Ak nájde vyhovujúce kritérium plánu, tak bude postupovať podľa neho.

Častokrát ale agent nemá vhodný zapamätaný plán a musí si vytvoriť nový na dosiahnutie požadovaného cieľa. Na nájdenie správnej sekvencie akcií použijeme algoritmus GPS (*General Problem Solver*), teoreticky popísaný v prehľadovej kapitole. GPS poslúži na redukciu diferencií medzi súčasným a požadovaným stavom prostredia. Diferencie sa dajú redukovať vykonaním vhodných akcií. V našom modeli sú stavy prostredia popísané rámcami, ktoré reprezentujú jednotlivé objekty a diferencie sú zmeny medzi objektami (formálne je zmena tiež rámec). Teraz bližšie popíšem našu implementáciu GPS.

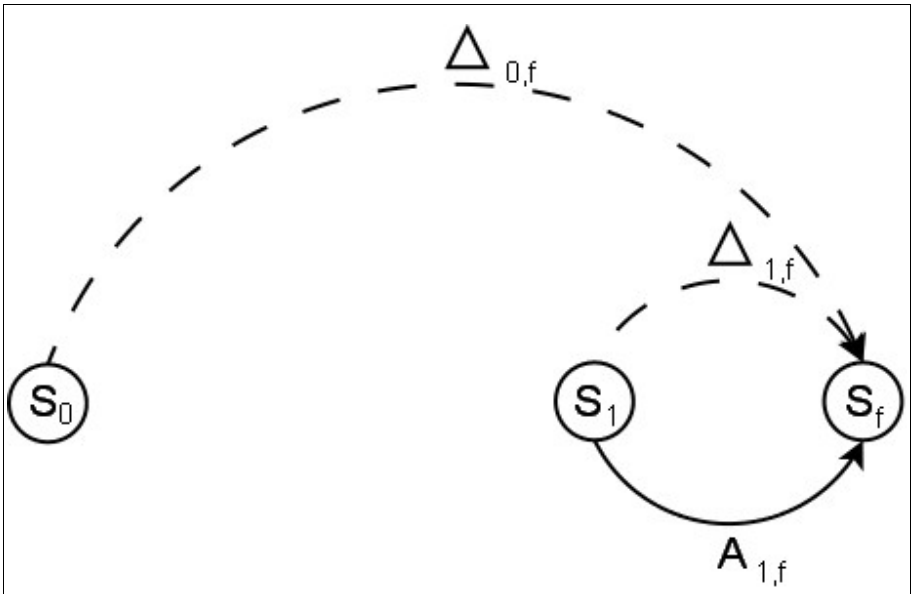
Obrázok 6.1. Začiatok plánovacieho procesu.

Agent sa potrebuje dostať zo súčasnej do požadovanej situácie. Požadovaná situácia zodpovedá aktuálnemu zámeru agenta. Na obrázku 6.1 je zobrazený začiatok plánovacieho procesu, S_0 symbolizuje súčasnú situáciu, S_f je požadovaná situácia (finálna z pohľadu jednej etapy plánovača).

Agent ďalej nájde diferenciu medzi súčasnou a požadovanou situáciou. Postup je veľmi jednoduchý. Finálna situácia je vyjadrená kritériom objektu spolu s identifikátorom objektu na scéne. Preto si agent aj z aktuálnej situácie všíma len objekt s daným ID. Z kritéria objektu, ktoré reprezentuje požadovanú situáciu, vypočítame priemernú hodnotu (prototyp) rámca[15]. Teraz už vieme určiť konkrétnu zmenu $\Delta_{0,f}$ medzi S_0 a S_f[16].

15 Na výpočet prototypu RK slúži metóda `meanCase()`.

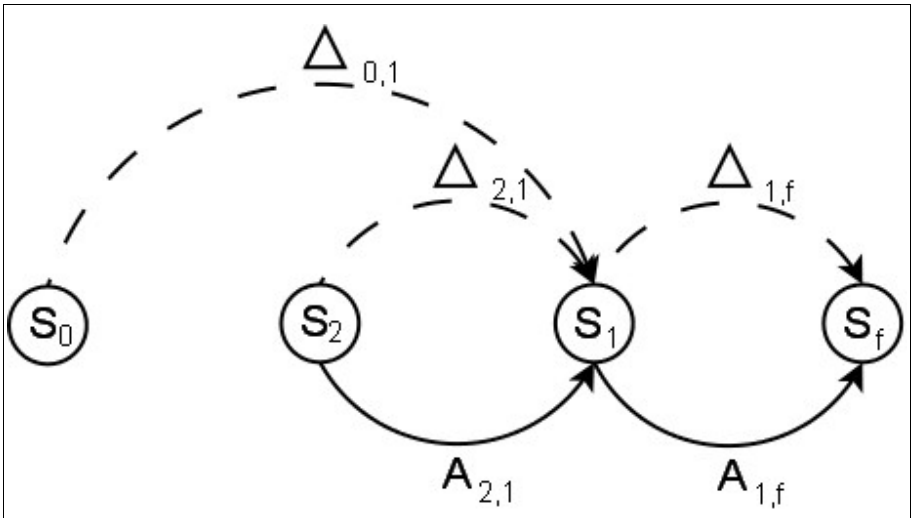
16 Zmenu zistíme pomocou metódy rámca `deltaFrame(Frame f1, Frame f2)`, ktorej zadáme ako argumenty rámce zodpovedajúce pôvodnému a požadovanému stavu objekta.



Obrázok 6.2. Hľadanie diferencie medzi aktuálnou a požadovanou situáciou. Nájdená akcia redukuje diferenciu len čiastočne.

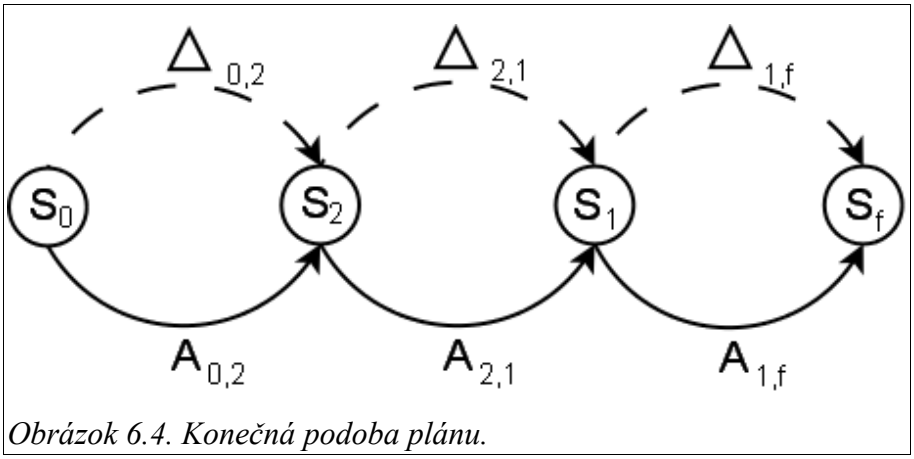
Ďalej sa snažíme nájsť v zozname RK akcií takú, ktorej RK zmeny je najviac aktívne pre zmenu $\Delta_{0,f}$. Ak sú kritériá akcií správne natrénované, môžeme predpokladať, že vybraná akcia bude najvhodnejším kandidátom na redukciu tejto diferencie. Na obrázku 6.2 ale vidíme, že najlepšej nájdenej akcii zodpovedala diferencia $\Delta_{1,f}$, ktorá nás síce dostane do stavu S_f, ale začínať musíme v stave S_1 (ktorý zatiaľ nepoznáme) a nie v našom aktuálnom stave S_0. Toto RK akcie si teda vložíme na koniec pripravovaného plánu, plánovací proces však ešte zďaleka neskončil.

Pomocou predpokladanej zmeny $\Delta_{1,f}$ a požadovaného stavu S_f vypočítame situáciu S_1, ktorú môžeme chápať ako podcieľ v plánovacom procese. Najskôr si potrebujeme overiť, či situácia S_1 vyhovuje predpokladom nájdenej akcie $A_{1,f}$. Ak sú predpoklady splnené, hľadáme akciu, ktorá nás dovedie z aktuálnej situácie (stále S_0) do situácie S_1. Je to rekurzívne volanie plánovača, sme v stave ako na obrázku 6.1, len namiesto S_f dáme S_1. Ak predpoklady akcie $A_{1,f}$ nie sú splnené, snažíme sa nájsť akciu, ktorá nás dostane do takej situácie, v ktorej budú predpoklady platiť (t.j. akcia bude aplikovateľná).



Obrázok 6.3. Hľadanie diferencie na splnenie podcieľa a akcie na jej redukciu.

Na obrázku 6.3 je znázornený ten istý problém ako v predchádzajúcom prípade. Chceme redukovať diferenciu $\Delta_{0,1}$, akcia $A_{2,1}$ s najaktívnejším RK zmeny je ale aplikovateľná v stave S_2 a nie aktuálnom S_0. Opäť rekurzívne voláme funkciu plánovača, aby sme sa dostali do S_2.



Obrázok 6.4. Konečná podoba plánu.

Nájdeme teda znova rozdiel medzi aktuálnou situáciou S_0 a požadovanou S_2. Tentokrát sme už našli akciu, ktorá je aplikovateľná v súčasnej situácii a redukuje danú diferenciu (obrázok 6.4).

Vytvorený plán vložíme do štruktúry plánovača a asociujeme ho so zámerom, na ktorého splnenie je určený. Naraz môže byť v plánovači uložených viacero plánov, každý asociovaný s iným zámerom. Keďže priority zámerov sú dynamické, môže sa stať, že naplánujeme sekvenciu akcií na splnenie istého zámeru, počas vykonávania

plánu sa ale objaví zámer s vyššou prioritou a preto mu musíme dať prednosť – plánovač je teda schopný robiť revízie. Starý zámer aj s plánom na jeho realizáciu ale ostáva v plánovači a v budúcnosti sa k nemu môžeme vrátiť. Už sme spomínali, že zámery sú analogické procesom v operačných systémoch, uviedli sme príklady bežiacich (v predošlom príklade nový plán) a pozastavených (starý plán) zámerov.

V algoritme môžeme veľakrát rekurzívne volať metódu, ktorá má nájsť akciu na prechod zo súčasného stavu do požadovaného (či už je to samotný zámer alebo podcieľ). Ako pri každej rekurzii, aj tu hrozí nebezpečenstvo zacyklenia. Vyvarujeme sa toho definovaním maximálnej povolenej hĺbky rekurzívneho volania. V prípade, že je táto hodnota prekročená, doposiaľ nájdený plán prehlásime za neplatný a vykonáme náhodnú akciu. Takýto prístup sa nám zdá byť vhodný pre náš model. Agent nemá dostatočné deduktívne schopnosti, aby dokázal určiť, že daný cieľ nie je možné splniť, preto ciele nazamieta. Ak by ale namiesto vykonania náhodnej akcie nevykonal nič, v ďalšom kroku by sa opäť snažil nájsť plán na splnenie toho istého cieľa. Plánovací mechanizmus je ale deterministický, takže plán by zasa nenašiel a už by nikdy nevykonal žiadnu akciu.

Ako je v popise plánovacieho algoritmu vidno, pri konštrukcii plánu postupujeme spätne od požadovanej sitácie ku aktuálnej. Plánovanie je teda orientované na cieľ. Správanie agenta však musí byť do istej miery aj reaktívne, to okrem iného znamená, že musí počas vykonávania plánu reagovať aj na aktuálne vstupy. Tejto téme sa venujeme v nasledujúcej kapitole.

6.2 Vykonávanie plánu

Pri vykonávaní plánu hrá dôležitú úlohu záväzok (*commitment*) k vybranému plánu. Agent nehľadá zbytočne nový plán, pokiaľ pri vykonávaní akcií plánu postupuje predpokladaným smerom. Niekedy je ale nútený záväzok nedodržať a vykonať revízie. Existujú dva hlavné dôvody na odmietnutie plánu. Externým dôvodom je meniace sa prostredie. Agent vytvoril plán v nejakom stave, tento plán ale už nemusí byť vhodný v modifikovanom prostredí. Interným dôvodom je, že agent disponuje iba predpokladmi o dôsledkoch akcií, v skutočnosti ale tieto presvedčenia nemusia vždy platiť.

Pri rozhodovaní o dodržaní alebo odmietnutí plánu sa agent riadi dvoma faktormi.

Jednak sú to predpoklady, ktoré sú v RK akcie asociované s akciou. Pred vykonaním akcie si agent overí, či sú predpoklady v aktuálnej situácii splnené. Ak nie je súčasný stav dostatočne podobný RK objektu (predpokladom), plán je odmietnutý a agent plánuje odznova zo situácie, do ktorej sa dostal vykonávaním predchádzajúceho plánu. Týmto spôsobom agent plán reviduje a opätovne sa snaží naplniť svoj zámer.

V druhej fáze sa porovná predpokladaná a skutočná zmena po vykonaní predchádzajúcej akcie. Ak sú si zmeny dosť podobné, môžeme sa domnievať, že zmeny boli naozaj spôsobené agentom. V každom prípade ale vieme, že plán postupuje správne a môžeme teda náš záväzok dodržať. Naopak, ak nastala iná zmena, než sme očakávali, plán zamietneme a revízia pokračuje ako v predošlom prípade.

Keď sú uvedené podmienky splnené, môžeme vykonať ďalšiu akciu z plánu. Konkrétna akcia na vykonanie je prototypovým príkladom RK popisu akcie[17].

6.3 Dosiahnutie zámeru

Agent si na začiatku fázy sense, hneď po vnímaní scény, overuje, či dosiahol aktuálny zámer. Zistí to tak, že aplikuje RK objektu predstavujúce zámer na objekt, na ktorý sa zámer vzťahuje. Ak je zámer splnený, overí si, či má s daným zámerom asociovaný nejaký plán (agent nemusí mať žiadny aktuálny plán, zámer mohol byť dosiahnutý náhodne vybranou akciou). Ak plán nájde a ten bol vykonaný korektne až do konca, vloží si agent tento plán do zoznamu RK plánov, aby ho mohol aj neskôr využiť pri plánovaní. Aktuálny zámer vyhodí zo zoznamu (presnejšie prioritnej fronty) zámerov a usmerní svoju pozornosť na dosiahnutie ďalšieho zámeru. Ak už žiadny ďalší zámer nemá, vyberie si želanie s najvyššou prioritou a vloží ho medzi zámery. Ak nemá ani želanie, znamená to, že nemusí v ďalšom kroku vykonávať nič plánovito (môže ale vykonať náhodnú akciu).

6.4 Zhrnutie

Popísali sme, ako prebieha plánovanie, čo agent robí pri vykonávaní jednotlivých krokov plánu a ako si overuje dosiahnutie cieľa. Plánovanie a vykonávanie plánu

17 Prototyp opäť získame volaním metódy `meanCase()` rozlišovacieho kritéria.

umiestnime do fázy *select* životného cyklu agenta, čím nahradíme náhodný výber akcií. Splnenie cieľových podmienok sa overuje na začiatku fázy *sense*.

Na plánovanie agent využíva RK akcií a plánov. Pri vykonávaní plánu sa agent musí presvedčiť, či sú splnené predpoklady pre ďalšiu akciu a či sa pomocou doposiaľ vykonaných akcií uberá správnym smerom. Po dosiahnutí zámeru upriami svoju pozornosť na ďalší zámer v poradí.

7 Simulačné experimenty

7.1 Experiment 1 – pohyb po dvojrozmernej mriežke

V tomto experimente sa agent pohybuje po dvojrozmernej mriežke. Okrem agenta sa v prostredí nevyskytujú žiadne iné objekty ani agenty. Cieľom agenta je dostať sa na pozície, ktoré mu buď zadá užívateľ, alebo si ich sám vygeneruje.

7.1.1 Popis

Prostredie má tvar 2D mriežky s rozmermi 40x40 bodov. Na začiatku experimentu je agent umiestnený v strede plochy, na pozícii [20,20]. Agent je reprezentovaný rámcom s atribútmi pre X-ovú a Y-ovú súradnicu. Na začiatku má agentov rámec tvar `{posX: 20; posY: 20; }`.

Agent má k dispozícii jednu akciu, *move*, pomocou ktorej dokáže meniť svoju pozíciu. Atribúty akcie označujú zmenu X-ovej a Y-ovej súradnice. Náhodná akcia je napr. `{actionType: move; posX: -7; posY: 8; }`. Najskôr sa zadefinuje typ akcie, za ním nasledujú parametre. Parametre akcie sú z rozsahu <-9;10>. O samotné vykonanie akcie sa stará prostredie. Ak chce agent vykonať akciu, ktorá by ho posunula mimo mriežku, prostredie mu v tom zabráni a umiestni ho na okraj mriežky.

Cieľ (želanie) agenta je určený RK objektu reprezentujúcim pozíciu, na ktorú sa agent má dostať. V našom experimente má agent na začiatku štyri želania, ktoré mu

zadá užívateľ. Má navštíviť všetky štyri rohy plochy. Množinu želaní môžeme symbolicky zapísať nasledovne:

$$\{[0,0],[39,0],[39,39],[0,39]\}$$

Priority jednotlivých želaní sú nastavené tak, aby ich agent dosahoval v uvedenom poradí. Priority sú v tomto prípade statické, nemenia sa počas experimentu. Ak agent splní tieto štyri želania, sám si generuje pozície, na ktoré sa chce dostať.

Priebeh experimentu

V súlade s paradigmou *starting small* agent spočiatku vôbec neplánuje, vykonáva iba náhodné akcie. Konkrétne, celý experiment pozostáva z 3000 iterácií, fáza bez plánovania trvá prvých 1000 iterácií. V prvej fáze si agent iba vytvára RK akcií pomocou asociačného algoritmu (asociuje náhodné akcie s pozorovanými predpokladmi a zmenami). Samozrejme, aj počas druhej fázy, s plánovaním, si agent naďalej prenastavuje RK akcií, aby sa čo najviac so svojimi presvedčeniami priblížil realite.

7.1.2 Výsledky

V našom experimente agent v každom kroku vykonal práve jednu akciu, dokopy teda má 3000 pozorovaní akcií a ich predpokladov a dôsledkov. Zaujíma nás, ako tieto pozorovania využil pri vytvorení vnútornej reprezentácie kauzality. Ďalej je taktiež zaujímavé, koľkokrát sa agentovi podarilo splniť jeho želanie, a či dosiahnutiu cieľa predchádzala plánovaná alebo len náhodná akcia.

Najskôr rozoberieme beh experimentu z pohľadu výsledného počtu jednotlivých druhov RK. Experiment sme nechali zbehnúť dvakrát – raz sme nastavili zlučovanie blízkych kategórií (*merging*), druhý krát bez zlučovania (*no merging*)[18]. V tabuľke vidíme, že ak sme použili zlučovanie, znížil sa počet kategórií približne o 40% (nerátajúc kritériá plánov).

18 Samozrejme, tu by bolo lepšie nechať experiment pri každom nastavení zbehnúť viackrát, výsledné počty kritérií spriemerniť a priemerné hodnoty porovnať. Nás ale viac zaujíma kvalitatívne vyhodnotenie experimentu, chceme rozanalyzovať konkrétne vytvorené rozlišovacie kritériá.

	Bez zlučovania	So zlučovaním
RK akcií	1557	918
Elementárne kritériá	520	297
RK popisov akcií	190	117
RK plánov	14	6

Tabuľka 7.1.1. Porovnanie počtu vytvorených RK bez zlučovania a so zlučovaním kategórií.

Keď sa však pozrieme, koľko cieľov sa agentovi podarilo dosiahnuť, zistíme, že v prípade bez zlučovania kategórií to bolo 21 splnených zámerov, v prípade so zlučovaním len 11. Preto sa pri podrobnej analýze kategórií sústredíme na prípad bez zlučovania – v našom konkrétnom experimente dopadol lepšie.

Z 21 splnených zámerov iba trom predchádzala náhodná akcia, to znamená, že zvyšných 18 bolo dosiahnutých plánovito. Agent si dokázal vytvoriť 14 RK plánov. Ako sme uviedli v kapitole o plánovači, agent si ukladá len úspešné plány, čiže 14 krát postupoval presne podľa plánu (t.j. vykonal všetky akcie plánu a dosiahol zamýšľaný cieľ). Zvyšné štyri razy mal síce vytvorený plán, ktorý pozostával zo zreťazenia viacerých akcií, nemusel ho však vykonávať až do konca, pretože už počas jeho vykonávania dosiahol cieľ.

Zamerajme teraz našu pozornosť na rozbor konkrétnych RK. Výhodou symbolového prístupu[19] je, že môžeme nahliadnuť dovnútra reprezentácie a podrobne ju rozanalyzovať. Sústreďme sa najprv na kritériá plánov – pri tých sme si istí, že nás doviedli do cieľa, takže by mali najvernejšie odrážať realitu (narozdiel od iných presvedčení, ktoré nemusia vždy platiť). Nebudeme uvádzať všetky vytvorené RK plánov, popíšeme iba jedno z nich:

```
Precondition:
DCVarObj: {posX:7,0#0,0#1.0; posY:6,0#0,0#1.0}
Actions:
ActionDesc:
DCVarObjActDesc: {posX:6,0#2,2#243.0; posY:6,5#1,7#243.0},
actionType:(move)
Precondition:
DCVarObj: {posX:21,7#11,4#1355.0; posY:25,8#10,1#1355.0}
```

19 Ako sme už spomínali, náš prístup nie je čisto symbolový, významy sú však rozhodne reprezentované na symbolovej báze.

```
Change:
DCVarChange: {posX:6,3#0,4#66.0; posY:5,5#1,5#66.0}
Goal:
DCVarObj: {posX:13,0#0,0#1.0; posY:11,0#0,0#1.0}
```

Každý plán sa skladá z troch častí – predpokladu, akcie a cieľa. Na zaznamenanie hodnôt atribútov využívame reprezentáciu založenú na varianciách – pre každý atribút sa zaznamená jeho stredná hodnota, rozptyl a početnosť. Napríklad `posX:6,0#2,2#243.0` znamená, že stredná hodnota X-ovej pozície je 6,0, rozptyl je 2,2 a hodnota bola vypočítaná z 243 príkladov.

Predpoklady plánu sú `{posX:7,0#0,0#1.0; posY:6,0#0,0#1.0}`. Keďže rozptyl oboch hodnôt je 0 a početnosť 1, predpoklad bol vytvorený z jedného príkladu – agent bol pred vykonávaním plánu zjavne na pozícii [7,6] .

Cieľ je `{posX:13,0#0,0#1.0; posY:11,0#0,0#1.0}`, agent sa teda chcel dostať na pozíciu [13,11] .

Postupnosť akcií v tomto prípade pozostáva iba z jednej akcie. Pred akciou by sa mal agent nachádzať na pozícii spĺňajúcej predpoklad `{posX:21,7#11,4#1355.0; posY:25,8#10,1#1355.0}`. Ako vidíme, tento predpoklad nie je ťažké splniť, pretože rozptyl hodnôt je dosť veľký.

RK popisu akcie definuje akciu typu *move* s parametrami `{posX:6,0#2,2#243.0; posY:6,5#1,7#243.0}`, zmena vyvolaná akciou je `{posX:6,3#0,4#66.0; posY:5,5#1,5#66.0}`. Vidíme, že hodnoty parametrov akcie a hodnoty zmien jednotlivých súradníc si navzájom zodpovedajú. Presne toto sme chceli – agent sa naučil, aké hodnoty parametrov má dať akcii aby dosiahol požadovanú zmenu. Na okrajoch mriežky môže byť správanie odlišné (prostredie reguluje zmeny, aby sa agent nedostal mimo plochy), predpoklady akcie však určujú, že sa agent nachádza niekde v strede mriežky.

Stredné hodnoty atribútov akcie v konečnom dôsledku určujú, s akými parametrami sa akcia zavolá. V našom príklade by to bola akcia *move(6,0;6,5)*. Aplikovaním tejto akcie na pôvodnú pozíciu agenta [7,6] dostaneme po zaokrúhlení cieľovú pozíciu [13,11].

Pozrime sa ešte bližšie na vybrané elementárne RK objektov. Najzaujímavejšie sú tie, ktoré majú najväčšiu početnosť, t.j. boli najviac krát aktívne. Uvedieme len tie, ktorých početnosť presahuje 100:

```
{posX:32,1#7,1#255.0; posY:18,4#7,7#255.0}
{posX:21,7#11,4#1355.0; posY:25,8#10,1#1355.0}
{posX:19,8#9,3#270.0; posY:8,4#7,1#270.0}
```

Vidíme, že všetky hodnoty majú dosť veľké rozptyly a stredné hodnoty sú niekde v strede plochy, nie na okrajoch. Správanie v strede je štandardné, čiže na predpokladoch nezáleží, pri okrajoch je výsledný efekt viac závislý od predpokladov.

Pri zmenách musel agent viac rozlišovať medzi jednotlivými prípadmi, preto nevznikli až také početné kategórie. RK zmien sú totiž smerodajné pri plánovaní akcií. Uvediem RK zmien, ktorých početnosť presahuje 30:

```
{posX:-0,2#4,0#2112.0; posY:-0,7#3,9#2112.0}
{posX:6,3#0,4#66.0; posY:5,5#1,5#66.0}
{posX:-7,9#0,8#52.0; posY:-8,0#0,8#52.0}
{posX:4,4#1,1#32.0; posY:-8,6#0,5#32.0}
{posX:9,5#0,5#38.0; posY:-0,6#0,9#38.0}
{posX:4,2#0,8#31.0; posY:9,0#0,9#31.0}
{posX:-8,6#0,5#31.0; posY:3,4#0,5#31.0}
{posX:0,0#0,0#31.0; posY:9,0#0,8#31.0}
```

Prvé RK má veľmi vysokú početnosť, ale definuje len malý pohyb ľubovolným smerom. Naopak, ostatné kritériá majú oveľa nižšie početnosti, zato však presne špecifikujú smer a veľkosť pohybu. Pri plánovaní sa dajú efektívne využiť tie RK akcií, ktoré obsahujú RK zmeny presne definujúce zmenu scény.

Pri RK plánov sme už uviedli aj RK akcie. Väčšina akcií sa týka štandardného vykonania akcie, t.j. agent sa nachádza ďalej od okraja plochy. Niektoré akcie ale odrážajú špecifické vlastnosti prostredia na okraji plochy, napríklad:

```
ActionDesc:
DCVarObjActDesc: {posX:9,2#0,7#33.0; posY:-8,3#0,7#33.0},
actionType:(move)
Precondition:
DCVarObj: {posX:19,1#0,5#10.0; posY:0,0#0,0#10.0}
Change:
DCVarChange: {posX:9,5#0,5#38.0; posY:-0,6#0,9#38.0}
```

Predpoklady určujú, že agent sa nachádza niekde okolo pozície [19,0]. Chce vykonať akciu *move(9,2;-8,3)*. Táto akcia by ho ale dostala mimo plochy (v smere osi Y-ovej), preto simulátor prostredia zasiahne a agent ostane na mriežke. Agent si ale tento zásah prostredia všimne a asociuje si s akciou a predpokladom správnu zmenu so strednými hodnotami [9,5;-0,6] (za normálnych okolností by zmena mala stredné hodnoty približne [9,2;-8,3], reflektujúc tak parametre akcie).

7.1.3 Zhrnutie

Podrobne sme si rozobrali experiment, v ktorom sa agent pohybuje po 2D mriežke a navštevuje želané pozície. Ukázali sme si, ako vyzerajú vybrané RK plánov, akcií, popisov akcií, objektov a zmien. Nahliadli sme, že agent si dokázal vytvoriť plány, ktoré ho viedli až k splneniu aktuálneho zámeru.

7.2 Experiment 2 – presun objektov

V druhom experimente bolo úlohou agenta rozmiestniť objekty na vopred stanovené pozície na 2D mriežke. Rozdiel oproti predchádzajúcemu experimentu je, že okrem agenta sa na mriežke nachádzajú aj iné objekty, čo môže spôsobiť isté ťažkosti pri učení dôsledkov akcií.

7.2.1 Popis

Agent sa pohybuje v rovnakom prostredí ako v predchádzajúcom prípade, na 2D mriežke s rozmermi 40x40 bodov. Agent je opäť popísaný rámcom, ktorého hodnota je na začiatok `{posX: 20; posY: 20; }`. Na ploche sú aj ďalšie tri objekty, dva trojuholníky a jeden štvoruholník:

```
{color: 0; posX: 20; posY: 20; vertices: 3; }
{color: 2; posX: 0; posY: 0; vertices: 3; }
{color: 2; posX: 5; posY: 34; vertices: 4; }
```

Ako vidíme, objekty sa líšia farbou, začiatočnou pozíciou a počtom vrcholov. Samozrejme, každý objekt (vrátane agenta) má svoj jednoznačný identifikátor. Cieľom agenta je umiestniť prvý trojuholník do pravého dolného rohu (t.j. na pozíciu blízku bodu [39,0]), druhý trojuholník do pravého horného rohu (bod [39,39]) a štvoruholník do ľavéh horného rohu (bod [0,39]). Nakoniec sa má sám dostať do ľavého dolného rohu (bod [0,0]). Objekty po umiestnení na správne miesto z prostredia zmiznú. Tento model pripomína jednoduchú hru, v ktorej má hráč rozmiestniť objekty na hracej ploche a nakoniec sa sám dostať na vyznačené miesto.

Agent má tentokrát k dispozícii mierne modifikovanú akciu, *moveObject*, s parametrami definujúcimi zmenu X-ovej a Y-ovej súradnice. Modifikácia spočíva v tom, že pri vykonaní musí agent prostrediu oznámiť, na akom objekte (s akým ID) má

byť akcia vykonaná. Predtým to nebolo potrebné, pretože bol na ploche sám a každá akcia sa automaticky vzťahovala na neho. Seba taktiež presúva pomocou akcie *moveObject*, keď ju zavolá so svojim ID.

Priebeh experimentu

Tentokrát sme nechali simuláciu bežať 2000 iterácií, pričom opäť prvých 1000 iterácií agent neplánoval akcie. V druhej polovici sa agent snažil splniť svoje ciele. Zaujímalo nás, či to aj v zmenených podmienkach dokáže.

7.2.2 Výsledky

Experiment sme nechali bežať viackrát, pričom sme menili buď vlastnosti objektov na scéne (farba, počet vrcholov, počiatočná pozícia) alebo agentove ciele (t.j. kam má daný objekt umiestniť). Väčšinou sa agentovi podarilo splniť ciele, uvádzame výsledky takéhoto úspešného pokusu. V diskusii sa snažíme prísť na dôvody (a možné riešenia) neúspechov.

Nebudeme už podrobne analyzovať všetky typy kritérií, zameriame sa len na RK plánov. Agentovi sa podarilo každý cieľ dosiahnuť pomocou vopred pripraveného plánu. Uvediem plán, ktorý agent využil na splnenie prvého zámeru, teda umiestnenie jedného trojuhlníka do pravého dolného rohu.

```
Precondition:
DCVarObj: {color:0,0#0,0#1.0; posX:31,0#0,0#1.0;
posY:0,0#0,0#1.0; vertices:3,0#0,0#1.0}
Actions:
ActionDesc:
DCVarObjActDesc: {posX:7,7#1,7#1242.0;
posY:0,9#5,3#1242.0}, actionType:(moveObject)
Precondition:
DCVarObj: {posX:23,1#12,2#4291.0; posY:22,6#12,6#4291.0}
Change:
DCVarChange: {posX:8,0#1,3#254.0; posY:-0,7#3,2#254.0}
Goal:
DCVarObj: {posX:39,0#0,0#1.0; posY:0,0#0,0#1.0}
```

Predpoklady plánu sa zjavne týkajú trojuholníka, je to prvý trojuholník, lebo atribút farba (*color*) má hodnotu 0. Trojuholník sa pred vykonaním plánu nachádzal na pozícii [31,0].

Použitá akcia má stredné hodnoty parametrov 7,7 a 0,9, asociovaná zmena má

podobné atribúty 8,0 a -0,7. Agent teda túto akciu dobre asocioval s pozorovanou zmenou. Nepríjemné na tomto kritériu popisu akcie je, že agent túto akciu príliš často používal, jej početnosť je 1242, teda viac ako polovica všetkých akcií spadala do tejto kategórie. Prílišná všeobecnosť vzniknutých kategórií zrejme nepridá na sile nášho modelu (týmto problémom sa bližšie zaoberáme v diskusii).

Na tomto príklade je zaujímavé aj to, že predpoklady akcií obsahujú len atribúty pre X-ovú a Y-ovú pozíciu, ostatné hodnoty (*color*, *vertices*) ignorujú. Agent teda správne kategorizoval, ktoré atribúty sú pre akciu *moveObject* podstatné.

7.2.3 Zhrnutie

V tomto experimente sme agenta vložili na plochu spolu s ďalšími objektami. Agent dokázal cielene umiestňovať objekty aj seba samého. Žiaľ, pri niektorých nastaveniach vlastností a cieľov agenta sa mu cieľ nepodarilo dosiahnuť.

7.3 Diskusia

Vykonané simulačné experimenty nám umožnili prakticky overiť navrhovanú reprezentáciu. Experimenty sme spúšťali už súbežne s teoretickým návrhom modelu. Vtedy sme zistili, že asociovanie vykonania akcie s viacerými objektami so sebou prináša nečakané komplikácie, na vyriešenie ktorých zatiaľ v modeli nie sú dosť silné prostriedky. V modeli hlavne nie je implementovaný algoritmus učenia komplexných RK situácií, ktoré by sa dali využiť ako predpoklady v RK akcií. Preto sme zvolili zjednodušenie a asociovali sme akcie len s jedným objektom.

V tomto prípade už experimenty dopadli lepšie. Podarilo sa nám vytvoriť plánovač operujúci s RK akcií a plánov, čo sme ukázali v príklade s pohybom agenta po mriežke alebo presunom objektov. Agent sa (väčšinou) dokázal vysporiadať aj s viacerými objektami na scéne. Niektoré objekty obsahovali aj atribúty, ktoré agent dokázal pri kategorizácii vhodne ignorovať. V plánovacom procese agent značne využíval revízie, čo bolo zapríčinené jeho nepresnými znalosťami o kauzalite. Neúplnosť a nepresnosť agentovej reprezentácie prostredia sme predpokladali, zahrnutie revízií do plánovacieho mechanizmu sa aj experimentálne ukázalo byť viac ako opodstatnené.

Vráťme sa teraz k problému z druhého experimentu, keď sa agentovi niekedy

nepodarilo dosiahnuť cieľ. V prípade, že agent svoje ciele nesplnil, nepodarilo sa mu správne vytvoriť vnútornú reprezentáciu, t.j. rozlišovacie kritériá. Obyčajne to bolo spôsobené tým, že si s danou akciou asocioval zlú zmenu. Vytvoril si totiž napríklad RK zmeny, ktoré zachytávalo stálosť pozície objektu a často ho preferoval pred pohybujúcimi sa objektami. Táto prílišná preferencia jedného kritéria je zrejme spôsobená zle nastavenými prahovými hodnotami použitými jednak v asociačnom algoritme, ako aj v plánovači. Tieto hodnoty sú buď statické, alebo sa dynamicky menia vopred stanoveným spôsobom. Hoci sa dajú empiricky nastaviť pre konkrétny experiment, vhodnejšie by bolo nájsť všeobecný postup na ich modifikáciu. Keďže prahové hodnoty agenta a funkcie na ich modifikáciu považujeme za analógiu genetickej výbavy u človeka, dali by sa na nastavenie optimálnych hodnôt použiť evolučné princípy. Vo viacgeneračnom modeli by si agent teda sám vytváral vnútornú reprezentáciu (bez genetického prenosu), vnútorné nastavenia učiacich algoritmov by sa však evolučne prenášali z rodičov na potomkov (samozrejme s miernymi modifikáciami – napr. kríženie, mutácia). Posledná myšlienka je zatiaľ len hypotetickej povahy, nemáme ju ešte experimentálne overenú.

8 Záver

Diplomová práca nadväzuje na model kognitívnej sémantiky rozlišovacích kritérií, ktorá slúži na vnútornú reprezentáciu konceptov agenta pohybujúceho sa v dynamických a otvorených prostrediach. Cieľom vlastnej práce bolo obohatiť model o plánovací a kauzálny modul agenta, a zároveň vykonať a vyhodnotiť simulačné experimenty.

Základným stavebným prvkom kauzálneho modulu agenta sú rozlišovacie kritériá akcií. Slúžia na asociáciu akcie s jej predpokladmi a dôsledkami. V implementácii sme sa venovali akciám ovplyvňujúcim práve jeden objekt na scéne, navrhli sme však aj rozšírenie modelu chápania kauzality na ľubovolný počet objektov. Načrtli sme tiež prepojenie kauzálneho a jazykového modulu.

Pri tvorbe plánovača sme využili tzv. BDI architektúru agenta. Agent disponuje presvedčeniami (*beliefs*), želaniami (*desires*) a zámermi (*intentions*), ktoré využíva v plánovacom mechanizme. Agent si dokáže vytvoriť plány na dosiahnutie aktuálnych zámerov. Plán tvorí zreťazenie individuálnych akcií. Agent je schopný plán počas jeho vykonávania monitorovať a v prípade, že sa nevyvíja správnym smerom, dokáže ho refaktorovať. Práve v schopnosti robenia revízií spočíva sila plánovacieho modulu, pretože predpokladáme, že sa agent bude pohybovať v dynamickom prostredí. Jeho znalosť prostredia nebude vždy dokonalá a korektná, takže vytvorená reprezentácia akcií nemusí presne odzrkadlovať realitu.

Vykonali sme simulačné experimenty, ktoré mali overiť vhodnosť našej implementácie. Zistili sme, že agent je schopný využiť vzniknuté sémantické štruktúry v plánovacom procese. Nahliadli sme dovnútra agentovej reprezentácie prostredia a podrobne rozanalyzovali jednotlivé typy kategórií.

V modeli ostáva viacero otvorených problémov. Treba doriešiť učenie rozlišovacích kritérií akcií – v našom zjednodušenom prípade agent pokladal všetky pozorované zmeny v prostredí za dôsledky svojej vlastnej činnosti. Taktiež by bolo vhodné asociovať vykonávanie akcií s viacerými objektami na scéne.

Príloha A – Implementácia

Teoreticky navrhnutý model implementujem v programovacom jazyku JAVA. Pri vývoji používam integrované vývojové prostredie Eclipse. Staviam na existujúcom projekte *Jarmila* od môjho školiteľa Martina Takáča. Projekt sa skladá z niekoľkých balíčkov, popíšem logickú štruktúru hierarchie balíčkov a bližšie sa zameriam na triedy, ktoré som sám doplnil.

Balíček `Jarmila.agents` obsahuje rôzne typy agentov – základný typ `Agent` a jeho potomkov `HumanTeacher` a `InventiveAgent`. Doplnil som triedu `PlanningAgent`, ktorý je priamym potomkom `InventiveAgent`. Tento predstavuje BDI rozšírenie architektúry agenta a dokáže plánovať.

Balíček `Jarmila.criteria` obsahuje všetky typy rozlišovacích kritérií, ako aj triedy, v ktorých sú tieto kritériá uložené. Obsahuje aj podbalíčky s rôznymi typmi RK – kvalitatívne (vracajú 0 alebo 1), založené na varianciách (používané v tejto práci) resp. kovarianciách (sofistikovanejšia verzia kritérií), a tzv. rule based (preddefinované kritériá, ktoré sa nemenia).

V balíčku `Jarmila.lexicon` je uložená trieda implementujúca slovník agenta – asociácie slov a významov.

Doplnil som balíček `Jarmila.planner` zahŕňajúci triedy, ktoré využíva plánovač. Je to samotný plánovač (`Planner`) a triedy reprezentujúce želania (`Desire`) a zámery (`Intention`). Popíšem ich neskôr.

Balíček `Jarmila.simulation` slúži na simuláciu prostredia. Balíček `Jarmila.structures` obsahuje pomocné triedy a rozhrania používané v ostatných triedach projektu.

Teraz upriamim pozornosť na triedy, ktoré som do projektu doplnil. Vytvoril som dva nové typy rozlišovacích kritérií – RK akcií a plánov. RK akcií (`DCAction`) spájajú popis akcie (`DCActionDescription`), predpoklady (`DCObject`) a dôsledky akcie

(`DCChange`). Trieda obsahuje aj metódu `similarityWith(Frame actionDesc, Item object, Frame change)` na zistenie podobnosti medzi konkrétnou akciou a zapamätaným kritériom.

Všetky akcie sú uložené v systéme akcií (`DCActionSystem`). Vieme vybrať akciu, ktorá najviac zodpovedá pozorovanej zmene alebo všetkým aspektom akcie (popis akcie, predpoklad, dôsledok).

RK plánu (`DCPlan`) je spájaný zoznam RK akcií, obsahuje aj predpoklad (`DCObject`) a cieľ (tiež `DCObject`) plánu. Keďže RK plánu slúži nielen na reprezentáciu všeobecného kritéria, ale aj na popis konkrétneho plánu, máme tu aj aktuálny predpoklad a cieľ (oba inštancie triedy `Frame`). V prípade, že bol plán správne použitý (dosiahli sme cieľ), aktualizujú sa RK zodpovedajúce predpokladu a cieľu na základe konkrétnych rámcov pomocou metódy `update()`. Sú tu aj metódy na pridanie akcie (používa sa pri plánovaní) a výber ďalšej akcie (pri vykonávaní plánu). Podobne, ako máme systém RK akcií, máme aj systém RK plánov (`DCPlanSystem`).

Vytvoril som nový typ agenta – `PlannningAgent`. Okrem systému elementárnych kritérií, ktoré obsahoval už aj základný typ `Agent`, tento obsahuje aj systém akcií, plánov a popisov akcií. `PlanningAgent` si pamätá ostatnú vykonanú akciu (používa pri učení RK akcií). V prípade, že plánuje, pamätá si tiež aktuálny zámer (`Intention`) a plán (`DCPlan`) na jeho dosiahnutie. Plán som kvôli zjednodušeniu priamo implementoval kritériom plánu, `DCPlan` teda zastrešuje všeobecné RK, ako aj konkrétnu inštanciu plánu. A samozrejme má každý agent svoj plánovač (`Planner`).

Životný cyklus agenta je riadený metódou `live()`, ktorá v sebe postupne volá metódy `sense()`, `select()` a `act()`, predstavujúce rovnomenné fázy životného cyklu. Tieto fázy sú teoreticky popísané v kapitole o kauzalite.

Plánovací proces je riadený metódou `plan()`. Tu najskôr hľadáme pomocou plánovača aktuálny plán asociovaný s aktuálnym zámerom. Na vytvorenie nového plánu slúži metóda `planToDCObject(DCObject dcObj, String objId)`. Parameter `dcObj` získame zo zámeru (pripomínam, že zámer je reprezentovaný RK objektu) a `objId` je identifikátor objektu, na ktorý sa zámer vzťahuje. V prípade, že sme našli (resp. vytvorili) plán, monitorujeme ho, a ak nie je treba vykonať žiadne

revízie, plán vykonáme pomocou metódy `executePlan()` – čo je vlastne len zavolanie ďalšej akcie plánu.

Plánovač (`Planner`) je uložený v balíčku `Jarmila.planner`. Obsahuje zoznamy želaní a zámerov – oba implementované pomocou prioritnej fronty (trieda `BinaryHeapMax` z balíčka `Jarmila.structures`). Zoznam aktuálne vybraných plánov je v `TreeMap`, asociujú sa tu zámery s plánmi. Ako už bolo spomenuté v teoretickom návrhu architektúry, agent môže mať naraz vytvorených viac plánov, realizuje sa ten, ktorý je asociovaný s aktuálnym zámerom. Aktuálny zámer sa nachádza na prvom mieste v prioritnej fronte. Ďalší zámer získame buď zo zoznamu zámerov, alebo zo zoznamu želaní (ak už ďalší zámer nemáme) pomocou verejnej metódy `nextIntention()`.

Želanie je implementované triedou `Desire`. Obsahuje inštanciu triedy `DCObject` reprezentujúcu konkrétne želanie a reťazec špecifikujúci ID objektu, na ktorý sa želanie vzťahuje. S každým želaním je spojená priorita, ktorá slúži na usporiadanie želaní v prioritnej fronte. Trieda `Desire` implementuje natívne JAVA rozhranie `Comparable`, inštancie sa porovnávajú práve podľa priority.

Zámer (`Intention`) je potomkom triedy `Desire`. Obsahuje navyše jednoznačný identifikátor, ktorý slúži v plánovači na asociovanie zámeru s plánom (s plánom sa teda neasociuje celý zámer, ale len jeho ID). Okrem toho je tu metóda na zistenie, či bol zámer splnený.

Popísal som základnú štruktúru projektu *Jarmila*, ktorý som využil v simulačných experimentoch na overenie funkčnosti navrhovaného modelu.

Príloha B – Obsah CD

Priložené CD obsahuje samotnú diplomovú prácu v elektronickej podobe, ako aj zdrojové texty projektu *Jarmila*, ktorý som naprogramoval a následne využil v simulačných experimentoch. Na CD je tiež podrobné logovanie zo simulačných experimentov.

V adresári ***doc*** sa nachádzajú súbory s diplomovou prácou vo formátoch *.odt* (textový dokument Open Office) a *.pdf* (Portable Document Format).

V adresári ***src*** užívateľ nájde zdrojové texty implementovaného projektu.

V adresári ***log*** sa nachádzajú podadresáre ***experiment1*** a ***experiment2*** s logovacími súbormi z jednotlivých experimentov.

Referencie

[1] Šefránek, J.: *Kognícia bez mentálnych procesov.* In: Rybár, J., Beňušková, Ľ., Kvasnička, V. (eds.): Kognitívne vedy. Kalligram, Bratislava 2002

[2] Takáč, M.: *Kognitívna sémantika rozlišovacích kritérií*. In: Kelemen, J., Kvasnička, V. (eds.): Kognice a umělý život VI. Slezská univerzita, Opava 2006

[3] Takáč, M.: *Modelovanie kultúrneho prenosu a jeho úloha v evolúcii jazyka*. In: Rybár, J., Kvasnička, V., Farkaš, I. (eds.): Jazyk a kognícia. Kalligram, Bratislava 2005

[4] Harnad, S.: *The Symbol Grounding Problem*. Physica D 42, 1990

[5] Hopcroft, J. E., Ullman, J.D.: *Formálne jazyky a automaty*. Alfa, Bratislava 1978

[6] Parisi, D., Cangelosi, A., Falcetta, I.: *Verbs, nouns, and simulated language games*. Journal of Italian Linguistics.

[7] Kvasnička, V. et al.: *Úvod do teórie neurónových sietí*. IRIS, Bratislava 1997

[8] Gänderfors, P.: *Conceptual Spaces*. MIT Press, Cambridge 2000

[9] Russell, S., Norvig, P.: *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, New Jersey 1995

[10] Rao, A. S., Georgeff, M. P.: *BDI Agents: From Theory to Practice*. In: Proceedings of the First International Conference on Multi-Agent Systems. San Francisco, USA 1995

[11] Kelemen, J. et al.: *Základy umelej inteligencie*. Alfa, Bratislava 1992

[12] Takáč, M.: *Categorization by Sensory-Motor Interaction in Artificial Agents*. In: Fum, D., Del Missier, F., Stocco, A. (eds.): Proceedings of the 7th International Conference on Cognitive Modeling. Edizioni Goliardiche, Trieste, Italy 2006

[13] Piaget, J., Inhelder, B.: *La psychologie de l'enfant*. PUF, Paris 1966

[14] Dastani, M., van Riemsdijk, M. B., Meyer, J.-J. Ch.: *Goal Types in Agent Programming*. In: Proceedings of the 17th European Conference on Artificial Intelligence. IOS Press, 2006

[15] Georgeff, M. P., Ingrand, F. F.: *Decision-Making in an Embedded Reasoning*

System. In: Proceedings of the Eleventh International Joint Conference on Artificial Intelligence. Detroit, MI 1989

[16] Elman, J. L.: *Learning and Development in Neural Networks: The Importance of Starting Small*. In: Cognition 48, 1993