# Logické programovanie a reprezentácia znalostí

Ján Šefránek

Ústav informatiky, Fakulta matematiky, fyziky a informatiky Univerzity Komenského, Bratislava, e-mail: sefranek@fmph.uniba.sk

**Abstrakt** O reprezentácii znalostí sa veľmi často rozmýšľa na pomerne nízkej úrovni (rámce, siete, pravidlá atď.). Ide v podstate o úroveň dátových štruktúr, s ktorými manipulujú nejaké procedúry. Inferencia sa vtedy chápe čisto procedurálne. Jedným z cieľov príspevku je poukázať na slabiny takéhoto prístupu. Hlavným cieľom však bude predstaviť teóriu reprezentácie znalostí, založenú na rôzne silných jazykoch logického programovania.

## 1 Úvod

Výskum v oblasti reprezentácie znalostí prešiel minimálne troma vývojovými fázami. V počiatočnom období umelej inteligencie (päťdesiate a šesťdesiate roky) dominovala predstava, že stačí opísať nejakú doménu pomocou formúl prvorádovej logiky a pomocou automatického dokazovania získať dôsledky, potrebné pre riešenie daného problému. Skúsení logici videli hneď v počiatkoch úskalia tohto programu.[1] Ostatní museli na základe skúsenosti zistiť, že bariéra nevypočítateľnosti, a principiálna neúplnosť našich poznatkov odsudzujú takýto program na neúspech.

Jedným z najznámejších kritikov programu bol Minski [13], ktorý v opozícii k dôrazu na dedukciu prišiel s ideou nemonotónnosti usudzovania a zdôrazňoval zhlukovanie poznatkov do celkov, ktoré nazval *rámcami*. Neskorší výskum a implementácie však ideu rámca zosunuli do úrovne údajových štruktúr a k odvodzovaniu sa pristupovalo procedurálne. Spravidla absentovala sémantická charakterizácia procedúr, manipulujúcich s reprezentovanými poznatkami. Navyše, reprezentácia poznatkov a odvodzovanie sa chápali ako odlišné a oddelené domény a schopnosti. (Kvôli zjednodušeniu vyjadrovania budeme o tomto prístupe neskôr hovoriť ako o *klasickom prístupe* k reprezentácii znalostí.)

Nemonotónne usudzovanie sa stalo predmetom záujmu teoretikov už koncom sedemdesiatych rokov [2]. Postupne sa ukázala tesná súvislosť tohto výskumu s výskumom v logickom programovaní a deduktívnych databázach [16]. Význam výskumu v oblasti nemonotónneho usudzovania a logického programovania pre reprezentáciu poznatkov sa zdôrazňuje už od začiatku deväťdesiatych rokov [3]. Dôraz sa kládol na fundamentálne problémy reprezentácie znalostí - sémantiku, vyjadrovaciu silu a výpočtovú zložitosť. Vytvoril sa jednotiaci pohľad na bázu poznatkov a odovodzovanie. Položili sa pevné sémantické základy hypotetickému

[1] Pozri Bar Hillelov komentár k McCarthyho vízii, publikovaný v [12].

usudzovaniu, ktoré je kľúčové z hľadiska spracovania znalostí. Od polovice deväťdesiatych rokov sa veľká pozornosť venuje implemenáciám [14, 7, 11, 15, 4]. O tomto prístupe budeme ďalej hovoriť ako o prístupe, založenom na logickom programovaní.

*Problém.* Medzi klasickým prístupom a prístupom, založenom na logickom programovaní možno vidieť latentné napätie. Ide predovšetkým o napätie medzi zdôrazňovaním a zanedbávaním sémantických základov znalostných systémov. Prirodzeným dôsledkom zdôrazňovania (zanedbávania) sémantiky je jednotiaci (ostro oddeľujúci) pohľad na reprezentáciu a odvodzovanie. Oba prístupy existujú popri sebe, ale bez vzájomného ovplyvňovania. Najmä v našom prostredí je prístup, založený na logickom programovaní, takmer neznámy.

*Cieľ článku.* Článok je prehľadový. V istom zmysle slova je aj osvetový. Jeho cieľom je informovať o relevancii výskumu v logickom programovaní a nemonotónnom usudzovaní pre pochopenie reprezentácie poznatkov. Oveľa podrobnejšie o téme pojednáva [17]. Domnievame sa, že v našom prostredí je takýto prehľad nanajvýš potrebný. Motivácia pre jeho napísanie sa vynorila počas komunikácie s popredným predstaviteľom komunity, sústredenej okolo série konferencií Znalosti.

## 2 Usudzovanie ako kľúč k reprezentácii znalostí

Tradične sa pomerne ostro rozlišuje medzi reprezentáciou znalostí na jednej strane a usudzovaním na strane druhej. Takáto ostrá hranica však nie je dobre zdôvodnená.

Tu nás zaujímajú systémy, ktoré používajú pri realizácii výpočtových úloh znalosti. Nie je dôležité, či znalosť, potrebnú na vykonanie nejakej úlohy (zodpovedanie nejakej otázky) systém získa prehľadávaním bázy znalostí alebo ju odvodí. Prehľadávanie bázy znalostí je iba triviálne odvodenie (počiatočná fáza procesu usudzovania). Navyše, očakáva sa, že riešenie úloh s využitím znalosti má byť do značnej miery flexibilné, schopné zvládať výnimky, neobvyklé situácie, nedostatok potrebných informácií. Inými slovami: má využívať hypotetické usudzovanie.

Pre tento typ úloh je výhodný deklaratívny jazyk na vyjadrenie, zápis (reprezentáciu) znalostí a na formulovanie požiadaviek. Štúdiu a charakterizácii takýchto jazykov sa venuje logika. Úloha logiky pre pochopenie inteligencie, založenej na spracovaní znalostí sa často prirovnáva k úlohe matematickej analýzy pre fyziku. Logické programovanie má popri dobre definovanej sémantike aj tú výhodu, že umožňuje zapisovať vykonateľné programy. Tým sa stáva dobrým kandidátom na deklaratívny jazyk (presnejšie, triedu deklaratívnych jazykov) pre reprezentáciu znalostí.

Budeme ďalej predpokladať, že výpočtové systémy nebudú mať všetky potrebné znalosti explicitne zapísané v nejakej databáze (báze znalostí), ale budú schopné odvodiť si aj dôsledky toho, čo je explicitne zapísané v báze znalostí, prípadne vygenerovať hypotézy, zlučiteľné s obsahom bázy znalostí.

Predstavu, ktorá nás vedie k tomu, že nie je dôležité rozlišovať medzi znalosťami reprezentovanými explicitne a znalosťami získanými usudzovaním, zhrnieme takto:

Znalosti, ktoré má agent k dispozícii, sú vyjadrené (reprezentované) v nejakom jazyku $J$. Nejaká množina $E$ viet (formúl) jazyka $J$ reprezentuje tie znalosti, ktoré sú explicitne zapísané (v báze). Znalosti, získané usudzovaním, sú opäť vety jazyka $J$. Vety, ktoré agent odvodí z $E$ označíme ako $Cn(E)$. Pritom $Cn$ požadujeme za *operátor odvodenia*, ktorý množinám viet priraďuje množiny viet. Je pritom definovaná nejaká *sémantika* $\sigma$, ktorá výrazom jazyka priraďuje nejaké objekty. *Sémantickou špecifikáciou* operátora odvodenia $Cn$ možno nazvať špecifikáciu nejakého vzťahu medzi $\sigma(E)$ a $\sigma(Cn(E))$ pre ľubovoľné $E$.

Obvykle sa $\sigma(E)$ definuje ako nejaká trieda modelov. Ak teda $\sigma(E)$ je nejaká trieda modelov množiny $E$, potom sa $Cn(E)$ zvykne definovať ako množina všetkých viet pravdivých v každom modeli zo $\sigma(E)$, teda $\sigma(E) = \sigma(Cn(E))$.[2] Uvedenou šablónou sa pozrime na dedukciu: *Dedukcia* (z nejakej teórie, či bázy poznatkov) sa charakterizuje pomocou triedy *všetkých* jej modelov. Keď $\sigma(E)$ je trieda všetkých modelov bázy $E$, potom množina deduktívnych dôsledkov množiny $E$, označme ju $Th(E)$, je množinou všetkých viet pravdivých v každom modeli zo $\sigma(E)$. Neskôr sa budeme venovať sémantickej špecifikácii ne-deduktívneho oddvodenia.

Môžeme teda uzavrieť: Ak si zvolíme nejakú sémantiku explicitne reprezentovaných poznatkov, potom tá si vynucuje, akú množinu poznatkov musíme akceptovať. Inak nami definovanú sémantiku neberieme vážne. Možno treba ešte venovať pozornosť otázke, či vôbec treba nejakú sémantiku.

## 3 Sémantiky nemonotónneho usudzovania

Rozlíšime dva základné typy sémantickej charakterizácie hypotetického (nemonotónneho)[3] usudzovania. Tento typ usudzovania má totiž kľúčovú úlohu z hľadiska reprezentácie poznatkov. Treba však pripomenúť, že ide o schematický pohľad na množstvo rôznych prístupov.

Už dávno sa v teórii a praxi znalostných systémov akceptuje, že realistický prístup k problému reprezentácie poznatkov spočíva na uvedomení si neúplnosti poznania, ktoré je k dispozícii. Preto sa považuje za nutné pracovať s predbežnými závermi, odvolateľnými hypotézami, revidovať bázy poznatkov, vrátane poznatkov, ktoré sú reprezentované implicitne. To znamená, že potrebujeme usudzovanie, ktoré je nemonotónne: po pridaní nových poznatkov do bázy sme niekedy nútení vzdať sa niektorých predbežných záverov, ak nie sú zlučiteľné s novými poznatkami.

Možno rozlišovať dve triedy sémantík nemonotónneho usudzovania. Prvá z nich – nazvime ju *preferenčnou sémantikou* – vychádza z toho, že niektoré modely

[2] Treba tu iba poznamenať, že operátory odvodenia, zaujímavé z hľadiska reprezentácie znalostí majú často iné vlastnosti. Zaujímavou vlastnoťou je alternatívnosť, pozri napríklad [17].

[3] Oba termíny považujeme v tomto texte za synonymné.

sú viac dôležité, iné menej. Formálne, definuje sa nejaká relácia preferencie $\prec$ na modeloch. $M$, model bázy $E$ je maximálne preferovaný vtedy, keď neexistuje jej model $N$, ktorý by bol preferovanejší, $(\neg\exists N) N \prec M$. Typickým príkladom sú minimálne modely (ten, kto si pripravuje plán cestovania železnicou, pracuje s minimálnym modelom: za pravdivé považuje iba to a presne to, čo je napísané v cestovnom poriadku).

Keď $\sigma_{\prec}(E)$ je trieda všetkých maximálne preferovaných modelov bázy $E$, potom množina viet, ktoré preferenčne vyplývajú z množiny $E$, označme ju $Cn_{\prec}(E)$, je množinou všetkých viet pravdivých v každom modeli zo $\sigma_{\prec}(E)$.

Prejdime k druhej zo spomínaných tried sémantík, nazvime si ju *sémantikou pevných bodov*. Intuitívne ide o to, že sa konštruujú množiny tých hypotéz, ktoré dohromady dávajú zmysel, neodporujú si a ktoré vytvárajú čo najúplnejší obraz danej oblasti. Trochu viac technicky: Definuje sa operátor, ktorý množine viet priradí množinu viet. Tento operátor formalizuje dve dôležité črty - *nasýtenosť* ("výstupná" množina obsahuje všetky relevantné vety) a *koherenciu* (výstupné vety si vzájomne neodporujú). Pevné body takýchto operátorov sú význačnými sémantickými objektami.[4] Keďže ide o komplikovanejšiu konštrukciu, uvedieme príklad - stabilné modely logických programov. Prezentovaný výklad z [18] sa líši od pôvodnej teórie Gelfonda a Lifschitza [9].

Predpokladáme nejakú množinu atómov $\mathcal{A}$ výrokovej logiky.[5] Literál je atóm $A \in \mathcal{A}$ alebo negácia atómu $notA$, kde $A \in \mathcal{A}$. Negácie atómov nazvime negatívnymi literálmi. V logickom programovaní sa, intuitívne, literál $notA$ považuje za pravdivý, ak nie je známe, že by bol pravdivý $A$. Teda $notA$ sa prijíma ako hypotéza dovtedy, kým nie je vyvrátená (pomocou $A$). Jedným z upresnení tejto intuície je práve sémantika stabilných modelov. Pravidlo je formula $r$ tvaru

$$A \leftarrow A_1, \ldots, A_k, notB_1, \ldots, notB_m,$$

kde $A$, $A_i$, $B_j$ sú atómy (pre každé $i = 1, \ldots, k$, $j = 1, \ldots, m$). Množinu literálov $\{A_1, \ldots, A_k, notB_1, \ldots, notB_m\}$ nazývame telom pravidla $r$ (značenie $body(r)$) a atóm $A$ nazývame hlavou pravidla ($head(r)$). Zamýšľaný význam pravidla: ak sú všetky literály $A_1, \ldots, A_k, notB_1, \ldots, notB_m$ pravdivé, potom je aj atóm $A$ pravdivý.

Kripkeho štruktúra asociovaná s programom $P$ je dvojica $(W, \rho)$. $W$ je množina konzistentných množín literálov, doplnená o symbol $w_{\perp}$, reprezentujúci každú nekonzistentnú množinu literálov. Relácia dostupnosti $\rho$ je množina dvojíc $(w_1, w_2)$ takých, že existuje pravidlo $r \in P$, pre ktoré $w_1 \models body(r)$ a $w_2 = w_1 \cup \{head(r)\}$, ak $w_1 \cup \{head(r)\}$ je konzistentná množina literálov. V opačnom prípade je $w_2 = w_{\perp}$. O prvkoch relácie dostupnosti budeme hovoriť ako o hranách a o ich postupnostiach ako o cestách.

[4] Množina hypotéz, ktorá je pevným bodom takéhoto operátora, spĺňa predstavu o nasýtenej a koherentnej množine hypotéz. Nasýtenosť zaručuje, že nič relevantného sme neopomenuli, koherencia, že dohromady dávajú rozumný zmysel.

[5] Sémantika stabilných modelov berie do úvahy miesto prvorádového programu množinu všetkých jeho základných inštancií, teda – vlastne – výrokovologický program, možno i nekonečný.

Množina literálov $S$ je *úplná*, ak pre každý atóm $A \in \mathcal{A}$ buď $A \in S$ alebo *not* $A \in S$. Úplnú množinu literálov $S$ voláme *stabilným modelom* výrokovologického programu $P$ práve vtedy, keď je splnená nasledujúca podmienka:

– v Kripkeho štruktúre asociovanej s $P$ existuje cesta, ktorej koreňom je $S^-$, množina všetkých negatívnych literálov z $S$
– táto cesta je ukončená v $S$, t.j. neexistuje hrana $(S, w_\perp)$.

Idea: Ako hypotézy predpokladáme všetky negatívne literály z $S^-$. Pomocou programu $P$ z nich postupne odvodzujeme všetky možné dôsledky. Ak dosiahneme tak nasýtenú množinu literálov, že z nich už nemožno pomocou $P$ odvodiť nič nového, ani naraziť na nekonzistentosť, potom $S$ možno považovať za význam programu $P$. $S$ je pevným bodom operátora, ktorý interpretáciám priraďuje interpretácie (využívajúc pritom pravidlá programu $P$). Inými slovami: ak $P$ považujeme za návod, ako meniť obraz o možnom stave sveta (reprezentovaný množinou literálov, pravdivých v tom stave), potom možný obraz sveta $S$ je stabilným modelom $P$ vtedy, keď $P$ nedokáže transformovať $S$ do inej množiny literálov (do iného obrazu sveta).

## 4 Reprezentácia poznatkov na úrovni údajových štruktúr a procedúr

V tejto časti uvedieme niekoľko príkladov, demoštrujúcich nedostatočnosť takých prístupov k reprezentácii poznatkov a inferencii, ktoré nie sú založené na nejakej sémantike.

**Algoritmus najkratšej cesty**

V umelej inteligencii sa ako prostriedok na reprezentáciu znalostí intenzívne uplatňujú sémantické siete, v ktorých sa dedia znalosti z viacerých nadtried, môžu sa vyskytovať konflikty medzi nimi a pripúšťajú sa výnimky zo vzťahu podtrieda/nadtrieda. V takejto sieti možno odvodiť (reprezentovať!) vzájomne kontradiktorické tvrdenia. Folklórny je príklad o tvorovi menom Tweety, ktorý by mal lietať, pretože je vták a súčasne by nemal lietať, lebo je pštros.

Na riešenie týchto konfliktov vymyslel Scott Fahlman v sedemdesiatych rokoch minulého storočia algoritmus najkratšej cesty [6]. Tento algoritmus zisťoval, či (napríklad) Tweety lieta alebo nelieta tak, že odštartoval z vrcholu *Tweety* a paralelne prechádzal všetkými hranami krok po kroku k susedným vrcholom. Ak došiel k vrcholu *lieta* skôr po pozitívne označenej hrane, prijal sa záver, že Tweety lieta, ak prišiel skôr po negatívne označenej hrane, prijal sa záver, že Tweety nelieta. Ak naraz, zdržal sa úsudku (vo vtedajšej terminológii: obe cesty sa vzájomne neutralizovali).

Ako vidno, ide o čisto procedurálne definovaný prístup k inferencii. Tento algoritmus však zlyháva ako prostriedok na spracovanie poznatkov. Náhodné hrany v alternatívnych reprezentáciách môžu podstatne ovplyvniť dĺžku cesty. V komplikovanejších sieťach niektoré vrcholy (na najkratšej ceste k cieľovému vrcholu) môžu byť neutralizované, a teda by nemali poskytovať podporu pre ďalšie závery.

Podstatné, čo kritici tohto algoritmu časom pochopili, bolo však to, že vrcholy (presnejšie – výroky, ktoré zodpovedajú vrcholom a hranám medzi nimi), ležiace na najkratšej ceste k cieľovému vrcholu, nemusia nutne tvoriť množinu hypotéz, ktoré dohromady dávajú zmysel. V technickejších termínoch – netvoria extenziu defaultovej teórie, ktorá zodpovedá danej sémantickej sieti.[6]

Videli sme, že čisto procedurálny prístup k manipulácii so štruktúrou nízkej úrovne nepostačuje na adekvátne spracovanie poznatkov, na korektné odpovedanie na otázky adresované báze poznatkov, lebo nevenuje dostatočnú pozornosť sémantickej špecifikácii usudzovania.

**Sémantické siete - procedurálna a deklaratívna reprezentácia**

Ku grafovým reprezentáciám sémantických sietí možno priradiť logický jazyk, v ktorom sa dá formulami vyjadriť presne to, čo v grafovom jazyku. A naopak, ľubovoľnú sémantickú sieť vyjadrenú v logickom jazyku, vhodnom na reprezentáciu hierarchií konceptov a ich rolí, možno ekvivalentne vyjadriť v grafovom jazyku. To vôbec nie je prekvapujúce. Iba o málo zaujímavejšie je, že indexovanie, ktoré je prirodzene implicitné v sémantických sieťach (i rámcoch), možno zaviesť aj na formulách, pozri napríklad [17]. Preto aj ľubovoľný výpočet na grafovej reprezentácii možno previesť na výpočet na zodpovedajúcej logickej reprezentácii (a aj naopak).

Preto veľmi prekvapuje, keď niektorí výskumníci hovoria (prípadne hovorili) o grafovej reprezentácii sémantických sietí (kombinovanej s procedurálnymi inferenčnými technikami) ako o výpočtovo efektívnejšej, hoci menej rigorózne definovanej (v porovnaní s logickou reprezentáciou a "deklaratívnymi inferenčnými technikami"). Pozri napríklad [1], s. 320 (zdôrazňujeme, že ide o prvé vydanie tejto učebnice, frekventovane používanej na celom svete, v druhom vydaní sa táto pasáž už neobjavila).

Treba pripomenúť, že systematickému štúdiu reprezentácií toho typu, čo predstavujú sémantické siete a rámce, sa venujú logiky deskripcie (description logic). Usudzovanie pomocou logík deskripcie možno realizovať aj pomocou prekladu do AnsPrologu [4].

**TMS**

Ďalší príklad prístupu, opísaného a vytvoreného na (nízkej) úrovni, blízkej implementácii, je Truth Maintenance System [5]. Autor, Jon Doyle, tento prístup opísal na úrovni uzlov, zoznamov a procedúr, ktoré s nimi pracujú. Kľúčová bola Truth Maintenance procedúra, ktorej cieľom bolo udržiavať množinu presvedčení systému (v pôvodnej formulácii: množinu uzlov, ktoré sú IN) v adekvátnom stave po každej modifikácii. Doyle charakterizoval túto procedúru ako efektívnu. Po abstraktnejšom opise TMS v logickom jazyku s presnou sémantikou ukázal Elkan [8], že TM procedúra je vlastne procedúrou výpočtu stabilného modelu, čo je NP-úplný problém.

[6] Tradičné formalizmy nemonotónneho usudzovania – defaultová logika, cirkumskripcia, autoepistemická logika – možno preložiť do rôznych jazykov logického programovania a, navyše, ich sémantika korešponduje sémantike logických programov [16]. Preto je pozornosť tohto textu sústredená na logické programovanie ako na predstaviteľa systematického prístupu k reprezentácii znalostí a k usudzovaniu.

**Zhrňme**

Všetky uvedené príklady mali ukázať, že reprezentácia znalostí na úrovni údajových štruktúr a inferencia na úrovni procedúr bez sémantickej špecifikácie majú kritické nedostatky. Nezaručujú korektné spracovania poznatkov a ich adekvátne využitie pri riešení problémov a realizovaní požiadaviek

Navyše, údajové štruktúry v jazyku bez jasne definovanej sémantiky sa obvykle v priebehu implementácie upravujú tak, aby sa ľahšie zvládli prípady, s ktorými sa vopred nerátalo. Podobne sa upravujú procedúry, ktoré nad týmito štruktúrami pracujú. Stráca sa prehľad o vlastnostiach a fungovaní systému. Sotva možno pri takomto prístupe riešiť fundamentálne problémy typu čo je odvoditeľné, či je implementované odvodenie korektné (a v akom zmysle), či je úplné na nejakého kritéria atď. V takom prípade nie je jasné, či odpovede na otázky sú správne, či rovnakú odpoveď na tú istú otázku nad tou istou bázou znalostí zaručene dostaneme aj pri ďalšom výpočte a pod. Ad hoc modifikácie procedurálne definovanej inferencie zatemňujú a komplikujú návrh reprezentácie, výpočty nad ňou a – predovšetkým – jej sémantiku.

## 5 Záver

V posledných rokoch sa oblasťou intenzívneho výskumu stalo programovanie pomocou odpoveďových množín (ďalej uprednostníme anglický termín answer set programming). Pojem odpoveďovej množiny (answer set) je zovšeobecnením pojmu stabilného modelu [10, 4]. Ide o variant logického programovania, ktorého cieľom je riešiť problémy (a odpovedať na otázky) pomocou odpoveďovej množiny [11, 15]. Prvé implementácie pochádzajú z polovice deväťdesiatych rokov [7, 14]. Medzi hlavné aplikačné oblasti answer set programming patrí reprezentácia znalostí. Monografia [4] pojednáva o teoretických základoch i o aplikáciách na reprezentáciu znalostí.

Treba podotknúť, že z výpočtového hľadiska ide o ťažké problémy. Je prirodzené, že ťažké problémy, súvisiace s usudzovaním a reprezentáciou znalostí, vyžadujú takýto nástroj. Výskum v oblasti venuje veľkú pozornosť optimalizácii výpočtov a dosiahnuté výsledky sú sľubné: riešenia pre programy s desiatkami tisíc pravidiel sa dajú dosiahnuť v rozumnom čase. Pracuje sa na experimentálnych implementáciách z oblasti plánovania, reprezentácie akcií a zmien, diagnostikovania, právneho usudzovania, usudzovania o hierarchických vzťahoch, o typických prípadoch, výnimkách atď. V plnej verzii tohoto textu [19] je stručná informácia o riešení niektorých elementárnych problémov reprezentácie znalostí prostriedkami answer set programming a o vyjadrovacích možnostiach rôznych jazykov pre answer set programming.

Logické formalizmy majú vlastnosť, ktorá je mimoriadne dôležitá z hľadiska reprezentácie znalostí. Ide o to, že dovoľujú relatívne pohodlné adaptácie, modifikácie jazyka na reprezentáciu znalostí s cieľom zosilniť ich tak, aby zvládali ďalšie úlohy. Mnohé výrazné vlastnosti spred modifikácie ostávajú nedotknuté. Táto schopnosť sa nazýva v angličtine elaboration tolerance.

Zosumarizujme: Prístup k reprezentácii znalostí založený na logickom programovaní sa opiera o dobre definovanú sémantiku. Vďaka nej umožňuje jasne špecifikovať, čo je odvoditeľné. Usudzovanie sa stáva prirodzene súčasťou reprezentácie znalostí. Riešenie problémov je založené na dobre definovanom sémantickom konštrukte. Prístup dovoľuje presne charakterizovať vyjadrovaciu silu rôznych (pod)jazykov na reprezentáciu znalostí a charakterizxovať výpočtovú zložitosť programov v týchto jazykoch.

Informovanosť o výsledkoch výskumu v tejto oblasti môže byť užitočná aj pre odborníkov rozvíjajúcich a používajúcich klasický prístup k reprezentácii znalostí.

**Poďakovanie.** Autor ďakuje jednému z anonymných recenzentov za podrobné a cenné pripomienky.

## Literatúra

1. Allen, J. *Natural Language Understanding.* Benjamin/Cummings Publ.Comp.1987
2. Artificial Intelligence, vol. 13, No 1-2
3. Baral, C., Gelfond, M. *Logic Programming and Knowledge Representation.* Journal of Logic Programming, 1994
4. Baral, C. *Knowledge representation, reasoning and declarative problem solving.* Cambridge 2003
5. Doyle, J. *A truth maintenance system.* Artificial Intelligence 12:231-272, 1979
6. Fahlman, S. *NETL: A System for Representing and Using Real World Knowledge.* MIT Press, Cambridge, MA, 1979
7. Eiter, T. et al. *The dlv system.* In Minker (ed.), Pre-prints of workshop on Logic-based AI. 2000
8. Elkan, C. *A rational reconstruction of non-monotonic truth maintenance systems.* Artificial Intelligence, 43(2):219-234, 1990
9. Gelfond, M., Lifschitz, V. *The stable model semantics for logic programming.* Proc. ICLP 1988, 1070-1080
10. Gelfond, M., Lifschitz, V. *Logic programs with classical negation.* Proc. ICLP 1990, 579-597
11. Marek, W., Truszczynski, M. *Stable models and an alternative logic programming paradigm.* In The logic programming paradigm a 25-year perspective, 375-398, Springer, 1999
12. Mc Carthy, J. *Programs with common sense,* 1959, `http://www-formal.stanford.edu/jmc/mcc59.html`
13. Minski, M. *A Framework for Representing Knowledge.* MIT, (1974)
14. Niemelä, I., Simons, P. *Efficient implementation of the well-founded and stable model semantics.* Proc. ICLP 1996, 289-303
15. Niemelä, I. *Logic programs with stable models semantics as a constraint programming paradigm.* Annals of Matahematics and Artificial Intelligence, 1999, 241-271
16. Przymusinski, T. *Non-monotonic formalisms and logic programming.* Proc. of 6th Int. Conf. on Logic Programming. MIT Press 1989, 655-674
17. Šefránek, J. *Inteligencia ako výpočet.* IRIS Bratislava 2000
18. Šefránek, J. *A Kripkean semantics for dynamic logic programming.* LPAR 2000, Springer, 469-486
19. Šefránek, J. *Reprezentácia znalostí prostriedkami logického programovania* http://www.ii.fmph.uniba.sk/ sefranek/online/z2004b.ps