Univerzita Komenského
Fakulta Matematiky Fyziky a Informatiky

Parakonzistentná Sémantika pre Multidimenzionálne Dynamické Logické Programy

Diplomová práca

2017 **Bc. Boris Kruľ**

Univerzita Komenského
Fakulta Matematiky Fyziky a Informatiky

Parakonzistentná Sémantika pre Multidimenzionálne Dynamické Logické Programy

Diplomová práca

Študijný program: Informatika
Študijný obor: 2508 Informatika
Katedra: Katedra Informatiky
Vedúci: RNDr. Martin Baláž

Bratislava, 2017

Bc. Boris Kruľ

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Boris Kruľ
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Parakonzistentná sémantika pre multidimenzionálne dynamické logické programy
Paraconsistent semantics for multidimensional dynamic logic programs

Cieľ: Cieľom práce je zadefinovať parakonzistentnú sémantiku pre multidimenzionálne logické programy, vysloviť a dokázať tvrdenia o jej vlastnostiach.

Anotácia: Stabilné modely sa používajú na definovanie deklaratívnej sémantiky pre logické programy. V prípade konfliktu však nemusí existovať stabilný model logického programu. Napriek tomu znalostná báza môže obsahovať užitočné informácie, ktoré nesúvisia s konfliktom.
Bolo navrhnutých viacero prístupov, ktoré sa vysporiadavajú s neexistenciou stabilných modelov. Napríklad parakonzistentné stabilné modely používajú ďalšie dve pravdivostné hodnoty na modelovanie neúplných a nekonzistentných poznatkov. Multidimenzionálne dynamické logické programy používajú reláciu preferencie na vyriešenie konfliktov medzi pravidlami z rôznych zdrojov.
Cieľom diplomovej práce je skombinovať oba prístupy - použiť sémantiku parakonzistentných stabilných modelov pre multidimenzionálne dynamické logické programy. Ak sú konfliktné pravidlá porovnateľné, menej preferované pravidlo je zamietnuté a viac preferované pravidlo ponechané. Ak nie sú porovnateľné, ďalšia pravdivostná hodnota sa využije na identifikáciu nekonzistentnej časti bázy znalostí.

Kľúčové slová: stabilný model, parakonzistencia, multidimenzionálny dynamický logický program

Vedúci: RNDr. Martin Baláž
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 16.10.2013

Dátum schválenia: 22.10.2013 prof. RNDr. Branislav Rovan, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Chcel by som poďakovať svojmu vedúcemu RNDr. Martinovi Balážovi za cenné rany a podporu pri písaní tejto práce.

Abstrakt

Autor: Bc. Boris Kruľ

Názov práce: Parakonzistentná Sémantika pre Multidimenzionálne Dynamické Logické Programy

Škola: Univerzita komenského v Bratislava

Fakulta: Fakulta matematiky, fyziky a informatiky

Vedúci práce: RNDr. Martin Baláž

Rozsah práce: 35 strán
Bratislava, 2017

Statické programy nestačia na modelovanie sveta ak chceme zahrnúť aj zmenu situácie v čase a preto bola predstavené paradigma dynamického logického programovania. Sémantiky zaoberajúce sa dynamickým logickým programovaním sú často založené na stabilných modeloch a snažia sa predchádzať konfliktom. V prípade konfliktu, ktorému nevedia predísť, však nemusí existovať stabilný model logického programu. Napriek tomu znalostná báza môže obsahovať užitočné informácie, ktoré nesúvisia s konfliktom. Boli predstavené sémantiky, napríklad parakonzistentné stabilné modely, používajúce ďalšie pravdivostné hodnoty, čím sa vysporiadavajú s nekonzistentnou informáciou, ale tieto programy sú iba statické programy. My predstavujeme skĺbenie týchto dvoch prístupov, teda dynamické logické programy zamietajúce konfliktné pravidlá na základe ich priority a používajúce viac pravdivostných hodnôt na vyrovnanie sa s nekonzistenciou spôsobenou konfliktnými pravidlami, ktorých prioritu nevieme porovnať. Taktiež predstavujeme spôsob ako rozlíšiť informáciu odvodenú na základe nekonzistencie od informácie, v ktorej odvodení nekonzistencia nie je.

kľúčové slová stabilný model, parakonzistencia, multidimenzionálny dynamický logický program

Abstrakt

Author: Bc. Boris Kruľ

Thesis title: Paraconsistent semantics for multidimensional dynamic logic programs

University: Comenius University, Bratislava

Faculty: Faculty of Mathematics, Physics and Informatics

Advisor: RNDr. Martin Baláž

Thesis lengt: 35 pages
Bratislava, 2017

Static programs are not enough for modelling the world if we want to include the changes in time therefore a paradigm of dynamic logic programming was introduced. Semantics for dynamic logic programs are often based on stable models and intend to prevent conflicts from occuring. In case a conflict, they can't prevent, stable model may not exists. But the knowledge base may contain useful information not related to conflict. There were semantics introduced, for example paraconsistent stable models, using another truth values, which enables them to cope with inconsistent information, but those programs are static programs. We introduce a combination these two approaches - dynamic logic programs rejecting rules based on their priority and using of more truth values to cope with inconsistency caused by rules whose priority could not be compared. We also introduce a way to distinguish an information deduced from inconsistency from information deduced without inconsistency.

kľúčové slová stable model, parakonzistency, multidimensional dynamic logic program

Obsah

Kapitola 1

Úvod

Dlhú dobu bola väčšina prác v oblasti logického programovania zameraná na statické informácie. Toto sa neskôr stalo nepostačujúcim pre reprezentovanie sveta a dynamickej informáciemeniacej sa v čase a preto bola predstavené paradigma dynamického logického programovania. Zo začiatku bol prístup taký, že vypočítame model programu ktorý aktualizujeme novým programom. Čo viedlo k počítaniu veľkého množstva modelov. Neskôr sa začali aktualizovať samotné programy a počítal sa až model výsledného programu. Pri aktualizovaní však môže dochádzak k nekonzistentnej informácii, čomu sa predchádza zamietnutím starej informácie v podobe zamietnutie staršieho z konfliktných pravidiel. Dynamické logické programy boli postupnosť programov čo sa tiež neskôr ukázalo ako nepostačujúce pre modelovanie sveta. Preto boli predstavené multidimenzionálne dynamické logické programy, pri ktorých program a aktualizácie tovrili acyklický orientovaný graf. Toto však malo za následok, že prioritu niektorých pravidiel nebolo možné porovnať. Keďže tieto dynamické logické programy (a aj multidimenzionálne dynamické logické programy) fungujú nad dvojhodnotovou logikou, nevedia sa vysporiadať s nekonzistenciou, ktorá môže nastať práve pri konflikte pravidiel ktorým nevieme porovnať prioritu. V takom prípade neexistuje dvojhodnotový model a strácame cenné informácie, ktoré nemusia súvisieť s nekonzistenciou.

K nekonzistencii je možné, okrem zamietania pravidiel, pristupovať aj tak, že sa s ňou jednoducho vysporiadame, teda budeme používať viachodnotovú logiku. Takto fungujú napríklad aj parakonzistentné stabilné modely, ktoré však sú sú iba statické programy a preto nie sú vhodné na prácu s informáciami, ktoré sa menia v čase.

Predstavíme teda skĺbenie týchto dvoch prístupov - sémantiku ktorá dokáže pracovať s informáciou meniacou sa v čase, predchádzajúcou konfliktom medzi rpavidlami zamietaním pravidiel z niššou prioritou tam dke sa to dá. Táto sémantika bude schopná vysporiadať sa aj s konfliktami spôsobenými pravidlami, ktorých prioritu sme nevedeli porovnať. Budeme, podobne ako parakonzistentné stabilné modely, používať viac pravdivostných hodnôt aby sme sa vysporiadali s nekonzistenciou. Taktiež predstavíme spôsob ako rozlíšiť informáciu odvodenú na základe nekonzistencie od informácie, v ktorej odvodení nekonzistencia nie je.

Kapitola 2

Prehľad

2.1 Zakladné pojmy

V tejto časti si spomenieme základné pojmy z oblasti logického programovania.

Pod *programom* budeme rozumieť množinu pravidiel tvaru:

$$L_0 \leftarrow L_1 \wedge \cdots \wedge L_n$$

Pričom časť nalavo od $\leftarrow$, teda L_0, voláme hlava pravidla a časť napravo od $\leftarrow$, teda $\leftarrow L_1 \wedge \cdots \wedge L_n$, voláme telo pravidla. Pravidlo, ktoré má prázdnu hlavu, voláme podmienka a pravidlo s prázdnym telom voláme fakt. Podla ďalších podmienok, ktoré kladieme na pravidlá, ďalej rozdelujeme programy. Ešte spomenieme, že literál je atóm L alebo defaultne negovaný atóm $not L$. *Definítny logický program* je program, ktorého pravidlá obsahujú iba atómy. Ak pravidlá programu obsahujú literály v tele, ale v hlave je atóm, voláme tento program *normálny logický program*. Ak umožníme pravidlám, aby obsahovali literály v tele a aj v hlave, tak program obsahujúci takéto rpavidlá voláme *zovšeobecnený logický program* (označovaný aj *GLP*). Pre poriadok dodáme, že každý definitívny logický program je aj normálny logický program a každý normálny logický program je aj rozšírený logický program.

Interpretácia je množina atómov. Atóm A je pravdivý v interpretácii I, označujeme $I \models A$, ak $A \in I$, inak je A nepravdivý. Defaultový literál $not A$

je pravdivý v I, označujeme to $I \models notA$, ak $A \notin I$, inak je $notA$ nepravdivý. Hovoríme, že interpretácia M je model zovšeobecneného logickéhop programu P vtedy a len vtedy, keď pre každé pravidlo $r \in P$, r má tvar $L_0 \leftarrow L_1 \wedge \cdots \wedge L_n$, platí, že ak $M \models L_1 \wedge \cdots \wedge L_n$, tak platí aj $M \models L_0$. Model M programu P je minimálny ak neexistuje model N taký, že $N \subset M$.

2.2 Dvojhodnotové sémantiky

2.2.1 Stabilné modely

Stabilné modely boli definované v [7].
Nech P je logický program. Predpokladajme, že každé pravidlo obsahujúce premennú je nahradené všetkými jeho zagroundovanými inštanciami, takže všetky atómy v P sú zagroundované. (Keďže nie je požadované, aby P bol konečný, premenné môžu byť eliminované týmto spôsobom aj ak program používa funkcné symboly a jeho Herbrandovské universum je nekonečné.)
Pre ľubovolnú množinu M atómov z P, nech P_M je program získaný z P vymazaním (GL-transformácia):

(i) každého pravidla obsahujúceho negatívny literál $\neg L$ vo svojom tele ak $L \in M$

(ii) všetky negatívne literály v telách ostatných pravidiel

Zjavne, P_M neobsahuje negácie, a teda P_M ma unikátny minimálny Herbrandovský model. Ak sa tento model zhoduje s M, potom hovoríme, že M je *stabilny model* P.

Veta 2.2.1 *Ľubovoľný stabilný mnodel P je minimálny Herbrantovský model P.*

2.2.2 Rozšírenie stabilnomodelovej sémantiky pre zovšeobecnené logické programy

Pre reprezentovanie negatívnej informácie v logických programoch sa nám hodia rozšírenie logických programov také, ktoré umožní negáciu $notA$ v hlave pravidla (hovoríme o defaultovej negácii). Takéto programy voláme zovšeobecnené logické programy. GL-transformácia pracuje s pravidlami ktoré v hlavne nemajú defaultovú negáciu, takže potrebujeme sa najskôr postarať o takéto pravidlá, presnejšie ich hlavy. V [11] bol popísaný spôsob čo s takouto hlavou robiť a to že ju presunieme do tela bez defaultovej negácie. Takto však dostaneme prázdnu hlavu pravidla (akoby tam bolo false) a takýmto pravidlám hovoríme podmienky. Definícia pre SM nepracovala s programami obsahujúcimi podmienky, ale aj toto vieme riešiť. Nájdeme stabilný model tradičným spôsobom pre program v ktorom sme sa postarali o pravidlá s defaultovou negáciou v hlave ale nebudeme uvažovať podmienky (vzniknuté ani prípadné pôvodné) a následne zahodíme všetky stabilné modely nespĺňajúce tieto podmienky.

2.2.3 Dobre-podporené (Well-supported) modely

Táto sémantika bola popísaná v [4].Základom dobre podporených modelov je funkcia mapovania úrovní (level mapping) $\ell : \mathcal{L} \to \mathbb{N}$. Rozšírime túto funkcio pre negatívne literály formy $notA$, kde A je prvok $\mathcal{L}$, položením $\ell(notA) = \ell(A)$. Pre konjunkciu $C = L_1 \wedge \cdots \wedge L_n$ ďalej rozšírime ℓ tak, že $\ell(C) = max(\{\ell(L_i) \mid L_i \in C\})$. Pre jednoduchosť ešte priradíme hodnotu -1 prádznej konjunkcii literálov.

Definícia 2.2.1 *Nech P je normálny logický program. Interpretácia M je* dobre podporený model *programu P ak:*

- *M je model programu P*
- *existuje mapovanie úrovní (level mapping) ℓ také, že pre každý atóm A v M existuje pravidlo $A \leftarrow A_1, \ldots, A_n, notB_1, \ldots, notB_m$ také, že $M \models A_1, \ldots, A_n, notB_1, \ldots, notB_m$ a $\ell(A) > \ell(A_i)$ pre $1 \leq i \leq n$.*

Veta 2.2.2 *Nech P je normálny logický program. Interpretácia M je dobre podporený model programu P práve vtedy, keď je stabilným model programu P.*

2.3 Parakonzistentné sémantiky

2.3.1 Dvojzväzy

V tejto časti popíšeme konštrukcie, ktoré budeme používať pri našej sémantike. Tieto konštrukcie boli popísané v [6] a [5]. Jeden z dôvodov prečo budeme používať tieto konšrukcie v našej sémantike je to, že ak naše pravdivostné hodnoty budú spĺňať isté vlastnosti, môžeme používať 4 pravdivostné hodnoty rovnako dobre ako pravdivostné hodnoty nad intervalom $< 0, 1 >$ realnych čísel.

Definícia 2.3.1 Pre-bilattice *je štruktúra $\mathcal{B} = (B, \leq_t, \leq_k)$ kde $\mathcal{B}$ je neprázdna množina a $\leq_t$ a $\leq_k$ sú čiastočné usporiadania, každé tvoriace s $\mathcal{B}$ štruktúru zväzu.*

Definícia 2.3.2 Pre-bilattice $(B, \leq_t, \leq_k)$ *je* úplný *ak všetky meety a joiny existujú vzhľadom na obe usporiadania. Označme nekonečný meet a join vzhľadom na $\leq_t$ $\bigwedge$ a $\bigvee$ a pre usporiadanie $\leq_k$ to bude $\prod$ a $\sum$*

Definícia 2.3.3 Pre-bilattice $(B, \leq_t, \leq_k)$ *je:*

1. interlaced *dvojzväz ak každý z operátorov $\wedge$, $\vee$, $\oplus$ a $\otimes$ je monotónny vzhľadom na obe usporiadania*

2. *nekonečný* interlaced *dvojzväz ak je úplný a všetky 4 infinitary meet a join operátory sú monotónne vzhľadom na obe usporiadania*

3. *distributívny dvojzväz ak platí všetkých 12 distributívnych pravidiel pre $\wedge$, $\vee$, $\oplus$ a $\otimes$*

4. *nekonečný distributívny dvojzväz ak je úplný a nekonečné aj konečné distributívne zákony sú platné.*

Majme dva zväzy $\mathcal{L}_1 = (L_1, \leq_1)$ a $\mathcal{L}_2 = (L_2, \leq_2)$, kde prvky $\mathcal{L}_1$ môžu poskytnúť pozitívny dôkaz, s $\leq_1$ ako porovnávajúcou reláciou a podobne s $\mathcal{L}_2$ ako negatívnym dôkazom. Zväzy nemusia byť rovnaké.

Definícia 2.3.4 Bilattice product $\mathcal{L}_1 \odot \mathcal{L}_2$ *je štruktúra* $(\mathcal{L}_1 \times \mathcal{L}_2, \leq_t, \leq_k)$ *kde:*

1. $(x_1, x_2) \leq_t (y_1, y_2)$ *ak* $x_1 \leq_1 y_1$ *a* $y_2 \leq_2 x_2$
2. $(x_1, x_2) \leq_k (y_1, y_2)$ *ak* $x_1 \leq_1 y_1$ *a* $x_2 \leq_2 y_2$

Definícia 2.3.5 Intervalová konštrukcia *je druhý prístup ako získať dvojzväz. Nech* L *je zväz.* $\mathcal{K}(L)$ *je štruktúra* $(\mathcal{I}(L), \leq_t, \leq_k)$ *kde:*

- $< a, b > \leq_t < c, d >$ *ak* $a \leq_L c$ *a* $b \leq_L d$
- $< a, b > \leq_k < c, d >$ *ak* $< c, d > \subseteq < a, b >$

Definícia 2.3.6 *Dvojzväz má operátor* negácie *ak existuje mapovanie,* $\neg$*, otáčajúce usporiadanie* $\leq_t$ *a usporiadanie* $\leq_k$ *zachováva nezmenené a* $\neg\neg x = x$*. Dvojzväz má operátor* konflácie *ak existuje mapovanie,*$-$*, otáčajúce usporiadanie* $\leq_k$ *a usporiadanie* $\leq_t$ *zachováva nezmenené a* $--x = x$*. Ak má dvojzväz obe tieto operátory, tak su komutatívne ak* $-\neg x = \neg - x$ *pre všetky* x*.*

Definícia 2.3.7 *Predpokladajme, že* $\mathcal{B}$ *je dvojzväz s konfláciou. Voláme* $x \in \mathcal{B}$ presným *ak* $x = -x$ *a* konzistentným *ak* $x \leq_k -x$*.*

Definícia 2.3.8 *Nech* $\mathcal{B} = (B, \leq_t, \leq_k)$ *je pre-bilattice a nech* S *je nejaká neprázdna množina. Potom* $\mathcal{B}^{\mathcal{S}}$ *je množina všetkých funkcií z* S *do* $\mathcal{B}$*. Relácie usporiadania sú na* $\mathcal{B}^{\mathcal{S}}$ *definované priamočiaro a ak* $\mathcal{B}$ *má negáciu alebo konfláciu tak aj tieto sú na* $\mathcal{B}^{\mathcal{S}}$ *taktiež rozšírené priamočiaro:*

- $v \leq_t w$ *ak* $v(s) \leq_t w(s)$ *pre všetky* $s \in S$
- $v \leq_k w$ *ak* $v(s) \leq_k w(s)$ *pre všetky* $s \in S$
- *ak* $\mathcal{B}$ *má negáciu, tak operácia negácia je na* $\mathcal{B}^{\mathcal{S}}$ *definovaná takto:* $\neg v$ *je mapovanie také, že* $(\neg v)(s) = \neg(v(s))$

- *ak $\mathcal{B}$ má konfláciuu, tak operácia konflácie je na $\mathcal{B}^{\mathcal{S}}$ definovaná takto: $-v$ je mapovanie také, že $(-v)(s) = -(v(s))$*

Je zjavné, že týmto je aj $\mathcal{B}^{\mathcal{S}}$ pre-bilattice. Ak S je množina atomických formúl nejakého formálneho jazyka a $\mathcal{B}$ je dvojzväz pravdivostných hodnôt, $\mathcal{B}^{\mathcal{S}}$ je priestor valuácií a má garantované užitočné algebraické vlastnosti. Ďalej ak $\mathcal{B}^{\mathcal{S}}$ je priestor valuácií, kde $\mathcal{B}$ je dvojzväz nejkého konkrétneho typu tak aj $\mathcal{B}^{\mathcal{S}}$ je dvojzväz toho istého typu.

Definícia 2.3.9 *Nech v_1 a v_2 sú vaulácie v dvojzväze $\mathcal{B}$ s negáciou. Definujme pseudovauláciu $v_1 \triangle v_2$ v $\mathcal{B}$ nasledovne. Pre zagroundovaný atóm A platí:*

$$(v_1 \triangle v_2)(A) = v_1(A)$$

$$(v_1 \triangle v_2)(\neg A) = \neg v_2(A)$$

2.3.2 Parakonzistentné stabilné modely

Príkladom parakonzistentnej sémantiky je sémantika, na ktorú sa teraz pozrieme a zoberieme si ju ako základ pre našu prácu. Bola popísaná v [11]. Táto sémantika bola pôvodne popísaná pre triedu disjunktívnych programov avšak táto trieda je pre nás zbytočne všeobecná a ďalšia práca s ňou by bola prehnane komplikovaná a tak sa obmedzíme na podtriedu teda programy s pravidlami, ktoré v hlave nemajú disjunkciu (teda špeciálny prípad pravidiel z disjunktývnych programov).

Definícia 2.3.10 *Pre jednoduchosť sa najskôr pozrieme na pozitívne programy teda pravidlá nebudú obsahovať negáciu, takže pod pozitívnym programom budeme rozumieť konečnú množinu klauzúl formy:*

$$L_1 \leftarrow L_2 \wedge \cdots \wedge L_m \ (m \geq 2)$$

kde L_i sú pozitívne alebo negatívne literály. Ak L je pozitívny (resp. negatívny) literál, $\neg L$ označuje jeho opačný negatívny (resp. pzitívny) literál a platí $L = \neg\neg L$.

Pravdivostné hodnoty štvor-hodnotovej logiky sú definované ako $IV = \{t, f, \top, \bot\}$, kde $t, f, \top, \bot$ označujú *pravda, nepravda, kontradikcia, nedefinované*. Množina pravdivostných hodnôt IV tvorí úplný zväz s usporiadaním $\preceq$ takým, že $\bot \preceq x \preceq \top$ kde $x \in \{t, f\}$. Tento zväz je tiež známy ako Belnapova štvor-hodnotová logika.
Interpretácia je definovaná ako funkcia $I : \mathcal{L}_\mathcal{P} \to IV$ taká, že pre každý literál $L \in \mathcal{L}_\mathcal{P}$ platí:

$$I(L) = t \text{ ak } L \in I \text{ a } \neg L \notin I,$$

$$I(L) = f \text{ ak } \neg L \in I \text{ a } L \notin I,$$

$$I(L) = \top \text{ ak } L \in I \text{ a } \neg L \in I,$$

$$I(L) = \bot \text{ inak.}$$

Definícia 2.3.11 *Nech P je pozitívny rozšírený program a I je interpretácia. Potom,*

1 *pre každý literál $L \in \mathcal{L}_\mathcal{P}$*

   a $I \models L$ *práve vtedy, keď* $t \preceq I(L)$

   b $I \models \neg L$ *práve vtedy, keď* $f \preceq I(L)$

2 *pre ľubovolnú konjunkciu zagroundovaných literálov $G = L_1 \wedge \cdots \wedge L_n$, $I \models G$ práve vtedy, keď $I \models L_i$ pre všetky i $(1 \leq i \leq n)$*

3 *pre ľubovolnú zagroundovanú klauzulu $C = F \leftarrow G$, $I \models C$ práve vtedy, keď $I \models F$ alebo $I \not\models G$. Špeciálne, $I \models \leftarrow G$ práve vtedy, keď $I \not\models G$ a $I \models F \leftarrow$ práve vtedy, keď $I \models F$.*

Interpretácia I je *model* programu P ak $I \models C$ pre všetky zagroundované klauzuly C programu P. Usporiadanie $\preceq$ na prvdvostných hodnotách je taktiež definované aj medzi interpretáciami. Pre interpreácie I a J, $I \preceq J$ práve vtedy, keď $I(L) \preceq J(L)$ pre všetky $L \in \mathcal{L}_\mathcal{P}$. Usporiadania $\succeq$, $\prec$ a $\succ$ sú definované obvyklým spôsobom. Na interpretáciu je možné pozerať sa aj ako na množinu, ktorej generátor je interpretácia (definovaná ako funkcia), preto

$I \preceq J$ práve vtedy, keď $I \subseteq J$. Model I je *minimálny* ak neexistuje model J taký, že $J \prec I$. Model I je *najmenší* ak $I \preceq J$ pre všetky modely J. Na rozlíšenie týchto pojmov od štandartného logického programovania budeme minimálny/najmenší model volať *parakonzistentný minimálny/najmenší model* (skrátene *p-minimálny/p-najmenší model*). Model budeme volať *konzistentným modelom* ak $I(L) \neq \top$ pre všetky $L \in \mathcal{L}_\mathcal{P}$, inak bude I *nekonzistentný model*. Program je *konzistentný* ak má konzistentný model, inak je *nekonzistentný*.

Propozícia 2.3.1 *Ak pozitívny rozšírený program má model, tak má aspoň jeden p-minimálny model.*

Propozícia 2.3.2 *Konzistentný pozitívny rozšírený program má konzistentný p-minimálny model.*

Definícia 2.3.12 *Rozšírený disjunktívny program je konečná množina klauzúl formy:*

$$L_1 \leftarrow L_2 \wedge \cdots \wedge L_m \wedge notL_{m+1} \wedge \cdots \wedge notL_n \ (n \geq m \geq 2)$$

kde L_i sú literály a not reprezentuje defaoultovú negáciu*. Pre potreby defaultovej negácie rozšírime definíciu 2.3.11 o tieto dve vlastnosti:*

- $I \models notL$ *práve vtedy, keď* $I(L) \preceq f$
- $I \models not\neg L$ *práve vtedy, keď* $I(L) \preceq t$

Samozrejme nestačí obmedziť sa iba na pozitívne programy.

Definícia 2.3.13 *Nech P je rozšírený program a I je interpretácia.* Redukt *programu P vzhľadom na I je pozitívny rozšírený disjunktívny program P^I taký, že klauzula*

$$L_1 \leftarrow L_2 \wedge \cdots \wedge L_m$$

je v P^I práve vtedy, keď existuje zagroundovaná klauzula

$$L_1 \leftarrow L_2 \wedge \cdots \wedge L_m \wedge notL_{m+1} \wedge \cdots \wedge notL_n$$

z P taká, že $\{L_{m+1}, \ldots, L_n\} \cap I = \emptyset$. Potom I nazývame parakonzistentný stabilný model (skrátene p-stabilný model) programu P ak I je p-minimálny model P^I.

Propozícia 2.3.3 *P-stabilný model je p-minimálny model.*

Keď program obsahuje nekonzistentnú informáciu, je užitočné vedieť rozlíšiť fakty ovplyvnené takouto informáciou od ostatných. Preto boli pridané dve pravdivostné hodnoty *st* a *sf* predstavujúce podozrivo pravdivý (suspiciously true) a podozrivo nepravdivý (suspiciously false). Tieto dve rpavdivostné hodnoty rozšírili zväz IV na zväz VI pričom $\bot \preceq sx \preceq x \preceq \top$ kde $x \in \{t, f\}$.

Nech $\mathcal{L}_P^f = \mathcal{L}_P \cup \{L^s \mid L \in \mathcal{L}_P\}$ a I^s je podmnožina $\mathcal{L}_P^f$, kde L^s označuje podozrivý literál.

Definícia 2.3.14 *Interpretácia je funkcia $I^s : \mathcal{L}_P^f \to VI$ taká, že pre každý literál $L \in \mathcal{L}_P$ plati, že $I^s(L) = lub\{x = t \ ak \ L \in I^s, \ x = f \ ak \ \neg L \in I^s, \ x = st \ ak \ L^s \in I^s, \ x = sf \ ak \ \neg L^s \in I^s, \ x = \bot \ inak\}$.*

Takže pravdivostná hodnota každého literálu $I^s(L)$ je definovaná ako najmenšia horná hranica (*lub*) každej hodnoty x, ktorá je určená výskytom L v I^s. Preto $I^s(L) = \top$ práve vtedy, keď $L \in I^s$ a $\neg L \in I^s$ alebo $L^s \in I^s$ a $\neg L^s \in I^s$ alebo $L^s \in I^s$ a $\neg L \in I^s$ alebo $L \in I^s$ a $\neg L^s \in I^s$. Treba si všimnúť, že $I^s(L) = st$ práve vtedy, keď $I^s(\neg L) = sf$. Pri logike VI je podporenie literálov a feaultovej negácie definované nasledovne:

- $I^s \models L$ práve vtedy, keď $st \preceq I^s(L)$
- $I^s \models \neg L$ práve vtedy, keď $sf \preceq I^s(L)$
- $I^s \models notL$ práve vtedy, keď $I^s(L) \preceq f$
- $I^s \models not\neg L$ práve vtedy, keď $I^s(L) \preceq t$

Podporenie klauzúl je rovnaké ako predtým.

Veta 2.3.4 *Nech P je rozšírený program, potom podozrivý* p-stabilný *model je model P.*

2.4 Dynamické Logické Programy

2.4.1 Dynamické logické programy

Na opísanie zmien sveta nestačia statické programy. Jeden zo spôsobov ako sa s týmto problémom vyrovnať je prostredníctvom aktualizácií programov čo viedlo k paradigme *Dynamické Logické Programovanie* (DLP). DLP je postupnosť logických programov, kde každý reprezentuje časový interval (stav) a obsahuje znalosti, ktoré by mali byť v danom stave pravdivé. Od predstavenia aktualizácií vzniklo niekoľko sémantík, popísaných v [10][1][2][8], s cieľom prekonať nedostatky predchádzajúcich. My si zhrnieme 4 sémantiky: *Dynamické stabilné modely* (DSM), *Dynamické odôvodnené aktualizácie* (DJU), *Rafinované dynamické stabilné modely* (RDSM) a *Rafinované dynamické odôvodnené aktualizácie* (RDSM).

Spoločnou črtou týchto sémantík pre aktualizácie je, že novšie pravidlo, ktoré je v konflikte so starším, ho môže zamietnuť, aby sa predišlo nekonzistencii. Dve pravidlá r a r' sú konfliktné, označíme $r \bowtie r'$ práve vtedy, keď $Head(r) = not\, Head(r')$.

Definícia 2.4.1 *Nech* $\mathcal{P} = (P_1, \ldots, P_n)$ *je DLP a* M *je interpretácia, potom:*

$$Rejected(\mathcal{P}, M) = \{r \mid r \in P_i, \exists r' \in P_j, i < j, r \bowtie r', M \models Body(r')\}$$

$$Rejected^{+}(\mathcal{P}, M) = \{r \mid r \in P_i, \exists r' \in P_j, i \leq j, r \bowtie r', M \models Body(r')\}$$

Kde Body(r) *predstavuje telo pravidla* r.

Množiny *Rejected* a $Rejected^{+}$ sú dve verzie zamietania pravidiel.

Definícia 2.4.2 *Nech* $\mathcal{P} = (P_1, \ldots, P_n)$ *je DLP a* M *je interpretácia, potom:*

$$Defaults(\mathcal{P}, M) = \{not\, L \mid \nexists r \in \rho(\mathcal{P}), Head(r) = L, M \models Body(r)\}$$

$$Defaults^{*}(\mathcal{P}, M) = \{not\, L \mid L \notin M\}$$

Kde Body(r) *predstavuje telo pravidla* r *a* Head(r) *predstavuje hlavu pravidla* r. $\rho(\mathcal{P})$ *je multimnožina obsahujúca všetky pravidlá v* $P_1, \ldots, P_n$.

Definícia 2.4.3 *Nech* $\mathcal{P} = (P_1, \ldots, P_n)$ *je DLP a* M *je interpretácia, potom:*

DSM M *je* dynamický stabilný model $\mathcal{P}$ *práve vtedy keď* $M' = least(\rho(\mathcal{P}) \setminus Rejected(\mathcal{P}, M) \cup Defaults(\mathcal{P}, M))$

DJU M *je* dynamická odôvodnená aktualizácia $\mathcal{P}$ *práve vtedy keď* $M' = least(\rho(\mathcal{P}) \setminus Rejected(\mathcal{P}, M) \cup Defaults^*(\mathcal{P}, M))$

RDSM M *je* rafinovaný dynamický stabilný model $\mathcal{P}$ *práve vtedy keď* $M' = least(\rho(\mathcal{P}) \setminus Rejected^+(\mathcal{P}, M) \cup Defaults(\mathcal{P}, M))$

RDJU M *je* rafinovaná dynamická odôvodnená aktualizácia $\mathcal{P}$ *práve vtedy keď* $M' = least(\rho(\mathcal{P}) \setminus Rejected^+(\mathcal{P}, M) \cup Defaults^*(\mathcal{P}, M))$

Kde $\rho(\mathcal{P})$ *je multimnožina obsahujúca všetky pravidlá v* $P_1, \ldots, P_n$.

2.4.2 Sémantiky multi-dimenzionálnych dynamických logických programov založené na stabilných modeloch

Keďže postupnosť updatov, z predchádzajúcej časti, je lineárna, ľahko sa s ňou pracuje, ale má svoje obmedzenia a preto bola rozšírená [9], aby sme sa zbavili tejto lineárnosti.

Definícia 2.4.4 *Nech* $D = (V, E)$ *je acyklický graf. Nech* $v \in V$. graf relevantnosti *grafu* D *vzhľadom na* v *je* $D_v = (V_v, E_v)$ *kde:*

- $V_v = \{v_i : v_i \in a\ , v_i \leq v\}$
- $E_v = \{(v_i, v_j) : (v_i, v_j) \in E\ a\ v_i, v_j \in V\}$

Intuitívne graf relevantnosti *grafu* D *je podgraf grafu* D*, obsahujúci všetky vrcholy a hrany vo všetkých cestách smerujúcich do* v.

Definícia 2.4.5 Multi-dimenzionálny Dynamický Logický Program, $\mathcal{P}$, *je dvojica* $(\mathcal{P}_D, D)$, *kde* $D = (V, E)$ *je acyklický orientovaný graf a* $\mathcal{P}_D = \{P_v : v \in V\}$ *je množina zovšeobecnených logických programov indexovaných vrcholmi* $v \in V$ *grafu* D.

Definícia 2.4.6 *Nech* $(\mathcal{P}_D, D)$ *je* Multi-dimenzionálny Dynamický Logický Program, *kde* $\mathcal{P}_D = \{P_v : v \in V\}$ *a* $D = (V, E)$. *Interpretácia* M_s *je stabilný model multi-dimenzionálneho updatu v stave* $s \in V$ *práve vtedy, keď:*

- $M_s = least([\mathcal{P}_s \setminus Reject(s, M_s)] \cup Default(\mathcal{P}_s, M_s))$

kde

- $\mathcal{P}_s = \bigcup_{i \leq s} P_i$
- $Reject(s, M_s) = \{r \in P_i | \exists r' \in P_j, i \leq j \leq s, head(r) = not\, head(r') \wedge M_s \models body(r')\}$
- $Default(\mathcal{P}_s, M_s) = \{notA |\ \nexists r \in \mathcal{P}_s : (head(r) = A) \wedge M_s \models body(r)\}$

2.4.3 Sémantika dobre-podporených (Well-supported) modelov pre multidimenzionálne dynamické logické programy

Sémantika dobre podporených modelov pre multidimenzionálne dynamické logické programy bola popísaná v [3].

Definícia 2.4.7 *Nech* $\mathcal{MP}$ *je ľubovolný* MDyLP *nad jazykom* $\mathcal{L}$, *nech* P_n *je update* $\mathcal{MP}$ *a nech* ℓ *je ľubovolný mapovacia funkcia nad* $\mathcal{L}$ *a nech* M *je ľubovolná 2-hodnotová interpretácia nad* $\mathcal{L}$. *Konečná množina zamietnutých pravidiel s ohľadom na* ℓ *taká, že*

$$Rej_l(\mathcal{MP}, M, n) = \{\tau \in P_i | \exists \eta \in P_j : i \prec j \preceq n, \tau \bowtie \eta, M \models body(\eta), \ell(head(\eta)) > \ell(B(\eta))\}$$

Definícia 2.4.8 *Nech $\mathcal{MP}$ je ľubovolný* MDyLP *nad nejakým jazykom $\mathcal{L}$ a nech* M *je interpretácia nad $\mathcal{L}$, P_n je update $\mathcal{MP}$. Hovoríme, že* M *je well-supported model P_n práve vtedy, keď existuje mapovacia funkcia (level maping) ℓ nad $\mathcal{L}$ taký, že i)* M *je model $p(P)\backslash Rej(M,n)$ a ii) $\forall A \in M \exists \tau \in p(P)\backslash Rej(M,n)$ také, že $head(\tau) = A, \ell(A) > B(\tau)$ a τ je podporené* M.

Veta 2.4.1 *Nech $\mathcal{MP}$ je ľubovolný* MDyLPs *nad jazykom $\mathcal{L}$, P_n je update $\mathcal{MP}$ a* M *je well-supported model $\mathcal{MP}$ v P_n. Potom* M *je tiež model $\mathcal{MP}$ v ľubovolnej sémantike definovanej v (DyLP, MDyLP, Disjunctive logic programs with inheritance)*

Kapitola 3

Prínos

Zavedenie aktualizácií pre logické programy spôsobilo viac priestoru pre nekonzistenciu spôsobenú zmenou konfliktom pravidiel, ktoré mohli vychádzať z rôznych okolností. Toto sa zvykne riešiť revíziou znalostí teda detekovaním konfliktných pravidiel a zamietnutím tých z konfliktu, ktoré majú menšiu prioritu. Avšak tento prístup sa vie vysporiadať iba s konfliktami, kde vieme jednotlivým pravidlám vieme priradiť prioritu a vieme určiť, ktoré z konfliktných pravidiel je menej prioritné. Ak z nejakého dôvodu toto nie je možné, nevieme určiť ktoré pravidlo zamietnúť a konfliktu sme nepredišli. Sémantiky zaoberajúce sa dynamickým logickým programovaním v takomto prípade zahodia model, pretože by obsahoval nekonzistenciu. Chceme preto sémantiku ktorá, pokiaľ je to možné, predíde konfliktom zamietaním menej prioritných pravidiel a vprípade konfliktov, ktorým nemožno týmto spôsobom predísť bude pokračovať vo výpočte modelu aj za cenu, že tento model bude obsahovať nekonzistenciu. Aby toto bolo možné, nemôžeme sa obmedziť na dvojhodnotovú logiku, keďže potrebujeme túto nekonzistenciu reprezentovať. Toto nám umožní získať aj výsledky ktoré nemusia byť ovplyvnené lokálnou nekonzistenciou. Taktiež je možné takto získať aj výsledky, pri ktorých sa v odvodení nekonzistencia objavila, ale tieto výsledky môžu byť tiež zaujímavé.

Majme programy P_1 a P_2

kde $P_1 = \{A \leftarrow B,\ C \leftarrow\}$

kde $P_2 = \{notA \leftarrow B,\ D \leftarrow,\ B \leftarrow\}$

Keby sme hovorili o modeloch programov P_1 a P_2 tak zjavne vyzerajú takto $M_1 = \{C\}$ a $M_2 = \{B,\ D\}$. Avšak ak sa na tieto dva programy pozrieme ako na postupnosť $\mathcal{P} = \{P_1,\ P_2\}$ tak výsledný model nebude existovať lebo A by mal byť pravdivý a zároveň nepravdivý, čím sme však stratili informácie o tom že B, C a D sú pravdivé. Bolo by pekné mať možnosť zachovať túto informáciu a teda zachovať takýto model aj za ceny nekonzistencie, preto potrebujeme sémantiku pracujúcu s programami a ich aktualizáciami a zároveň schopnú počítať s nekonzistenciou. Takže by sme chceli nejako spracovať aktualizácie a zároveň predísť nekonzistencii, ktorej sa vieme zbaviť. Následne získať model, ktorý bude potencionálne obsahovať nekonzistenciu, ktorej sme sme nevedeli predísť. Pre jednoduchosť celý proces rozdelíme na dve časti:

- Spracovanie aktualizácií predchádzaním nekonzistencii tam, kde to vieme. Spracovaním aktualizácií zároveň získame jeden program bez aktualizácií ako medzivýsledok.

- Hľadanie modelu programu, ktorý sme získali v predchádzajúcom kroku ako medzivýsledok, pričom tento model nemusí byť konzistentný.

Okrem jednoduchosti a prehľadnosti nám toto rozdelenie umožní aj to, že môžeme spracovávanie aktualizácií prispôsobiť našim potrebám.

Predchádzanie nekonzistencii spočíva v revízii pravidiel v aktualizáciách a zamietaním pravidiel v prípade konfliktu. Aby sme nestratili informáciu, ktorá je reprezentovaná pravidlom, zamietame vždy iba jedno pravidlo z konfliktu a vždy to, ktoré má menšiu prioritu. V prípade *DSM*, *DJU*, *RDSM* a *RDJU*, kde máme postupnosť aktualizácií, je priorita pravidiel určená poradím v postupnosti, teda pravidlo z novšej aktualizácie má vyššiu prioritu. V prípade multidimenzionálných dynamických logických programov založených na stabilných modeloch alebo dobre podporených modeloch dostávame informáciu o priorite z orientovaného acyklického grafu, v ktorom vrcholy predstavujú jednotlivé logické programy. Ak z programu P_i vedie cesta do programu P_j tak má program P_j vyššiu prioritu ako P_i (budeme to označovať

$P_i \leq_D P_j$, kde D je orientovaný acyklický graf).

Po zamietnutí všetkých konfliktných pravidiel s nižšou prioritou sme predišli konfliktom kde sme vedeli porovnať prioritu pravidiel. Avšak predchádazjúci krok nevie riešiť situáciu ak jedna aktualizácia obsahovala konfliktné pravidlá alebo ak sme pri konfliktných pravidlách nevedeli porovnať prioritu (ak v grafe neexistuje cesta z P_i do P_j a ani opačne). V tom prípade sa nekonzistencii v modeli nemusíme vedieť vyhnúť a teda daný model by sme bežne zamietli. Takýto zamietnutý model však môže obsahovať informácie, ktoré cheme a preto chceme ten model zachovať aj napriek tomu že obsahuje nekonzistenciu. Na to však treba inú sémantiku, aby sa z nekonzistenciou vedela vysporiadať. Našli sme viaceré prístupy, ktoré vyzerali pre tonto účel použiteľne a z nich dva boli veľmi vhodné a dobre sa s nimi pracovalo:

Dvojzväzy popísané v 2.3.1. Dvojzväzy majú veľmi pekné vlastnosti a dve usporiadania uľahčili prácu. Je tu však tendencia vracať aj modely obsahujúce nekonzistenciu aj v prípadoch kde by sme sa chceli takým modelom vyhnúť. Ako príklad môžeme uviesť jednoduchý program $P = \{A \leftarrow notB,\ B \leftarrow notA\}$. Modely ktoré získame sú $M_1 = \{A,\ notA,\ B,\ notB\}$, $M_2 = \{A,\ notB\}$, $M_3 = \{notA,\ B\}$, $M_4 = \emptyset$, pričom v M_4 majú oba literály hodnotu *nedefinované* alebo $\perp$. Modely M_2 a M_3 sú stabilné modely tohto programu a niekedy je žiadúci výsledok aj model M_4, avšak model M_1 je v praxi menej preferovaný, keďže nám dáva až primoc veľa informácií. Nekonzistencii v modeli M_1 hovoríme, že je nevynútená a je preferencia získavať modely, ktoré neobsahujú nevynútenú nekonzistenciu. Keďže ich výsledkom sú aj také modely, ktoré nie sú pre naše účely potrebné a chceli by sme im predísť a nepodarilo sa nájsť spôsob, ktorým by sme sa zbavili nevynútenej nekonzistencie, rozhodli sme sa pre druhú možnosť.

Druhou možnosťou sú parakonzistentné stabilné modely popísané v 2.3.2. Táto sémantika vychádza zo stabilných modelov a dalo by sa povedať že stabilné modely rozširuje tak, aby vedeli pracovať s nekonzistenciou. Podobne ako pri stabilných modeloch, aj tu sa urobí redukt, ku ktorému sa nájde model. Vzhľadom na podobnosť so stabilnomodelovou sémantikou sa parakonzistentné stabilné modely javia byť presne to čo potrebujeme.

3.1 Zavedenie sémantiky

V tejto časti si ukážeme detaily, teda samotné definície. Budeme potrebovať viac pravdivostných hodnôt ako iba *pravda* a *nepravda* a najmenší rozumný dvojzväz ktorý môžeme použiť je $IV = \{t, f, \top, \bot\}$, kde $t, f, \top, \bot$ označujú *pravda, nepravda, kontradikcia, nedefinované*. Keďže hovoríme o dvojzväze potrebujeme spomenúť aj dve usporiadania. V tomto prípade to budú $\leq_t$ a $\leq_k$ teda pravdivostné a znalostné usporiadanie, ktoré naše pravdivostné hodnoty usporiadajú nasledovne $f \leq_t \bot, \top \leq_t t$ a $\bot \leq_k f, t \leq_k \top$. Usporiadania definujeme $<_t$, $\geq_t$, $>_t$, $<_k$, $\geq_k$ a $>_k$ obvyklým spôsobom. Výhodou použitia dvojzväzu je to, že nemúsíme mať iba tieto pravdivostné hodnoty, ale môžeme použiť aj iný dvojzväz s rovnakými vlastnosťami a príslušne upraviť interpretáciu a valuáciu a revízia znalostí, teda zamietanie konfliktných pravidiel, bude fungovať aj bez zmeny. Taktiež nikde nevyžadujeme aby bol počet pravdivostných hodnôt konečný. My použijeme IV pre jednoduchosť a lepšiu prehladnosť.

Pod interpretáciou budeme rozumieť množinu $I \subseteq (\mathcal{L} \cup \overline{\mathcal{L}})$, kde $\overline{\mathcal{L}} = \{\overline{L} \mid L \in \mathcal{L}\}$ a $\overline{L}$ reprezentuje nepravdivý literál L.

Definícia 3.1.1 *Nech* I *je interpretácia. Valuácia* val_I *je funkcia* $val_I : \mathcal{L}_\mathcal{P} \to IV$ *taká, že pre každý literála* $L \in \mathcal{L}_\mathcal{P}$ *platí:*

- $val_I(L) = t$ *ak* $L \in I$ *a* $\overline{L} \notin I$
- $val_I(L) = f$ *ak* $L \notin I$ *a* $\overline{L} \in I$
- $val_I(L) = \top$ *ak* $L \in I$ *a* $\overline{L} \in I$
- $val_I(L) = \bot$ *inak*
- $val_I(L_1 \wedge \cdots \wedge L_n) = min_t\{val_I(L_1), \ldots, val_I(L_n)\}$, *kde* min_t *je minimum vzhľadom na usporiadanie* $\leq_t$
- $val_I(A \leftarrow B) = t$ *ak* $val_I(B) \leq_t val_I(A)$

Definícia 3.1.2 *Nech* I *je interpretácia, potom pre každý literál* $L \in \mathcal{L}_\mathcal{P}$ *platí, že:*

- $I \models L$ *práve vtedy, keď* $t \leq_k val_I(L)$ *a zároveň* $\top \leq_t val_I(L)$
- $I \models notL$ *práve vtedy, keď* $val_I(L) \leq_k f$ *a zároveň* $val_I(L) \leq_t \top$

Chceme reprezentovať štyri pravdivostné hodnoty, takže nebude stačiť v interpretácii mať iba pravdivé literály a predpokladať, že ostatné budú nepravdivé. Preto budeme v interpretácii mať literál A ak je tento literál pravdivý a literál $\overline{B}$ ak je literál B nepravdivý. Podobne pre nekonzistentný ($\top$) literál C bude v interpretácii C aj $\overline{C}$ a ak bude literál D mať hodnotu nedefinované alebo $\bot$ sa nebude vyskytovať v interpretácii v ani jednej podobe.

Keďže jeden program môže mať viac interpretácií, veľmi by sa nám hodila možnosť provnať a usporiadať tieto interpretácie. Preto si teraz zavedieme usporiadanie na interpretáciách. Pri stabilných modeloch sme preferovali minimálny model vzhľadom na inklúziu, čo znamenalo že sme preferovali množinu, ktorá obsahovala menej literálov, teda menej pravdivých literálov, keďže ostatné literály boli nepravdivé. Chceli sme mať v modeli pravdivé iba tie literály, ktoré vieme podporiť pravidlami. To isté by sme radi dosiahli aj teraz a preto usporiadame interpretácie vzhľadom na pravdivostné usporiadanie $\leq_t$.

Definícia 3.1.3 *Nech* I *je interpretácia, potom:*

- $I^+ = \{L \mid L \in I\}$
- $I^- = \{L \mid \overline{L} \in I\}$

Tieto dve množiny nám poslúžia na porovnanie intepretácií. Množina I^+ obsahuje literály, ktoré sú pravdivé v I a podobne množina I^- obsahuje literály, ktoré sú nepravdivé v I.

Definícia 3.1.4 *Nech* I *a* J *sú interpretáciu, potom hovoríme, že* $I \preceq_t J$ *ak:*

- $I^+ \subseteq J^+$
- $J^- \subseteq I^-$

Usporiadania definujeme $\prec_t$, $\succeq_t$ a $\succ_t$ obvyklým spôsobom. Podobne si pripravíme aj usporiadanie interpretácií vzhľadom na znalostné usporiadanie $\leq_k$.

Definícia 3.1.5 *Nech* I *a* J *sú interpretáciu, potom hovoríme, že* $I \preceq_k J$ *ak:*

- $I^+ \subseteq J^+$
- $I^- \subseteq J^-$

Usporiadania definujeme $\prec_k$, $\succeq_k$ a $\succ_k$ obvyklým spôsobom.

Keď sme celý proces rozdelili na spracovanie aktualizácií a hľadanie modelu, spomínali sme, že toto rozdelenie nám umožní byť flexibilnejší pri výbere spôsobu, akým budeme riešiť konflikt pravidiel. Preto si uvedieme viacero definícií, ako ukážku ako si môžeme zvoliť sémantiku, ktorá bude základom pre zamietanie konfliktných pravidiel. Pre ukážku sme zvolili *DSM*, *DJU*, *RDSM*, *RDJU*, *MDyLP* založené na *stabilných modeloch* a *MDyLP* založené na *dobre podporených modeloch.*

Definícia 3.1.6 *Nech I je interpretácia a* $\mathcal{P}$ *je dynamický logický program, teda program a jeho aktualizácie, potom:*

- $Defaults(\mathcal{P}, I) = \{not\, L \mid \nexists r \in P,\ Head(r) = L,\ M \models Body(r),\ P \in \mathcal{P}\}$

Definícia 3.1.7 *Nech I je interpretácia a* $\mathcal{P}$ *je dynamický logický program, teda program a jeho aktualizácie, potom:*

- $Defaults^*(\mathcal{P}, I) = \{not\ L \mid L \notin I,\ \overline{L} \in I\}$

Definícia 3.1.8 *Nech I je interpretácia a* $\mathcal{P}$ *je dynamický logický program, teda program a jeho aktualizácie, potom:*

- $Rejected(\mathcal{P}, I, i) = \{r \in P_i \mid \exists r' \in P_j,\ i < j,\ Head(r) \bowtie Head(r'),\ val_I(notBody(r')) <_t val_I(Body(r)),\ P_i, P_j \in \mathcal{P}\}$

Definícia 3.1.9 *Nech I je interpretácia a $\mathcal{P}$ je dynamický logický program, teda program a jeho aktualizácie, potom:*

- $Rejected^{+}(\mathcal{P}, I, i) = \{r \in P_i \mid \exists r' \in P_j,\ i \leq j,\ Head(r) \bowtie Head(r'),\ val_I(notBody(r')) <_t val_I(Body(r)),\ P_i, P_j \in \mathcal{P}\}$

Definícia 3.1.10 *Nech $D = (V, E)$ je orientovaný acyklický graf a nech $\mathcal{P}_{\mathcal{D}} = \{P_v \mid v \in V\}$ je množina zovšeobecnených logických programov, potom:*

- $Rejected^{*}(\mathcal{P}, I, i) = \{r \in P_i \mid \exists r' \in P_j,\ i <_D j,\ Head(r) \bowtie Head(r'),\ val_I(notBody(r')) <_t val_I(Body(r)),\ P_i, P_j \in \mathcal{P}_{\mathcal{D}}\}$ *kde $i <_D j$ označuje, že existuje cesta z vrchola i do vrchola j v grafe D*

Definícia 3.1.11 *Majme fukncIu $\ell : (L) \to \mathbb{N}$ takú, že $\ell(L) = \ell(notL)$. Nech $C = L_1, \ldots, L_n$, rozšírime ℓ tak, že $\ell(C) = max(\{\ell(L_i) \mid L_i \in C\})$ a pre jednoduchosť prázdnej konjunkcii literálov priradíme -1. Nech $D = (V, E)$ je orientovaný acyklický graf a nech $\mathcal{P}_{\mathcal{D},\ell} = \{P_v \mid v \in V\}$ je množina zovšeobecnených logických programov, potom:*

- $Rejected_{\ell}(\mathcal{P}_{\mathcal{D}}, I, i) = \{r \in P_i \mid \exists r' \in P_j,\ i <_D j,\ Head(r) \bowtie Head(r'),\ val_I(notBody(r')) <_t val_I(Body(r)),\ \ell(Head(r') > \ell(Body(r'),\ P_i, P_j \in \mathcal{P}\}$

Funkcia $Head(r)$ už podla názvu vráti hlavu pravidla teda lavú časť pravidla. Formálnejšie nech r je pravidlo tvaru $L_1 \leftarrow L_2 \wedge \cdots \wedge L_n$, potom $Head(r) = L_1$. Funkcia $Body(r)$ už podla názvu vráti Telo pravidla teda pravú časť pravidla. Formálnejšie nech r je pravidlo tvaru $L_1 \leftarrow L_2 \wedge \cdots \wedge L_n$, potom $Body(r) = L_2 \wedge \cdots \wedge L_n$. operácia $\bowtie$ vyjadruje konflikt pravidiel presnejšie konflikt ich hláv, teda $Head(r) \bowtie Head(r')$ práve vtedy, keď $Head(r) = notHead(r')$.

To, že sú hlavy dvoch pravidiel v konflikte ešte neznamená že musíme jedno z pravidiel zamietnuť. To či budeme zamietať záleží aj od tiel pravidiel, presnejšie od toho, akú hodnotu majú $val_I(Body(r))$ a $val_I(Body(r'))$. Telá konfliktných pravidiel potrebujeme porovnať, aby sme vedeli, či chceme

menejprioritné pravidlo zamietnuť. Toto porovnanie a teda aj celá podmienka by mala byť zovšeobecnením podmienky používanej v sémantikách *DLP* a *MyDLP*. Taktiež by sme chceli, aby táto myšlienka nebola našitá na dvojzväz *IV*, keďže aj v tomto smere by sme chceli zachovať flexibilitu. Myšlienka, ku ktorej sme dospeli je, že menejprioritné pravidlo chceme zamietnuť ak došlo k nelogickej situácii, napríklad keby horná hranica intervalu bola menšia ako dolná hranica intervalu. Teraz sa pozrime ako sme dostali podmienku $val_I(notBody(r')) <_t val_I(Body(r))$

(1) z podmienky $Head(r) \bowtie Head(r')$ dostaneme $Head(r) = notHead(r')$ a kedže $notnotA = A$ tak je jedno v hlave, ktorého pravidla sa negácia nachádzala

(2) $Head(r) \leftarrow Body(r)$ paltí ak $val_I(Body(r)) \leq_t val_I(Head(r))$

(3) z (1) a (2) dostaneme, že aj $val_I(Body(r)) \leq_t val_I(notHead(r'))$

(4) $val(notL) = notval(L)$

(5) z (3) a (4) dostaneme $val_I(Head(r')) \leq_t val_I(notBody(r))$

(6) a teda $val_I(Body(r)) \leq_t val_I(Head(r)) \leq_t val_I(notBody(r'))$

(7) takže $val_I(notBody(r'))$ a $val_I(Body(r))$ tvoria horný a dolný odhad pre $val(Head(r))$ a pravidlo zamietneme ak bude horný odhad menší ako dolný odhad teda ak $val_I(notBody(r')) <_t val_I(Body(r))$

Povedali sme, že v prvom kroku je možné vybrať si sémantiku, podla ktorej budeme zamietať konfliktné pravidlá a preto máme viac verzií množín *Defaults* a *Rejected*. Treba ich však skombinovať a dostať medzivýsledok, teda program P', ktorému budeme v ďalšom kroku hľadať model:

- *DSM*:
  $Residue(\mathcal{P}, I) = \bigcup_{1 \leq i \leq n} [P_i \setminus Rejected(\mathcal{P}, I, i)]$, kde n je počet programov v $\mathcal{P}$ a $P' = Residue(\mathcal{P}, I) \cup Defaults(\mathcal{P}, I)$

- *DJU*:
  $Residue(\mathcal{P}, I) = \bigcup_{1 \leq i \leq n} [P_i \setminus Rejected(\mathcal{P}, I, i)]$, kde n je počet programov v $\mathcal{P}$ a $P' = Residue(\mathcal{P}, I) \cup Defaults^*(\mathcal{P}, I)$

- *RDSM*:
  $Residue(\mathcal{P}, I) = \bigcup_{1 \leq i \leq n} [P_i \setminus Rejected^+(\mathcal{P}, I, i)]$, kde n je počet programov v $\mathcal{P}$ a $P' = Residue(\mathcal{P}, I) \cup Defaults(\mathcal{P}, I)$

- *RDJU*:
  $Residue(\mathcal{P}, I) = \bigcup_{1 \leq i \leq n} [P_i \setminus Rejected^+(\mathcal{P}, I, i)]$, kde n je počet programov v $\mathcal{P}$ a $P' = Residue(\mathcal{P}, I) \cup Defaults^*(\mathcal{P}, I)$

- *MDyLP* založené na *stabilných modeloch*:
  $Residue^*(\mathcal{P}_\mathcal{D}, I) = \bigcup_{v \in V} (\bigcup_{i \leq_D v} [P_i \setminus Rejected(\mathcal{P}_\mathcal{D}, I, i)])$ a $P' = Residue(\mathcal{P}_\mathcal{D}, I) \cup Defaults(\mathcal{P}_\mathcal{D}, I)$

- *MDyLP* založené na *dobre podporených modeloch*:
  $Residue_\ell(\mathcal{P}_\mathcal{D}, I) = \bigcup_{v \in V} (\bigcup_{i \leq_D v} [P_i \setminus Rejected(\mathcal{P}_\mathcal{D}, I, i)])$ a keďže táto sémantika nepoužíva množinu *Defaults*, tak $P' = Residue(\mathcal{P}_\mathcal{D}, I)$

V druhej časti chcem nájsť model programu P', ktorý sme dostali ako medzivýsledok po skončení prvej časti. Keďže tento program môže obsahovať konflikty pravidiel, ktoré nebolo možné vyriešiť v prvej časti (napríklad ak sme nevedeli určiť menej prioritné pravidlo), možeme dostať nekonzistentný model. Preto sa budeme opierať o parakonzistentné stabilné modely spomínané v 2.3.2. Program, ktorý máme môže obsahovať pravidlá s negáciou v hlave a tomu sa budeme venovať ako prvému. Z pravidla tvaru $notL \leftarrow L_1 \wedge \cdots \wedge L_n$ urobíme pravidlo $\leftarrow L \wedge L_1 \wedge \cdots \wedge L_n$ pričom hlava bude prázdna čo znamená, že pravidlo sa bude správať akoby hlava tohto pravidla bola nepravda čo znamená, že aby bolo pravidlo splnené telo nesmie byť pravdivé. Takto dostaneme podmienku, ktorú musí kandidát M na model spĺňať. Budeme hladať model pre program bez týchto podmienok a zamietneme všetkých kandidátov nespĺňajúce podmienky. V prípade, že v prvom kroku sme si pre zamietanie pravidiel zvolili *MDyLP* založené na

dobre podporených modeloch, musí kandidát M na model spĺňať aj $\forall L \in M$ (resp. $\forall \overline{L} \in M$) $\exists r \in P'$ také, že $Head(r) = L$ (resp. $Head(r) = notL$), $\ell(Head(r)) > \ell(Body(r))$ a $M \models r$. Urobíme redukt P'_M programu P' podľa kondidáta na model M, tak že:

- vynecháme všetky pravidlá obsahujúce $notL$ v tele ak $L \in M$
- vynecháme všetky $notL$ z tiel pravidiel také, že $\overline{L} \in M$
- všetky pravidlá uzemníme

Definícia 3.1.12 *Majme GLP program P. Hovoríme, že M je Parakonzistentný stabilný model pre DLP (resp. MDyLP) vtedy a len vtedy, keď M je model P'_M, ktorý vznikol ako redukt programu P a neexistuje model M' programu P'_M, taký, že $M' \prec_t M$.*

3.2 Nekonzistencia v odvodení

Pri práci s programami obsahujúcimi nekonzistentnú informáciu je vhodné vedieť rozlíšiť, či sa v odvodení literálu objavila nekonzistencia. V niektorých prípadoch môže byť informácia o pravdivosti takéhoto literálu menej dôveryhodná. Niekedy nie je žiadna informácia odvodená z nekonzistencie použitelná, ale sú situácie kedy nekonzistencia v odvodení nemení nič na závere, k akému sme odvodením dospeli. Keďže je toto všetko závislé od situácie, nechceme odvodenie obsahujúce nekonzistenicu a všetky výsledky tohto odvodenia zahadzovať, ale páčilo by sa nám keby sme mohli na skutočnosť, že nejaký výsledok sme bez nekonzistencie nevedeli dosiahnuť, mohli upozorniť. Preto sa pokúsime nájsť spôsob ako zachovať informáciu o príadnej nekonzistencii v odvodení. Môžeme sa pokúšať objaviť nekonzistenciu v odvodení počas hľadania modelu alebo vypočítať model a následne skúmať za akých okolností sme k jednotlivým záverom prišli. Ukážeme si dva spôsoby.

Prvý spôsob pridal nové pravdivostné hodnoty, ktoré budú reprezentovať pravdu a nepravdu avšak také, ku ktorým sme dospeli pomocou nekonzistencie. Teda literály budú fungovať ako doteraz v telách pravidiel a budú slúžiť rovnako na odvodenie iných literálov, ale keď sa objavia v odvodení tak aj

odvodený literál bude mať priradenú novú pravdivostnú hodnotu, pretože niekde v ich odvodení bola nekonzistencia.

Druhý spôsob nepridáva pravdivostné hodnoty, ale používa nové literály a nové pravidlá aby objavil nekonzistenciu v tele pravidiel. Tento spôsob nijako nemení odvodenia pôvodných literálov. Pre každý literál zisťuje, či existuje pravidlo, ktorého hlavou je daný literál, ktoré neobsahuje nekonzistenciu alebo iný menej dôveryhodný literál. Taktiež pre pravidlá zisťuje či v ich tele bola nekonzistencia alebo menej dôveryhodný literál. Aby toto všetko mohol zistiť, pre každé pravidlo vytvorí pomocný literál a pomocné pravidlo tvorené z nových literálov. Taktiež pre každý literál vytvorí pomocný literál, ktorý je používaný na určenie či príslušný pôvodný literál je menej dôveryhodný. Nový literál na to používa už spomínané pomocné literály prislúchajúce pravidlám.

Ako bolo spomenúté, prvý spôsob je pridať pravdivostné hodnoty, ktoré budú reprezentovať, že v odvodení bola nekonzistencia alebo menej dôveryhodný literál. Tieto pravdivostné hodnoty označíme f' a t' a dostaneme tak dvojzväz $VI = \{f, f^\top, t, t^\top, \bot, \top\}$ s usporiadaniami: $f \leq_t f^\top \leq_t \bot, \top \leq_t t^\top \leq_t t$ a $\bot \leq_k f^\top, t^\top \leq_k f, t \leq_k \top$. Usporiadania $<_t$, $\geq_t$, $>_t$, $<_k$, $\geq_k$ a $>_k$ definujeme obvyklým spôsobom. V interpretácii potrebujeme reprezentovať menej dôveryhodný literál a použijeme na to symboly $L^\top$ a $\overline{L^\top}$. Valuáciu upravíme nasledovne:

Definícia 3.2.1 *Nech $I^\top$ je interpretácia. Valuácia $val_{I^\top}^\top$ je funkcia $val_{I^\top}^\top : \mathcal{L}_\mathcal{P}^\top \to VI$, kde taká, že pre každý literála $L \in \mathcal{L}_\mathcal{P}$ platí:*

- $val_{I^\top}^\top(L) = t$ *ak* $L \in I^\top$ *ale* $L^\top \notin I^\top$
- $val_{I^\top}^\top(L) = f$ *ak* $\overline{L} \in I^\top$ *ale* $\overline{L^\top} \notin I^\top$
- $val_{I^\top}^\top(L) = t^\top$ *ak* $L^\top \in I^\top$
- $val_{I^\top}^\top(L) = f^\top$ *ak* $\overline{L^\top} \in I^\top$
- $val_{I^\top}^\top(L) = \top$ *ak* $L \in I \wedge \overline{L} \in I^\top$ *alebo* $L^\top \in I \wedge \overline{L^\top} \in I$ *alebo* $L \in I \wedge \overline{L^\top} \in I$ *alebo* $\overline{L} \in I \wedge L^\top \in I$

- $val^{\top}_{I^{\top}}(L) = \bot$ *ak* $L \notin I \ \wedge \ \overline{L} \notin I^{\top} \ \wedge \ L^{\top} \notin I \ \wedge \ \overline{L^{\top}} \notin I$
- $val^{\top}_{I^{\top}}(L_1 \wedge \cdots \wedge L_n) = min_t\{val^{\top}_{I^{\top}}(L_1), \ldots, val^{\top}_{I^{\top}}(L_n)\}$, *kde* min_t *je minimum vzhľadom na usporiadanie* $\leq_t$
- $val^{\top}_{I^{\top}}(A \leftarrow B) = t$ *ak* $val^{\top}_{I^{\top}}(B) \leq_t val^{\top}_{I^{\top}}(A)$

Definícia 3.2.2 *Nech* $I^{\top}$ *je interpretácia, potom pre každý literál* $L \in \mathcal{L}_{\mathcal{P}}$ *platí, že:*

- $I^{\top} \models L$ *práve vtedy, keď* $t^{\top} \leq_k val^{\top}_{I^{\top}}(L)$ *a zároveň* $\top \leq_t val^{\top}_{I^{\top}}(L)$
- $I^{\top} \models notL$ *práve vtedy, keď* $val^{\top}_{I^{\top}}(L) \leq_k f$ *a zároveň* $val^{\top}_{I^{\top}}(L) \leq_t \top$

Keďže sme rozšírili interpretáciu, pozmeníme aj usporiadania na interpretáciách.

Definícia 3.2.3 *Nech* $I^{\top}$ *je interpretácia, potom:*

- $I^{\top +} = \{L \mid L \in I^{\top} \vee L^{\top} \in I^{\top}\}$
- $I^{\top -} = \{L \mid \overline{L} \in I^{\top} \vee \overline{L^{\top}} \in I^{\top}\}$

Množina I^{+} obsahuje dôveryhodné aj menej dôveryhodné literály, ktoré sú pravdivé v I^{+} a podobne množina I^{-} obsahuje dôveryhodné aj menej dôveryhodné literály, ktoré sú nepravdivé v I^{+}.

Definícia 3.2.4 *Nech* $I^{\top}$ *a* $J^{\top}$ *sú interpretáciu, potom hovoríme, že* $I^{\top} \preceq^{\top}_{t} J^{\top}$ *ak:*

- $I^{\top +} \subseteq J^{\top +}$
- $J^{\top -} \subseteq I^{\top -}$

Usporiadania definujeme $\prec^{\top}_{t}$, $\succeq^{\top}_{t}$ a $\succ^{\top}_{t}$ obvyklým spôsobom.

Definícia 3.2.5 *Nech* $I^{\top}$ *a* $J^{\top}$ *sú interpretáciu, potom hovoríme, že* $I^{\top} \preceq^{\top}_{k} J^{\top}$ *ak:*

- $I^{\top +} \subseteq J^{\top +}$

- $I^{\top^-} \subseteq J^{\top^-}$

Usporiadania definujeme $\prec_k^\top$, $\preceq_k^\top$ a $\succ_k^\top$ obvyklým spôsobom.

Definícia 3.2.6 *Interpretácia M je parakonzistentný stabilný model pre DLP (resp. MDyLP) s dôveryhodnosťou vtedy a len vtedy, keď M je model P a neexistuje model M' programu P, taký, že $M' \prec_t M$.*

Druhý spomínaný spôsob je zaviesť pre každý literál L vyrobíme nový pomocný literál $L^\top$, ktorý bude slúžiť iba na udržiavanie informácie, či je literál L sám nekonzistentný, prípadne či sa v pravidlách, ktoré majú L v hlave pravidla, nachádza nekonzistentný literál, prípadne iný literál odvodený pomocou menej dôveryhodného literálu. Nechcem však uvažovať pravidlo r s L v hlave pravidla, ktoré nemá podporené telo, hoci obsahuje nekonzistenciu, alebo menej dôveryhodný literál. Pravidlo r, by mohlo spôsobiť, že literál L, na základe pomocného literálu $L^\top$, bude označený za menej dôveryhodný, hoci pravidlo r nie je pre odvodenie L dôležité. Aby sme predišli takejto situácii, ponúkajú sa dve takmer rovnaké možnosti. Potrebujeme dôveryhodnoť literálu určovať na základe pravidiel, ktoré sa podielajú na jeho odvodení, teda takých, ktoré sú splnené. Môžeme počkať na výpočet modelov pôvodného programu a na základe výsledného modelu určiť relevantné pravidlá vzhľadom na model a iba tieto pravidlá použiť na určenie dôveryhodnosti literálov. Druhá možnosť je tá, že použijeme interpretáciu, ktorá je kandidátomna model, na vygenerovanie našich pomocných literálov a pravidiel už počas hľadania modelu po spracovaní aktualizácií a zamietaní konfliktných pravidiel s nízkou prioritou. Ide o ten istý postup a vlastne tú istú metódu a rozdiel je v tom, či počítame modely dvoch programov alebo model jedného väčšieho programu.

Definícia 3.2.7 *Nech program P' je program získaný (ako medzivýsledok) po zamietaní konfliktných pravidiel a nech M je model (resp. kandidát na model) programu P', potom:*

(1) $R_0^M = \{L \leftarrow | \; \forall L \in M\} \cup \{\overline{L} \leftarrow | \; \forall \overline{L} \in M\}$

(2) $R_1^M = \{L^\top \leftarrow L \wedge \overline{L} \mid \forall L \in P'\}$

(3) $R_2^M = \{notL^\top \leftarrow notL_{r_1} \wedge \cdots \wedge notL_{r_n} \mid r_1, \ldots, r_n$ *sú všetky také pravidlá z* P'_M*, ktorých hlava je* L *a sú podporené* $M\} \cup \{not\overline{L^\top} \leftarrow notL_{r_1} \wedge \cdots \wedge notL_{r_n} \mid r_1, \ldots, r_n$ *sú všetky také pravidlá z* P'_M*, ktorých hlava je* $\overline{L^\top}$ *a sú podporené* $M\}$

(4) $R_3^M = \{notL_{r_i} \leftarrow notL_1^\top \wedge \cdots \wedge notL_n^\top \mid Body(r_i) = L_1 \wedge \cdots \wedge L_n\}$

Kde pravidlá $r_i \in P'$ *a literály* $L^\top$ *a vštky literály* L_{r_i} *sú nové literály.*

Takže $L^\top$ bude pravdivý, ak je literál L nekonzistentný alebo ak $\exists r \in P'$ a $M \models r$ také, že $Head(r) = L$ (resp. $Head(r) = \overline{L}$) a telo pravidla r obsahuje nekonzistenciu alebo menej dôveryhodný literál. Pre každé pravidlo r zavedieme pomocný literál L_r a pravidlo $notL_r \leftarrow notL_1^\top \wedge \cdots \wedge notL_n^\top$, ktoré nám toto umožní. Ak zisťovanie dôveryhodnosti robíme po nájdení modelov pre program P', pridáme informácie o literáloch v podobe pravidiel bez tela.

Definícia 3.2.8 *Nech program* P *je program získaný (ako medzivýsledok) po zamietaní konfliktných pravidiel a nech* I *je interpretácia. Vytvoríme program s dôveryhodnosťou* $P_I^\top$*, pridaním nových literálov a pravidiel,* $P_I^\top = P \cup R_1^M \cup R_2^M \cup R_3^M$.

Definícia 3.2.9 *Nech program* P *je program získaný (ako medzivýsledok) po zamietaní konfliktných pravidiel a nech* M *je model* P*. Vytvoríme program počítajúci dôveryhodnosť literálov programu* P*:* $P_M^\top = R_0^M \cup R_1^M \cup R_2^M \cup R_3^M$.

3.3 Vlastnosti

Chceli by sme, aby naša sémantika bola rozšírením sémantík, o ktoré sa opiera. Inými slovami chceli by sme, aby naša sémantika vygenerovala všetky tie modely, ktoré by vygenerovali sémantiky, ktoré sme si zobrali za základ, a v prípade niektorých vstupov aj nejaké rozumné modely navyše. Vhľadom na to že celý proces je možné rozdeliť na dve fázy a to zamietanie konfliktných

pravidiel v aktualizáciách a hľadanie modelu novovzniknutého programu, tak si to aj teraz takto rozdelíme.
Takže chceme čosi ako spätnú kompatibilitu so zvolenou sémantikou teda chceme ukázať, že vyrobíme rovnaké množiny *Defaults* a *Rejected*, čím dostaneme aj rovnakú množinu *Residue* ako by vyrobila pôvodná sémantika.

Máme dve verzie množiny Defaults a to *Defaults* a *Defaults**. Množina *Defaults* je bez zmeny a teda bude u nás obsahovať tie isté literály ako by obsahovala pri použití *DSM*, *RDSM* a *MDyLP* založené na *stabilných modeloch*. Rozdiel je pri množine *Defaults**. Pôvodne, v *DJU* a *RDJU*, táto množina obsahovala všetky literály, ktoré nie sú v modeli. Toto bolo možné preto, že pri dvojhodnotovej logike je zvyk, že model obsahuje pravdivé literály a literály, ktoré nie sú v modeli sú nepravdivé. Takže množina *Defaults** pôvodne obsahovala nepravdivé literály. Takže o to isté sme sa snažili aj my, ale vo viachodnotovej logike je ťažké povedať, že všetko, čo nie je v modeli, musí byť nepravdivé. Naša reprezentácia (pri ukázžkovej štvorhodnotovej logike) hovorí, že ak je literál L pravdivý, tak $L \in M$. Ak je literál L nepravdivý, tak $\overline{L} \in M$. Ak je literál L nekonzistentný, pravdivostná hodnota $\top$, tak $L \in M$ a aj $\overline{L} \in M$ a podobne ak má literál L pravdivostnú hodnotu $\bot$, tak $L \notin M$ a aj $\overline{L} \notin M$. Takže ak chceme mať množinu nepravdivých literálov, tak chceme literály, pre ktoré platí, že $\overline{L} \in M$ a zároveň $L \notin M$ čím sme sa dostali k našej množine *Defaults**.

Zaujímavejšie to je s množinou *Rejected*. Hoci jednotlivé sémantiky, ktoré používame na zamietanie pravidiel majú rozdielne množiny *Rejected*, ten hlavný rozdiel, na ktorý sa teraz pozrieme je naša podmienka $val_M(notBody(r')) <_t val_M(Body(r))$. Pri dvojhodnotovej logike sme požadovali iba $M \models Body(r')$, teda aby zamietajúce pravidlo bolo podložené modelom. My sme túto podmienku zovšeobecnili, aby zohladňovala telá oboch pravidiel a hlavne aby sme mohli požívať rôzne viachodnotové logiky. Treba sa však uistiť, že táto podmienka je naozaj zovšeobecnením pôvodnej a teda ak by sme dostali ten istý program a aktualizácie, budeme mať rovnakú množinu *Rejected*. Pôvodná podmienka pracovala s dvojhodnotovou logikou a vyžadovala aby

$val_M(Body(r')) = t$ teda aby telo zamietajúceho pravidla bolo pravdivé. Na telo zamietaného pravidla r nemáme požidavky, takže môže byť pravdivé $val_M(Body(r)) = t$ alebo nepreavdivé $val_M(Body(r)) = f$. Z čoho dostaneme, že $val_M(Body(r)) \leq_t val_M(Body(r'))$. Teraz si zoberieme našu podmienku $val_M(not\ Body(r')) <_t val_M(Body(r))$ a vyberieme negáciu pred zátvorku a dostaneme $not\ val_M(Body(r')) <_t val_M(Body(r))$. Následne presunieme negáciu na druhú stranu a dostaneme $val_M(Body(r')) \geq_t not\ val_M(Body(r))$ alebo $not\ val_M(Body(r)) \leq_t val_M(Body(r'))$, čo sa už nápadne podobá $val_M(Body(r)) \leq_t val_M(Body(r'))$ a ešte raz spomeňme, že v pôvodnej podmienke sme nemali žiadne podmienky na telo zamietaného pravidla r a teda môže byť pravda aj nepravda alebo po znegovaní nepravda alebo pravda.

Ešte nám stáva pozrieť sa na model pôvodných sémantík a náš model. Sémantiky *DJU*, *DSM*, *RDJU*, *RDSM* a *MDyLP* založené na *stabilných modeloch* hľadajú model rovnakým spôsobom a sémantika *dobre podporených modelov* pre *MDyLP* požaduje aby M bol model a aby bola splnená podmienka pre ℓ, teda $\forall L \in M$ (resp. $\forall \overline{L} \in M$) $\exists r \in P'$ také, že $Head(r) = L$ (resp. $Head(r) = notL$), $\ell(Head(r)) > \ell(Body(r))$ a $M \models r$. My sme pri hľadaní modelunajskôr odstránime negáciu z hláv pravidiel tak, že prerobíme takéto pravidlá na podmienky presunutím hláv do tela pravidla (a znagovaním týchto hláv). Takže z rozšíreného logického programu dostaneme normálny logický program s podmienkami. K tomuto programu nájdeme stabilné modely štandartnou *GL-redukciou* a zahodíme modely nespĺňajúce podmienky (a v prípne zahodíme aj modely nespĺňajúce podmienku pre dobrepodporené modely). Ak máme konzistentný *DLP* alebo *MDyLP* dostaneme aj konzistentný model a to taký ako by našla príslušná sémantika.

Stabilné modely, a teda aj parakonzistentné stabilné modely, majú vlastnosť, že sa vyhýbajú nevynútenej nekonzistencii. Napríklad program $P = \{A \leftarrow notB,\ B \leftarrow notA\}$ má 2 stabilné modely: $M_1 = \{A\}$ a $M_2 = \{B\}$. Tieto modely sú preferované a o modeli $M_3 = \{A,\ notA,\ B,\ notB\}$ hovoríme, že obsahuje nevynútenú nekonzistenciu. Takáto nevynútená nekonzistencia vznikne keď budeme mať podobný cyklus pravidiel párnej dĺžky.

V našom prípade modely M_1 a M_2 vyzerajú nasledovne $M_1 = \{A,\ \overline{B}\}$ a $M_2 = \{\overline{A},\ B\}$ a model $M_3 = \{A,\ \overline{A},\ B,\ \overline{B}\}$ a je zjavné, že $M_1 \subset M_3$ a aj $M_2 \subset M_3$, takže M_3 nie je *parakonzistentný stabilný model* pre *DLP* (resp. *MDyLP*).

Navrhli sme možnosť rozlíšiť, či sa v odvodení nejakého literálu objavila nekonzistencia. Chceme ukázať, že toto rozlíšenie nám nepokazilo hľadané modely, ktoré by sme získali bez tohto rozlišovanie. Ukážeme, že *parakonzistentný stabilný model* pre *DLP* (resp. *MDyLP*) s *dôveryhodnosťou* je model P'.

Veta 3.3.1 *Nech P je* DLP *(resp.* MDyLP*) a P' je program získaný z P po zamietaní konfliktných pravidiel. Potom* parakonzistentný stabilný model *pre* DLP *(resp.* MDyLP*) s* dôveryhodnosťou *je model programu P'.*

Dôkaz 3.3.1.1 *Nech $M^\top$ je* parakonzistentný stabilný model *pre* DLP *(resp.* MDyLP*) s* dôveryhodnosťou *a nech M je korešpondujúci* parakonzistentný stabilný model *pre* DLP *(resp.* MDyLP*), v ktorom ku každému literálu $L^\top$ prislúcha literál L a k literálu $\overline{L^\top}$ prislúcha literál $\overline{L}$. Z definície $M^\top \models L$ práve vtedy, keď $M \models L$ a $M^\top \models notL$ práve vtedy, keď $M \models notL$. Preto $M^\top$ podporí každú klauzulu podporenú modelom M.*

Ku každému *parakonzistentnému stabilnému modelu* pre *DLP* (resp. *MDyLP*) s *dôveryhodnosťou* $M^\top$ programu P existuje $M = \{L \mid M^\top \models L^\top\} \cup \{\overline{L} \mid M^\top \models \overline{L^\top}\}$, ktorý je *parakonzistentný stabilný model* pre *DLP* (resp. *MDyLP*) programu P. Podobne ku každému *parakonzistentnému stabilnému modelu* pre *DLP* (resp. *MDyLP*) N programu P existuje *parakonzistentný stabilný model* pre *DLP* (resp. *MDyLP*) s *dôveryhodnosťou* $N^\top$ programu P taký, že $N = \{L \mid N^\top \models L^\top\} \cup \{\overline{L} \mid N^\top \models \overline{L^\top}\}$.

Zisťovanie dôveryhodnosti novými literálmi a pravidlami je druhý spôsob ako overiť či sa v odvodení nejakého literálu objavila nekonzistencia, ktorý nevyžadoval pridávanie pravdivostných hodnôt.

Veta 3.3.2 *Nech M je model programu P, potom existuje model val_{M_1} programu $P_1 = P \cup R_1^M \cup R_2^M \cup R_3^M$ taký, že $val_M(L) = val_{M \cup M_1}(L)$ pre všetky literály z programu P a množín pravidiel programu P.*

Teda zisťovanie dôveryhodnosti pôvodných literálov nemení ich pravdivostné hodnoty.

Dôkaz 3.3.2.1 *Program P_1 obsahuje všetky pravidlá programu P a teda inú pravdivostnú hodnotu pôvodných literálov by spôsobovali nové pravidlá. Avšak v nových pravidlách sa pôvodné literály objavujú iba v množine R_1^M a aj v tejto množine sa objavujú iba v tele pravidiel. Takže žiadne nové pravidlo nemá v hlave pôvodný literál a teda nemôžu meniť ich pravdivostnú hodnotu.*

Veta 3.3.3 *Nech P je program a M je jeho model. Vytvoríme program s dôveryhodnosťou $P_1 = P \cup R_1^M \cup R_2^M \cup R_3^M$ pridaním nových literálov a pravidiel. Nech M_1 je model programu P_1, taký že $val_M(L) = val_{M \cup M_1}(L)$. Vytvoríme program $P_2 = R_0^M \cup R_1^M \cup R_2^M \cup R_3^M$ tiež počítajúci dôveryhodnosť literálov programu P. Existuje model M_2 programu P_2 taký, že $val_{M_1}(L) = val_{M \cup M_2}(L)$ pre všetky literály z programu P a množín pravidiel R_1^M, R_2^M a R_3^M.*

Teda oba programy zisťujúce dôveryhodnosť literálov programu P označia rovnaké literály za menej dôveryhodné.

Dôkaz 3.3.3.1 *Keďže pravidlá majúce v hlave nové literály sú zhodné v programe P_1 a P_2 budú aj pravdivostné hodnoty pre tieto literály v modeloch M_1 a M_2 zhodné. Taktiež vieme, ža pravdivostné hodnoty pôvodných literálov sú v modeloch M a M_1 zhodné. Takže jediné čo môže tieto rpavdivostné hodnoty pokaziť je množina R_0^M v programe P_2. Avšak tieto pravidlá sú fakty, ktoré dosahujú rovnakú pravdivostnú hodnotu povodných literálov ako majú tieto literály v modeli M.*

3.4 Ďalšie pokračovanie

Pri hľadaní modelu vyberáme najmenší model vzhľadom na usporiadanie $\prec_t$ pretože sme si ako základ vybrali stabilné modely. Keby sme si však vybe-

rali najmenší model vzhľadom na usporiadanie $\prec_k$ dostali by sme iný typ modelov, ktorý tiež môže mať zaujímave vlastnosti. Toto usporiadanie by pravdivostné hodnoty literálov netlačilo k nepravde ale k $\perp$. Pri usporiadaní $\prec_t$ je každý literál nepravda ak nie je podporený pravidlom, ktoré je podporené modelom. Pri usporiadaní $\prec_k$ by sa však situácia zmenila tak, že literál by bol nedefinovaný, teda jeho pravdivostná hodnota by bola $\perp$ a inú by mohol nadobudnúť iba ak by bol podporený pravidlom, ktoré je podporené modelom. Táto vlastnosť pripomína dobre postavené (wellfounded) modely. Na to aby sme vedeli povedať či by sme takto dosiahli vlastnosti wellfounded modelov, teda či by išlo o rozšírenie týchto modelov, ako aj ďalšie vlastnosti, však bude potrebné ďalšie skúmanie.

Kapitola 4

Záver

Podarilo sa nám predstaviť sémantiku pracujúcu s pragramami a ich aktualizáciami, ktorá predchádza konfliktom zamietaním menejprioritných pravidiel. Umožnili sme istú voľnosť pri výbere spôsobu akým chceme zamietať konfliktné pravidlá. Aj keď sa vo výslednom programe objaví nekonzistencia, ktorej sme nedokázali predísť, budeme počítať model a získame tak užitočné informácie, ktoré by sme inak stratili. Naša sémantika umožňuje použitie aj viachodnotovej logiky, jediné požiadavky ktoré kladieme na pravdivostné hodnoty sú aby tvorili dvojzväz s dvoma usporiadaniami. Navrhli sme aj dva spôsoby ako rozlíšiť fakty ovplyvnené nekonzistenciou od faktov, ktoré s nekonzistenciou vôbec nesúvisia. Toto môže byť v niektorých situáciách dôležitá vlastnosť, ale aj fakty, v ktorých odvodení sa nekonzistencia objavila môžu byť užitočné a preto ich chceme zachovať avšak aj rozlíšiť. Naznačili sme možnosť ako zmeniť model ktorý budeme hľadať, na taký v ktorom defaultná hodnota nebude nepravda ale nedefinované.

Literatúra

[1] J. J. Alferes, J. A. Leite, L. M. Pereira, H. Przymusinska, and T. C. Przymusinski. Dynamic logic programming, 1998.

[2] José Júlio Alferes, Federico Banti, Antonio Brogi, and João Alexandre Leite. The Refined Extension Principle for Semantics of Dynamic Logic Programming. *Studia Logica*, 79(1):7–32, February 2005.

[3] F. Banti, J. J. Alferes, A. Brogi, and P. Hitzler. P.: The well supported semantics for multidimensional dynamic logic programs. In *LPNMR 2005, LNCS 3662*, pages 356–368. Springer, 2005.

[4] François Fages. A new fixpoint semantics for general logic programs compared with the well-founded and the stable model semantics. *New Generation Computing*, 9(3):425–443, 1991.

[5] Melvin Fitting. The family of stable models, 1993.

[6] Melvin Fitting. Bilattices are nice things, 2002.

[7] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. pages 1070–1080. MIT Press, 1988.

[8] Alexandre Leite. On Some Differences Between Semantics of Logic Program Updates.

[9] João Alexandre Leite, José Júlio Alferes, and Luís Moniz Pereira. Multi-dimensional dynamic logic programming, 2000.

[10] João Alexandre Leite and Luís Moniz Pereira. Iterated Logic Program Updates. 1998.

[11] Chiaki Sakama and Katsumi Inoue. Paraconsistent stable semantics for extended disjunctive programs. *Journal of Logic and Computation*, 5:265–285, 1995.