Katedra aplikovanej informatiky
Fakulta matematiky, fyziky a informatiky
Univerzita Komenského v Bratislave

Martin Baláž

Kripkeho štruktúry nad dvojzväzmi

Minimová práca

Školiteľ: Doc. PhDr. Ján Šefránek, CSc.

Bratislava December 2004

Obsah

Časť I

Kapitola 1

Úvod

Pozadie

Logické programy sú silným nástrojom na reprezentáciu znalostí človeka. Defaultová negácia umožňuje vyjadriť mnohé znalosti o svete, ktoré nie sú ľahko vyjadriteľné v klasickej logike. Nemonotónnosť negácie je však jedným z hlavných dôvodov, prečo vznikli rôzne sémantiky pre logické programy. Význam defaultovej negácie je založený na predpoklade uzavretého sveta. Pokiaľ nemáme evidenciu o platnosti niektorého faktu, predpokladáme, že neplatí. Nie vždy sa nám to hodí, často potrebujeme vyjadriť aj to, že niečo priamo neplatí. Preto bol zavedný druhý typ negácie - explicitná negácia.

Prvé sémantiky pre logické programy boli klasické dvojhodnotové. Gelfond a Lifschitz [11] zaviedli stabilnú sémantiku, ktorú neskôr rozšírili na logické programy s explicitnou negáciou [12, 13]. Przymusinski zaviedol trojhodnotové alebo čiastočné stabilné modely [23], ktoré môžu obsahovať neúplú informáciu. Ukázal, že každý normálny logický program má aspoň jeden čiastočne stabilný model a dobre založený model je najmenší z nich [22]. Nakoniec boli navrhnuté štvorhodnotové logiky, ktoré môžu obsahovať nekonzistencie [4]. Všetky tieto prístupy majú veľa spoločného.

M. Ginsberg zaviedol elegentný pojem dvojzväz, ktorý je prirodzeným rozšírením dvoj, troj a štvorhodnotovej logiky [14]. Belnapova logika je najjednoduchším netriviálnym príkladom dvojzväzu. Obsahuje všetky podmnožiny množiny pravdivostných hodnôt t a f. Jednoprvková množina $\{t\}$ predstavuje pravdivú hodnotu, $\{f\}$ nepravdivú. Neúplná informácia je reprezentovaná ako prázdna množina a nekonzistentná informácia ako množina $\{t, f\}$. M. Fitting zovšeobecnil Gelfond-Lifschitzov operátor pre dvojzväzy [7, 10, 8]. Ukázal, že dvojhodnotová a trojhodnotová stabilná sémantika je špeciálnym prípadom zovšeobecneného prístupu.

Jedna zo základných čŕt uvažovania človeka je narábanie s dynamickými poznatkami, ktoré prichádzajú v rôznom čase a z rôznych zdrojov. Preto by bolo vhodné rozšíriť logické programy tak, aby boli schopné tieto zmeny zachytávať. Množina modelov logického programu obsahuje menej informácie ako samotný program. Modely neobsahujú závislosti medzi literálmi, preto je lepšie hovoriť o zmene logického programu ako o zmene interpretácie sveta [3, 19]. Multidimenzionálny logický program pozostáva z množiny logických programov a relácie preferencie. Preferovanejšie logické programy menia menej preferované. Všetky

sémantiky zavedené v [20, 5, 18] sú založené na princípe kauzálneho zamietania. Ak majú dve pravidlá opačné hlavy a splnené telo, uprednostňujeme pravidlo z preferovanejšieho programu. Stabilný model sa potom overuje vzhľadom na pravidlá, ktoré neboli zamietnuté.

Problémy

Ako Fitting ukázal vo svojich prácach, existencia veľkého počtu stabilných sémantík nie je nevyhnutná. Nakoľko majú veľa spoločného, dajú sa charakterizovať pomocou spoločného stabilného operátora. Rozšírenie, ktoré navrhol, nepokrýva však dostatočne veľkú triedu logických programov. Neumožňuje charakterizovať stabilné modely pre logické programy s explicitnou negáciou. Navyše stabilný operátor je zadefinovaný iba pre normálne logické programy, ktoré neobsahujú pravidlá s defaultovou negáciu v hlavách.

Myšlienka zmeny logických programov sa dá aplikovať nielen na dvojhodnotové logiky. Princíp kauzálneho zamietania sa dá rozšíriť aj na viachodnotové logické programy. Podobne ako v statickom prípade, zovšeobecnený prístup zjednocuje viacero sémantík, ktoré sú jeho špeciálnym prípadom.

Stabilné modely pre multidimenzionálny logický program zdieľajú rovnaký problém so stabilnými modelmi pre statický prípad. Neobsahujú závislosti medzi jednotlivými literálmi. Pokiaľ budujeme sémantiku na princípe kauzálneho zamietania, tak množina modelov nám postačuje. Vieme rozhodnúť o zamietnutí pravidla iba na základe informácií obsiahnutých v modeli. Pokiaľ by sme sa chceli rozhodovať aj na základe závislosti medzi literálmi, tak nám množina stabilných modelov nepostačuje. Medzi pravidlami $a \leftarrow b$ a $b \leftarrow \sim a$ neexistuje konflikt v zmysle princípu kauzálneho zamietania. Napriek tomu by sme cyklickú závislosť literálu a od opačného literálu $\sim a$ vedeli vyriešiť zamietnutím menej preferovaného pravidla.

Riešenie

Defaultová negácia v hlavách pravidiel zohráva v multidimenzionálnych logických programoch dôležitú úlohu. Umožňuje zamietať pravidlá z menej preferovaných programov. Explicitná negácia zase zvyšuje vyjadrovaciu silu jazyka. Poskytuje nástroje na explicitné vyjadrenie nepravdy bez predpokladu uzavretého sveta. Preto rozšírime stabilný operátor na triedu zovšeobecnených logických programov s explicitnou negáciou.

Kripkeho štruktúry [24] poskytujú sémantickú štruktúru obsahujúcu závislosti medzi literálmi. Môžeme sa na ne pozerať ako na orientovaný graf. Vrcholy tvoria čiastočné interpretácie. Hrana existuje medzi dvomi interpretáciami, ak sa vieme dostať aplikáciou niektorého pravidla z prvej interpretácie do druhej. Výpočet stabilného modelu prislúcha ceste zakorenenej v množine defaultov a terminujúcej v úplnej interpretácii. Pokiaľ by sme chceli v prípade konfliktu aplikovať princíp kauzálneho zamietania, preferovali by sme pravidlo z vyššieho programu bez ohľadu na to, akou cestou sme nekonzistentnú interpretáciu dosiahli. Sémantiky založené na závislostiach medzi literálmi môžu zobrať do úvahy nielen preferenciu pravidiel, ale aj cesty vedúce do interpretácie a počiatočné množiny defaultov [25].

Zadefinujeme Kripkeho štruktúry asociované s viachodnotovým logickým programom rozšíreným o explicitnú negáciu. Predstavíme existujúce dvojhodno-

tové sémantiky multidimenzionálneho logického programu založené na princípe kauzálneho zamietania. Zovšeobecnenie týchto sémantík pre viachodnotový prípad, definícia Kripkeho štruktúry asociovanej s multidimenzionálnym logickým programom a sémantiky založené na závislostiach medzi literálmi sú mimo rozsahu tejto práce, avšak neskôr uvedené výsledky k týmto oblastiam smerujú.

Prehľad

V druhej kapitole zadefinujeme jazyk pre viachodnotové logické programy s explicitnou negáciou. Ukážeme, aké formuly môže obsahovať logický program. V tretej kapitole predstavíme algebraické štruktúry zväz a dvojzväz ako priestor pravdivostných hodnôt. Potom zadefinujeme interpretácie resp. aproximácie, pomocou ktorých budeme priraďovať formulám jazyka ich pravdivostnú hodnotu. V piatej kapitole rozšírime stabilný operátor na triedu zovšeobecnených logických programov s explicitnou negáciou, pomocou ktorého neskôr vybudujeme Kripkeho štruktúru. Ukážeme, ako v nej môžeme identifikovať výpočet stabilných modelov. Nakoniec spravíme prehľad existujúcich sémantík multidimenzionálnych logických programov založených na princípe kauzálneho zamietania.

Kapitola 2

Syntax

V tejto kapitole predstavíme syntax logických programov [21]. Rozšírime prvorádový jazyk tak, aby sme ho mohli použiť pre viachodnotové logické programy. Zavedieme logické konštanty, ktoré nám umožnia narábať aj s inými pravdivostnými hodnotami ako sú pravda alebo nepravda.

V praxi sa často ukazuje, že je užitočné rozlišovať dva typy negácie - defaultovú a explicitnú. Defaultová negácia je pravdivá vtedy, pokiaľ nemáme dôvod veriť jej opaku. Naopak platnosť explicitnej negácie musí byť priamo podopretá. Rozšírenie jazyka o druhý typ negácie zväčšuje jeho vyjadrovaciu silu.

2.1 Jazyk

Prvorádový jazyk pozostáva z formúl skonštruovaných nad abecedou symbolov. Najprv zadefinujeme abecedu rozšírenú o logické konštanty a druhý typ negácie. Potom ukážeme, ako vyzerajú formuly rozšíreného prvorádového jazyka.

Definícia 2.1. *Abeceda* $\mathcal{A}$ obsahuje premenné, konštanty, funkčné a predikátové symboly, logické konštanty, unárne $\{not, \neg\}$ a binárne $\{\wedge, \vee, \rightarrow\}$ logické spojky, kvantifikátory $\{\forall, \exists\}$ a interpunkčné symboly.

Každému funkčnému a predikátovému symbolu je priradené nenulové prirodzené číslo určujúce jeho aritu. Na konštanty sa môžeme pozerať ako na funkčné symboly s aritou 0, na logické konštanty ako na nulárne logické spojky. Symbol *not* označuje defaultovú negáciu, symbol $\neg$ explicitnú negáciu. Konjunkciu označujeme symbolom $\wedge$, disjunkciu $\vee$ a implikáciu $\rightarrow$. Symbol $\forall$ nazývame všeobecný kvantifikátor, $\exists$ existenčný kvantifikátor. Interpunkčné symboly sú zátvorky (,) a čiarka. Logické spojky a kvantifikátory nazývame *logické symboly*. Interpunkčné symboly spolu s logickými symbolmi okrem logických konštánt tvoria pevnú časť každej abecedy. Množiny premenných, konštánt a logických konštánt, funkčných a predikátových symbolov môžu byť pre každú abecedu iné.

Definícia 2.2. *Term* je definovaný pomocou indukcie nasledovne:

- Premenná alebo konštanta je term.
- Ak f je funkčný symbol arity n a $t_1, \ldots, t_n$ sú termy, potom $f(t_1, \ldots, t_n)$ je term.

Definícia 2.3. Ak p je predikátový symbol arity n a $t_1, \ldots, t_n$ sú termy, potom $p(t_1, \ldots, t_n)$ je *atóm*. *Objektívny literál* je atóm alebo explicitná negácia atómu. *Defaultový literál* je defaultová negácia objektívneho literálu. *Literál* je objektívny alebo defaultový literál.

Definícia 2.4. *Formula* je definovaná pomocou indukcie nasledovne:

- Objektívny literál je formula.
- Logická konštanta je formula.
- Ak F je formula, potom $(not\ F)$ je formula.
- Ak F a G sú formuly, potom $(F \wedge G)$, $(F \vee G)$, $(F \rightarrow G)$ sú formuly.
- Ak F je formula a x je premenná, potom $(\forall x F)$, $(\exists x F)$ sú formuly.

Aby sme predišli používaniu priveľa zátvoriek, budeme predpokladať hierarchiu logických spojok s najvačšou preferenciou navrchu:

$$not, \forall, \exists$$
$$\vee$$
$$\wedge$$
$$\rightarrow$$

Definícia 2.5. *Rozšírený prvorádový jazyk* $\mathcal{L}$ nad abecedou $\mathcal{A}$ je množina formúl vytvorených zo symbolov abecedy $\mathcal{A}$.

Príklad 2.1. $(\forall x(\exists y((\neg p(x, y)) \wedge \frac{1}{4} \rightarrow (not\ q(x)))))$ je formula, ktorá obsahuje defaultovú negáciu $\sim$ aj explicitnú negáciu $\neg$. Takisto obsahuje logickú konštantu $\frac{1}{4}$. Keď odstránime zátvorky, ktoré nespôsobia nejednoznačnosť, a použitím predchádzajúcej preferencie logických spojok môžeme zjednodušiť formulu na $\forall x \exists y(\neg p(x, y) \wedge \frac{1}{4} \rightarrow not\ q(x))$.

2.2 Logický program

Definícia 2.6. *Dosah* kvantifikátora $\forall x$ (resp. $\exists x$) vo formule $(\forall x)F$ (resp. $(\exists x)F$) je F. *Viazaný výskyt* premennej vo formule je výskyt hneď za kvantifikátorom alebo výskyt v dosahu kvantifikátora, ktorý má rovnakú premennú hneď za kvantifikátorom. Každý iný výskyt premennej je *voľný*.

Definícia 2.7. *Uzavretá formula* je formula bez voľných výskytov ľubovolnej premennej.

Definícia 2.8. *Klauza* je uzavretá formula tvaru

$$\forall x_1 \ldots \forall x_s(L_1 \wedge \cdots \wedge L_m \rightarrow L_{m+1} \vee \cdots \vee L_n)$$

kde L_i sú literály alebo nulárne spojky a $x_1, \ldots, x_s$ sú všetky premenné, ktoré sa vyskytujú v $L_1, \ldots, L_n$.

Klauzy sú typické pre logické programovanie a používa sa pre ne špeciálna notácia

$$L_{m+1}, \ldots, L_n \leftarrow L_1, \ldots, L_m$$

Predpokladá sa, že všetky premenné sú univerzálne kvantifikované. Čiarky v $L_1, \ldots, L_m$ prestavujú konjunkciu, v $L_{m+1}, \ldots, L_n$ disjunkciu. Namiesto klasickej implikácie $\rightarrow$ sa používa jej obrátená verzia $\leftarrow$.

Definícia 2.9. *Pravidlo* je klauza tvaru $L \leftarrow L_1, \ldots, L_n$, kde L je literál. L nazývame *hlava* a $L_1, \ldots, L_n$ *telo* pravidla.

Definícia 2.10. *(Zovšeobecnený) logický program* je konečná množina pravidiel. *Definitný logický program* obsahuje pravidlá bez defaultových literálov. *Normálny logický program* neobsahuje pravidlá s defaultovým literálom v hlave.

Logické programy obsahujúce druhý typ negácie sa zvyknú označovať ako rozšírené logické programy. Nakoľko v tejto práci budeme pracovať iba s rozšírenými logickými programami, budeme tento prívlastok vynechávať.

Príklad 2.2. Pravidlo programu $P = \{flu(X) \leftarrow 0.75, fever(X), cough(X)\}$ hovorí, že ak má pacient horúčku a kašeľ, tak diagnóza chrípky je správna v 75% prípadov.

Kapitola 3

Pravdivostné hodnoty

Dvojhodnotové logiky narábajú iba s pravdivostnými hodnotami 0 a 1. Ich vzťahy sú celkom jednoduché. Hodnota 0 znamená nepravda, hodnota 1 pravda. 0 a 1 sú navzájom opačné hodnoty, 0 je pravdivostne menšia ako 1. Ako by malo vyzerať pravdivostné usporiadanie pre ľubovolnú množinu pravdivostných hodnôt? Aké vlastnosti by mal mať operátor negácie priraďujúci opačné hodnoty? V nasledujúcej časti odpovieme na tieto otázky.

3.1 Zväzy

Pre priestor pravdivostných hodnôt sa zvyčajne používajú lineárne usporiadané množiny, vo všeobecnosti nám stačí čiastočné usporiadanie. To znamená, že nie všetky prvky musia byť porovnateľné.

Definícia 3.1. *Čiastočne usporiadaná množina* $(L, \leq)$ je množina L s binárnou reláciou $\leq$ na L, ktorá spĺňa nasledujúce vlastnosti

- $x \leq x$ (reflexivita)
- $x \leq y, y \leq x \Rightarrow x = y$ (antisymetria)
- $x \leq y, y \leq z \Rightarrow x \leq z$ (tranzitivita)

Lineárne usporiadaná množina navyše spĺňa vlastnosť

- $x \leq y$ alebo $y \leq x$ (linearita)

Nech K je neprázdna podmnožina čiastočne usporiadanej množiny $(L, \leq)$, $x \in L$. Potom x je *horné ohraničenie* K, ak $y \leq x$ pre všetky $y \in K$. Horné ohraničenie x množiny K je *minimálne horné ohraničenie* K, ak neexistuje horné ohraničenie y množiny K rôzne od x také, že $y \leq x$. Horné ohraničenie x množiny K je *najmenšie horné ohraničenie* K (alebo *suprémum* K), ak pre každé horné ohraničenie y množiny K platí $x \leq y$ (značíme $\bigvee K$). Suprémum dvoch prvkov x, y budeme označovať $x \vee y$. Podobne x je *dolné ohraničenie* K, ak $x \leq y$ pre všetky $y \in K$. Dolné ohraničenie x množiny K je *maximálne dolné ohraničenie* K, ak neexistuje dolné ohraničenie y množiny K rôzne od x také, že $x \leq y$. Dolné ohraničenie x množiny K je *najväčšie dolné ohraničenie* K (alebo

infímum K), ak pre každé dolné ohraničenie y množiny K platí $y \leq x$ (značíme $\bigwedge S$). Infímum dvoch prvkov x, y budeme označovať $x \wedge y$.

Suprémum a infímum budeme označovať rovnakým symbolom ako konjunkciu a disjunkciu. O ktorý symbol pôjde bude jasné z kontextu.

Neformálny význam pravidla je, že hodnota hlavy má byť väčšia alebo rovná ako hodnota tela. Pokiaľ zoberieme všetky pravidlá s rovnakou hlavou, potom hodnota hlavy by mala byť väčšia alebo rovná ako minimále horné ohraničenie hodnôt tiel pravidiel. Vo všeobecnosti takýchto minimálnych ohraničení môže byť viac. Kvôli jednoduchosti výpočtov budeme požadovať práve jedno najmenšie horné (resp. najväčšie dolné) ohraničenie.

Definícia 3.2. Čiastočne usporiadaná množina $(L, \leq)$ je *zväz*, ak pre ľubovolné $x, y \in L$ existuje $x \vee y$ a $x \wedge y$.

Zväz nám zaručuje existenciu supréma a infíma iba pre konečné množiny. Zoberme si napríklad reálne čísla z otvoreného intervalu $(0, 1)$. Pre každú konečnú podmnožinu existuje suprémum a infímum. Napriek tomu podmnožina $\{1 - \frac{1}{n}\}_{n=2}^{\infty}$ nemá suprémum a podmnožina $\{\frac{1}{n}\}_{n=2}^{\infty}$ nemá infímum v intervale $(0, 1)$.

Veta 3.1. *Čiastočne usporiadaná množina $(L, \leq)$ je zväz práve vtedy, keď suprémum a infímum existujú pre každú neprázdnu konečnú podmnožinu L.*

Dôkaz. Ak suprémum a infímum existujú pre každú neprázdnu konečnú podmnožinu, potom existujú aj pre dvojprvkovú.

Nech K je neprázdna podmnožina zväzu L. Ak $K = \{a\}$, potom na základe reflexivity $\leq$ a definície supréma a infíma platí $\bigvee\{a\} = \bigwedge\{a\} = a$.

Nech $K = \{a, b, c\}, d = a \vee b, e = d \vee c$. Treba ukázať, že $e = \bigvee\{a, b, c\}$. Platí, že $a \leq d, b \leq d$ a $d \leq e, c \leq e$. Vďaka tranzitivite platí aj $a \leq e, b \leq e$. Navyše ak f je horné ohraničenie $\{a, b, c\}$, tak $a \leq f, b \leq f, c \leq f$. Preto aj $d \leq f$ a $e \leq f$. Teda e je suprémum K.

Ak $K = \{a_0, a_1, \ldots, a_n\}, n \leq 0$, tak pomocou indukcie a predchádzajúcich vzťahov sa dá ukázať, že $(\ldots(a_0 \vee a_1) \cdots \vee a_n)$ je suprémum K.

Podobne sa dá dokázať tvrdenie pre infímum. □

Definícia 3.3. Zväz $(L, \leq)$ nazývame *úplný*, ak suprémum a infímum existujú pre ľubovolnú neprázdnu podmnožinu L.

Dôsledkom úplnosti je aj to, že množina L má najmenší a najväčší prvok. Najmenší prvok budeme označovať $\bot$, najväčší $\top$. Naviac zadefinujeme pre prázdnu množinu $\bigvee \emptyset = \top$ a $\bigwedge \emptyset = \bot$.

Infímum budeme používať ako význam pre konjunkciu, suprémum pre disjunkciu. Chýba nám však operátor pre negáciu. Nie všetky zväzy majú vhodnú symetriu, aby sa na nich dal zadefinovať operátor negácie, ktorý priraďuje prvkom ich opačný význam.

Definícia 3.4. Zväz L má *operátor negácie*, ak existuje zobrazenie $\sim\colon L \mapsto L$ také, že

- $\sim\sim x = x$
- $x \leq y \Rightarrow \sim y \leq \sim x$

Veta 3.2. *Pre úplný zväz s operátorom negácie platí $\sim \top = \bot$, $\sim \bot = \top$.*

Dôkaz. Nech existuje $x \in L$ také, že $\sim \top = x \neq \bot$. Keďže $\bot \leq x$, potom aj $\top = \sim x \leq \sim \bot$. $x \neq \bot$, teda ani $\top \neq \sim \bot$, čo je v spore s predpokladom, že $\top$ je najväčší prvok L. Podobne pre $\sim \bot = \top$. □

Veta 3.3. *Pre zväz s operátorom negácie platí* $\sim (x \wedge y) = \sim x \vee \sim y$ *a* $\sim (x \vee y) = \sim x \wedge \sim y$.

Dôkaz. Nech $a = x \wedge y$. Potom $a \leq x$ a $a \leq y$ a preto aj $\sim x \leq \sim a$ a $\sim y \leq \sim a$. Nech $\sim x \leq \sim b, \sim y \leq \sim b, \sim a \not\leq \sim b$. Potom $b \leq x, b \leq y$ a teda $b \leq a$. Preto $\sim a \leq \sim b$, čo je v spore s predpokladom. Takže $\sim a = \sim x \vee \sim y$.

Podobne pre $\sim (x \vee y) = \sim x \wedge \sim y$. □

Príklad 3.1. Dvojprvková množina $\{0, 1\}$ s usporiadaním $0 \leq 1$ je zväz. Nakoľko je to konečná množina, je to zároveň aj úplný zväz. 0 je najmenší prvok, 1 je najväčší prvok. Má jediný operátor negácie $\sim 0 = 1, \sim 1 = 0$.

Otvorený interval $(0, 1)$ s usporiadaním na reálnych číslach je tiež zväz. Ako už bolo spomenuté, nie je úplný. Nemá najmenší ani najväčší prvok. Má však nekonečne veľa operátorov negácie, napríklad $x \mapsto 1-x$, $x \mapsto 1-\sqrt{1-(x-1)^2}$.

Množina funkcií, ktoré priraďujú prvkom ľubovolnej množiny pravdivostné hodnoty zo zväzu, tvorí tiež zväz. Rovnako je zachovaná vlastnosť úplnosti a dá sa priamočiaro rozšíriť operátor negácie. Linearita však nemusí byť zachovaná, ak je pôvodné usporiadanie lineárne, v množine funkcií vo všeobecnosti nie je.

Definícia 3.5. Nech $(L, \leq)$ je zväz, S je neprázdna množina. Nech L^S je množina všetkých zobrazení z S do L a $f, g \in L^S$. Potom

- $f \leq g$, ak $f(s) \leq g(s)$
- ak L má operátor negácie $\sim$, tak určuje operátor $(\sim f)(s) = \sim f(s)$

pre všetky $s \in S$.

Lema 3.1. *Nech* $(L, \leq)$ *je zväz,* S *je neprázdna množina. Nech* L^S *je množina všetkých zobrazení z* S *do* L *a* $f, g \in L^S$. *Potom*

- $(f \wedge g)(s) = f(s) \wedge g(s)$
- $(f \vee g)(s) = f(s) \vee g(s)$

pre všetky $s \in S$.

Veta 3.4. *Nech* $(L, \leq)$ *je zväz,* S *je neprázdna množina. Nech* L^S *je množina všetkých zobrazení z* S *do* L. *Potom*

1. *Potom* L^S *je zväz.*
2. *Ak* L *je úplný zväz, potom aj* L^S *je úplný zväz.*
3. *Ak* L *má operátor negácie, tak určuje operátor negácie pre* L^S.

Dôkaz. Všetky tvrdenia vyplývajú z lemy 3.1 alebo priamo z definícií. □

Úplné zväzy s operátorom negácie sú algebraické štruktúry, ktoré budú vyhovovať našim požiadavkám. Obsahujú pravdivostné usporiadanie, ľubovolná podmnožina má infímum a suprémum. Infímum použijeme pre význam konjunkcie, suprémum pre disjunkciu a operátor negácie bude určovať význam defaultovej negácie. Množiny funkcií, ktoré priraďujú pravdivostné hodnoty z úplného zväzu, zdieľajú všetky spomenuté vlastnosti.

3.2 Dvojzväzy

Pre každý objektívny literál L sa môžu v logickom programe vyskytovať dva druhy pravidiel - pozitívne pravidlá s objektívnym literálom L v hlave alebo negatívne pravidlá s defaultovým literálom $\sim L$ v hlave. Hodnota literálu L musí byť teda väčšia alebo rovná ako suprémum hodnôt tiel pozitívnych pravidiel. Na druhej strane hodnota literálu $\sim L$ musí byť väčšia alebo rovná ako suprémum hodnôt tiel negatívnych pravidiel. Vďaka vlastnostiam operátora negácie hodnota literálu L musí byť menšia alebo rovná ako infímum negácie hodnôt tiel negatívnych pravidiel. Pozitívne pravidlá nám určujú spodnú hranicu a negatívne hornú hranicu. Literál L môže potom nadobúdať hodnoty z intervalu $(x, y) = \{z \mid x \leq z, z \leq y\}$. Ak hranica x alebo y klesá, klesá aj stupeň pravdivosi prípustných hodnôt. Čím bližšie sú hranice x a y k sebe, tým máme väčší stupeň informácie a menej prípustných hodnôt. Takto je určené pre intervaly pravdivostné a informačné usporiadanie. Teraz zadefinujeme algebraickú štruktúru, pre ktorú bude množina intervalov špeciálnym prípadom.

Definícia 3.6. Trojicu $(B, \leq_t, \leq_i)$ nazývame *dvojzväz*, ak $(B, \leq_t)$ a $(B, \leq_i)$ sú zväzy. Usporiadanie $\leq_t$ nazývame *pravdivostné* a usporiadanie $\leq_i$ *informačné*. $(B, \leq_t, \leq_i)$ je *úplný dvojzväz*, ak $(B, \leq_t)$ a $(B, \leq_i)$ sú úplné zväzy.

Každý zo zväzov $(B, \leq_t)$ a $(B, \leq_i)$ má svoje infímum $\vee_t, \vee_i$ a suprémum $\wedge_t, \wedge_i$. Ak sú to úplné zväzy, majú takisto najväčší prvok $\top_t, \top_i$ a najmenší prvok $\bot_t, \bot_i$.

Ak z môže nadobúdať hodnoty z intervalu (x, y), potom $\sim z$ nadobúda hodnoty z intervalu $(\sim y, \sim x)$. Takýto typ operátora zamieňa pravdivostné usporiadanie a zachováva informačné.

Definícia 3.7. Dvojzväz $(B, \leq_t, \leq_i)$ má *operátor negácie*, ak existuje zobrazenie $\sim: B \mapsto B$ také, že

- $\sim\sim x = x$
- $x \leq_t y \Rightarrow \sim y \leq_t \sim x$
- $x \leq_i y \Rightarrow \sim x \leq_i \sim y$

Niekedy je zaujímavé sledovať, ako sa zmení množina prípustných hodnôt (x, y), ak zameníme hranice na (y, x). Pokiaľ $x = y$, tak $(x, y) = (y, x)$ obsahuje iba jeden prvok. Ak $x \leq y$, tak (x, y) je neprázdna množina a (y, x) je prázdna. Ak $y \leq x$, tak (x, y) je prázdna množina a (y, x) je neprázdna. Ak x a y sú neporovnateľné, potom (x, y) a (y, x) sú prázdne množiny. Tento typ operátora zamieňa informačné usporiadanie a zachováva pravdivostné.

Definícia 3.8. Dvojzväz $(B, \leq_t, \leq_i)$ má operátor konflácie, ak existuje zobrazenie $-: B \mapsto B$ také, že

- $--x = x$
- $x \leq_t y \Rightarrow -x \leq_t -y$
- $x \leq_i y \Rightarrow -y \leq_i -x$

Definícia 3.9. Nech $(B, \leq_t, \leq_i)$ je dvojzväz s operátorom konflácie $-$. Prvok $x \in B$ je konzistentný, ak $x \leq_i -x$. Prvok $x \in B$ je exaktný, ak $x = -x$.

Teraz môžeme zadefinovať dvojzväz intervalov nad zväzom L. Ďalšie dve tvrdenia hovoria o tom, že definícia je korektná. L^2 zachováva úplnosť. Ak má L operátor negácie, potom aj L^2 má operátor negácie. Operátor konflácie nie je závislý od L.

Definícia 3.10. Nech $(L, \leq)$ je zväz. Potom

- $(x_1, y_1) \leq_t (x_2, y_2) \Leftrightarrow x_1 \leq x_2, y_1 \leq y_2$
- $(x_1, y_1) \leq_i (x_2, y_2) \Leftrightarrow x_1 \leq x_2, y_2 \leq y_1$
- ak L má operátor negácie $\sim$, tak určuje operátor $\sim (x, y) = (\sim y, \sim x)$
- $-(x, y) = (y, x)$

Lema 3.2. *Nech $(L, \leq)$ je zväz. Potom*

1. $(x_1, y_1) \wedge_t (x_2, y_2) = (x_1 \wedge x_2, y_1 \wedge y_2)$
2. $(x_1, y_1) \vee_t (x_2, y_2) = (x_1 \vee x_2, y_1 \vee y_2)$
3. $(x_1, y_1) \wedge_i (x_2, y_2) = (x_1 \wedge x_2, y_1 \vee y_2)$
4. $(x_1, y_1) \vee_i (x_2, y_2) = (x_1 \vee x_2, y_1 \wedge y_2)$

Veta 3.5. *Nech $(L, \leq)$ je zväz. Potom*

- *$(L^2, \leq_t, \leq_i)$ je dvojzväz.*
- *ak L má operátor negácie, tak určuje operátor negácie pre L^2.*
- *$-$ je operátor konflácie L^2.*
- *ak L je úplný zväz, tak L^2 je úplný dvojzväz.*

Dôkaz. Všetky tvrdenia vyplývajú z lemy 3.2 alebo priamo z definícií. □

Interval (x, y) je exaktný, ak $(x, y) = (y, x) \Leftrightarrow x = y$. Exaktný interval obsahuje práve jeden prvok. Interval (x, y) je konzistentný, ak $(x, y) \leq_i (y, x) \Leftrightarrow x \leq y$. Konzistentný interval obsahuje aspoň jeden prvok.

Podobne ako pri zväzoch, množina funkcií, ktoré priraďujú prvkom ľubovolnej množiny pravdivostné hodnoty z dvojzväzu, tvorí tiež dvojzväz. Zachováva vlastnosť úplnosti, dá sa na ňu rozšíriť operátor negácie aj konflácie.

Definícia 3.11. Nech $(B, \leq_t, \leq_i)$ je dvojzväz, S je neprázdna množina. Nech B^S je množina všetkých zobrazení z S do B a $f, g \in B^S$. Potom

- $f \leq_t g$, ak $f(s) \leq_t g(s)$
- $f \leq_i g$ ak $f(s) \leq_i g(s)$
- ak B má operátor negácie $\sim$, tak určuje operátor $(\sim f)(s) = \sim f(s)$
- ak B má operátor konflácie $-$, tak určuje operátor $(-f)(s) = -f(s)$

pre všetky $s \in S$.

Lema 3.3. *Nech $(B, \leq_t, \leq_i)$ je dvojzväz, S je neprázdna množina. Nech B^S je množina všetkých zobrazení z S do B a $f, g \in B^S$. Potom*

1. $(f \wedge_t g)(s) = f(s) \wedge_t g(s)$
2. $(f \vee_t g)(s) = f(s) \vee_t g(s)$
3. $(f \wedge_i g)(s) = f(s) \wedge_i g(s)$
4. $(f \vee_i g)(s) = f(s) \vee_i g(s)$

pre všetky $s \in S$.

Veta 3.6. *Nech* B *je dvojzväz,* S *je neprázdna množina. Potom*

- $(B^S, \leq_t, \leq_i)$ *je dvojzväz.*
- *ak* B *je úplný dvojzväz, tak* B^S *je úplný dvojzväz.*
- *ak* B *má operátor negácie, tak určuje operátor negácie pre* B^S.
- *ak* B *má operátor konflácie, tak určuje operátor konflácie pre* B^S.

Poznámka 3.1. Odteraz budeme predpokladať, že množina logických konštánt L jazyka $\mathcal{L}$ je usporiadaná v úplnom zväze $(L, \leq)$, ktorý má operátor negácie $\sim$. Tým je určený aj úplný dvojzväz intervalov $(L^2, \leq_t, \leq_i)$ spolu s operátorom negácie aj konflácie.

Kapitola 4

Sémantika

V prvej kapitole sme zadefinovali formuly rozšíreného prvorádového jazyka, v ďalšej kapitole sme predstavili zväzy resp. dvojzväzy ako priestor pravidvostných hodnôt. Teraz ukážeme, akým spôsobom budeme formulám jazyka priraďovať ich význam.

4.1 Interpretácie

Na priradenie významu formúl potrebujeme okrem priestoru pravdivostných hodnôt aj doménu pre významy konštánt a funkčných symbolov. Vo všeobecnosti takýchto domén existuje nekonečne veľa. My sa budeme zaoberať iba Herbrandovými interpretáciami, pretože všetky výsledky v tejto práci sa dajú priamočiaro rozšíriť na ľubovolné interpretácie.

Herbrandove interpretácie interpretujú termy rovnako pre každý jazyk. Jediné, čo je variabilné, je priraďovanie významu predikátom. V dvojhodnotových logikách stačí vymenovať všetky základné atómy, ktoré sú pravdivé, aby sme vedeli interpretovať celý jazyk. Na interpretáciu sa môžeme pozerať aj ako na zobrazenie, ktoré priradí hodnotu pravda všetkým základným atómom, ktoré sú pravdivé a hodnotu nepravda všetkým základným atómom, ktoré sú nepravdivé. Tento prístup sa dá použiť aj pre viachodnotové logiky.

Definícia 4.1. *Základný term* je term, ktorý neobsahuje premenné. Podobne *základná formula* je formula, ktorá neobsahuje premenné.

Definícia 4.2. *Herbrandovo univerzum* $\mathcal{U}(\mathcal{L})$ je množina základných termov jazyka $\mathcal{L}$. *Herbrandova báza* $\mathcal{B}(\mathcal{L})$ je množina základných objektívnych literálov jazyka $\mathcal{L}$.

Definícia 4.3. *Uzemnená verzia* logického programu P je množina základných pravidiel získaných z P nahradením všetkých premenných prvkami z Herbrandovho univerza všetkými možnými spôsobmi.

Pokiaľ je Herbrandovo univerzum nekonečná množina a program obsahuje pravidlo s premennou, uzemnená verzia má nekonečne veľa pravidiel. Na každý atóm sa môžeme pozerať ako na novú výrokovú premennú. Odteraz budeme pod logickým programom myslieť jeho uzemnenú verziu. V príkladoch namiesto prvorádových formúl budeme preto používať výrokové formuly.

Definícia 4.4. *Herbrandova interpretácia* jazyka $\mathcal{L}$ je zobrazenie z $\mathcal{B}(\mathcal{L})$ do množiny logických konštánt L.

Vďaka vete 3.4 množina všetkých interpretácií jazyka $\mathcal{L}$ tvorí tiež úplný zväz, ktorý má operátor negácie. Najmenším prvkom je interpretácia, ktorá každému základnému objektívnemu literálu priraďuje pravdivostnú hodnotu $\bot$, najväčším prvkom je interpretácia priraďujúca pravdivostnú hodnotu $\top$.

Často sa zvykne hovoriť o interpretácii množiny formúl S namiesto jazyka, ktorý dané formuly obsahuje. Predpokladá sa, že rozšírený prvorádový jazyk je definovaný konštantami, funkčnými a predikátovými symbolmi, ktoré sa vyskytujú v S. Preto budeme hovoriť o Herbrandovom univerze $\mathcal{U}(S)$ a Herbrandovej báze $\mathcal{B}(S)$ množiny S a takisto o Herbrandovej interpretácii množiny S.

Teraz ukážeme, ako sa interpretácia základných atómov môže rozšíriť na interpretáciu ľubovolnej uzavretej formuly.

Definícia 4.5. Nech I je Herbrandova interpretácia jazyka $\mathcal{L}$. *Ohodnotenie* v_I uzavretých formúl jazyka $\mathcal{L}$ je zobrazenie do množiny logických konštánt L definované nasledovne:

- Ak F je základný objektívny literál, potom $v_I(F) = I(F)$.
- Ak F je logická konštanta, potom $v_I(F) = F$.
- Ak F je uzavretá formula, potom $v_I(not\ F) = \ v_{\sim I}(F)$.
- Ak F a G sú uzavreté formuly, potom
  - $v_I(F \wedge G) = v_I(F) \wedge v_I(G)$
  - $v_I(F \vee G) = v_I(F) \vee v_I(G)$
  - $v_I(F \rightarrow G) = \top$, ak $v_I(F) \leq v_I(G)$, inak $v_I(F \rightarrow G) = \bot$
- Ak F je formula a x je premenná, potom
  - $v_I(\forall x F) = \bigwedge_{t \in \mathcal{U}(\mathcal{L})} v_I(F\{x/t\})$
  - $v_I(\exists x F) = \bigvee_{t \in \mathcal{U}(\mathcal{L})} v_I(F\{x/t\})$

  kde $\{x/t\}$ je substitúcia, ktorá zamení každý výskyt premennej x za základný term t

Definícia 4.6. Nech I je Herbrandova interpretácia jazyka $\mathcal{L}$. Hovoríme, že I je *modelom klauzy* F, ak $v_I(F) = \top$, a *modelom logického programu* P, ak I je modelom každej klauzy F z P.

Doteraz sme nehovorili o vzťahu atómu a objektívneho literálu. Nasledujúca vlastnosť koherencie spája ich význam.

Definícia 4.7. Nech I je Herbrandova interpretácia jazyka $\mathcal{L}$. Hovoríme, že I je *koherentná*, ak $I(\neg L) \leq \sim I(L)$ pre všetky objektívne literály $L \in \mathcal{B}(\mathcal{L})$.

Príklad 4.1. Pokiaľ vieme, že má pacient Peter aspoň na 75% chrípku (formálne $I(flu(Peter)) = 0,75$), naša vedomosť o opaku môže byť maximálne 25% ($I(\neg flu(Peter)) \leq 0,25$). Pokiaľ by to tak nebolo, tak by sme nekonzistentne tvrdili, že Peter má a zároveň aj nemá s nenulovou pravdepodobnosťou chrípku.

4.2 Aproximácie

Aproximácie nepriraďujú základným objektívnym literálom priamo ich pravdivostnú hodnotu, ale interval prípustných hodnôt.

Definícia 4.8. *Herbrandova aproximácia* jazyka $\mathcal{L}$ je zobrazenie z $\mathcal{B}(\mathcal{L})$ do množiny dvojíc logických konštánt L^2.

Na Herbrandovu aproximáciu sa môžeme pozerať aj ako na prvok dvojzväzu interpretácií $(L^{\mathcal{B}(\mathcal{L})})^2$, nakoľko je izomorfný s dvojzväzom $(L^2)^{\mathcal{B}(\mathcal{L})}$.

Vráťme sa teraz opäť k pravidlám logického programu. Nech sú hodnoty objektívnych literálov ohraničené konzistentnou aproximáciou $A = (I, J)$. Pokiaľ chceme mať najslabšie ohraničenie pre hlavu pravidla, hodnota tela musí byť čo najmenšia. Objektívne literály L budeme teda ohodnocovať najmenšou hodnotou $I(L)$, defaultové literály *not* L negáciou najväčšej hodnoty $\sim J(L)$. Takýmto spôsobom môžeme zadefinovať pseudo-ohodnotenie, ktoré priraďuje uzavretým formulám ich najmenšiu prípustnú hodnotu.

Definícia 4.9. Nech $A = (I, J)$ je Herbrandova aproximácia jazyka $\mathcal{L}$. *Pseudo-ohodnotenie* v_A uzavretých formúl jazyka $\mathcal{L}$ je zobrazenie do množiny logických konštánt L definované nasledovne:

- Ak F je objektívny literál, potom $v_A(F) = I(F)$.
- Ak F je logická konštanta, potom $v_A(F) = F$.
- Ak F je uzavretá formula, potom $v_A(not\ F) = v_{\sim A}(F)$.
- Ak F a G sú uzavreté formuly, potom
  - $v_A(F \wedge G) = v_A(F) \wedge v_A(G)$
  - $v_A(F \vee G) = v_A(F) \vee v_A(G)$
  - $v_A(F \rightarrow G) = \top$, ak $v_A(F) \leq v_A(G)$, inak $v_A(F \rightarrow G) = \bot$
- Ak F je formula a x je premenná, potom
  - $v_A(\forall x F) = \bigwedge_{t \in \mathcal{U}(\mathcal{L})} v_A(F\{x/t\})$
  - $v_A(\exists x F) = \bigvee_{t \in \mathcal{U}(\mathcal{L})} v_A(F\{x/t\})$

  kde $\{x/t\}$ je substitúcia, ktorá zamení každý výskyt premennej x za základný term t

Definícia 4.10. Nech A je Herbrandova aproximácia jazyka $\mathcal{L}$. Hovoríme, že A je *modelom klauzy* F, ak $v_A(F) = \top$, a *modelom logického programu* P, ak A je modelom každej klauzy F z P.

Definícia 4.11. Nech A je Herbrandova aproximácia jazyka $\mathcal{L}$. Hovoríme, že A je *koherentná*, ak $A(\neg L) \leq_t \sim A(L)$ pre všetky objektívne literály $L \in \mathcal{B}(\mathcal{L})$.

Koherentná aproximácia ohraničuje iba koherentné interpretácie.

Lema 4.1. *Nech $A = (I, J)$ je koherentná aproximácia, $I \leq K \leq J$ je interpretácia. Potom K je koherentná interpretácia.*

Dôkaz. Platí, že $I(\neg L) \leq K(\neg L) \leq J(\neg L)$ a $\sim J(L) \leq \sim K(L) \leq \sim I(L)$. Vďaka koherentnosti aproximácie $I(\neg L) \leq \sim J(L)$ a $J(\neg L) \leq \sim I(L)$. □

Kapitola 5

Stabilné modely

Ukázali sme, ako interpretácie určujú význam uzavretých formúl. Nás budú zaujímať iba tie interpretácie, ktoré modelujú logický program a spĺňajú podmienku koherencie. Navyše budeme požadovať, aby sa dala každá pravdivostná hodnota objektívneho literálu vypočítať "zdola" pomocou pravidiel a defaultových predpokladov. Množina všetkých takýchto interpretácií bude tvoriť význam logického programu.

5.1 Stabilné interpretácie

Najprv sa pozrime na triedu definitných logických programov. Nakoľko neobsahujú defaultové literály, výpočet spočíva iba v aplikovaní informácie obsiahnutej v pravidlách programu. Zedefinujeme rozšírený van Emden-Kowalského operátor T_P [26] pre viachodnotový prípad:

Definícia 5.1. Nech P je definitný logický program, $\mathcal{I}$ je množina všetkých interpretácií P.

Operátor $T_P : \mathcal{I} \mapsto \mathcal{I}$ je definovaný nasledovne:

$$T_P(I)(L) = v_I(L) \vee \bigvee_{\substack{c \in P \\ head(c)=L}} v_I(body(c))$$

Operátor T_P ráta najslabšie ohraničenie objektívnych literálov na základe pravidiel definitného programu. Pokiaľ sa nájdu pravidlá, pre ktoré hodnota hlavy nie je väčšia ako hodnota tela, nová interpretácia priradí najmenšiu hodnotu takú, aby boli nerovnosti splnené.

Výpočet "zdola" znamená postupné aplikovanie operátora T_P na najmenšiu interpretáciu $\bot$.

Definícia 5.2. Nech T_P je operátor z predchádzajúcej definície. Potom
$T_P \uparrow 0(I) = I$
$T_P \uparrow \alpha(I) = T_P(T_P \uparrow (\alpha - 1)(I))$, ak α je ordinál nasledovníka
$T_P \uparrow \alpha(I) = \bigvee_{\beta<\alpha} T_P \uparrow \beta(I)$, ak α je limitný ordinál

Veta 5.1. *Nech P je definitný logický program. Potom $T_P \uparrow \omega(\bot)$ je najmenší model P.*

Definitný logický program má jednoznačne definovaný význam - svoj minimálny model. Normálny logický program však môže obsahovať defaultové negácie v telách pravidiel. Výsledná interpretácia by teda mohla obsahovať defaultové predpoklady. Ak by sme zamenili všetky defaultové literály za ich pravdivostnú hodnotu, získali by sme definitný logický program. Pokiaľ by minimálny model korešpondoval s danou interpretáciou, naše defaultové predpoklady by boli opodstatnené. Takto sa dá zovšeobecniť Gelfond-Lifschitzov operátor [11] na viachodnotový prípad.

Definícia 5.3. Nech P je normálny logický program, I je interpretácia P. P *modulo* I je program P_I získaný z P zámenou všetkých defaultových literálov L, ktoré sa vyskytujú v telách pravidiel, za logickú konštantu $v_I(L)$.

Definícia 5.4. Interpretácia I normálneho logického programu P je *stabilná*, ak $T_{P_I} \uparrow \omega(\bot) = I$.

Príklad 5.1. $P = \{a \leftarrow \frac{1}{4}, b \leftarrow \frac{1}{4}, a \leftarrow \sim b, b \leftarrow \sim a, \neg a \leftarrow a\}$ je normálny logický program. $S_1 = \{a \mapsto \frac{1}{4}, \neg a \mapsto \frac{1}{4}, b \mapsto \frac{3}{4}, \neg b \mapsto \bot\}$ je prvým kandidátom na stabilnú interpretáciu. Program P modulo S_1 je $P_{S_1} = \{a \leftarrow \frac{1}{4}, b \leftarrow \frac{1}{4}, b \leftarrow \frac{3}{4}, \neg a \leftarrow \frac{1}{4}\}$. S_1 je stabilná interpretácia a tiež koherentná, pretože $S_1(\neg a) \leq \sim S_1(a)$ a $S_1(\neg b) \leq \sim S_1(b)$. Pozrime sa na druhého kandidáta $S_2 = \{a \mapsto \frac{3}{4}, \neg a \mapsto \frac{3}{4}, b \mapsto \frac{1}{4}, \neg b \mapsto \bot\}$. Program P modulo S_2 je $P_{S_2} = \{a \leftarrow \frac{1}{4}, b \leftarrow \frac{1}{4}, a \leftarrow \frac{3}{4}, \neg a \leftarrow \frac{3}{4}\}$. S_2 je stabilná interpretácia, avšak nie je koherentná, pretože $S_2(\neg a) \not\leq \sim S_2(a)$.

Veta 5.2. *Nech I je stabilná interpretácia normálneho logického programu P. Potom I je model P.*

Dôkaz. Keďže I je stabilná interpretácia P, potom I je model P_I. Preto I je model aj P. □

Definícia 5.5. Nech P je zovšeobecnený logický program. P^+ je program, ktorý obsahuje všetky pravidlá P s objektívnym literálom v hlave, P^- je program, ktorý obsahuje všetky pravidlá P s defaultovým literálom v hlave.

Definícia 5.6. Interpretácia I zovšeobecneného logického programu P je *stabilná*, ak $T_{P_I^+} \uparrow \omega(\bot) = I$.

Príklad 5.2. $P = \{a \leftarrow \frac{1}{4}, b \leftarrow \frac{1}{4}, a \leftarrow \sim b, b \leftarrow \sim a, \sim a \leftarrow a\}$ je zovšeobecnený logický program. $S_1 = \{a \mapsto \frac{1}{4}, \neg a \mapsto \bot, b \mapsto \frac{3}{4}, \neg b \mapsto \bot\}$ je stabilná interpretácia P a zároveň aj modelom P. $S_2 = \{a \mapsto \frac{3}{4}, \neg a \mapsto \bot, b \mapsto \frac{1}{4}, \neg b \mapsto \bot\}$ je stabilná interpretácia P ale nie je modelom P, pretože $v_{S_2}(\sim a \leftarrow a) = \bot$.

Stabilné interpretácie definitných logických programov korešpondujú s najmenším modelom. Pri normálnych logických programoch sú tiež modelom. V prípade zovšeobecnených logických programoch to tak už nemusí byť. Pre všetky tri triedy programov stabilné interpretácie nemusia byť koherentné. Ako sa dajú charakterizovať stabilné interpretácie, aby vždy modelovali program a zachovávali vlastnosť koherencie?

5.2 Aproximácie stabilných interpretácií

Stabilné interpretácie nemusia byť modelom logického programu P, pretože nemusia modelovať P^-. Operátor T_P definovaný na množine interpretácií nepoužíva informácie obsiahnuté v negatívnych pravidlách. Zadefinujeme teraz operátor na množine aproximácií. Spodná hranica bude určená pozitívnymi pravidlami, horná negatívnymi.

Definícia 5.7. Nech P je logický program, $\mathcal{A}$ je množina všetkých aproximácií P a $\mathcal{I}$ je množina všetkých interpretácií P.

Operátor $T_P : \mathcal{A} \mapsto \mathcal{I}$ je definovaný nasledovne:

$$T_P(A)(L) = v_A(L) \vee \bigvee_{\substack{c \in P \\ head(c)=L}} v_A(body(c))$$

Operátor $F_P : \mathcal{A} \mapsto \mathcal{I}$ je definovaný nasledovne:

$$F_P(A)(L) = \sim v_A(not\ L) \wedge \bigwedge_{\substack{c \in P \\ head(c)=notL}} \sim v_A(body(c))$$

Operátor $\Psi_P : \mathcal{A} \mapsto \mathcal{A}$ je definovaný nasledovne:

$$\Psi_P(A) = (T_P(A), F_P(A))$$

Výpočet "zdola" bude znamenať postupnú aplikáciu operátora Ψ_P na aproximáciu $(\bot, J)$. Interpretácia J predstavuje defaultové predpoklady.

Definícia 5.8. Nech A je aproximácia logického programu P. Potom
$\Psi_P \uparrow 0(A) = A$
$\Psi_P \uparrow \alpha(A) = \Psi_P(\Psi_P \uparrow (\alpha - 1)(A))$, ak α je ordinál nasledovníka
$\Psi_P \uparrow \alpha(A) = \bigvee_{\beta<\alpha}{}_i\ \Psi_P \uparrow \beta(A)$, ak α je limitný ordinál

Teraz ukážeme, ako pomocou operátora Ψ_P môžeme charakterizovať stabilné modely.

Lema 5.1. *Nech I, J sú interpretácie logického programu P. Potom $F_P(I, J) \leq J$.*

Dôkaz. Pre $F_P(A)(L) = \sim v_A(not\ L) \wedge \bigwedge_{\substack{c \in P \\ head(c)=notL}} \sim v_A(body(c))$ platí, že $J(L) = \sim v_A(not\ L) \leq F_P(I, J)(L)$. Preto $J \leq F_P(I, J)$. □

Lema 5.2. *Nech $I \leq J$ sú interpretácie logického programu P, J je model P. Potom $F_P(I, J) = J$.*

Dôkaz. Ak $I \leq J$, tak $v_{(I,J)}(body(c)) \leq v_J(body(c))$. Ak J je model P, tak $v_J(body(c)) \leq v_J(head(c))$. Potom pre $c \in P^-$ platí, že $v_{(I,J)}(body(c)) \leq v_J(head(c)) = v_{(I,J)}(head(c))$. Preto $F_P(I, J) = J$. □

Veta 5.3. *Interpretácia J logického programu P je stabilný model P práve vtedy, keď $\Psi_P \uparrow \omega(\bot, J) = (J, J)$.*

Dôkaz. Nech $I_\alpha = T_{P_J^+} \uparrow \alpha(\bot)$ je taká postupnosť interpretácií, že $\bigvee\limits_{\alpha<\omega} I_\alpha = J$. Keďže $I_\alpha \leq J$, vďaka leme 5.2 $F_P(I_\alpha, J) = J$. Potom $T_P(\Psi_P \uparrow \alpha(\bot, J)) = T_{P_J^+} \uparrow (\alpha+1)(\bot)$ a preto $\Psi_P \uparrow \omega(\bot, J) = (J, J)$.

Nech $(I_\alpha, J_\alpha) = \Psi_P \uparrow \alpha(\bot, J)$ je taká postupnosť aproximácií, že platí $\bigvee\limits_{\alpha<\omega}{}_i\,(I_\alpha, J_\alpha) = (J, J)$. Vďaka leme 5.1 $J_\alpha = J$. Potom $T_{P_J^+} \uparrow (\alpha+1)(\bot) = T_P(\Psi_P \uparrow \alpha(\bot, J))$ a preto $T_{P_J^+} \uparrow \omega(\bot) = J$. □

Ako sme už spomenuli, stabilný model nemusí byť koherentný ani pre definitné logické programy. Pokúsime sa teda rozšíriť operátor tak, aby zachovával vlastnosť koherencie.

Definícia 5.9. Nech P je logický program, $\mathcal{A}$ je množina všetkých aproximácií P a $\mathcal{I}$ je množina všetkých interpretácií P.

Operátor $F_P^* : \mathcal{A} \mapsto \mathcal{I}$ je definovaný nasledovne:

$$F_P^*(A)(L) = \sim v_A(not\ L) \wedge \bigwedge_{\substack{c\in P \\ head(c)\in\{notL,\neg L\}}} \sim v_A(body(c))$$

Operátor $\Psi_P^* : \mathcal{A} \mapsto \mathcal{A}$ je definovaný nasledovne:

$$\Psi_P^*(A) = (T_P(A), F_P^*(A))$$

Definícia 5.10. Nech A je aproximácia logického programu P. Potom
$\Psi_P^* \uparrow 0(A) = A$
$\Psi_P^* \uparrow \alpha(A) = \Psi_P^*(\Psi_P^* \uparrow (\alpha-1)(A))$, ak α je ordinál nasledovníka
$\Psi_P^* \uparrow \alpha(A) = \bigvee\limits_{\beta<\alpha}{}_i\ \Psi_P^* \uparrow \beta(A)$, ak α je limitný ordinál

Lema 5.3. *Nech A je koherentná aproximácia logického programu P. Potom $\Psi_P^*(A)$ je koherentná aproximácia.*

Dôkaz. Nech $A = (I, J)$ je koherentná aproximácia logického programu P. Potom $I(\neg L) \leq \sim J(L)$. Treba ukázať, že $T_P(A)(\neg L) \leq \sim F_P^*(A)(L)$.

$$T_P(A)(\neg L) = I(\neg L) \vee \bigvee_{\substack{c\in P \\ head(c)=\neg L}} v_A(body(c))$$

$$\sim F_P^*(A)(L) = \sim J(L) \vee \bigvee_{\substack{c\in P \\ head(c)=\neg L}} v_A(body(c)) \vee \bigvee_{\substack{c\in P \\ head(c)=notL}} v_A(body(c))$$

Keďže $I(\neg L) \leq \sim J(L)$, tak platí aj $T_P(A)(\neg L) \leq \sim F_P^*(A)(L)$. □

Keďže aproximácia $(\bot, \top)$ je koherentná, iterácia operátora $\Psi_P^* \uparrow \omega$ vypočíta iba koherentnú interpretáciu. Podobne, ako sme pomocou operátora Ψ_P charakterizovali stabilné modely, pomocou operátora Ψ_P^* môžeme charakterizovať koherentné stabilné modely.

Lema 5.4. *Nech I, J sú interpretácie logického programu P. Potom $F_P^*(I, J) \leq J$.*

Dôkaz. Pre $F^*_P(I,J)(L) = \sim v_{(I,J)}(not\ L) \wedge \bigwedge_{\substack{c\in P \\ head(c)\in\{notL,\neg L\}}} \sim v_{(I,J)}(body(c))$ platí, že $J(L) = \sim v_{(I,J)}(not\ L) \leq F^*_P(I,J)(L)$. Preto $J \leq F^*_P(I,J)$. □

Lema 5.5. *Nech I, J sú interpretácie logického programu P, J je koherentný model P. Potom $F^*_P(I,J) = J$.*

Dôkaz. Ak $I \leq J$, tak $v_{(I,J)}(body(c)) \leq v_J(body(c))$. Ak J je model P, tak $v_J(body(c)) \leq v_J(head(c))$. Ak J je koherentná interpretácia, tak $v_J(\neg\ L) = J(\neg L) \leq \sim J(L) = v_J(not\ L)$. Potom pre $c \in P^-$ platí, že $v_{(I,J)}(body(c)) \leq v_J(head(c)) = v_{(I,J)}(head(c))$ a taktiež pre $c \in P^+$ platí, že $v_{(I,J)}(body(c)) \leq v_J(head(c)) \leq v_J(not\neg head(c)) = v_{(I,J)}(not\neg head(c))$. Preto $F^*_P(I,J) = J$. □

Veta 5.4. *Interpretácia J logického programu P je koherentný stabilný model P práve vtedy, keď $\Psi^*_P(\bot, J) = (J, J)$.*

Dôkaz. Nech $I_\alpha = T_{P^+_J} \uparrow \alpha(\bot)$ je taká postupnosť interpretácií, že $\bigvee_{\alpha<\omega} I_\alpha = J$. Keďže $I_\alpha \leq J$, vďaka leme 5.5 $F^*_P(I_\alpha, J) = J$. Potom $T_P(\Psi^*_P \uparrow \alpha(\bot, J)) = T_{P^+_J} \uparrow (\alpha+1)(\bot)$ a preto $\Psi^*_P \uparrow \omega(\bot, J) = (J, J)$.

Nech $(I_\alpha, J_\alpha) = \Psi^*_P \uparrow \alpha(\bot, J)$ je taká postupnosť aproximácií, že platí $\bigvee_{\alpha<\omega}{}_i\,(I_\alpha, J_\alpha) = (J, J)$. Vďaka leme 5.4 $J_\alpha = J$. Potom $T_{P^+_J} \uparrow (\alpha+1)(\bot) = T_P(\Psi^*_P \uparrow \alpha(\bot, J))$ a preto $T_{P^+_J} \uparrow \omega(\bot) = J$.

Treba ešte ukázať, že J je koherentná interpretácia. Vďaka leme 5.3 sú všetky aproximácie (I_α, J) koherentné a teda platí $I_\alpha(\neg L) \leq \sim J(L)$. Keďže J je suprémum I_α, potom aj $J(\neg L) \leq \sim J(L)$. □

Máme teda zadefinovaný operátor Ψ^*_P, ktorý nám charakterizuje koherentné stabilné modely. Množina všetkých takýchto modelov bude tvoriť význam zovšeobecnených logických programov rozšírených o explicitnú negáciu.

Kapitola 6

Kripkeho štruktúry

V dnešnom svete sme neustále vystavovaní prísunu rôznych informácií. Nie vždy sú však konzistentné. Konflikty medzi nimi zvykneme riešiť na základe rôznych preferencií. Napríklad novšie informácie preferujeme pred staršími alebo dôveryhodnejšie zdroje pred menej dôveryhodnými. Multidimenzionálny logický program využíva ideu, že logické programy môžu byť menené logickými programami [3]. Pozostáva z množiny logických programov, ktoré sú usporiadané reláciou preferencie.

Definícia 6.1. Nech $(V, <)$ je čiastočne usporiadaná množina, $\mathcal{P} = \{P_i \mid i \in V\}$ je množina logických programov. Dvojica $(\mathcal{P}, V)$ sa nazýva *multidimenzionálny logický program*. Ak je množina $(V, <)$ usporiadaná lineárne, $(\mathcal{P}, V)$ sa nazýva *dynamický logický program*.

Budeme často používať jednoduchšie označenie $\mathcal{P}$ namiesto $(\mathcal{P}, V)$, pričom budeme implicitne predpokladať existenciu čiastočne usporiadanej množiny V.

Dynamický logický program je špeciálnym prípadom multidimenzionálneho logického programu. Na lineárne usporiadané programy sa môžeme pozerať aj ako na vývoj nášho poznania v čase. Nie vždy však preferujeme poznatky iba na základe času, staršia informácia od dôveryhodného zdroja môže byť uprednostňovaná pred novšou od zdroja, ktorý je nedôveryhodný. Preferencia na základe viacerých kritérií vyžaduje zložitejšie usporiadanie ako iba lineárne.

Logický program zachytáva viac informácií ako množina jeho modelov. Obsahuje závislosti medzi literálmi, ktoré sa stratia, pokiaľ sa obmedzíme iba na modely [19]. Ukážeme si to na nasledujúcom príklade.

Príklad 6.1. Zoberme si program P, ktorý hovorí o tom, že niekto je pacifista a pacifista je rozumná osoba:

$$\begin{aligned} pacifist &\leftarrow \\ reasonable &\leftarrow pacifist \end{aligned}$$

Tento program má jediný stabilný model $M_P = \{pacifist, reasonable\}$. Neskôr bude preferovanejší program U tvrdiť, že je vojna a vtedy dotyčná osoba nie je pacifista.

$$\begin{aligned} war &\leftarrow \\ not\ pacifist &\leftarrow war \end{aligned}$$

Pokiaľ by sme menili množinu modelov, dostali by sme nový model $M = \{war, reasonable\}$. Napriek tomu, že spomínaná osoba už nie je pacifista, ešte stále ju považujeme za rozumnú. Ak sa však pozrieme na logické programy, usúdime, že M nie je intuitívnym modelom, pretože osoba bola rozumná kvôli tomu, že bola pacifista, nepravdivosť predpokladu pacifista by malo mať za následok nepravdivosť literálu rozumná osoba. Intuitívnym modelom by mal byť model M'.

Predchádzajúci príklad nám naznačuje, že je lepšie hovoriť o konflikte medzi pravidlami ako o konflikte medzi literálmi. Uprednostníme to pravidlo, ktoré je z preferovanejšieho programu.

Výpočet koherentného stabilného modelu logického programu P spočíva v postupnom aplikovaní operátora Ψ^*_P na aproximáciu s defaultovými predpokladmi. Na množinu takýchto výpočtov sa môžeme pozerať aj ako na orientovaný graf - uzly tvoria aproximácie programu P a hrana exstuje medzi dvomi aproximáciami práve vtedy, keď sa vieme dostať z jednej do druhej aplikáciou operátora Ψ^*_P. Potom každá cesta začínajúca v aproximácii iba s defaultovými predpokladmi $(\bot, J)$ a končiaca v exaktnej aproximácii (J, J) počíta stabilný model J.

Operátor Ψ^*_P v každom kroku aplikuje všetky možné pravidlá z programu P. Takýto hrubozrnný postup však nie je vhodný pre dynamický prípad, pretože aplikácia ľubovolného pravidla môže byť zamietnutá konfliktnou informáciou z preferovanejšieho programu. Preto potrebujeme postup zjemniť tak, aby sa v každom kroku aplikovalo iba jedno pravidlo.

Definícia 6.2. Kripkeho štruktúra $\mathcal{K}_P$ asociovaná s logickým programom P je usporiadaná dvojica (W, ρ), kde $W = \mathcal{A}$ je množina všetkých aproximácií P, $\rho \subseteq W \times W$ je relácia dostupnosti taká, že

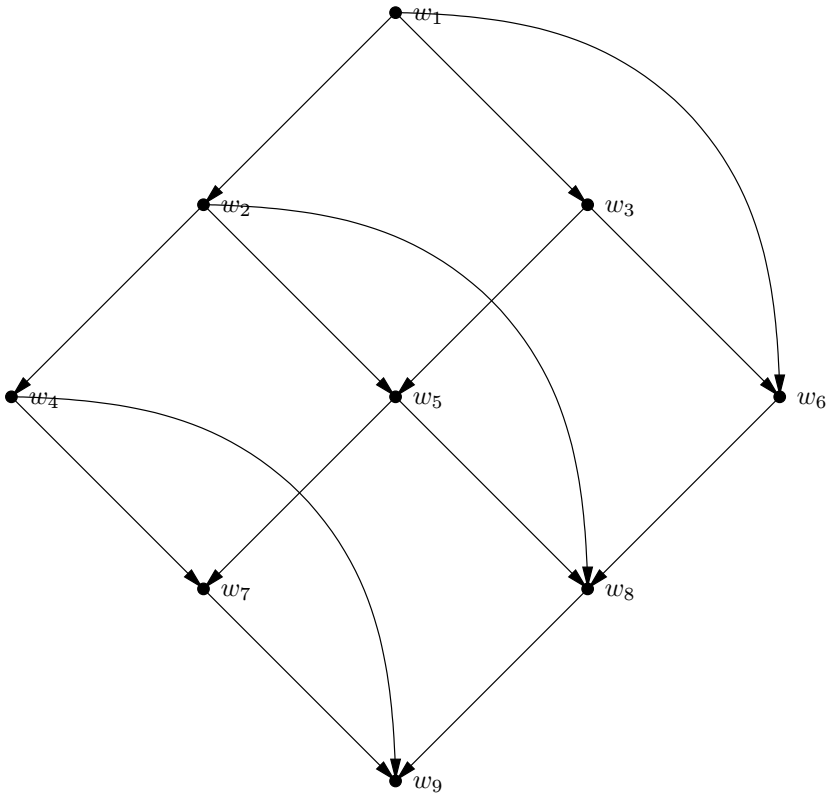
$$(A_1, A_2) \in \rho \Leftrightarrow \exists c \in P : A_2 = \Psi^*_{\{c\}}(A_1) \neq A_1$$

Príklad 6.2. Zoberme program P s množinou logických konštánt $\{0, \frac{1}{4}, \frac{3}{4}, 1\}$ a operátorom negácie $\sim x = 1 - x$.

$$\begin{aligned} a &\leftarrow \tfrac{1}{4} \\ \neg b &\leftarrow \tfrac{1}{4} \\ a &\leftarrow not\ \neg b \\ \neg b &\leftarrow not\ a \\ \neg a &\leftarrow a \end{aligned}$$

Kandidátom na koherentný stabilný model je interpretácia $M = \{a \mapsto \frac{1}{4}, \neg a \mapsto \frac{1}{4}, b \mapsto 0, \neg b \mapsto \frac{3}{4}\}$. Nasledujúci obrázok predstavuje relevantnú časť Kripkeho štruktúry pre výpočet modelu M:

Obr. 6.1: Relevantná časť Kripkeho štruktúry



$$
\begin{aligned}
w_1 &= (\bot, M) = \\
&= \{a \in (0,\tfrac{1}{4}), \neg a \in (0,\tfrac{1}{4}), b \in (0,0), \neg b \in (0,\tfrac{3}{4})\} \\
w_2 &= \Psi^*_{\{a \leftarrow \frac{1}{4}\}}(w_1) = \Psi^*_{\{a \leftarrow not\, \neg b\}}(w_1) = \\
&= \{a \in (\tfrac{1}{4},\tfrac{1}{4}), \neg a \in (0,\tfrac{1}{4}), b \in (0,0), \neg b \in (0,\tfrac{3}{4})\} \\
w_3 &= \Psi^*_{\{b \leftarrow \frac{1}{4}\}}(w_1) = \\
&= \{a \in (0,\tfrac{1}{4}), \neg a \in (0,\tfrac{1}{4}), b \in (0,0), \neg b \in (\tfrac{1}{4},\tfrac{3}{4})\} \\
w_4 &= \Psi^*_{\{\neg a \leftarrow a\}}(w_2) = \\
&= \{a \in (\tfrac{1}{4},\tfrac{1}{4}), \neg a \in (\tfrac{1}{4},\tfrac{1}{4}), b \in (0,0), \neg b \in (0,\tfrac{3}{4})\} \\
w_5 &= \Psi^*_{\{b \leftarrow \frac{1}{4}\}}(w_2) = \Psi^*_{\{a \leftarrow \frac{1}{4}\}}(w_3) = \Psi^*_{\{a \leftarrow not\, \neg b\}}(w_3) = \\
&= \{a \in (\tfrac{1}{4},\tfrac{1}{4}), \neg a \in (0,\tfrac{1}{4}), b \in (0,0), \neg b \in (\tfrac{1}{4},\tfrac{3}{4})\} \\
w_6 &= \Psi^*_{\{\neg b \leftarrow \sim a\}}(w_1) = \Psi^*_{\{\neg b \leftarrow \sim a\}}(w_3) = \\
&= \{a \in (0,\tfrac{1}{4}), \neg a \in (0,\tfrac{1}{4}), b \in (0,0), \neg b \in (\tfrac{3}{4},\tfrac{3}{4})\} \\
w_7 &= \Psi^*_{\{b \leftarrow \frac{1}{4}\}}(w_4) = \Psi^*_{\{\neg a \leftarrow a\}}(w_5) = \\
&= \{a \in (\tfrac{1}{4},\tfrac{1}{4}), \neg a \in (\tfrac{1}{4},\tfrac{1}{4}), b \in (0,0), \neg b \in (\tfrac{1}{4},\tfrac{3}{4})\} \\
w_8 &= \Psi^*_{\{\neg b \leftarrow \sim a\}}(w_2) = \Psi^*_{\{\neg b \leftarrow \sim a\}}(w_5) = \Psi^*_{\{a \leftarrow \frac{1}{4}\}}(w_6) = \Psi^*_{\{a \leftarrow not\, \neg b\}}(w_6) = \\
&= \{a \in (\tfrac{1}{4},\tfrac{1}{4}), \neg a \in (0,\tfrac{1}{4}), b \in (0,0), \neg b \in (\tfrac{3}{4},\tfrac{3}{4})\} \\
w_9 &= \Psi^*_{\{\neg b \leftarrow \sim a\}}(w_4) = \Psi^*_{\{\neg b \leftarrow \sim a\}}(w_7) = \Psi^*_{\{\neg a \leftarrow a\}}(w_8) = (M, M) = \\
&= \{a \in (\tfrac{1}{4},\tfrac{1}{4}), \neg a \in (\tfrac{1}{4},\tfrac{1}{4}), b \in (0,0), \neg b \in (\tfrac{3}{4},\tfrac{3}{4})\}
\end{aligned}
$$

Vidíme, že výpočet koherentného stabilného modelu M zodpovedá ceste v Kripkeho štruktúre, ktorá začína iba s defaultami $(\bot, M)$ a končí v exaktnej aproximácii (M, M).

Odteraz budeme predpokladať, že uzemnená verzia logického programu P je konečná množina.

Definícia 6.3. *Cesta* σ v Kripkeho štruktúre $\mathcal{K}_P = (\mathcal{A}, \rho)$ asociovanej s logickým programom P je postupnosť $\{A_\alpha\}_{\alpha=1}^{n}$, kde $(A_\alpha, A_{\alpha+1}) \in \rho$. Hovoríme, že cesta σ je *zakorenená* v A_0. Cesta σ *terminuje* v A_n, ak neexistuje hrana $(A_n, A) \in \rho$.

Veta 6.1. *Nech $\mathcal{K}_P$ je Kripkeho štruktúra asociovaná s logickým programom P, J je koherentný stabilný model P. Potom existuje cesta $\sigma \in \mathcal{K}_P$ zakorenená v $(\bot, J)$ a terminujúca v (J, J).*

Dôkaz. Nech J je koherentný stabilný model logického programu P, $\{\Psi_P^* \uparrow \alpha(\bot, J)\}_{\alpha<\omega} = \{A_\alpha\}_{\alpha<\omega}$ je postupnosť aproximácií. Teraz ukážeme, že to, čo vieme odvodiť aplikáciou viacerých pravidiel v jednom kroku operátora Ψ_P^*, vieme odvodiť aj postupnou aplikáciou pravidiel. Pre každú iteráciu operátora Ψ_P^* zadefinujeme také usporiadanie na literáloch, že aplikácia pravidla s menším literálom v hlave neovplyvní pravdivostnú hodnotu tela pravidla s väčším literálom v hlave.

Nech $C_L^\alpha = \{c \in P \mid head(c) = L, v_{A_\alpha}(head(c)) < v_{A_\alpha}(body(c))\}$ je taká minimálna množina, že $T_P(A_\alpha)(L) = v_{A_\alpha}(L) \vee \bigvee_{c \in C_L} v_{A_\alpha}(body(c))$. Pre každé α vieme usporiadať objektívne literály do postupnosti tak, že $\forall c \in C_{L_1}^\alpha : L_2 \in body(c) \Rightarrow L_1 <_\alpha^* L_2$.

Nech $L_1 <_\alpha L_2$, ak existuje $c \in C_{L_1}^\alpha : L_2 \in body(c)$. Keďže pre všetky $c \in C_L^\alpha$ platí $v_{A_\alpha} head(c) < v_{A_\alpha} body(c)$, potom aj $I(L_1) < I(L_2)$. Relácia $<_\alpha$ je teda ireflexívna aj asymetrická. Tranzitívny uzáver $<_\alpha^*$ je ostré čiastočné usporiadanie. Pre každú konečnú čiastočne usporiadanú množinu existuje lineárne usporiadanie, ktoré zachováva pôvodné čiastočné usporiadanie.

Teraz môžeme bezpečne aplikovať pravidlá podľa usporiadania literálov. Nech $A_{\alpha,0} = A_\alpha$. Nové $A_{\alpha,i+1}$ získame postupnou aplikáciou pravidiel z $C_{L_{\alpha,i}}$. Pretože každá množina C_L je minimálna, $A_{\alpha,j} < A_{\alpha,j+1}$ pre všetky $c \in C_L$. Vďaka usporiadaniu literálov platí $\bigvee_i A_{\alpha,i} = A_{\alpha+1}$. □

Veta 6.2. *Nech $\mathcal{K}_P$ je Kripkeho štruktúra asociovaná s logickým programom P, $\sigma \in \mathcal{K}_P$ je cesta zakorenená v $(\bot, J)$ a terminujúca v (J, J). Potom J je koherentný stabilný model.*

Dôkaz. Nech $\sigma \in \mathcal{K}_P$ je cesta zakorenená v $(\bot, J)$ a terminujúca v (J, J). Keďže σ terminuje v (J, J), potom $\Psi_P^*(J, J) = (J, J)$.

Z definície operátora Ψ^* vyplýva, že pre ľubovolné aproximácie $A \leq_i B$ a $c \in P$ platí $\Psi_{\{c\}}^*(A) \leq_i \Psi_P^*(B)$. Matematickou indukciou vzhľadom na dĺžku cesty $\sigma = \{A_\alpha\}_{\alpha=1}^{n}$ sa dá ukázať, že $A_\alpha \leq_i \Psi_P^* \uparrow \alpha(\bot, J)$. Preto $(J, J) \leq_i \Psi_P^* \uparrow \omega(\bot, J)$.

Operátor Ψ_P^* je monotónny vhľadom na informačné usporiadanie. Potom $\Psi_P^* \uparrow \alpha(\bot, J) \leq_i \Psi_P^* \uparrow \alpha(J, J) = (J, J)$. Preto $\Psi_P^* \uparrow \omega(\bot, J) \leq_i (J, J)$. □

Dôsledok 6.1. *Nech $\mathcal{K}_P$ je Kripkeho štruktúra asociovaná s logickým programom P. J je koherentný stabilný model P práve vtedy, keď existuje existuje cesta $\sigma \in \mathcal{K}_P$ zakorenená v $(\bot, J)$ a terminujúca v (J, J).*

Kripkeho štruktúra asociovaná s logickým programom nám môže slúžiť ako výpočtový model. Navyše poskytuje nástroje na to, aby každá aplikácia pravidla mohla byť prebitá pravidlom z preferovanejšieho programu. Zovšeobecnenie na multidimenzionálne logické programy je mimo rozsahu tejto práce. V ďalšej kapitole však predstavíme sémantiky založené na princípe kauzálneho zamietania, pre ktoré môže byť definícia Kripkeho štruktúry rozšírená.

Kapitola 7

Princíp kauzálneho zamietania

V predchádzujúcej časti sme videli, že logický program obsahuje viac informácií ako množina jeho modelov. Obsahuje závislosti medzi literálmi, ktoré sú pre dynamické logické programovanie dôležité. Budeme preto radšej hovoriť o konflikte medzi závislosťami ako o konflikte medzi literálmi. Konflikt medzi dvomi pravidlami môže nastať vtedy, ak v hlavách obsahujú navzájom opačné literály. Konflikt medzi literálmi L a $notL$ budeme označovať ako $notL \bowtie L$. Vzniklo viacero sémantík, ktoré rôzne definujú, kedy pravidlo z vyššieho programu prebíja pravidlo z nižšieho programu. Jedna z prvých sémantík pre dynamické logické programy bola zavedená v článku [20].

Definícia 7.1. Nech $\mathcal{P} = \{P_i \mid i \in V\}$ je multidimenzionálny logický program, M je interpretácia. Nech

$$\begin{aligned} Rejected(M, P_i) &= \{c \in P_i \mid M \models body(c), \exists d \in P_j : \\ &\quad i < j, M \models body(d), head(c) \bowtie head(d)\} \\ Residue(M, \mathcal{P}) &= \bigcup_{i=1}^{n} (P_i \setminus Rejected(M, P_i)) \end{aligned}$$

M je *DJU model* (Dynamic Justified Update) $\mathcal{P}$, ak M je najmenší model logického programu $Residue(M, \mathcal{P}) \cup M^{-}$.

Príklad 7.1. Zoberme programy P a U z príkladu 6.1. Chceme overiť, či interpretácia $M^{'} = \{war\}$ je *DJU model* dynamického logického programu $P \oplus U$.

$$\begin{aligned} M^{'-} &= \{not\ pacifist \leftarrow;\ not\ reasonable \leftarrow\} \\ Rejected(M^{'}, U) &= \emptyset \\ Rejected(M^{'}, P) &= \{pacifist \leftarrow\} \\ Residue(M^{'}, \mathcal{P}) &= \{reasonable \leftarrow pacifist;\ war \leftarrow;\ not\ pacifist \leftarrow war\} \end{aligned}$$

$M^{'}$ je najmenší model programu $Residue(M^{'}, \mathcal{P}) \cup M^{'-}$, preto je aj DJU model.

Interpretácia $M = \{war, pacifist\}$ nie je DJU model. Množina defaultov pre M je $\{not\ reasonable \leftarrow\}$, množina $Residue(M, \mathcal{P})$ je rovnaká ako pre model $M^{'}$. Neexistuje žiadne pravidlo, ktoré by mohlo podoprieť literál $pacifist$.

Eiter [5] sa pokúsil zadefinovať sémantiku tak, aby zamietol čo najmenej pravidiel a nezahodil tak niektoré modely. Nepovoľuje preto zamietať pravidlami, ktoré už boli sami zamietnuté.

Definícia 7.2. Nech $\mathcal{P} = \{P_i \mid i \in V\}$ je multidimenzionálny logický program, M je interpretácia. Nech

$$\begin{aligned} Rej(M, P_i) &= \{c \in P_i \mid M \models body(c), \exists d \in P_j \setminus Rej(M, P_j) : \\ &\quad i < j, M \models body(d), head(c) \bowtie head(d)\} \\ Res(M, \mathcal{P}) &= \bigcup_{i=1}^{n} (P_i \setminus Rej(M, P_i)) \end{aligned}$$

M je *BDJU model* (Backward Dynamic Justified Model) $\mathcal{P}$, ak M je najmenší model logického programu $Res(M, \mathcal{P}) \cup M^{-}$.

Príklad 7.2. Rozdiel medzi DJU a BDJU sémantikou vzniká pri cyklických zmenách. Zoberme napríklad postupnosť logických programov $P_1 = \{a \leftarrow\}$, $P_2 = \{not\ a \leftarrow\}$, $P_3 = \{a \leftarrow a\}$. Pravidlo v treťom programe nám neprináša žiadnu novú informáciu. Intuitívne by sme teda predpokladali, že významom postupnosti bude interpretácia $M = \{a\}$.

$$\begin{aligned} M^{-} &= \emptyset \\ Rej(M, P_3) = Reject(M, P_3) &= \emptyset \\ Rej(M, P_2) = Reject(M, P_2) &= \{not\ a \leftarrow\} \\ Rej(M, P_1) &= \emptyset \\ Reject(M, P_1) &= \{a \leftarrow\} \end{aligned}$$

Cyklická zmena $a \leftarrow a$ umožnila v prípade BDJU sémantiky ponechať pôvodný predpoklad $a \leftarrow$, preto M je BDJU model. DJU sémantika umožní prebyť $a \leftarrow$ faktom $not\ a \leftarrow$ z vyššieho programu, preto M nie je DJU model.

Napriek tomu, že *DJU* sémantika sa v niektorých prípadoch vie vysporiadať s cyklickými zmenami, neplatí to vo všeobecnosti. Autori článku [18] zadefinovali novú sémantiku, ktorá je odolná voči tautológiám vo viacerých prípadoch ako *DJU* sémantika. Avšak tiež existujú prípady, kedy dáva neintuitívne výsledky.

Definícia 7.3. Nech $\mathcal{P} = \{P_i \mid i \in V\}$ je multidimenzionálny logický program, M je interpretácia. Nech

$$\begin{aligned} Defaults(M, \mathcal{P}) &= \{not\ a \mid \nexists c \in \mathcal{P} : head(c) = a, M \models body(c)\} \\ Rejected(M, P_i) &= \{c \in P_i \mid M \models body(c), \exists d \in P_j : \\ &\quad i < j, M \models body(d), head(c) \bowtie head(d)\} \\ Residue(M, \mathcal{P}) &= \bigcup_{i=1}^{n} (P_i \setminus Rejected(M, P_i)) \end{aligned}$$

M je *DSM model* (Dynamic Stable Model) $\mathcal{P}$, ak M je najmenší model logického programu $Residue(M, \mathcal{P}) \cup Defaults(M, \mathcal{P})$.

Príklad 7.3. Ukážeme príklad, kedy DJU sémantika nie je odolná voči tautológiám, ale DSM sémantika je. Nech $P = \{a \leftarrow\}$, $U = \{not\ a \leftarrow not\ a\}$. Keďže

pravidlo $not\ a \leftarrow not\ a$ neprináša žiadnu novú informáciu, interpretácia $M = \emptyset$ nezodpovedá našim intuíciám.

$$\begin{aligned} M^{-} &= \{not\ a\} \\ Defaults(M, \mathcal{P}) &= \emptyset \\ Reject(M, P) &= \{a \leftarrow\} \\ Reject(M, U) &= \emptyset \\ Residue(M, \mathcal{P}) &= \{not\ a \leftarrow not\ a\} \end{aligned}$$

M je najmenší model $Residue(M, \mathcal{P}) \cup M^{-}$, ale nie je najmenším modelom $Residue(M, \mathcal{P}) \cup Defaults(M, \mathcal{P})$.

Pre úplnosť uvádzame aj štvrtú sémantiku, ktorá vznikla spojením predchádzajúcich dvoch.

Definícia 7.4. Nech $\mathcal{P} = \{P_i \mid i \in V\}$ je multidimenzionálny logický program, M je interpretácia. Nech

$$\begin{aligned} Defaults(M, \mathcal{P}) &= \{not\ a \mid \nexists c \in \mathcal{P} : head(c) = a, M \models body(c)\} \\ Rej(M, P_i) &= \{c \in P_i \mid M \models body(c), \exists d \in P_j \setminus Rej(P_j, M) : \\ &\quad i < j, M \models body(d), head(c) \bowtie head(d)\} \\ Res(M, \mathcal{P}) &= \bigcup_{i=1}^{n} (P_i \setminus Rej(M, P_i)) \end{aligned}$$

M je *BDSM model* (Backwards Dynamic Stable Model) $\mathcal{P}$, ak M je najmenší model logického programu $Res(M, \mathcal{P}) \cup Defaults(M, \mathcal{P})$.

Príklad 7.4. Tento príklad ukazuje, že ani BDSM sémantika a ani DSM sémantika nie sú odolné voči tautológiám. Nech $P_1 = \{a \leftarrow, not\ a \leftarrow\}$, $P_2 = \{a \leftarrow a\}$. Samotný program P_1 nemá stabilný model. Program P_2 obsahuje iba tautológiu, ktorá neprináša žiadnu novú informáciu. Očakávame teda, že program $P_1 \oplus P_2$ nebude mať dynamický stabilný model. Pre všetky spomínané sémantiky však platí $DJU(\mathcal{P}) = BDJU(\mathcal{P}) = DSM(\mathcal{P}) = BDSM(\mathcal{P}) = \{\{a\}\}$.

Všetky štyri sémantiky sú porovnávané v práci [17]. Platí medzi nimi nasledujúci vzťah.

Veta 7.1. *Nech $\mathcal{P}$ je dynamický logický program. Potom*

- $DSM(\mathcal{P}) \subseteq DJU(\mathcal{P})$
- $BDSM(\mathcal{P}) \subseteq BDJU(\mathcal{P})$

Veta 7.2. *Nech $\mathcal{P}$ je dynamický logický program. Potom*

- $DJU(\mathcal{P}) \subseteq BDJU(\mathcal{P})$
- $DSM(\mathcal{P}) \subseteq BDSM(\mathcal{P})$

Príklad 7.4 naznačuje, v akých prípadoch zlyháva DSM sémantika. Pokiaľ jeden program obsahuje konfliktné pravidlá, nie sme schopný zamietnuť napriek tomu, že hlava jedného z nich je podopretá aj vo vyššom programe. Zamietanie pravidiel aj na úrovni jedného programu bolo podnetom pre vznik ďalších sémantík pre dynamický logický program [2]:

Definícia 7.5. Nech $\mathcal{P} = \{P_i \mid i \in V\}$ je dynamický logický program, M je interpretácia. Nech

$$\begin{aligned} Reject^R(M, P_i) &= \{c \in P_i \mid M \models body(c), \exists d \in P_j : \\ &\quad i \leq j, M \models body(d), head(c) \bowtie head(d)\} \\ Residue^R(M, \mathcal{P}) &= \bigcup_{i=1}^{n} (P_i \setminus Reject^R(M, P_i)) \end{aligned}$$

M je *RDJU model* (Refined Dynamic Justified Update) $\mathcal{P}$, ak M je najmenší model logického programu $Residue^R(M, \mathcal{P}) \cup M^-$.

Definícia 7.6. Nech $\mathcal{P} = \{P_i \mid i \in V\}$ je dynamický logický program, M je interpretácia. Nech

$$\begin{aligned} Defaults(M, \mathcal{P}) &= \{not\ a \mid \nexists c \in \mathcal{P} : head(c) = a, M \models body(c)\} \\ Reject^R(M, P_i) &= \{c \in P_i \mid M \models body(c), \exists d \in P_j : \\ &\quad i \leq j, M \models body(d), head(c) \bowtie head(d)\} \\ Residue^R(M, \mathcal{P}) &= \bigcup_{i=1}^{n} (P_i \setminus Reject^R(M, P_i)) \end{aligned}$$

M je *RDSM model* (Refined Dynamic Stable Model) $\mathcal{P}$, ak M je najmenší model logického programu $Residue^R(M, \mathcal{P}) \cup Defaults(M, \mathcal{P})$.

Definície RDJU a RDSM sémantík nie sú priamočiaro rozšíriteľné na multidimenzionálne logické programy, pretože vzniká problém pri zamietaní pravidiel medzi neporovnateľnými programami [16]. RDJU sémantika zdieľa niektoré nedostatky DJU sémantiky a preto nie je odolná voči cyklickým zmenám. RDSM sémantika však túto vlastnosť má.

Príklad 7.5. Vráťme sa k príkladu 7.4. Ukážeme, že $M = \{a\}$ je $RDSM$ model. Keďže obe pravidlá v programe $P_1 = \{a \leftarrow;\ not\ a \leftarrow\}$ sú navzájom konfliktné, tak sa zamietnu. Množina defaultov je prázdna a teda M je najmenší model $Residue^R(M, \mathcal{P}) \cup Defaults^R(M, \mathcal{P}) = P_2 = \{a \leftarrow\}$.

Veta 7.3. *Nech $\mathcal{P}$ je dynamický logický program. Potom*

- $RDSM(\mathcal{P}) \subseteq DSM(\mathcal{P})$
- $RDJU(\mathcal{P}) \subseteq DJU(\mathcal{P})$

Všetky spomínané sémantiky sú porovnávané v diplomovej práci [15]. Autor ukázal, že pre acyklické programy sú všetky tieto sémantiky zhodné. V súčasnej dobe sa preferuje v lineárnom prípade RDSM sémantika práve kvôli jej odolnosti voči cyklickým zmenám.

Kapitola 8

Záver

Multidimenzionálne logické programy potrebujú obsahovať aj pravidlá s negatívnou hlavou, aby mohli zamietať pozitívne literály. Preto sme potrebovali stabilné modely nielen pre normálne logické programy [8], ale aj pre zovšeobecnené logické programy. Rovnako sme obohatili jazyk o explicitnú negáciu, ktorá zväčšuje vyjadrovaciu silu.

Pomocou stabilného operátora sme zadefinovali Kripkeho štruktúru. Je založená na postupnom aplikovaní ohraničení obsiahnutých v pravidlách programu na aproximácie. Výpočet stabilných modelov sa dá potom identifikovať ako cesta zakorenená v aproximácii obsahujúcej iba defaulty a terminujúca v exaktnej aproximácii.

Definícia Kripkeho štruktúry sa dá rozšíriť aj pre sémantiky viachodnotových multidimenzionálnych logických programov založených na princípe kauzálneho zamietania. Najprv je však potrebné tieto sémantiky charakterizovať pomocou pevných bodov operátorov. Potom sa dajú identifikovať výpočty príslušných stabilných modelov v Kripkeho štruktúre a ukázať ekvivalencia obidvoch alternatívnych prístupov.

Časť II

Kapitola 9

Projekt dizertačnej práce

9.1 Východiská

Nemonotónnosť negácie je jedným z hlavných zdrojov veľkého počtu rôznych sémantík pre logické programy. Medzi najviac akceptované patrí bezpochyby stabilná sémantika [11]. Stabilné modely sú minimálne, každá pravdivostná hodnota atómu sa dá vypočítať postupnou aplikáciou pravidiel na množinu defaultov. Pojem stabilného modelu bol rozšírený aj pre logické programy obsahujúce druhý typ negácie - explicitnú negáciu [12, 13].

Fitting rozšíril dvojhodnotové stabilné modely pre viachodnotový prípad [6, 9]. Použil algebraickú štruktúru dvojzväz zavedenú Ginsbergom v [14] na zadefinovane zovšeobecneného operátora nielen pre stabilné modely, ale aj čiastočné a parakonzistentné stabilné modely [7, 8, 10]. Sémantikou logického programu sú modely, ktoré tvoria pevné body operátora.

Uvažovanie s dynamickými poznatkami je dôsledok vývoja nášho poznania v čase, získavanie informácií z rôznych zdrojov a pod. Logické programy zachytávajú naše vedomosti, avšak treba tento koncept rozšíriť tak, aby bol schopný zachytávať zmeny. Logické programy obsahujú viac informácie ako množina ich stabilných modelov [19]. Na rozdiel od modelov obsahujú aj závislosti medzi literálmi. Preto je lepšie hovoriť o zmene logického programu ako o zmene interpretácie sveta.

Bolo vytvorených viacero sémantík pre zmeny logických programov, ktoré sú založené na princípe kauzálneho zamietania. Nech $\mathcal{P}$ je množina logických programov s reláciou preferencie, $Rejected(\mathcal{P}, M)$ obsahuje pravidlá, ktoré sú zamietnuté pravidlami z preferovanejších programov a množina $Defaults(\mathcal{P}, M)$ obsahuje defaulty vzhľadom na interpretáciu M. Pravidlo kauzálneho zamietania hovorí, že M je stabilný model $\mathcal{P}$ práve vtedy, keď M je stabilný model $(\mathcal{P} \setminus Rejected(\mathcal{P}, M)) \cup Defaults(\mathcal{P}, M)$. Autori článkov [5, 18, 20] rôznymi definíciami množín $Rejected(\mathcal{P}, M)$ a $Defaults(\mathcal{P}, M)$ zaviedli sémantiky Dynamic Justified Updates, Dynamic Stable Models, Backwards Dynamic Justified Updates a Backwards Dynamic Stable Models.

9.2 Ciele

Zovšeobecníme existujúce sémantiky multidimenzionálnych logických programov s explicitnou negáciou založené na princípe kauzálneho zamietania pre viachodnotové logiky. Ukážeme, že dvojhodnotový prípad zovšeobecneného prístupu korešponduje s existujúcimi sémantikami.

9.3 Metódy riešenia

V prvej časti bola interpretácia rozšíreného logického programu zadefinovaná ako zobrazenie z rozšírenej Herbrandovskej bázy do zväzu pravdivostných hodnôt. Porovnáme tento prístup s článkom [1], v ktorom autori definujú interpretáciu ako zobrazenie z Herbrandovskej bázy do dvojzväzu. Prvá zložka predstavuje hodnotu atómu a druhá zložka hodnotu objektívneho literálu. Symbol konjunkcie je interpretovaný ako infímum a symbol disjunkcie ako suprémum vzhľadom na informačné usporiadanie.

Ako priestor pravdivostných hodnôt zvolíme úplný dvojzväz. Rozšírime definíciu stabilného operátora pre zovšeobecnené logické programy, ktoré môžu obsahovať defaultovú negáciu v hlavách pravidiel.

Rozšírime princíp kauzálneho zamietania pre viachodnotový prípad. Pravidlá s opačnými hlavami budú v konflikte, ak interval určený pravdivostnými hodnotami ich tiel nebude obsahovať žiaden prvok. Priamočiarym zovšeobecnením množiny defaultov dostaneme DJU, DSM, BDJU a BDSM sémantiky pre viachodnotové logické programy.

9.4 Návrh postupu

Pre viachodnotové logické programy s explicitnou negáciou

1. zovšeobecníme van Emden-Kowalského operátor pre definitné logické programy
2. zovšeobecníme Gelfond-Lifschitzovu transformáciu pre normálne logické programy
3. zadefinujeme koherentné stabilné modely zovšeobecnených logických programov pomocou stabilného operátora
4. zadefinujeme DJU, DSM, BDJU a BDSM sémantiky pomocou princípu kauzálneho zamietania

Literatúra

[1] J. Alcantâra, C. V. Damásio, and L. M. Pereira. Paraconsistent logic programs. Italy, August 2002.

[2] J. J. Alferes, F. Banti, and J. A. Leite. Semantics for dynamic logic programming: A principle-based approach. In V. Lifshitz and I. Niemela, editors, *Proceedings of the Seventh International Conference on Logic Programming and Nonmonotonic Reasoning*. Springer-Verlag, 2004.

[3] J. J. Alferes and L. M. Pereira. Update-programs can update programs. In J. Dix, L. M. Pereira, and T. Przymusinski, editors, *Non-Montonic Extensions of Logic Programming, Selected papers from NMELP'96*, pages 110–131. Springer, LNAI 1216, 1997.

[4] H. A. Blair and V. S. Subrahmanian. Paraconsistent logic programming. volume 287.

[5] T. Eiter, G. Sabbatini, M. Fink, and H. Tompits. On updates of logic programs: Semantics and properties. Technical Report 1843-00-08, Institute of Information Systems, Vienna University of Technology, July 2001.

[6] M. Fitting. Bilattices in logic programming. In G. Epstein, editor, *The Twentieth International Symposium on Multiple-valued Logics*, pages 238–246, 1990.

[7] M. Fitting. The family of stable models. *Journal of Logic Programming 17*, pages 197–225, 1993.

[8] M. Fitting. Fixpoint semantics for logic programming a survey. *Theoretical Computer Science*, 278(1-2):25–51, 2002.

[9] M. C. Fitting. Bilattices and the semantics of logic programming. *Journal of Logic Programming*, 11(2):91–116, 1989.

[10] M. C. Fitting. Well-founded semantics, generalized. In *Logic Programming, Proceedings of the 1991 International Symposium*, pages 71–84, Cambridge, MA, 1991. MIT Press.

[11] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Fifth Int'l Conf.Symp. on Logic Programming*, pages 1070–1080. Seattle, 1988.

[12] M. Gelfond and V. Lifschitz. Logic programs with classical negation. pages 579–597. MIT Press, 1990.

[13] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, pages 365–385, 1991.

[14] M. L. Ginsberg. Multivalued logics: A uniform approach to reasoning in artificial intelligence. *Computational Intelligence*, 4:265–316, 1988.

[15] M. Homola. On relations of the various semantics approaches in multidimensional dynamic logic programming, 2004.

[16] J. Šiška. Refined semantics for multidimensionial dynamic logic programming, 2004.

[17] J. A. Leite. *Evolving Knowledge Bases: Specification and Semantics*, volume 81 of *Frontiers in Artificial Intelligence and Applications, Dissertations in Artificial Intelligence*. IOS Press, Amsterdam, 2003.

[18] J. A. Leite, J. J. Alferes, L. M. Pereira, H. Przymusinska, and T. C. Przymusinski. Dynamic logic programming. *Linkping Electronic Articles in Computer and Information Science*, 2(18), December 1997.

[19] J. A. Leite and L. M. Pereira. Generalizing updates: from models to programs. In J. Dix, L. M. Pereira, and T. Przymusinski, editors, *Logic Programming and Knowledge Representation, Selected Extended Papers from LPKR'97*, pages 224–246. Springer, LNAI 1471, 1998.

[20] J. A. Leite and L. M. Pereira. Iterated logic program updates. In J. Jaffar, editor, *Procs. of the 1998 Joint International Conference and Symposium on Logic Programming (JICSLP'98)*, pages 265–278. MIT Press, 1998.

[21] J. W. Lloyd. *Foundations of Logic Programming, Second Edition*. Springer-Verlag, 1987.

[22] T. Przymusinski. Well-founded semantics coincides with three-valued stable semantics. *Fundam. Inf.*, 13(4):445–463, 1990.

[23] T. C. Przymusinski. Stationary semantics for disjunctive logic programs and deductive databases. In *Proceedings of the 1990 North American conference on Logic programming*, pages 40–59. MIT Press, 1990.

[24] J. Šefránek. A Kripkean semantics for dynamic logic programming. *Lecture Notes in Computer Science*, 1955:469–486, 2000.

[25] J. Šefránek. Semantic consideration on rejection. 2004.

[26] M. van Emden and R. Kowalski. The semantics of predicate logic as a programming language. *Journal of the Assoc. for Comp. Mach.*, 23:733–742, 1976.