Sekvencie v prírode

Andrej Lúčny

MicroStep-MIS, Čavojského 1, 841 04 Bratislava 4
Katedra aplikovanej informatiky, FMFI UK, Mlynská Dolina, 842 48 Bratislava 4
E-mail: andy@microstep-mis.com

Abstrakt. Príspevok sa zaoberá modelovaním sekvencií v správaní živých organizmov. Východiskom je Minského predstava o tzv. skriptoch. Tá má ešte značné sklony k používaniu klasických algoritmov (procedúra, vývojový diagram), ku ktorým však v prírode nenachádzame biologické koreláty. Sekvenčné správanie sa preto snažíme modelovať ako dôsledok paralelného pôsobenia samostatne aktívnych modulov, tzv. agentov. Porovnávame rozdiely voči klasickým modelom a hľadáme také pozorovania v prírode, ktoré svedčia o relevantnosti takéhoto modelovania.

1 Úvod

Pokiaľ modelujeme správanie nejakého živočícha v počítači, je vhodné používať pritom softwarové komponenty, o ktorých si vieme predstaviť, že im prislúchajú určité biologické koreláty. Hoci je generovanie určitého správania v prírode i počítači založené na určitej interpretácií inštrukcií, povaha prostriedkov tejto interpretácie môže byť veľmi rozdielna. Na jednej strane príroda prevyšuje počítač čo do paralelnosti interpretačného procesu, počítač zase prírodu pravdepodobne prevyšuje v jeho univerzálnosti. Preto je veľmi zaujímavou otázka, na aké komponenty sa je vhodné pri modelovaní prírody v počítači obmedziť, aby sme postihli charakter modelovaného.

Pokiaľ by sme sa v ničom neobmedzili, hrozí nám, že vytvoríme model, ktorý dosahuje zhodu s pozorovaním tým, že modeluje samotné pozorovanie a nie pozorovaného tvora. Takýto model potom nedokáže povedať nič viac, než už vieme zo samotného pozorovania. Uvažujme napríklad modelovanie pohybu šesťnohého hmyzu. Môžeme urobiť pozorovanie a odmerať postupnosť uhlov, ktoré zvierajú jednotlivé kĺby, v jednotlivých časových okamihoch. Klasickým algoritmom sa generovanie takejto sekvencie veľmi ľahko implementuje. Pohyb modelu bude potom veľmi dobre zodpovedať pozorovaniu, ale netreba dodávať, že neprezrádza nič o povahe generovania pohybu.

Aké obmedzenia na používané komponenty by sme teda mali prijať? Najčistejšie riešenie tu predstavujú samozrejme štruktúry typu neurónových sietí, ktoré sú priamo odpozorované z biologického systému. Pritom dobre ukazujú ako príroda dokáže z ničoho urobiť niečo. Avšak ich nevýhodou je, že ich úspech je limitovaný a aj pokiaľ sa dostaví, neumožňujú preniknúť do logickej povahy procesu generovania správania. Odkryť podstatu tohto procesu vedia lepšie behaviorálne architektúry, pri ktorých výsledné správanie modelu je dôsledkom koordinácie medzi paralelnými modulmi symbolicko-výpočtovej povahy produkujúcimi určité elementárne správania.

Obzvlášť vhodné sa nám vidí použiť také moduly, ktoré disponujú nielen vlastným vláknom aktivity, ale v jeho v rámci neustále vykonávajú relatívne jednoduchú voľbu produkovaných akcií na základne vnímaného stavu ich prostredia a ich vnútorného stavu. Takéto moduly budeme nazývať reaktívnymi agentmi. Z hľadiska koordinácie týchto agentov, preferujeme tzv. nepriamu komunikáciu, teda,

že komunikujú podobne ako mravce pri chemotaxii [4]. Takto z modelu eliminujeme klasické algoritmy interpretujúce sekvenciu inštrukcií. Biologicky bude takýto model prijateľnejší. Avšak kým reaktívne správanie typu podnet-odozva sa v ňom bude dať veľmi dobre implementovať, správanie spočívajúce vo vykonávaní určitej sekvencie sa naopak bude implementovať ťažko.

Vezmeme si preto konkrétne správanie spočívajúce v generovaní určitej sekvencie. Vyjdeme z klasického modelu, za ktorý môžeme považovať tzv. skript zo známeho Minského societného modelu mysle [9]. Ukážeme ktoré stránky správania nedokáže zachytiť a potom budeme prezentovať ako sa analogickou úlohou popasuje náš prístup s obmedzeným arzenálom stavebných komponentov.

2 Príklad sekvenčného správania

Ideálnym kandidátom na sekvenciu hodnú modelovania je zakladanie potomstva kutavkou [2] [5]. Kutavka je totiž jednak pomerne jednoduchý hmyzí živočích, jednak toto konkrétne správanie je jedno z najzložitejších individuálnych správaní v ríši hmyzu vôbec. Ďalším povzbudzujúcim faktorom je, že toto správanie je evidentne vrodené (žiadna kutavka nevidí svoju matku zakladať potomstvo).

Kutavka je samotárska osa známa najmä pre hrôzostrašný spôsob akým zabezpečuje pre svoje larvy dostatok čerstvej potravy: žihadlom pichne svoju korisť (každý druh loví špecifickú obeť z radov chrobákov, napríklad sedlovku) tak, že ju ochrnie a odtiahne ju do podzemnej komôrky, kde na ňu nakladie vajíčko. Z toho sa vyliahne kutavčia larva, ktorá potom paralyzovanú obeť za živa zje, pričom jej to trvá niekoľko mesiacov. Kutavka pritom realizuje vždy rovnaký rituál[1]: najprv vyhrabe komôrku, potom uloví obeť, položí si ju pred komôrku, vlezie do komôrky skontrolovať ju, potom do nej vtiahne obeť, znesie na ňu vajíčko, vylezie z komôrky a zacelí jej vchod. K tomuto správaniu sa viaže niekoľko zaujímavých okolností.

Pokiaľ by ste kutavke vo fáze keď vlezie na kontrolu do komôrky sedlovku odtiahli, ale nie príliš ďaleko – aby ju kutavka našla, opäť by ju priniesla ku komôrke. Ale opäť by ju položila pred vchod a šla by vykonať kontrolu (ktorú už práve pred chvíľou vykonala), čiže vliezla by do komôrky. Takto má experimentátor možnosť opakovať odtiahnutie sedlovky od vchodu a udržiavať kutavku v bezvýchodiskovom cykle [2]. Kutavka 30-40 krát zakaždým sedlovku pritiahne ku vchodu komôrky, ale nepoučí sa a nevtiahne ju hneď dnu, naopak, ide vykonávať kontrolu, ktorú už vykonala mnoho krát a s pričinením experimentátora opäť o sedlovku príde. E. Gál položil v [1] provokujúcu otázku: čo chýba kutavke v porovnaní so správaním človeka, ktorý by sa takto nedal vodiť za nos?

Nejaké jednoduché vysvetlenie tu neobstojí, lebo na rozdiel od kontroly komorky, pri opätovnom nájdení odtiahnutej sedlovky, ju kutavka nejde druhý raz pichať žihadlom. Určité fázy vie teda zopakovať, iné preskočiť.

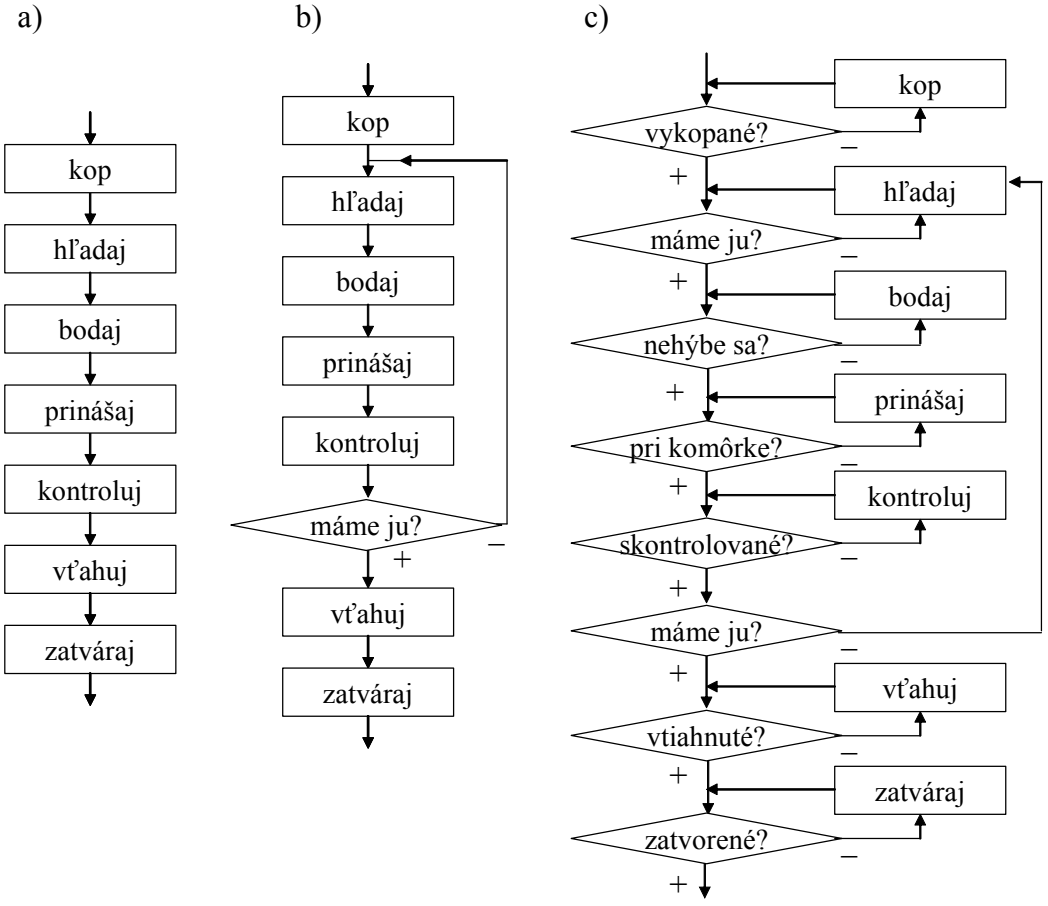
Navyše kutavka si evidentne dokáže zapamätať, že už vykopala komôrku a dokonca kde to bolo. Jej schopnosti učiť sa orientovať pri lete podľa význačných pozemných objektov, boli jednoznačne preukázané. S predstavou, že kutavka je z hľadiska zložitosti svojho riadiaceho mechanizmu len jednooký medzi slepými (teda relatívne jednoduchý tvor vykonávajúci pomerne zložité správanie) sa teraz pokúsime túto sekvenciu generovať umelým riadiacim systémom.

[1] Tento rituál sa odlišuje v rámci jednotlivých druhov kutaviek, tu uvádzame podobu z [2]

3 Minského skript

Pokiaľ by sme chceli opísané sekvečné správanie zaznamenať algoritmickým spôsobom, zrejme by sme na úvod vytvorili niečo podobné obrázku č. 1a. Pokiaľ by sme túto štruktúru použili v našom modeli kutavky, postupovali by sme podobne ako M. Minsky v [9], ktorý takéto štruktúry nazýva skriptami. Pokiaľ som dobre pochopil jeho ideu – tento autor ponecháva pomerne široké pole fantázii čitateľa – predstavuje si realizáciu skriptu ako dátami riadeného agenta, teda ako agenta, ktorý interpretuje určité dátové štruktúry (pričom tie sa môžu v čase vyvíjať).

Pokiaľ by sme to naozaj spravili takto, dostali by sme sa k už uvedenému problému zámeny modelovania pozorovania a pozorovaného. Skript založený na interpretácii algoritmu z obrázku č. 1a totiž jednej strane vykazuje výbornú zhodu s pozorovaním správania za štandardných okolností, avšak na strane druhej zlyhá pri ľubovoľnom porušení týchto podmienok. Napríklad keď vo fáze kontroly posunieme sedlovku kúsok od komôrky, vykonávanie skriptu sa načisto poruší, miesto toho aby sa prešlo na fázu hľadania. Náš model by mal tendenciu pri takom posunutí vygenerovať výnimku a nechať zakladanie potomstva tak. Vidíme, že záznam, ktorý uspokojí každého biológa, pre počítač hovorí príliš málo. Náš prístup sa môžeme samozrejme pokúsiť zachrániť a to obohatením nášho algoritmu o ošetrenie výnimiek. Dostaneme tak riešenie znázornené na obrázku č. 1b.

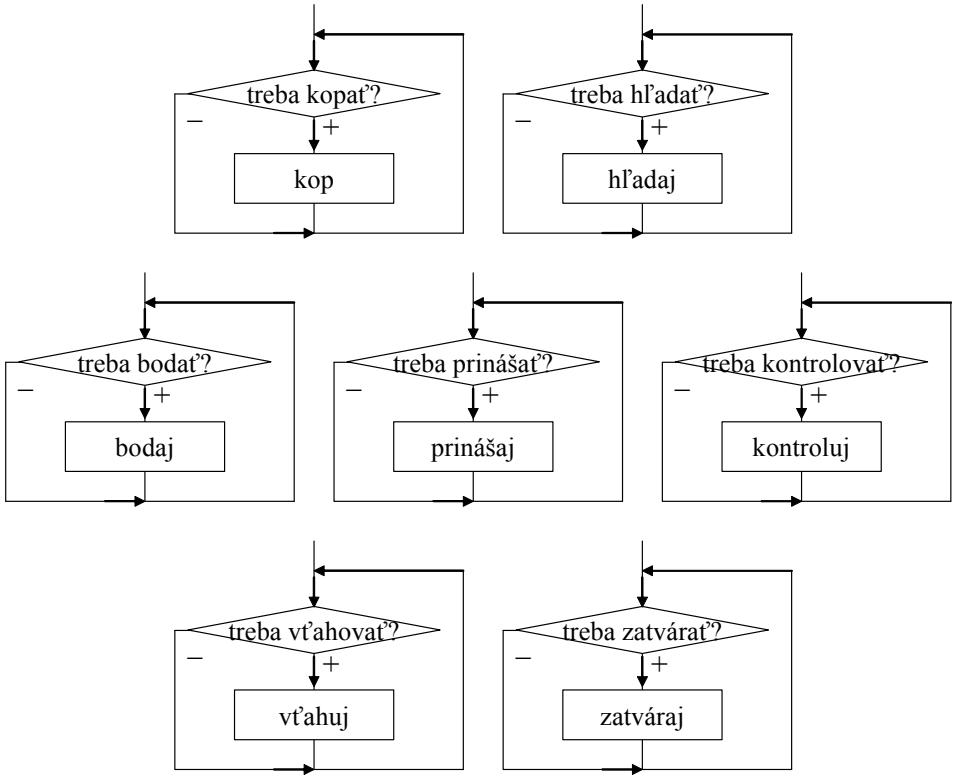


Obrázok 1. a) jednoduchý skript b) skript s návratom c) skript s návratom i preskočením

Teraz sa pri odtiahnutí sedlovky, model kutavky poslušne vráti k fáze hľadania. Keďže sme sedlovku odtiahli len kúsok, je vysoká pravdepodobnosť, že bude nájdená opäť tá istá sedlovka. Ale beda! Náš model sa opäť pustí do bodania, a to nezodpovedá pozorovaniu. Do nášho skriptu musíme vložiť aj schopnosť preskočiť určité fázy. Túto schopnosť je najlepšie oprieť o samotnú štruktúru vykonávania jednotlivých fáz. Riešenie je znázornené na obrázku 1c.

Takýto model - zachytávajúci návraty k určitým fázam aj preskakovanie určitých fáz – nám už umožňuje aj rozumne odpovedať na otázku postulovanú E. Gálom. Všimnime si, že každá podmienka rozhodujúca v ukončení, respektíve preskočení určitej fázy sa operia o operačné údaje – okrem jedinej výnimky a tou je kontrola komôrky. Kým napríklad po nájdení bodnutej sedlovky, je ľahké pozorovať, že je bodnutá, po vylezení z komôrky je nemožné pozorovať, že je skontrolovaná – to by si kutavka musela pamätať. Zrejme si teda pomáha tým spôsobom, že podmienka „skontrolované ?“ je implementovaná ako „trčím hlavou von z komôrky ?“.

Takéto vysvetlenie samozrejme má aj tienisté stránky – v prvom rade je to už spomínaná schopnosť kutavky zapamätať si veľa zaujímavých vecí. Na druhej strane však kutavka za normálnych okolností nenarazí na potrebu implementovať danú podmienku šikovanejšie. Zrejme by nebol principiálny problém kutavku v tomto ohľade vylepšiť, ale v prírode to proste nie je treba. Preferencia opierať sa o informácie z prostredia a nie o vlastnú pamäť môže mať taktiež súvis s tým, že toto správanie je vrodené.



Obrázok 2. Realizácia skriptu pomocou paralelných modulov, ktoré sú autonómne a pracujú na reaktívnom princípe.

Ďalšou nevýhodou tohto modelu je, že nedokáže vysvetliť to magické číslo 30-40, ktoré udáva počet opakovaní, po ktorých kutavka kontrolu vynechá a sedlovku vtiahne do komôrky bez nej. Toto číslo je príliš vysoké, aby sme tento počin mohli považovať za šikovné správanie, ktoré je dôsledkom schopností zapojiť do generovania správania anticipáciu. Avšak náš model nezopakuje proces 30-40 krát ale nekonečne veľa krát.

Hlavným nedostatkom je však potreba explicitného ošetrenia výnimiek, teda určenia do ktorej fázy sa treba vrátiť, pokiaľ sa vo vykonávaní nedá pokračovať. V tomto ohľade stále modelujeme svoje pozorovanie a nie pozorovaný objekt: pri zavádzaní jednotlivých výnimiek z modelu nikdy nedostaneme viac informácie než do neho vložíme.

4 Reaktívny model

Je možné vytvoriť model, v ktorom by ošetrovanie výnimiek implicitne vyplynulo z vlastností modulov realizujúcich jednotlivé fázy sekvencie ? Odpoveď je kladná. V prvom rade musíme dostať do nášho modelu paralelizmus, t.j. rozbiť jediný interpreter inštrukcií na autonómne moduly. Každý takýto modul bude disponovať vlastným vláknom aktivity. Povaha modulu môže byť v princípe rôzna, my sa však obmedzíme na špecifický druh modulov, ktoré sa vyznačujú dvomi vlastnosťami:

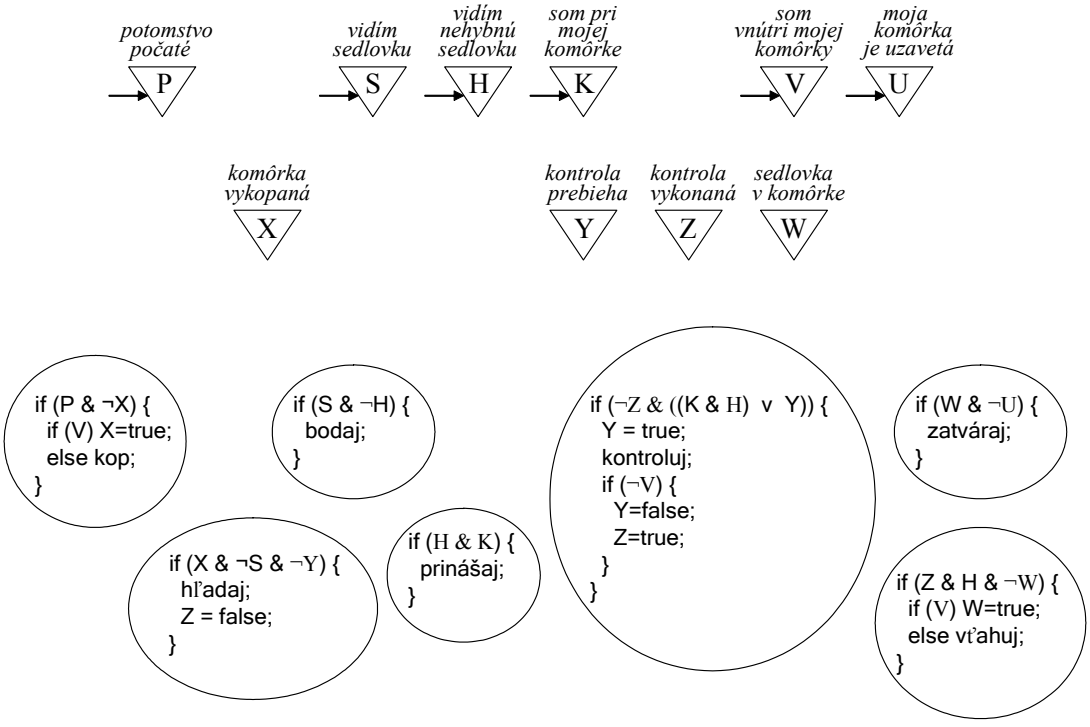
- ich správanie je generované na základe cyklickej voľby jednotlivých akcií (v informatickej terminológii hovoríme, že správanie je implementované tzv. reentrantným spôsobom.
- voľba akcií je púhou reakciou na externé informácie, ktoré môže modul získať a jeho vnútorný stav

Takéto moduly budeme nazývať reaktívnymi agentami. Zvolenie práve tejto štruktúry je pravda čistou špekuláciou, ale z formálneho hľadiska možno k tomu dodať len to, že buď budeme modelovať na báze štruktúry, z ktorej pri zavedení výsledku jednotlivého pozorovania nič nového implicitne nevyplýva alebo budeme hádať štrukturálne vlastnosti, ktoré nejaké implicitné následky vyvolávajú.

Základný tvar modelu pre zakladanie potomstva kutavkou môžeme vidieť na obrázku 2. Tento obrázok však vystihuje len „agentový“ charakter modelu, z aplikačného pohľadu je kľúčové ako budeme implementovať podmienky typu „treba X ?“. Tak napríklad „treba kopať ?“ musí byť pravdivé pokiaľ prebieha proces vykopania komôrky – v tomto smere sa zhoduje s negáciou podmienky „vykopané ?“ z obrázku 1c. Avšak okrem toho musíme zabezpečiť, aby sa modul neaktivizoval v inej chvíli, než tesne po počatí (čo je iniciačný podnet modelovanej sekvencie). To možno zabezpečiť napríklad tak, že kutavku vyzbrojíme schopnosťou zapamätať si informácie typu „mám vykopanú komôrku“. Podmienku „treba kopať ?“ potom implementujeme ako „počala som a ešte nemám vykopanú komôrku“. Dobre si však musíme rozmyslieť, ktoré z potrebných informácii sa budú získavať z prostredia a ktoré si budeme vnútorne počítať a pamätať. Pokiaľ budeme preferovať získavanie informácii z prostredia, dostaneme model, podobný tomu na obrázku č. 3. Pritom ešte výraznejšie než pri obyčajnom skripte narazíme na to, že implementovať kontrolu komôrky je náročnejšie než ktorúkoľvek inú fázu. Vyplýva to z faktu, že táto fáza nemá vplyv na prostredie a navyše ju štartujú podmienky, ktoré po jej odštartovaní prestanú platiť (pre kontrolou musíme mať pri sebe ulovenú sedlovku, ale opúšťame ju a bez nej lezieme do komôrky). Preto sa dá očakávať, že práve pri tejto fáze sa vyskytnú problémy.

Pokrokom tohto modelu je, že v sebe automaticky zahŕňa ošetrenie výnimiek návratom k predošlým fázam. Napríklad, keď počas prinášania, kutavke sedlovka vypadne, model automaticky prejde do fázy hľadania. Ak nájde inú – ešte neulovenú – sedlovku, dobodá ju, ak nájde tú istú, fázu bodania preskočí. V kritickom prechode od kontroly k vtiahnutiu sa na základe straty sedlovky vráti do fázy hľadania. Tejto fáze treba naopak brániť, aby nebola aktívna už počas kontroly, preto je hľadanie založené na podmienke X & ¬S & ¬Y a nie X & ¬S.

V rámci tohto modelu by sme teda na otázku, prečo kutavka tak zanovito kontrolu opakuje, odpovedali, že preto, lebo štartovacia podmienka vykonania kontroly, nie je počas jej vykonávania splnená. Tým sa táto fáza líši od akejkoľvek inej fázy. Vyžaduje pamätať si, že sme sa na kontrolu podujali (údaj Y) a pokračovať v nej výlučne na základe tohto rozhodnutia. Tým pádom ale v momente ukončenie kontroly musíme toto rozhodnutie zabudnúť, aby sme s ňou vôbec prestali. Potiaľto je všetko v poriadku. Ale, na to aby sme v tomto modeli dokázali na základe ukončenia kontroly aktivovať vtiahnutie, musíme mať ďalší pamäťový údaj Z, ktorý indikuje, že kontrola už bola vykonaná a zároveň zabraňuje, aby kutavka po skončení jednej kontroly, nezačala ihneď kontrolu novú. Problematické však je, že by pri opakovanom priebehu spôsobil preskočenie kontroly – čo odporuje pozorovaniu. Preto je nevyhnuté ho explicitne zabudnúť. V našom riešení ho zabúdame vo fáze hľadania, čo je hnus, ktorý nemožno biologicky zdôvodniť.



Obrázok 3. Reaktívny model generujúci analogické správanie ako skript. Hore sú informácie získavané rozpoznávaním prostredia, pod nimi informácie, ktoré je nevyhnutné si pamätať. Dolu agenty cyklicky vykonávajúce uvedenú voľbu akcii realizujúce jednotlivé fázy skriptu reentrantným spôsobom.

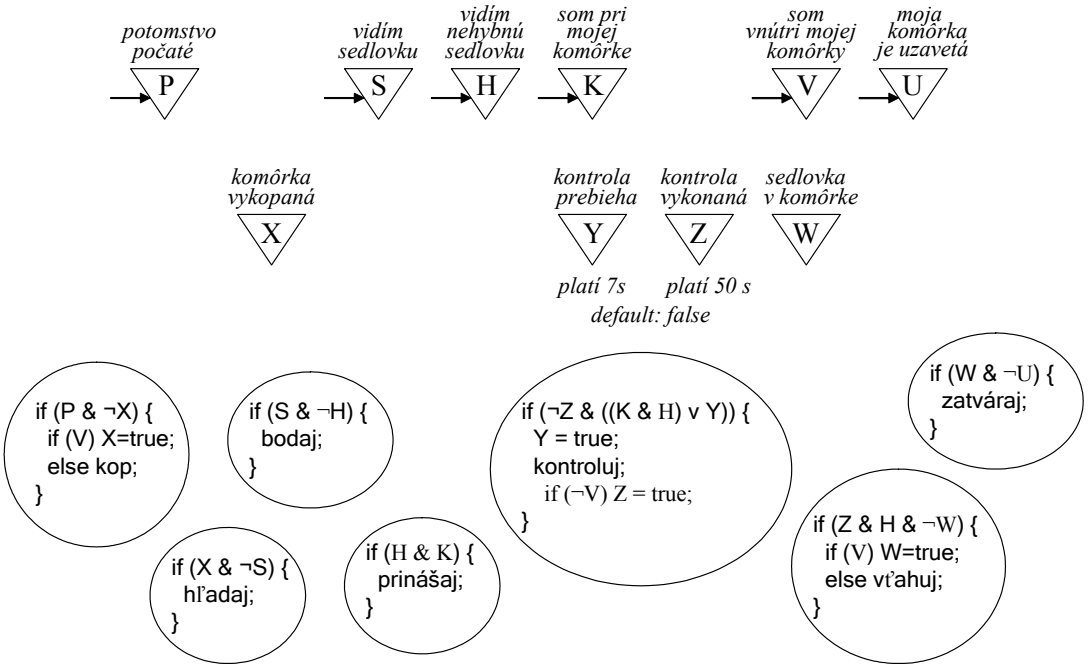
Nepekné na tom je, že sa schopnosť opakovania napriek implicitným mechanizmom musí aj tak podporiť explicitne. Avšak má to logiku aspoň v tom, že keby kutavke počas

kontroly sedlovku niekto nadobro uchmatol, musela by ísť loviť novú a pritom by sa dostatočne vzdialila od komôrky na to, aby ju bolo potrebné opäť kontrolovať. Zbytočnou kontrolou v nastrojených podmienkach teda platí za efektívne správanie v prirodzených podmienkach, čo iste rada zaplatí.

Tomuto modelu však každopádne niečo chýba.

5 Reaktívny model s časovou platnosťou

Predchádzajúcemu modelu však toho nechýba veľa. Stačí si uvedomiť, že na rozdiel od počítača sú živé tvory – z technických príčin – vybavené viacerými druhmi pamäti. Aj čitateľovi sa už možno stalo, že šiel do pivnice pre lekvár a keď tam prišiel nevedel si vybaviť čo vlastne chcel (autorovi sa to stáva častejšie, než by si želal). Pokiaľ budú informácie slúžiace na generovanie podmienok aktivácie agentov iba v krátkodobej pamäti, netreba explicitne implementovať ich zabúdanie. Isté informácie samozrejme zabudnúť neslobodno – napríklad, že sme vykopali komôrku, a takisto kde sme si ju vykopali. Iné si nemusíme pamätať vôbec, lebo sa dajú priamo vyčítať z prostredia – napríklad, že sedlovka je už ochrnutá - alebo sú permanentne prítomné kvôli stavu kutavky – napríklad, že počala. Ale pre informácie typu, že sme skontrolovali komôrku je najvhodnejšie definovať obmedzenú časovú platnosť, potrebnú len na to, aby prebehla príslušná aktivácia – aby jeden modul vyzval druhý na prevzatie riadenia.



Obrázok 4. Upravený reaktívny model využívajúci obmedzenú časovú platnosť údajov.

Model, ktorý zabúdanie nahradzuje definovaním časovej platnosti je na obrázku č. 4. Vďaka možnosti definovať obmedzenú časovú platnosť, sa dá model značne zjednodušiť. Hlavne z neho môžeme eliminovať vzťah medzi hľadaním a kontrolou a nechať hľadanie pôsobiť pri kontrole paralelne, nakoľko si tieto dve činnosti neodporujú. Predtým sme to nemohli urobiť jedine kvôli potrebe zabúdať pri hľadaní,

že kontrola prebehla. Teraz to však pri hľadaní zabudneme vďaka tomu, že hľadanie trvá zvyčajne dlhší čas, než je časová platnosť vykonania kontroly (údaj Z).

Pozoruhodné je, že keď som prvý raz vytvoril tento model (v [8]), zavrhol som ho ako odporujúci pozorovaniu. Tento model je síce elegantnejší, ale je v ňom implicitne skrytá možnosť, že kutavka preskočí fázu kontroly (napríklad keď fáza hľadania trvá príliš krátko), prípadne ak definujeme nejakú platnosť aj pre údaj o vykopaní komôrky, tak dokonca môže po veľkom počte opakovaní odletieť vykopať inú komôrku. Navyše, keď zvážime biologické možnosti implementácie časovej platnosti – časté zapisovanie takého údaju by zrejme viedlo k posilneniu schopnosti zapamätať si ho dlhšie a tým pádom by po väčšom počte opakovaní kutavka kontrolu vynechala – vychádza, že by sa model správal inteligentnejšie, než je obvykle uvádzané pri opisoch príslušných pozorovaní (napr. v [1]). Model bol však natoľko elegantnejší, že som pátral po pôvodnejších informáciách a naozaj v [2] sa uvádza, že po 30-40 opakovaniach kutavka kontrolu vynechá.

A tak tento model dostal zelenú a ukázalo sa, že je celkom slušne podnetný. Opakovanie kontroly pri odtiahnutí kutavky vysvetľuje tým, že kým kutavka sedlovku opäť nájde, zabudne, že komôrku skontrolovala. Pokiaľ by však sedlovku vrátila ku komôrke dostatočne rýchlo, môže si ešte kontrolu pamätať a v takom prípade kontrolu vynechá. Podobný efekt môže spôsobiť veľký počet opakovaní, lebo hoci kutavka bude sedlovku vracať ku komôrke kvôli únave čoraz pomalšie, jej pamäťová „stopa“ bude naopak čoraz vytrénovanejšia a zapamätá si dlhšie, že kontrola bola vykonaná.

Okrem toho vieme na základe tohto modelu navrhnúť niekoľko experimentov a predpovedať ich výsledok:

- čo sa stane, keď pri kontrole ochrnutú sedlovku odstránime a dáme miesto nej inú nebodnutú, ale zato napríklad uviazanú na nitku aby neušla ? Z modelu vychádza, že by ju mala bodnúť a vtiahnuť.
- aký vplyv bude mať vzdialenosť odtiahnutia sedlovky na počet opakovaní ? Tu nám vychádza, že ak bude vzdialenosť dostatočne malá, žiadne opakovanie sa nedostaví. Inak vo všeobecnosti, čím menšia vzdialenosť tým menej opakovaní.
- dá sa zväčšiť počet opakovaní z tých 30-40 ? Podľa modelu by sa to malo podariť tak, že budeme stále vhodne zväčšovať vzdialenosť o ktorú sedlovku odťahujeme.
- čo sa stane keď komôrku zakopeme ? Tu by sa zo značnej časovej platnosti údaju o vykopaní komôrky dalo súdiť, že kutavka sa dosť dlho bude správať načisto bezradne – bude hľadať neexistujúcu komôrku, potom si však vykope komôrku novú.

Keby mi nejaký etológ vedel zistiť – či už z existujúcich prác alebo vykonaním experimentu, nakoľko sú tieto predpovede úspešné, bol by som veľmi vďačný.

6 Záver

V tomto príspevku sme sa snažili čitateľa presvedčiť, že pri modelovaní živých systémov v počítači by sme mali používať princípy paralelizmu a reentrantnosti paralelných modulov a venovať pozornosť časovej platnosti údajov slúžiacich na ich koordináciu. Čitateľa, ktorý by mal záujem bližšie sa oboznámiť s detailami konkrétnej softwarovej architektúry, ktorá to umožňuje, odkazujem na [7] [8].

Treba ďalej spomenúť, že som v tomto príspevku obišiel určité detaily. V prvom rade som uvažoval len najvrchnejšiu úroveň riadenia a nezaoberal som sa tým, ako sú implementované voľby akcií „kop“, „hľadaj“, ... Tu by bolo treba zvážiť jednak

realizáciu hierarchie, keď je zložitejšia voľba implementovaná na základe „rekurzie“ niekoľkými agentami nižšej úrovne, jednak realizáciu subsumcie, keď sa určitý agent svojou činnosťou vtiera do činnosti iných agentov. Ako príklad hierarchie môžeme uviesť prinášanie: musíme sedlovku uchopiť, preniesť a pustiť. Ako príklad subsumpcie môžeme uvažovať samotné prinesenie, ktoré sa bude opierať o bežný „bezcieľny“ pohyb, ale bude ho máličko ovplyvňovať, aby sa kutavka hýbala skôr v smere, v ktorom sa nachádza vykopaná komôrka. Keby sme sa zaoberali týmito záležitosťami, ukázalo by sa, že zvolené vlastnosti reaktívneho modelu, nie sú čistou špekuláciou, ale predstavujú práve architektúru, ktorá hierarchiu a subsumpciu realizuje z istého hľadiska ideálnym spôsobom. Tieto vlastnosti teda vychádzajú z istých všeobecných princípov.

Ako vedľajší efekt sme sa snažili ukázať, že niektoré zaujímavé vlastnosti získavajú živé systémy stupídnejším spôsobom, než by sme si sami želali. Doslova ich majú vďaka svojej nedokonalosti. Jednoznačným dôsledkom tohto faktu je, že pokiaľ chceme zostrojiť stroje s podobnými vlastnosťami, musíme túto nedokonalosť použiť ako konštrukčný princíp. Toto zároveň poukazuje na rozdiely medzi strojmi a živými tvormi. Nepodobajú sa práve preto, že pri konštrukcii našich strojov sa snažíme o ich dokonalosť.

Literatúra

[1] Gál, E.: *Intuitívna psychológia a kognitívne vedy.* In: Kognitívne vedy III (Kvasnička V., Pospíchal J. eds.), CHTF, STU, Bratislava, 2000

[2] Hass, H.: *The Human Animal: The Mystery of Man's Behavior.* G. P. Putnam's Sons., 1970, pp. 27

[3] Kelemen, J.: *Strojovia a agenty.* Archa, Bratislava, 1994

[4] Lúčny, A.: *Reaktívny model inteligentného systému.* In: Kognitívne vedy II (Kvasnička, V. – Pospíchal, J., eds.), CHTF STU Bratislava, 1999, 75-84

[5] Lúčny, A.: *Hľadanie kvalitatívneho rozdielu.* In: Kognice a umělý život (Kelemen, J. – Kvasnička, V. – Pospíchal, J., eds.), FPF SU, Smolenice, 2001, 167-176

[6] Lúčny, A.: *Baldwinov efekt: Plyn a brzda evolúcie.* In: Kognice a umělý život III (Kelemen, J. ed.), FPF SU, Opava, 2003, 151-164

[7] Lúčny, A.: *Building Complex Systems with Agent-Space Architecture.* Computing and Informatics, Vol. 23 (2004), 1001-1036

[8] Lúčny, A.: *Tvorba inteligentných systémov na báze architektúry Agent-Space.* Rigorózna práca, Katedra aplikovanej informatiky, Matematicko-fyzikálna fakulta, Univerzita Komenského, Bratislava, 2005

[9] Minsky, M.: *The Society of Mind.* Simon&Schuster, New York, 1986

[10] Minsky, M.: *Konštrukcia mysle.* Archa, Bratislava, 1996

[11] Minsky, M.: *Emotion machine.* In Proc. EMCSR'2000 (Trappl, R. ed.), Vienna, 2000

[12] Minsky, M.: *Emotion machine.* http://web.media.mit.edu/~minsky/, 2004