Enti – Simulátor přirozeného prostředí lidského světa

Ondřej Bojar (obo@cuni.cz), Cyril Brom
Milan Hladík, Mikuláš Vejlupek, Vojtěch Toman, David Voňka

Abstrakt

Projekt Enti implementuje simulátor prostředí podobného přirozenému lidskému světu. V simulovaném prostředí se pohybují samostatné bytosti, některé jsou řízeny počítačem, v roli jiných může do simulace vstoupit lidský uživatel. Bytosti spolu mohou komunikovat v jednoduché variantě přirozeného jazyka (konkrétně češtiny). Chování bytostí řízených počítačem je dáno skripty v nově navrženém programovacím jazyce E. Výsledné prostředí je vhodnou „laboratoří umělé inteligence“, kde si uživatel může testovat vlastní algoritmy přirozeného lidského chování.

Klíčová slova: simulace lidského světa, umělý život, umělá inteligence, komunikace přirozenou řečí, prostředí pro ověřování algoritmů umělé inteligence

1 Cíle projektu Enti

Cílem projektu Enti bylo vytvořit simulátor prostředí podobného přirozenému lidskému světu. Návrh projektu se přitom soustředil především na následující rysy světa lidí:

- lidé jsou bytosti s nezávislým vědomím,
- lidé přijímají podněty z okolního světa,
- lidé mají (elementární) schopnost okolní svět ovlivňovat,
- lidé se ve světě nepohybují náhodně, ale s určitým cílem (přinejmenším s cílem existovat co nejdelší dobu),
- lidé spolu komunikují přirozenou řečí.

Bytosti v simulovaném prostředí nazývejme *enty*[1]. V roli enta může do simulovaného prostředí vstoupit i lidský uživatel.

Simulátor prostředí i entů řízených počítačem byl implementován pro platformu typu Unix. Finální verze simulátoru má podobu „laboratoře pro umělou inteligenci“: je zprovozněn model světa podobného jednomu rodinnému domku, jsou implementováni dva strojoví enti (zahradník pečuje o rostliny a hudebník má na starosti zábavu). K dispozici je rovněž grafický prohlížeč simulovaného prostředí. Implementace dalších strojových bytostí je ponechána uživateli systému Enti, inspirovat se přitom může jednak jednak hotovými enty, a jednak detailním popisem v *Příručce autora světa a skriptů*. Pro ladění chodu strojových entů je připraven též debugger s grafickým rozhraním.

V tomto textu je možné bohužel pouze podrobněji nastínit jednotlivé rysy projektu. Zájemce o podrobnější popis simulace světa proto odkazujeme na čtyřdílnou dokumentaci projektu.

1.1 Velká francouzská revoluce

Samostatné poznámky si zaslouží významný požadavek v projektu Enti: požadavek maximální rovnosti mezi člověkem a enty, příznačně označovaným jako ideál *Velké francouzské revoluce (VFR)*.

V návrhu projektu se ideál VFR podařilo dodržet, i když boj za tento ideál zasahoval i do velmi jemných možností specifikace světa entů. Výsledku však ideál přinesl hned dvě výhody:

[1] Jméno *ent* je odvozeno od pojmů samostatná *entita* či autonomní *agent*.

Platforma pro Turingův test. Ve světě se mohou pobyhovat bytosti řízené člověkem i počítačem, pro ostatní bytosti jsou podle vnějších vlastností i bezprostřední možnosti ovlivňovat svět naprosto nerozlišitelné. Odlišit je lze pouze na základě důvtipnosti v chování. (Dlužno poznamenat, že naši enti poměrně záhy v Turingově testu selhávají a jejich strojovost je prohlédnuta.)

Složitost lidského světa názorně. Ideál VFR nebyl při návrhu používán jednostranně: lidé ať mohou to, co enti, ale částečně i na stranu druhou: enti ať mohou žít jako lidé. To nás motivovalo navrhovat svět velmi bohatý a složitý, navzdory očekávané obtížnosti realizace. Návrh světa se v tomto ohledu zdařil a ve světě entů je možné doslova vrazit čelem do problémů, které zdravému lidskému rozumu žádné zvláštní potíže. Rozsah projektu pak ilustruje složitost lidského světa.

1.2 Princip zachování běhu světa

Nedílnou součástí projektu je navržení světa, tj. konkrétní definice prostředí, ve kterém se enti mají pohybovat. Klíčovou vlastností světa je *princip zachování běhu*. To znamená, že svět musí být navržen tak, aby během simulace neustále zůstával ve stavu podobném stavu počátečnímu, a to i když se v něm budou pohybovat strojoví enti.

Na druhou stranu není možné zajistit, aby svět „opravoval“ sám sebe, pokud ho bude záměrně poškozovat lidský ent.

1.2.1 Princip zachování běhu pro enty a základní fyzické vlastnosti

Princip zachování běhu platí i pro enty. Jestliže budou mít enti základní fyzické vlastnosti, které svým významem budou určovat, jestli je entům dobře nebo špatně, musí být svět navržen tak, aby entům mohlo být většinu času dobře. Rozhodli jsme, že enti budou mít tuto sadu základních fyzických vlastností: únava, hlad, žízeň, zdraví, pnutí[2].

Aby entům mohlo být dobře, je ve světě možné: vyspat se, napít se, najíst se, vyléčit se, dojít si na záchod; a to opakovaně. Ve světě proto existuje jídlo, voda, místo na spaní, záchod a něco na léčení, přičemž voda, jídlo a léčicí prostředky jsou obnovitelné.

Při „horších“ hodnotách fyzických vlastností bude entům špatně. Co přesně znamená *špatně*? Mohli bychom počítat nějaký agregát vlastností – celkový pocit enta – a říci, že *špatně* znamená, že agregát je menší než 13,5. Nám se ale jako lepší jeví přímé penalizace. Vhodnou penalizací je například smrt; ent může zemřít žízní, hladem nebo na následky zranění. Při příliš vysoké únavě ent usne a při příliš vysokém pnutí… inu, neudrží potřebu.

1.2.2 Smysl života entů – hledání svatého grálu

Entovým cílem samozřejmě není dosíci nirvány. Na druhou stranu není ani vhodné, aby ent po celou dobu své existence jen stál na místě a nic nedělal. Jakou činnost by však měl ent vykonávat?

Odpověď nám přímo nabízí náš princip zachování běhu světa: enti se budou o svět starat.

Budou to právě oni, kdo ho budou opečovávat a udržovat v takovém stavu, aby jim poskytl vše, co k životu potřebují (tj. aby jim bylo dobře). Svět tedy nebude, jak bylo řečeno na začátku této kapitoly, udržovat sám sebe ve stavu blízkém k počátečnímu; v takovém stavu ho budou udržovat enti.

Tuto „péči o svět“ musí entům někdo „naprogramovat“ – bude to programátor entích skriptů a použije pro tento účel speciálně navrženého jazyka E (viz kapitola 4.1 na straně 6). V souvislosti s základními fyzickými vlastnostmi světa je teoreticky možné rozdělit základní úkoly ve světě například takto:

- starost o jídlo (aby enti měli dost potravin)
- starost o bezpečí (aby entům nikdo neubližoval)
- starost o čistotu (aby enti „neumírali na tyfus“)
- starost o předměty (aby rozličné předměty fungovaly, aby jich bylo dost, aby tekla voda)

Rozhodli jsme se do světa zahrnout i prvky zábavy. Přibývá tedy:

[2] Jak moc se chce entovi na záchod.

- starost o zábavu

1.2.3 Život a smrt enta

Řekli jsme, že budeme vůči entům tak krutí, že je čas od času necháme zemřít; hladem, žízní anebo na následky zranění. Můžeme si však dovolit vyloučit enta po jeho smrti ze světa? Ent se o část světa stará: jestliže zemře, s jeho kusem světa to začne jít s kopce a v důsledku toho mohou zemřít další enti.

Jak by takovou situaci vyřešili lidé? Převzali by na sebe povinnosti zemřelého. To je ovšem dost složité: Jak vyřešit, kdo má dostatek času vzít si na bedra další závazky? Jak naplánovat nové činnosti mezi víc jedinců, aby si při jejich vykonávání vzájemně nepřekáželi? Jak zjistit, kdo daným činnostem rozumí? Jak zjistit, kdo by se je mohl naučit?

Nad těmito problémy jsme se zamýšleli, a nakonec jsme se rozhodli, že tímto směrem projekt rozšiřovat nebudeme a činnosti na jiné enty přenášet nebudeme.

To znamená, že úkoly buď nikdo vykonávat nebude, nebo vznikne nový ent, který bude mít přesně stejné úkoly jako předchozí. Jelikož má smrt sloužit jako penalizace jednoho enta, nikoli všech, rozhodli jsme se jít druhou cestou, a sice tím nejjednodušším možným způsobem: enta necháme za čas znovu „se narodit“.

Je tedy vidět, že smrt enta pro něj není zas takovou tragédií, jakou by se mohla jevit na první pohled.

1.3 Komunikace přirozenou řečí

Cílem projektu Enti je též simulovat komunikaci přirozenou řečí. Studujeme-li podrobněji účel komunikace mezi lidmi, odhalíme tyto pohnutky:

Mluvením nahradit vnímání. Komunikace s kolegou mi umožňuje nechodit na místo, kde lze něco vidět, pokud tam kolega už byl a může mi o tom referovat. Přitom informační zisk může být větší buď pro mne (já se ptám a kolega odpovídá) nebo pro kolegu (já kolegu varuji, že v místnosti je něco nebezpečného, nebo naopak informuji, že něco, co on potřeboval je již k dispozici).

Mluvením nahradit konání. Nemusím určitou práci ve světě provádět já, pokud se komunikací dohodnu s kolegou, že mne zastoupí.

Mluvení jako nástroj pro myšlení. Lidé provádějí rozumové konstrukce často (ne-li vždy) za použití vnitřní řeči, promluvou k sobě samému. Vnitřní řeč pomáhá člověku udržet po delší dobu pozornost zaměřenou na konkrétní problém. Umožňuje sledovat sám sebe při provádění analýzy problému a kontrolovat správnost postupu.

Mluvením nahradit myšlení. Komunikace s kolegou mi umožňuje neprovádět nějakou rozumovou konstrukci, pokud ji kolega už provedl a může pomocí jazyka stejnou konstrukci vyvolat v mé mysli.

Mluvením řídit spolupráci. Komunikace umožňuje více osobám zapojit se do práce za společným cílem. Spolupracující bytosti se napřed musí dohodnout na společném cíli, tj. vypracovat a každá přijmout jeho zadání. Při samotné práci komunikace umožňuje pohodlnější synchronizaci v čase a prostoru.

Návrh projektu Enti zahrnoval pouze první dva uvedené cíle: mluvením nahradit vnímání a konání. Další cíle byly již v počáteční fázi zamítnuty, především proto, že myšlenkové konstrukce v mozku enta nedokážeme v rámci toho projektu propracovat do takové bohatosti, aby se přirozený jazyk stal užitečným nástrojem pro jejich vedení, natož pak aby o nich bylo zajímavé a úsporné mluvit. (Proces porozumění popisu myšlenkové konstrukce ať již z vlastní vnitřní řeči nebo z řeči kolegy bude v našem případě bohužel vždy složitější, než provést celou konstrukci znova.) Rovněž spolupráci entů se nepodaří rozvinout v projektu do takové míry, aby komunikace přesáhla pouhé rozkazování kolegům (což je pokryto ve druhém bodu).[3]

[3] Z popsaného rozhodnutí vyplývá mj. to, že komunikaci s našimi enty nikdy nebude možné vést tak, aby člověk zjišťoval obecně platná tvrzení o entím světě. (Např. „Jsou mrkve oranžové?“) Naši enti nemohou být z časových důvodů vybaveni bohatším a obecnějším odvozovacím aparátem (např. sémantickými sítěmi či jinou logickou reprezentací), aby byli schopni na takové otázky odpovídat. Rádi bychom toto omezení v pozdější době odstranili, neboť je v současné době k dispozici řada teoretických nástrojů nabízejících se k ověření praxí.

Z časových důvodů byla implementace obou uvedených cílů omezena, takže jsou v současné době strojoví enti schopni zpracovávat a generovat pouze věty z předem definované množiny (tuto množinu může autor světa podle libosti obohacovat). Naproti tomu jsme se zaměřili na konkrétní problém: správné zpracování a generování (vágních) odkazů na konkrétní předměty v simulovaném světě. Věty tedy mohou obsahovat odkazy na předměty jako „parametry“ a lingvistický modul enta zajistí správné vyjádření a pochopení označovaných předmětů[4]. V případě nejasností má lingvistický modul oprávnění zjistit podrobnější informace o předmětu. Mezi enty tak může proběhnout například konverzace uvedená v ukázce 1.

Uživatel	Seber modrý šroubovák.
Ent	*Který modrý šroubovák, ten šroubovák v pokoji člověka nebo ten v šroubovák v dílně?*
Uživatel	Ten v pokoji člověka.
(Ent sebere šroubovák.)	
Uživatel	Seber žlutý šroubovák v pokoji člověka.
(Ent sebere druhý šroubovák.)	
Uživatel	Dej ho do skříně.
Ent	*Do které skříně, do té skříně v pokoji policajta. . . ?*
Uživatel	Do <tohoto předmětu>. *(ukažte na skříň)*
(Ent dá žlutý šroubovák do skříně.)	
Uživatel	Dej ten modrý na postel v pokoji člověka.
(Ent dá modrý šroubovák na postel.)	

Obrázek 1: Ukázka konverzace přirozeným jazykem.

Technické detaily i problémy spojené s přesnou identifikací předmětů podle vágních označení je opět možné najít v dokumentaci projektu.

2 Fyzikální realita simulovaného světa

Projekt Enti záměrně velmi zjednodušuje fyzikální vlastnosti světa lidí:

Diskrétní běh času. Simulace je prováděna po kolech, v každém kole všichni enti dostanou celý popis relevantní části světa, rozhodnou se, jaký krok učiní, a požádají o provedení elementární změny v prostředí.

Diskrétní prostor a objekty, dvě dimenze a obsahování. Prostor světa entů není spojitý, ale je rozdělen na malé dlaždice ve čtvercové síti. Dlaždice mohou obsahovat jednotlivé předměty, včetně entů samotných. Předměty mohou (podle určitých pravidel) zase obsahovat předměty (rozlišujeme umístění *v* předmětu a *na* předmětu a zvlášť sledujeme objemovou a povrchovou kapacitu předmětů). Pro zjednodušení komunikace mezi enty a serverem prostředí bude navíc prostor dlaždic rozdělen do samostatných úseků, tzv. místností. Enti tedy pravidelně dostávají informace o všech změnách, které proběhly v daném kole v místnosti, kde se ent nachází. Naopak ent nedostává žádné informace o změnách v jiné místnosti.

Omezený čas i prostor. Entí čas je zdola omezen startovním okamžikem projektu. Entí prostor je omezený, v simulovaném světě je dopředu dán počet místností, místnosti nemohou přibývat. Rovněž velikost místností je dána předem, v místnostech nemohou přibývat dlaždice. Naproti tomu není explicitně omezen počet předmětů na dlaždici, či hloubka vkládání předmětů do sebe. (Je zde ovšem přirozené omezení podle „velikosti“ předmětů, a rovněž omezení dané reprezentací identifikátorů předmětu.)

V průběhu práce na projektu se popsaná diskretizace mnohokrát ukázala jako zjednodušení doslova spásné. Pevná kategorizace předmětů, elementů času i prostoru dává totiž programátorům entů do ruky jistotu existence konečné odpovědi na každou otázku. Určujeme-li pro enta, zda něco vidí nebo nevidí, máme díky diskrétním místnostem jasnou odpověď. Ent vidí všechny předměty, které leží na dlaždicích v té místnosti, kde je on sám. Není možné, aby ent nějaký předmět viděl částečně

[4]Předměty lze označovat vlastními jmény (enti a místnosti), přesným popisem (*šroubovák na polici v dílně*) či „ukázáním na předmět“. Pokud o předmětu již byla řeč je možné použít též zájmen či odkazování typu *ten modrý šroubovák*.

(třeba proto, že leží relativně daleko od něj, ale ne zas tak daleko, aby ho neviděl vůbec). Není možné, aby ent sdělení kolegy „slyšel, ale trochu špatně“.

Diskrétní a omezený prostor navíc umožňuje případné dokazování tvrzení o světě entů. (Např. poskytuje zarážku pro matematickou indukci.) Princip matematického dokazování bohužel nelze v reálném světě použít právě proto, že zatím jsme v žádném směru nenarazili na nějakou hranici tohoto světa.

3 Architektura projektu

Simulace světa se zúčastňují tři základní typy komponent:

Server prostředí je centrálním prvkem simulace. Zajišťuje simulaci fyzikální podstaty světa a dohlíží na dodržování „fyzikálních zákonů“ (nedovolí překročit kapacitu dlaždice či předmětu ap.). Fyzikální zákony jsou v serveru prostředí vestavěny, konkrétní konfigurace světa (definice místností, základní rozmístění předmětů, počet entů v simulaci ap.) jsou načítány při startu z konfiguračního souboru.

Ent je program zajišťující simulaci jedné bytosti ve světě. Podrobněji viz 4 na následující straně.

Prohlížeč zajišťuje lidskému uživateli možnost vstoupit do simulovaného světa. Pomocí Prohlížeče uživatel nejen vidí stav místnosti v simulovaném světě, ale též ovládá jednoho z entů – konkrétně toho v modrých kalhotách a žluté mikině, viz obrázek 2.

Každá z těchto komponent byla implementována jako samostatný program. Server prostředí i procesy jednotlivých účastníků v simulaci mohou být spuštěny na odlišných počítačích; komunikují spolu prostřednictvím standardních unixových soketů a připojení je tedy možné např. i po síti Internet.

Obrázek 2: Pohled do simulovaného světa pomocí prohlížeče.

3.1 Společná řeč komponent – systém handlů

Diskrétní povaha simulovaného světa umožnila zavedení konceptu „handlů“, univerzálního číslování všech časových okamžiků, prostorových úseků, předmětů i jejich vlastností.

Při implementaci výhod existence univerzálního systému handlů rovněž využíváme: handly pro typy předmětů ap. jsou definovány ještě před samotnou kompilací všech komponent a ve zdrojových kódech je symbolicky můžeme označovat speciálními výrazy tvaru `HOzidleH` ap. Před kompilací jsou těmto symbolickým handlům jednotně pro všechny zdrojové kódy přiřazeny konkrétní čtyřbajtové číselné konstanty. Tímto způsobem je pohodlně zajištěna konzistence číselného označování objektů simulovaného světa ve všech zúčastněných programech, a to bez ohledu např. na konkrétní programovací jazyk, v němž jsou implementovány.

3.2 Chod simulace

Simulovaná bytost (strojový ent či lidský prohlížeč) v každém časovém okamžiku dostane na vstup uspořádaný seznam handlů, který kóduje aktuální stav světa v místnosti enta. Ent si pro další postup může pamatovat ze světa cokoli.

V rámci tohoto kola ent povinně reaguje odesláním elementárního požadavku (opět vyjádřeného formou handlu, tzv. *atomické instrukce*) na změnu světa. Ent může vyjádřit požadavek nedělat v tomto kole nic.

Server prostředí požadavky entů vyhodnotí a je-li to možné, uplatní změny v simulovaném světě. V dalším kole pak entům pošle společně s aktualizací stavu světa informaci o úspěchu či neúspěchu jejich předchozího požadavku. V případě konfliktních požadavků dvou entů (přesun na tutéž dlaždici ap.) server prostředí náhodně rozhodne, který z entů uspěl a který ne.

4 Umělý člověk – Ent

Chování každé umělé bytosti zapojené v simulaci v projektu ENTI zajišťuje samostatný proces, program `ent`. Program `ent` řeší především tyto úkoly: zpracovává údaje od serveru prostředí, vnímá a pamatuje si změny v prostředí, plánuje své chování v prostředí, pravidelně odesílá serveru prostředí své bezprostředně požadované akce, reaguje na zaslechnuté věty přirozeného jazyka, vyslovuje věty přirozeného jazyka.

Rozdílné úkoly programu `ent` bylo možné delegovat samostatným komponentám. Stručně popíšeme jednotlivé komponenty zde, podrobněji se jim věnují následující kapitoly:

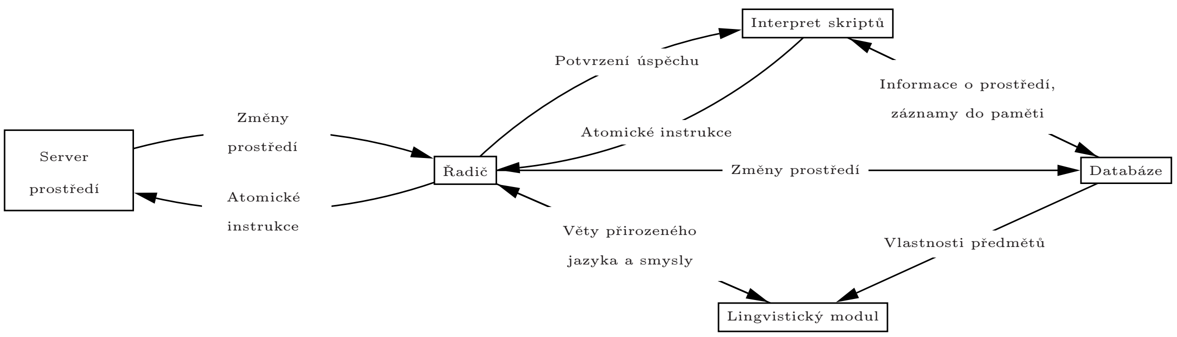
Řadič zajišťuje komunikaci se serverem prostředí. Načítání dat do pracovních datových struktur a předávání dat dalším komponentám.

Databáze zajišťuje entovi veškerou správu poznatků. Do databáze si ent ukládá, co právě vidí i viděl, co plánuje kdy provádět ap. Databáze je též využívána jako sdílená paměť, pokud si ostatní komponenty potřebují předat nějaké údaje.

Interpret skriptů realizuje samotný chod enta. Interpret načte skripty enta (viz 4.1) a postupně je vykonává. Skripty mohou provádět řadu vnitřních výpočtů, nahlížet do databáze, ověřovat aktuální podmínky v prostředí. Konečným úkolem skriptů je pak emitovat atomické instrukce, určovat, co má ent za daných okolností provést.

Lingvistický modul zajišťuje entovi schopnost porozumět zprávám přirozeného jazyka a tyto zprávy též generovat. Lingvistický modul je především překladač mezi přirozeným jazykem a jednoduchou vnitřní reprezentací významů vět.

Přehled komponent i jejich vzájemné vztahy jsou načrtnuty ve schématu 3.



Obrázek 3: Základní komponenty enta.

4.1 Jak enty programovat – Jazyk E

S ohledem na složitost simulovaného prostředí a „řídké pocity uspokojení“, jichž enti v simulovaném prostředí mohou dosáhnout[5], jsme se rozhodli chování umělých bytostí přímo programovat a nespoléhat se na automatické nalezení optimálních postupů např. metodou genetických algoritmů. Specifika simulovaného prostředí nás přiměla k návrhu vlastního programovacího jazyka, jazyka E.

Základ jazyka E tvoří:

[5]Ent musí udělat mnoho kroků, projít mnoho dveří a provést mnoho úkonů, než upeče bábovku, kterou může sníst.

Skripty. Skript je zápisem algoritmu enta v jazyku E. Skript si můžeme představit jako term nebo i jako proceduru – ve skutečnosti je něčím mezi tím. Skript je interpretem *postupně* (to je důležité – postupně, nikoli najednou!) rozkládán na akce, které ent bude provádět v dalším kole (tzv. atomické instrukce).

Máme tak např. skript na zalévání, na jezení, chůzi po dlaždicích v rámci místnosti, hledání konkrétního předmětu kdekoli v domě atd.

Interrupty. Při formulování algoritmů chodu enta nás pálí, že musíme v každém kole kontrolovat velké množství podmínek. Lze si všimnout, že tyto podmínky jsou vždy po určitou dobu stejné – například po celou dobu zalévání ent musí držet konev. Tyto podmínky jsou ve skutečnosti invarianty určitých částí skriptu. Místo, abychom v každém kroku kontrolovali, jestli invariant ještě platí, celý proces zautomatizujeme pomocí interruptu:

Interrupt je skript, který ent začne provádět po splnění určité podmínky. V jistou chvíli řekneme, že je interrupt *aktivní* („navešený“ – od této chvíle chceme, aby invariant platil) – a dokud neřekneme, že už aktivní není (invariant už nemusí platit), interpret jazyka se postará o automatické kontrolování jeho podmínky.

Můžeme tedy snadno zjištovat, jestli ent ješte konev drží – a co víc! – pokud nedrží, můžeme rovnou spustit skript, který situaci napraví.

Hodnotící funkce. Při hlubší analýze zjistíme, že ne všechny skripty jsou stejně „důležité“ – nadto jejich důležitost se v čase mění. Když se entovi zachce na záchod, bude pro něj jiste zajímavější skript, který mu uleví, než skript na čtení knihy.

Každý skript má hodnotící funkci, jež na základě stavu enta a jeho okolí spočítá, jak „důležitý“ zkoumaný skript v danou chvíli je. Na základě hodnotící funkce se interpret rozhodne, který skript má z několika alternativ spustit.

Polotransakční zpracování. Půjdeme-li ještě dál, zjistíme, že mnohé operace by bylo dobré provádět transakčně. To ale většinou není možné: enta může v jeho činnosti kdykoli cokoli přerušit, navíc když se k činnosti vrátí zpět, už nemusí být možné ji dokončit (uvažme případ, kdy ent zalévá záhon a dojde mu voda; ent si vodu dojde naplnit, ale někdo mu mezitím záhon zalije). A samozřejmě také není možné provádět nějaké akce „soukromě“ a teprve na závěr je „potvrdit“ – všechny změny se do světa entů promítají okamžitě.

Z transakčních nástrojů zavádíme modifikovaný `COMMIT` (činnost okamžitě končí úspěchem) a `FAIL` (činnost okamžitě končí neúspěchem). Dále `RERUN` (začni celou akci znovu); to ale někdy není možné, počáteční podmínky pro daný skript už neplatí. V takovém případě bychom potřebovali `UNDO`, aby ent uvedl vše do původního stavu nebo po nedokončeném skriptu alespoň „uklidil“. Je navržena *záchranná varianta* skriptu, která se spouští právě pro případy `UNDO` – ve stávající verzi interpretu však není implementovaná.

Skripty samotné se podobají prologovským termům, interpret jazyka E umí vykonávat skripty s backtrackováním i bez něj. Pro další zjednodušení máme především pro nebacktrackující skripty zavedeny: obdobu for-cyklu, podmínky „if-then“, proměnné předávané odkazem a globální proměnné.

Příklad konkrétního skriptu naleznete v ukázce 4 na následující straně.

Ve stávající implementaci jsou formou knihovních funkcí připraveny skripty pro chůzi, manipulaci s předměty, hledání předmětů, péče o základní životní potřeby (jídlo, pití, únava, toaleta) a též skripty pro péči o zahradu (zalévání, sázení, okopávání) a kulturu (hra na hudební nástroj). Knihovnu entích skriptů je možné obohacovat, podrobný pois syntaxe jazyka i ovládání debuggeru je k dispozici v *Příručce autora světa a skriptů.*

5 Shrnutí

Představili jsme projekt Enti, jehož cílem bylo vytvořit simulátor prostředí podobného přirozenému lidskému světu. Podrobněji jsme se věnovali cílům projektu i výsledné architektuře softwarového díla, které vytčených cílů alespoň částečně dosahuje. Výsledný simulátor obsahuje úplné prostředí simulovaného světa, schopnosti simulovaných entů jsou však dosud do značné míry omezené. Simulátor je tedy vhodnou „laboratoří umělé inteligence“, kde je možné zkoušet algoritmy

```
// Dojde do místnosti, kde je daný ent. Pokud je příznak nastaven na "yes",
// snaží se dojít až k entovi. Pokud kolega utíká, stále jej ent sleduje.
jdiKEntovi(hEnt, bAzKNemu)
$- return 0.  // Hodnotící funkce vrací vždy 0, skript nemá více variant, mezi
              // nimiž by bylo třeba rozlišovat
:-
  if jeTuTaky(hEnt)
  then
    // Daný ent je ve stejné místnosti jako já
    if bAzKNemu == "no" then COMMIT fi, // a nemusím k němu dojít -> hotovo
    if cJsemU(hEnt) then COMMIT fi,     // už jsem u něj -> hotovo
    try dojdiKEntoviVTetoMistnosti(hEnt), // neuteče-li jinam, dojdu k němu
    RERUN  // Znovu spusťme tento skript, ať se ověří, že jsem u enta
  fi,
  // Ent není v této místnosti. Půjdu tam, kde kdysi býval, ale nastavím si
  // interrupt, který ohlídá, jestli enta nepotkám už cestou. Pokud ano,
  // restartuji tento skript, takže vyrazím hned za ním.
  getLocalPriority(jdiKEntoviPrio),
  iVidimEnta = _,
  localHook(
    { jeTuTaky(hEnt) },  // Test, podmínka interruptu
    jdiKEntoviPrio + 1,  // Priorita interruptu o jedna vyšší, než priorita
                     // tohoto skriptu
    { RERUN },           // Při splnění podmínky restaruj.
    iVidimEnta           // Identifikátor nově navěšeného interruptu
  ),
  if faktBytCoKdeOdkdy(hEnt, hKdeJe, _), striptoken(hKdeJe), jeMistnost(hKdeJe) then
    // Vím, že ent býval v nějaké místnosti.
    jdi(hKdeJe),  // Půjdeme tedy do té místnosti
    RERUN         // a tam se znovu rozhlédneme.
  fi,
  // Bohužel nevím, kde ent je, projdu postupně všechny místnosti a interrupt
  // mi ohlídá, jestli enta nepotkám.
  vsechnyMistnosti(sTempMistnosti),
  cNaplanujPohodlnyPruchod(sTempMistnosti, sMistnosti),
  usporadanaProchazka(sMistnosti),
  // Obešel jsem všechny místnosti a nikde ho nepotkal, selžu.
  FAIL.
```

Obrázek 4: Ukázka skriptu, který dovede enta k danému kolegovi.

chování umělých bytostí. Jazyk vhodný pro programování těchto bytostí jsme stručně popsali i zde, zájemce o podrobnější informace nezbývá než odkázat na podrobnou dokumentaci k projektu.

6 Poděkování

Práce na tomto tématu byla částečně podpořena grantem GAČR č. 201/02/1456 a grantem GAUK č. 300/2002/A INF/MFF.

7 Literatura

Ondřej Bojar, Cyril Brom, Milan Hladík, Mikuláš Vejlupek, Vojtěch Toman, David Voňka. *Dokumentace studentského softwarového projektu* ENTI*: Teoretická dokumentace, Uživatelská příručka, Příručka autora světa a skriptů, Programátorská dokumentace.* 2002. MFF UK, Praha.