Plánovanie na symbolových modeloch sveta so stabilnomodelovou sémantikou

Michal Čertický

Fakulta matematiky, fyziky a informatiky, Univerzita Komenského
Bratislava, Slovensko
certicky@fmph.uniba.sk

Abstrakt

Typickým postupom pri logickom plánovaní je v posledných rokoch definovanie akcií pomocou konceptu fluentov, ktoré sú doménovo špecifické a neoddeliteľné od zvyšku plánovacieho problému. Zistili sme, že v prípade učiacich sa agentov a rastúcej množiny akcií je však výhodnejšie reprezentáciu akcií separovať. Predstavujeme teda nový plánovací formalizmus bez fluentov IK-STRIPS, ktorý je inšpirovaný klasickým STRIPSom (1971), avšak je kompatibilný s moderným spôsobom reprezentácie znalostí so stabilnomodelovou sémantikou a použiteľný v komplexných doménach s neúplnými znalosťami. Taktiež uvádzame výsledky porovnania nášho plánovača s bežne používanou alternatívou DLV^K.

1 Plánovanie a logické modely sveta

Jednou z aktuálnych výziev vrámci problematiky agentovo-orientovaného programovania a modelovania je nájdenie efektívneho plánovacieho mechanizmu pracujúceho so znalosťami reprezentovanými symbolovo, t.j. pomocou jedného z vhodných logických formalizmov. Pojem plánovania chápeme ako algoritmický proces nachádzania takých postupností akcií agenta, ktoré dovedú buď jeho, alebo okolitý svet z počiatočného stavu do formálne popísaného požadovaného stavu.

Kvôli redukcii kognitívneho procesu plánovania (resp. iného typu usudzovania) na abstraktný algoritmus zvyčajne máme naše znalosti o svete reprezentované simplifikovanými logickými konštrukciami – modelmi sveta. Kvôli svojej vysokej vyjdrovacej sile je trendom v oblasti symbolovej reprezentácie znalostí (vrátane plánovania) v posledných rokoch paradigma Answer Set Programmingu (ASP) [1,2], kde pracujeme s poznatkami vo forme tzv. rozšírených logických programov (ELP) – množín pravidiel, faktov a integritných obmedzení pozostávajúcich z literálov so syntaxou prvorádovej predikátovej logiky. Sémantika ASP je vybudovaná okolo konceptu stabilného modelu log. programu (množiny literálov extrahovanej predpísaným spôsobom z pôvodného ELP) a operátora stabilnomodelového vyplývania (umožňujúceho dedukciu na týchto poznatkoch). Model sveta teda môžeme chápať ako jeden takýto rozšírený logický program.

Akcie chápeme ako abstraktné matematické operátory, ktoré nám z modelov sveta vytvárajú nové. Plánovanie je teda redukovateľné na klasický problém prehľadávania orientovaného grafu, kde vrcholy predstavujú jednotlivé modely a hrany zasa všetky aplikovateľné akcie.

2 Klasický prístup

Existuje niekoľko rôznych formalizmov na reprezentáciu akcií a plánovacích problémov všeobecne, z ktorých každý má svoje výhody a nevýhody. Typickým predstaviteľom je napríklad jazyk K [2,3].

```
fluents:
on(B,L) requires block(B), location(L).
occupied(B) requires location(B).

actions:
move(B,L) requires block(B), location(L).

always:
inertial on(B,L).
caused occupied(B) if on(B1,B), block(B).
caused on(B,L) after move(B,L).
caused -on(B,L1) after move(B,L), on(B,L1), L <> L1.
executable move(B,L) if not occupied(B),
        not occupied(L), B <> L.

NoConcurrency.


initially:      on(a,table). on(b,table). on(c,a).
goal:           on(c,b), on(b,a), on(a,table)? (3)
```

Fig. 1. Príklad plánovacieho problému v jazyku K.

Predchádzajúci príklad predstavuje jednoduchý ilustračný plánovací problém *Blocksworld domain* [4], v ktorom je cieľom preusporiadať niekoľko kociek na stole z jednej počiatočnej konfigurácie (*initially*) do inej cieľovej (*goal*). Vidíme tu reprezentáciu jedinej akcie (*actions*) `move(B,L)`, ktorá presúva kocku B na lokáciu L. Táto

akcia má jednu podmienku vykonateľnosti (*executable*) a tri efekty (*caused*).

3 Nedostatky a riešenie

Problém jazyka K (a iných moderných plánovacích formalizmov) vyplýva z použitia tzv. fluentov. Fluenty (v predchádzajúcom príklade časť *fluents*) slúžia na reprezentáciu všetkých predikátov, ktoré sa v danej doméne môžu meniť buď časom alebo pôsobením jednotlivých akcií. Fluenty sú teda viazané na konkretnú doménu a plánovací problém. Vidíme tiež, že podmienky použiteľnosti a efekty našej akcie sú potom definované s použitím týchto fluentov, a teda sú tiež závislé na zvyšku domény.

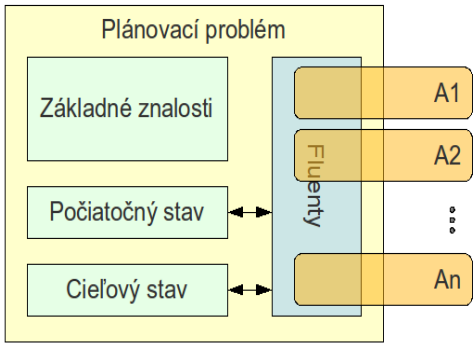

Fig. 2. Reprezentácia jednotlivých akcií s použitím fluentov v prípade jazyka K.

Problémom je, že v niektorých príkladoch je výhodné mať akcie separované a nezávislé na akejkoľvek konkrétnej doméne alebo probléme. Napríklad pri experimentoch s učiacimi sa agentami sme mali potrebu jednotlivé akcie separovať od zvyšku znalostí o svete, aby sme z nich mohli kombinovaním vytvárať nové zložitejšie akcie bez zásahu do reprezentácie ostatných druhov vedomostí. Ďalší problém nastáva, keď chceme počas fungovania agenta umelo pridávať nové akcie, ktoré by ovplyvňovali predikáty, ktoré zatiaľ nie sú fluentami. V typickom prípade nemáme možnosť a/alebo čas zložitým spôsobom revidovať a konvertovať celú bázu znalostí vždy keď sa má agent naučiť novú akciu.

Pre spomínané prípady je teda potrebný reprezentačný formalizmus, ktorý by jednotlivé akcie definoval separovane od zvyšku problému, teda bez použitia doménovo špecifických fluentov.

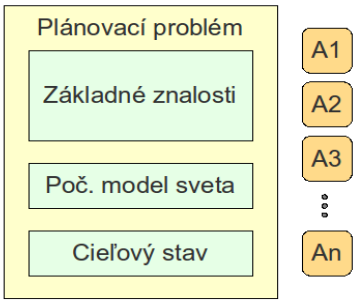

Fig. 3. Reprezentácia jednotlivých akcií bez použitia fluentov v prípade jazykov STRIPS a IK-STRIPS.

Príkladom formalizmu, ktorý reprezentoval akcie oddelene od zvyšku plánovacieho problému bol STRIPS [5,6] (Stanford Research Institute Problem Solver) z roku 1971. Namiesto fluentov ovplyvňovali akcie priamo modely sveta na ktoré boli aplikované. Podmienky vykonateľnosti boli reprezentované jednoduchou množinou literálov (resp. v jednoduchšom prípade atómov) a boli splnené vtedy, ak každý literál z nej bol obsiahnutý v danom modeli sveta. Efekty akcií boli definované ďalšími dvoma množinami (*add list* a *delete list*) a boli do daného modelu buď pridané alebo z neho odobrané.

Nedostatkom STRIPSu, ktorý ho robí nepoužiteľným pre plánovanie v reálnom svete je, že pracoval so zastaralou dvojhodnotovou predikátovou logikou. Tá je nedostatočná pre efektívne usudzovanie s neúplnými znalosťami (ktorým sa v dostatočne komplexných doménach nevyhneme). Vhodým spôsobom reprezentácie neúplných znalostí je už spomínané ASP so stabilnomodelovou sémantikou. STRIPS teda potreboval byť upravený a zdokonalený tak, aby bol kompatibilný s Answer Set Programmingom.

4 IK-STRIPS formalizmus

Jedným z výsledkov nášho výskumu je modifikácia STRIPS formalizmu použiteľná s neúplnými znalosťami reprezentovanými pomocou ASP nazvaná IK-STRIPS (Incomplete Knowledge STRIPS). Každá akcia v tomto formalizme pozostáva z troch hlavných častí:

1. Jej meno a parametre.
2. Dve množiny literálov – $preconditions^+$ a $preconditions^-$, ktoré opisujú pozitívne a negatívne podmienky vykonateľnosti akcie.
3. Dve množiny (*add list* a *delete list*) tzv. *effect descriptorov*, teda akýchsi nových konštruktov opisujúcich každý z efektov danej akcie.

Prvá časť – meno a parametre – zostáva oproti pôvodnému STRIPSu nezmenená. Meno je len jednoduchým identifikátorom akcie, ktorý sa neskôr využíva pri procese plánovania a parametre sú logické premenné, ktoré sú počas plánovania substituované konštantami z aktuálneho modelu sveta.

V druhej časti máme namiesto jednej množiny literálov dve. Akcia je potom splniteľná v danom modeli práve vtedy, ak z neho vyplýva každý literál z $preconditions^+$ a žiaden literál z $preconditions^-$. Toto rozdelenie nám umožňuje lepšie definovať podmienky v prípade že nemáme všetky informácie o momentálnom stave sveta. Ak by sme totiž v pôvodnom STRIPSe chceli povedzme vyjadriť, že akcia nie je vykonateľná ak platí literál a, tak by sme museli medzi podmienky zaradiť jeho negáciu $\neg a$. To ale znamená že akciu môžeme vykonať len vtedy ak disponujeme explicitnou informáciou o tom, že a neplatí. V IK-STRIPSE ale máme možnosť buď $\neg a$ zaradiť do preconditions+ (čo znamená to isté), alebo alternatívne zaradiť a do $preconditions^-$ (čo by znamenalo, že nám stačí nevedieť o tom, že by a platilo).

Treťou a poslednou časťou reprezentácie akcií sú množiny *add list* a *delete list*. Na rozdiel od pôvodného STRIPSu však tieto neobsahujú obyčajné literály, ale zložitejšie *effect descriptory*. Pre prácu s neúplnými znalosťami totiž nie je dostatočné mať efekty s pevne danými konštantnými parametrami (keďže nemôžme dopredu vedieť na aké objekty narazíme – nehovoriac o tom, že ich môže v komplexných doménach byť extrémne veľa). Potrebujeme teda definovať efekty pomocou premenných. Príjemným dôsledkom použitia premenných je aj to, že umožníme každému efektu ovplyvniť rôzne modely sveta rôznymi spôsobmi (v závislosti na aktuálnom stave sveta efekt môže byť trochu iný, keďže premenné sú substituované konštantami práve z daného modelu). Ďalšou novinkou je zavedenie individuálnych podmienok aplikovateľnosti pre každý efekt. Je len prirodzené, že jedna akcia má vo všeobecnosti viacero potenciálnych efektov, z ktorých sa v danej situácii niektoré aplikujú a iné nie. V pôvodnom STRIPSe sa toto riešilo pomocou rozdeľovania jednej akcie na viacero iných s mierne odlišnými podmienkami vykonateľnosti, čo je nie len neintuitívne a zbytočne sa odkláňajúce od prirodzeného jazyka, ale mohlo to potenciálne aj spomaliť proces plánovania kvôli väčšiemu počtu akcií.

Effect descriptory sa teda skladajú z nasledujúcich častí:

1. *Modifikátor*, čiže literál s premennými, ktoré sa počas plánovania substituujú za konštanty z aktuálneho modelu sveta.
2. Dve množiny literálov reprezentujúce pozitívne a negatívne podmienky aplikovateľnosti efektu.

Proces substituovania premenných v effect descriptoroch počas plánovania voláme inštanciáciou daného descriptoru. Premenné sa postupne nahrádzajú rôznymi kombináciami konštánt z modelu a vždy sa s nimi skontrolujú podmienky aplikovateľnosti. Ak sú podmienky pre danú substitúciu splnené (všetky literály z pozitívnych vyplývajú a žiadne z negatívnych nevyplývajú z modelu sveta), tak pomocou tejto substitúcie vytvátame tzv. inštanciu efektu (*effect instance*). Effect descriptor je teda aplikovateľný na daný model sveta, ak je v ňom množina jeho inštancií neprázdna.

Proces modifikácie modelu sveta M akciou A je teda počas plánovania nasledovný: Najskôr zistíme, či je A v modeli M vykonateľná tak, že skontrolujeme jej pozitívne a negatívne podmienky. Ak sú všetky podmienky splnené, tak sa pokúsime vygenerovať všetky inštancie jej effect descriptorov. Nakoniec pridáme k modelu M modifikátory všetkých inštancií descriptorov z add listu A a odoberieme z neho modifikátory inštancií descriptorov z delete listu.

Okrem reprezentácie akcií sa IK-STRIPS od svojho predchodcu mierne líši aj v definícii cieľov. Popis *cieľa* všeobecne hovorí o tom, čo má byť v danom modeli splnené a čo nie. Definícia má podobu, tak ako v predchádzajúcich prípadoch, dvojice množín. To, či je daný model sveta cieľovým, sa rozhodne podľa nich. *Cieľovým stavom* sveta teda nazývame akýkoľvek stav, v ktorého modeli je splnený aktuálny cieľ.

Výstupom procesu plánovania je potom *plán* – teda postupnosť akcií s príslušnými parametrami, ktorá po postupnej aplikácii na iniciálny model sveta vyprodukuje model jedného z cieľových stavov. Plán je navyše *korektný*, ak po aplikovaní každej jeho akcie je tá nasledujúca vykonateľná.

5 Plánovač a výsledky

Ďalším výsledkom nášho výskumu je experimentálna implementácia plánovača používajúceho IK-STRIPS ako formalizmus na reprezentáciu plánovacieho problému.

Vybraným programovacím jazykom bol Python a samotné plánovanie bolo redukované (ako už bolo naznačené v úvode) na problém prehľadávania orientovaného grafu. Zvoleným algoritmom bola mierna optimalizácia A* pathfindingu s jednoduchými heuristikami ohodnocujúcimi počet splnených a nesplnených cieľových podmienok.

Výsledky nášho plánovača sme porovnávali s často používanou alternatívou DLV^K , ktorá využíva ako svoj formalizmus spomínaný jazyk K. Riešenou úlohou boli postupne vždy zložitejšie inštancie Blocksworld domain problému. Ako máme možnosť vidieť na nasledujúcom grafe, IK-STRIPS plánovač je podstatne rýchlejší v prípade typických problémov s najkratšou dĺžkou plánu väčšou ako 10. Okrem tejto rýchlostnej prevahy disponuje v porovnní s DLV^K aj ďalšou výhodou – nepotrebuje na vstupe dostať očakávanú dĺžku plánu (ktorá je samozrejme pri bežnom využití vopred neznáma).

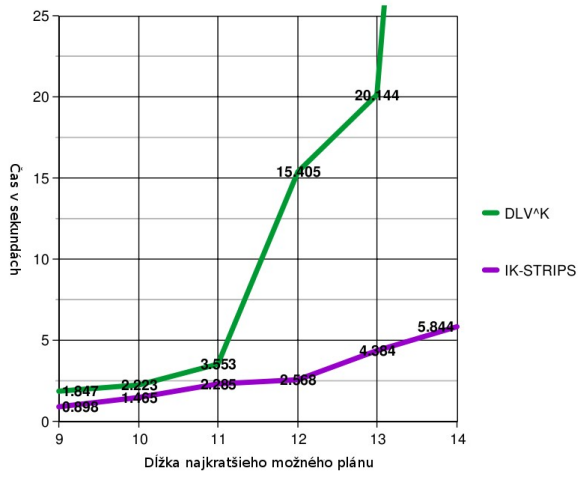

Fig. 4. Porovnanie výpočtových časov plánovača DLV^K a IK-STRIPS na rôznych inštanciách Blocksworld domain problému.

Literatúra

[1] V. Lifschitz: *Answer Set Programming and Plan Generation*, Artificial Intelligence vol. 138, 2002.

[2] T. Eiter, W. Faber, N. Leone, G. Pfeifer, A. Polleres: *Planning under Incomplete Knowledge*: Lecture Notes in Computer Science 2000, Springer Berlin / Heidelberg: 2000.

[3] A. Polleres: *The DLVK System for Planning with Incomplete Knowledge*. Masters thesis, Vienna University of Technology, Austria, 2001.

[4] J. Slaney, S. Thiebaux, *Blocks World revisited,* Artificial Intelligence 125: 119-153, 2001.

[5] R. E. Fikes, N. J. Nilsson, *STRIPS: A new approach to the application of theorem proving to problem solving*. Artificial Intelligence 2 : 1971.

[6] C. A. Cocosco, *A Review of "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving by R.E. Fikes, N.J. Nillson, 1971"*.