UNIVERZITA KOMENSKÉHO V BRATISLAVE
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

Čestné vyhlásenie

Čestne prehlasujem, že som túto diplomovú prácu vypracoval samostatne s použitím uvedených zdrojov.

V bratislave, dňa ....

Chcel by som sa 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: 53 strán
Bratislava, 2017

Statické logické programy stačia na modelovanie sveta, ale ak chceme zahrnúť aj zmenu situácie v čase sú menej praktické ako multidimenzionálne dynamické logické programy 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 vysporiadajú 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: 53 pages
Bratislava, 2017

Static logic programs are impractical for modelling the world if we want include changes in time and multidimensional dynamic logic programs are more practical for this task, 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 occurring. 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

1 Úvod **1**

2 Prehľad **3**

2.1 Základné pojmy 3

2.2 Dvojhodnotové sémantiky 4

2.2.1 Stabilné modely 4

2.2.2 Rozšírenie stabilnomodelovej sémantiky 5

2.2.3 Dobre-podporené (Well-supported) modely 6

2.3 Parakonzistentné sémantiky 6

2.3.1 Dvojzväzy 6

2.3.2 Parakonzistentné stabilné modely 12

2.4 Dynamické Logické Programy 17

2.4.1 Dynamické logické programy 17

2.4.2 Sémantika multi-dimenzionálnych dynamických logických programov založená na stabilných modeloch 20

2.4.3 Sémantika dobre-podporených (Well-supported) modelov pre multidimenzionálne dynamické logické programy 23

3 Prínos **27**

3.1 Zavedenie sémantiky 28

3.2 Nekonzistencia v odvodení 37

3.3 simulácia IV pomocou dvoj-hodnotovej logiky 39

3.4 Vlastnosti 46

3.5 Ďalšie pokračovanie 51

4 Záver **53**
Literatúra 53

Zoznam obrázkov

2.1 pohľad vedúceho výskumného pracovníka 22
2.2 pohľad študenta 23

3.1 extrémne prvky vzhľadom na usporiadania $\leq_t$ a $\leq_k$ 29
3.2 vyšetrenie 36
3.3 usporiadania $\leq_t$ a $\leq_k$ 43

Kapitola 1

Úvod

Dlhú dobu bola väčšina prác v oblasti logického programovania zameraná na statické informácie. Toto sa neskôr stalo nepraktickým pre reprezentovanie sveta a dynamickej informácie meniacej 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 dochádzať 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 tvorili 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 sklbenie týchto dvoch prístupov - sémantiku, ktorá dokáže pracovať s informáciou meniacou sa v čase, predchádzajúcou konfliktom medzi pravidlami zamietaním pravidiel z nižšou prioritou tam kde 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 Základné pojmy

Najskôr sa dohodneme ako budeme rozumieť jednotlivým pojmom logického programovania.

Pod *logickým programom* budeme rozumieť množinu pravidiel (ozn.: r) tvaru:

$$L \leftarrow L_1 \wedge \cdots \wedge L_n$$

Pričom časť naľavo od $\leftarrow$, teda L, voláme hlava pravidla (ozn.: $Head(r)$) a časť napravo od $\leftarrow$, teda $L_1 \wedge \cdots \wedge L_n$, voláme telo pravidla (ozn.: $Body(r)$). Pravidlo, ktoré má prázdnu hlavu, voláme ohraničenie a pravidlo s prázdnym telom voláme fakt. Podla ďalších podmienok, ktoré kladieme na pravidlá, ďalej rozdeľujeme programy. Ešte spomenieme, že literál je atóm A alebo defaultovo negovaný atóm *not L*. *Definitný 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 pravidlá voláme *zovšeobecnený logický program* (ozn.: *GLP*). Majme pravidlá r_1 a r_2 potom hovorím že pravidlá r_1 a r_2 sú v konflikte, zapisujeme $r_1 \bowtie r_2$, ak $Head(r_1) = not\ Head(r_2)$. Pre poriadok dodáme, že každý definitný 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. Defaultovo negovaný atóm $not\, A$ je pravdivý v I, označujeme to $I \models not\, A$, ak $A \notin I$, inak je $not A$ nepravdivý. Hovoríme, že interpretácia M je model zovšeobecneného logického programu P vtedy a len vtedy, keď pre každé pravidlo $r \in P$, r má tvar $L \leftarrow L_1 \wedge \cdots \wedge L_n$, platí, že ak $M \models L_1 \wedge \cdots \wedge L_n$, tak platí aj $M \models L$. Model M programu P je minimálny ak neexistuje model N taký, že $N \subset M$.

Hebrandovské univerzum H je definované dvoma pravidlami: 1) koštanty patria do H 2) ak $t_1 \in H, \ldots, t_n \in H$ potom aj funkcia f s n parametrami tvaru $f(t_1, \ldots, t_n)$ patrí do H. Atóm (resp. literál) získaný nahradením všetkých premenných prvkami H voláme *uzemnený atóm* (resp. *uzemnený literál*). Množina všetkých uzemnených atómov je *Herbrandovská báza*.

2.2 Dvojhodnotové sémantiky

2.2.1 Stabilné modely

Nech P je logický program. Predpokladajme, že každé pravidlo obsahujúce premennú je nahradené všetkými jeho uzemnenými inštanciami, takže všetky atómy v P sú uzemnené. (Keďže nie je požadované, aby P bol konečný, premenné môžu byť eliminované týmto spôsobom aj ak program používa funkčné symboly a jeho Herbrandovské univerzum je nekonečné.)

Definícia 2.2.1 ([7]) *Pre ľubovolnú množinu M atómov z P, nech P_M je program získaný z P vymazaním (GL-transformácia):*

(i) každého pravidla obsahujúceho negatívny literál $not\, L$ 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 *stabilný model* P.

Veta 2.2.1 ([7]) *Ľubovoľný stabilný model P je minimálny Herbrantovský model P.*

2.2.2 Rozšírenie stabilnomodelovej sémantiky

Pre reprezentovanie negatívnej informácie v logických programoch sa nám hodia rozšírenie logických programov také, ktoré umožní negáciu *not* A 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 [12] 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 ohraničenia. Aby takéto pravidlo (ohraničenie) bolo splnené, musí byť telo pravidla nepravdivé. Definícia stabilných modelov nepracovala s programami obsahujúcimi ohraničenia, 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 v tomto programe nebudeme pre účely hľadania stabilného modelu uvažovať ohraničenia (vzniknuté ani prípadné pôvodné). Následne z výslednej množiny modelov zahodíme všetky stabilné modely nespĺňajúce tieto ohraničenia.

Definícia 2.2.2 *Nech P je GLP. Nech P' je program obsahujúci pravidlá pôvodného programu P, ktoré majú neprázdnu hlavu pravidla a zároveň neobsahujú negáciu v hlave pravidla. Zavedieme množinu ohraničení* $Constraints_P = \{ \leftarrow A \wedge A_1 \wedge \cdots \wedge A_m \wedge not\ A_{m+1} \wedge \cdots \wedge not\ A_n \mid \exists r \in P,\ r = not\ A \leftarrow A_1 \wedge \cdots \wedge A_m \wedge not\ A_{m+1} \wedge \cdots \wedge not\ A_n, m, n \geq 0\} \cup \{r \mid r \in P,\ r = \leftarrow A_1 \wedge \cdots \wedge A_m \wedge not\ A_{m+1} \wedge \cdots \wedge not\ A_n\}$. *Potom model M budeme volať stabilný model programu P vtedy a len vtedy, keď M je stabilný model programu P' a spĺňa všetky ohraničenia v množine* $Constraints_P$.

2.2.3 Dobre-podporené (Well-supported) modely

Základom dobre podporených modelov je mapovacia funkcia (level mapping) $\ell : \mathcal{L} \to \mathbb{N}$. Rozšírime túto funkciu pre negatívne literály tvaru $notA$, 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 0 prádznej konjunkcii literálov.

Stabilné modely môžu byť charakterizované aj prostredníctvom mapovacej funkcie a v tejto podobe sú pomenované *dobre podporené modely* [4]. Model je dobre podporený práve vtedy, keď je možné definovať mapovaciu funkciu nad atómami takú, že atóm A patrí do modelu práve vtedy, keď existuje pravidlo ktorého hlavou je A, ktorého telo je pravdivé v danom modeli a mapovacia funkcia priradí atómu A vyššiu hodnotu ako ktorémukoľvek z atómov v tele.

Definícia 2.2.3 ([3]) *Nech P je normálny logický program. Interpretácia M je* dobre podporený model *programu P ak:*

- *M je model programu P*
- *existuje mapovacia funkcia (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 ([3]) *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

Pred tým bude zavedený dvojzväz, začneme preddvojzväzom.

Definícia 2.3.1 ([6]) *Preddvojzväz (pre-bilattice) je štruktúra* $\mathcal{B} = (B, \leq_t, \leq_k)$ *kde* B *je neprázdna množina a* $\leq_t$ *a* $\leq_k$ *sú čiastočné usporiadania, každé tvoriace s* B *štruktúru zväzu.*

K obom usporiadaniam existujú operátory prieseku a spojenia. Pri pravdivostnom usporiadaní $\leq_t$ sú to klasické operácie $\wedge$ a $\vee$. Pri znalostnom usporiadaní $\leq_k$ to sú operácie *konsenzus* (consensus) so symbolom $\otimes$ a *dôverčivosť* (gullability) so symbolom $\oplus$. $x \otimes y$ je možné chápať ako informácie na ktorých sa x a y zhodnú. Dôverčivá osoba uverí čomukoľvek a tak $x \oplus y$ bude kombinácia informácií od x s informáciami od y. Najmenší prvok vzhľadom na usporiadanie $\leq_k$ je označený $\bot$ a najväčší prvok je zonačený $\top$. O $\bot$ je možné uvažovať ako žiadnej informácií, zatiaľ čo $\top$ reprezentuje všetky informácie, čo môže zahŕňať nekonzistenciu. Podobne najmenší prvok vzhľadom na pravdivostné usporiadanie je označený ako *nepravda* a najväčší prvok je označený ako *pravda*. Keďže máme 4 operácie, existuje 12 distributívnych zákonov:

- $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$
- $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$
- $x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)$
- $x \oplus (y \otimes z) = (x \oplus y) \otimes (x \oplus z)$
- $x \wedge (y \otimes z) = (x \wedge y) \otimes (x \wedge z)$
- $x \otimes (y \wedge z) = (x \otimes y) \wedge (x \otimes z)$
- $x \vee (y \otimes z) = (x \vee y) \otimes (x \vee z)$
- $x \otimes (y \vee z) = (x \otimes y) \vee (x \otimes z)$
- $x \wedge (y \oplus z) = (x \wedge y) \oplus (x \wedge z)$
- $x \oplus (y \wedge z) = (x \oplus y) \wedge (x \oplus z)$
- $x \vee (y \oplus z) = (x \vee y) \oplus (x \vee z)$
- $x \oplus (y \vee z) = (x \oplus y) \vee (x \oplus z)$

Definícia 2.3.2 ([6]) *Preddvozjzväz (pre-bilattice)* $(B, \leq_t, \leq_k)$ *je* úplný *ak všetky prieseky a spojenia existujú vzhľadom na obe usporiadania. Označme nekonečný priesek a spojenie vzhľadom na* $\leq_t$ *symbolmi* $\bigwedge$ *a* $\bigvee$ *a vzhľadom na* $\leq_k$ *symbolmi* $\prod$ *a* $\sum$

Pozrime sa na vlastnosti dvozväzov.

Definícia 2.3.3 ([6]) Preddvojzväz $(B, \leq_t, \leq_k)$ *je:*

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

2. *nekonečný dvojzväz ak je úplný a všetky 4 operátory nekonečného prieseku a spojenia sú monotónne vzhľadom na obe usporiadania*

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

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

Poznáme základné vlastnosti dvojzväzov a teraz sa pozrieme ako dvojzväz môžeme získať. 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 ([6]) Dvojzväzový súčin $\mathcal{L}_1 \odot \mathcal{L}_2$ *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* $x_2 \leq_2 y_2$

Nech $\mathcal{L} = (L, \leq)$ je zväz a $a, b \in L$ sú jeho prvky, pod intervalom budeme rozumieť $\langle a, b\rangle = \{x \in L \mid a \leq_L x \leq_L b\}$. Triedu intervalov nad zväzom $\mathcal{L}$ budeme označovať $\mathcal{I}(L)$. Nepresné meranie x môže viesť k záveru, že x je medzi a a b a teda môžeme použiť $\langle a, b\rangle$ ako našu súčasnú informáciu o x. Zjavne lepšie meranie zmenši (zúži) interval.

Definícia 2.3.5 ([6]) Intervalová konštrukcia *je druhý prístup ako získať dvojzväz.*
Nech $\mathcal{L}$ je zväz. $\mathcal{K}(L)$ je štruktúra $(\mathcal{I}(\mathcal{L}), \leq_t, \leq_k)$ kde:

- $\langle a, b\rangle \leq_t \langle c, d\rangle$ ak $a \leq_L c$ a $b \leq_L d$
- $\langle a, b\rangle \leq_k \langle c, d\rangle$ ak $\langle c, d\rangle \subseteq \langle a, b\rangle$

Definícia 2.3.6 ([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 ([6]) *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 ([6]) *Nech $\mathcal{B} = (B, \leq_t, \leq_k)$ je preddvojzväz 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 konfláciu, 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 $\mathcal{B}^{\mathcal{S}}$ preddvojzväz. 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.

Dvojzväzom môžeme prispôsobiť aj valuácie.

Definícia 2.3.9 ([6]) *Nech v_1 a v_2 sú valuácie v dvojzväze $\mathcal{B}$ s negáciou. Definujme pseudovaluáciu $v_1 \triangle v_2$ v $\mathcal{B}$ nasledovne. Pre uzemnený atóm A platí:*

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

Na neliterály je rozšírenie pseudovaluácie zjavné.

Podobne ako valuácie, čosi si povieme aj o operátori priamych dôsledkov, pevných bodoch tohto operátora a ich vlastnostiach.

Definícia 2.3.10 ([5]) *Rozšírený operátor priamych dôsledkov, $\psi_P : \mathcal{V}(\mathcal{B}) \times \mathcal{V}(\mathcal{B}) \to \mathcal{V}(\mathcal{B})$ je definovaný nasledovne. Nech $v_1, v_2 \in \mathcal{V}(\mathcal{B})$, $\psi_P(v_1, v_2)$ je valuácia taká, že:*

- *ak uzemnený atóm A nie je hlavou pravidla $r \in P^\star$, $\psi_P(v_1, v_2)(A) = false$*
- *ak sa $A \leftarrow B$ nachádza v $P^\star$, tak $\psi_P(v_1, v_2)(A) = (v_1 \triangle v_2)(B)$*

kde $P^\star$ je množina uzemnených inštancií pravidiel programu P.

Definícia 2.3.11 ([5]) *Mapovanie f čiastočne usporiadaného priestoru je monotónne ak $x \leq y$ implikuje $f(x) \leq f(y)$ a anti-monotónne ak $x \leq y$ implikuje $f(x) \geq f(y)$.*

Ak predpokladáme, že platia podmienky štruktúrnosti (interlacing conditions) pre $\mathcal{B}$, pri $\leq_k$ usporiadaní, tak ψ_P je monotónne v oboch argumentoch. Pri $\leq_t$ usporiadaní je ψP monotónne pri prvom argumente a anti-monotónne pri druhom argumente. Ak pri $\leq_t$ zafixujeme druhý argument, budeme mať monotónny operátor s jedným (prvým) argumentom a teda budeme mať najmenší pevný bod.

Definícia 2.3.12 ([5]) *Jednovstupové mapovanie ψ'_P je operátor odvodený od operátora ψ_P. Najmenší pevný bod, $v \leq_t$ usporiadaní, mapovania $(\lambda x)\psi_P(x, v)$ je $\psi'_P(v)$.*

Definícia 2.3.13 ([5]) *Stabilná valuácia programu P je pevný bod operátora ψ'_P.*

Veta 2.3.1 ([5]) *Operátor ψ'_P je monotónny v $\leq_k$ usporiadaní a anti-monotónny v $\leq_t$ usporiadaní.*

Veta 2.3.2 ([5]) *Operátor ψ'_P má najmenší pevný bod, označujeme s^k_P, a najväčší pevný bod, označujeme S^k_P , pri $\leq_k$ usporiadaní.*

Teda každý program má aspoň jednu stabilnú valuáciu.

Definícia 2.3.14 ([5]) *Valuácia v je model programu P, ak pre každý uzemnený atóm A, ak $A \leftarrow B \in P^\star$ tak $v(A) = \bigvee\{v(B) \mid A \leftarrow B \in P*\}$ inak $v(A) = \bot$.*

Veta 2.3.3 ([5]) *Nech f je anti-monotónne na úplnom zväze L, potom existujú dva prvky L, μ a ν také, že:*

- *μ a ν sú najmenší a najväčší pevný bod f*[2]
- *f osciluje medzi μ a ν v zmysle, že $f(\mu) = \nu$ a $f(\nu) = \mu$*
- *ak x a y sú tiež body L medzi ktorými f osciluje, tak x a y ležia medzi μ a ν.*

Definícia 2.3.15 ([5]) *Extrémne oscilačné body operátora ψ'_P označujeme s^t_P a S^t_P, pričom $s^t_P \leq_t S^t_P$.*

Veta 2.3.4 ([5]) *Valuácie $s^t_P \otimes S^t_P$ a $s^t_P \oplus S^t_P$ sú pevné body ψ'_P a teda aj stabilnými modelmi.*

Veta 2.3.5 ([5]) *Extrémne pevné body ψ'_P pri usporiadaní $\leq_k$ majú nasledovný vzťah k extrémnym oscilačným bodom pri usporiadaní $\leq_t$:*

- $s_P^k = s_P^t \otimes S_P^t$
- $S_P^k = s_P^t \oplus S_P^t$
- $s_P^t = s_P^k \wedge S_P^k$
- $S_P^t = s_P^k \vee S_P^k$

Príklad 2.3.1 *Ako príklad ([6]) dvojzväzu si predstavme, že máme skupinu ľudí $\mathcal{P}$. Ak sa týchto ľudí spýtame na názor na vetu* X*, niektorí ju nazvú pravdou a niektorí nepravdou. Taktiež niektorí odmietnu odpovedať a niektorí si nemusia byť istí nazývajúc ju pravdou aj nepravdou. Takže vete* X *môžeme priradiť $< P, N >$, kde* P *je množina ľudí z $\mathcal{P}$, ktorí hovoria, že* X *je pravda a* N *je množina tých, ktorí hovoria, že* X *je nepravda. Ako sme naznačili, nevyžadujeme, aby $P \cup N = \mathcal{P}$ ani $P \cap N = \mathcal{P}$.*
Usporiadanie na tejto ľudskej štruktúre bude nasledovné:
$< P_1, N_1 > \leq_k < P_2, N_2 >$ ak $P_1 \subseteq P_2$ a $N_1 \subseteq N_2$ a podobne $< P_1, N_1 > \leq_t < P_2, N_2 >$ ak $P_1 \subseteq P_2$ a $N_2 \subseteq N_1$ (všimnime si zmenu). Takže budeme mať viac informácií ak viac ľudí vyjadrí pozitívny alebo negatívny názor a pravdivosť stúpa, ak menej ľudí vyjadrí negatívny názor alebo viac ľudí pozitívny názor. Dodajme ešte, že $\bot =< \emptyset, \emptyset >$, $\top =< \mathcal{P}, \mathcal{P} >$, nepravda $=< \emptyset, \mathcal{P} >$ a pravda $=< \mathcal{P}, \emptyset >$.
Samozrejme máme aj negáciu a konfláciu, teda $\neg < P, N >$ bude $< N, P >$ teda vymeníme tých, ktorí hovoria za a proti. $A - < P, N >$ bude $< \mathcal{P} \setminus N, \mathcal{P} \setminus P >$, čo je zmena ich pôvodných postojov, teda ľudia, ktorí potvrdzujú sú tí, čo pôvodne nepopierali.

2.3.2 Parakonzistentné stabilné modely

Klasický stabilný model je dvojhodnotový a ak sa v programe objaví nekonzistencia, tak stabilný model pre tento program neexistuje. Jeden zo spôsobou ako sa vyrovnať s nekonzistenciou je indetifikovanie tejto nekonzistencie a pokračovanie vo výpočte, pričom táto nekonzistencia neovplyvní nezávislú časť bázy znalostí. Tento prístup bol aplikovaný aj na stabilnomodelovú sémantiku, čím sme dostali rozšírenie stabilných modelov na *parakonzistentné stabilné modely* [12].

Definícia 2.3.16 ([12]) *Pre jednoduchosť sa najskôr pozrieme na pozitívne programy teda pravidlá nebudú obsahovať negáciu, takže pod pozitívnym programom budeme rozumieť konečnú množinu klauzúl tvaru:*

$$L \leftarrow L_1 \wedge \cdots \wedge L_m \ (m \geq 1)$$

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 $I : \mathcal{L}_\mathcal{P} \rightarrow IV$ 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 2.3.17 ([12]) *Nech P je pozitívny rozšírený program a I je interpretácia. Potom,*

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

$$I \models L \text{ práve vtedy, keď } t \preceq I(L)$$
$$I \models \neg L \text{ práve vtedy, keď } f \preceq I(L)$$

2 *pre ľubovolnú konjunkciu uzemnených 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 uzemnenú 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 uzemnenú klauzuly C programu P. Usporiadanie $\preceq$ na pravdivostných hodnotách je taktiež definované aj medzi interpretáciami. Pre interpretácie 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 2.3.6 ([12]) *Ak pozitívny rozšírený program má model, tak má aspoň jeden p-minimálny model.*

Propozícia 2.3.7 ([12]) *Konzistentný pozitívny rozšírený program má konzistentný p-minimálny model.*

Predstavíme si operátor priamych dôsledkov pre pozitívne logické programov a použijeme sémantiku pevných bodov tohto operátora na charakterizovanie parakonzistentného sémantiky minimálneho modelu.

Definícia 2.3.18 ([12]) *Nech P je pozitívny rozšírený logický program a $\mathcal{I}$ je množina interpretácií. Potom mapovanie $T_P : 2^{\mathcal{L}_P} \rightarrow 2^{\mathcal{L}_P}$ je definované ako*

$$T_P(\mathcal{I}) = \bigcup_{I \in \mathcal{I}} T_P(I)$$

kde mapovanie $T_P : 2^{\mathcal{L}_P} \to 2^{\mathcal{L}_P}$ *je definované nasledovne*

$$T_P = \begin{cases} \emptyset, \text{ ak } \{L_l, \ldots, L_m\} \subseteq I \\ \text{pre nejakú uzemnené ohraničenie} \\ \leftarrow L_l \wedge \ldots \wedge L_m \text{ z } P \\ \{J | \text{ pre každú uzemnenú klauzulu} \\ C_i : L_i \leftarrow L_{l_i} \wedge \ldots \wedge L_{m_i} \text{ z } P; \\ \text{takú, že } \{L_{l_i}, \ldots, L_{m_i}\} \subseteq I, \\ J = I \cup \{L_i\}, \text{ inak} \end{cases} \tag{2.1}$$

Definícia 2.3.19 ([12]) *Mocniny* $\mathcal{T}_P$ *sú definované nasledovne*

$$\mathcal{T}_P \uparrow 0 = \{\emptyset\}$$

$$\mathcal{T}_P \uparrow n+1 = \mathcal{T}_P(\mathcal{T}_P \uparrow n)$$

$$\mathcal{T}_P \uparrow \omega = \bigcup_{\alpha<\omega} \bigcap_{\alpha<\omega} \mathcal{T}_P \uparrow n$$

Veta 2.3.8 ([12]) $\mathcal{T}_P \uparrow \omega$ *je pevný bod.*

Lema 2.3.9 ([12]) *Nech* P *je pozitívny rozšírený logický program. Potom* I *je model* P *práve vtedy keď* $I \in \mathcal{T}_P(\{I\})$.

Lema 2.3.10 ([12]) *Ak* I *je p-minmálny model* P*, potom pre každý literál* L *v* I *existuje uzemnená klauzula* $L_1 \vee \ldots \vee L_l \leftarrow L_{l+1} \wedge \ldots \wedge L_m$ *z* P *taká, že* $\{L_{l+1}, \ldots, L_m\} \subseteq I$ *a* $L = L_i$ *pre nejaké* i $(1 \leq i \leq l)$.

Nech $\mu(\mathcal{T}_P \uparrow \omega) = \{I | I \in \mathcal{T}_P \uparrow \omega \text{ a } I \in \mathcal{T}_P(\{\emptyset\})\}$. Potom $\mu(\mathcal{T}_P \uparrow \omega)$ reprezentuje množinu modelov P, ktoré sú zahrnuté vo pevnom bode uzáveru. Taktiež, nech $min(\mathcal{I}) = \{I \in \mathcal{I} | \neg\exists J \in \mathcal{I}$ také, že $J \subset I\}$. Potom platí nasledujúci výsledok.

Veta 2.3.11 ([12]) *Nech* P *je pozitívny rozšírený logický program a* $\mathcal{PMM}_P$ *je množina všetkých p-minimálnych modelov* P*, potom*

$$\mathcal{PMM}_P = min(\mu(\mathcal{T}_P \uparrow \omega))$$

Dôsledok 2.3.12 ([12]) *Nech P je pozitívny rozšírený logický program. Potom $\mathcal{T}_P \uparrow \omega$ obsahuje unikátny p-najmenší model P.*

Teraz rozšírime sémantiku pridaním defaultovej negácie.

Definícia 2.3.20 ([12]) *Rozšírený logický program je konečná množina klauzúl tvaru:*

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

kde L_i sú literály a not reprezentuje defaultovú negáciu. *Pre potreby defaultovej negácie rozšírime definíciu 2.3.17 o tieto dve vlastnosti:*

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

Definícia 2.3.21 ([12]) *Nech P je rozšírený program a I je interpretácia.* Redukt *programu P vzhľadom na I je pozitívny rozšírený logický program P^I taký, že klauzula*

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

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

$$L \leftarrow L_1 \wedge \cdots \wedge L_m \wedge not\, L_{m+1} \wedge \cdots \wedge not\, L_n$$

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 2.3.13 ([12]) *P-stabilný model je p-minimálny model.*

Keď program obsahuje nekonzistentnú informáciu, je užitočné vedieť rozlíšiť fakty ovplyvnené takouto informáciou od ostatných. Je viacero spôsobov ako to dosiahnuť a jeden z nich je označiť fakty, v ktorých odvodení sa objavila nekonzistencia, za podozrivé. 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 $\mathcal{L}_P^s = \mathcal{L}_P \cup \{L^s \mid L \in \mathcal{L}_P\}$ a I^s je podmnožina $\mathcal{L}_P^s$, kde L^s označuje podozrivý literál.

Definícia 2.3.22 ([12]) *Interpretácia je funkcia $I^s : \mathcal{L}_P^s \to VI$ 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 $I^s(\neg L) = sf$.

Definícia 2.3.23 ([12]) *Pri logike* VI *je podporenie literálov a defaultovej 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 2.3.14 ([12]) *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.

Príklad 2.4.1 *Ukážme si jednoduchý príklad ([1]), na ktorom priblížime, čo od dynamických logických programov očakávame. Majme logický program P. Keď je tento program sám, tak jeho model je M. Pri postupnom pridávaní ďalších logických programov, akualizácií, U_1 a U_2 predstavujúcich zmenu, ktorá nastala, sa mení aj výsledný model.*

P $\quad sleep \leftarrow not\, tv\_on$

$watch\_tv \leftarrow tv\_on$

$tv\_on \leftarrow$

$M = \{tv\_on, watch\_tv\}$
Keďže je zapnutý televízor, tak nespíme, ale pozeráme televízor.

U_1 $\quad not\, tv\_on \leftarrow power\_failure$

$power\_failure \leftarrow$

$M = \{power\_failure, sleep\}$
Nastal výpadok prúdu a televízor už nie je zapnutý, hoci pravidlo v pôvodnom programe hovorí opak, ale toto pravidlo je staršie a teda menej prioritné. Keďže je výpadok prúdu, tak televízor nie je zapnutý, takže nepozeráme televízor, ale spíme.

U_2 $\quad not\, power\_failure \leftarrow$

$M = \{tv\_on, watch\_tv\}$
Pri poslednej aktualizácii zistíme, že výpadok prúdu už bol opravený, takže aj predchádzajúcu informácia o výpadku prúdu už nie je relevantná, takže nie je pravda, že televízor nie je zapnutý a preto nespíme ale pozeráme zapnutý televízor.

Od predstavenia aktualizácií vzniklo niekoľko sémantík, popísaných v [11][1][2][9][8], s cieľom prekonať nedostatky predchádzajúcich. My si zhrnieme 4 sémantiky: *dynamické stabilné modely* (DSM), *dynamické odôvodnené aktualizácie* (DJU), *zlepšené dynamické stabilné modely* (RDSM) a *zlepšené dynamické odôvodnené aktualizácie* (RDJU).

Spoločnou črtou týchto sémantík pre aktualizácie je, že novšie pravidlo, ktoré je v konflikte so starším, môže toto staršie pravidlo zamietnuť, aby sa predišlo nekonzistencii. Dve pravidlá r a r' sú v konflikte ($r \bowtie r'$) práve vtedy, keď $Head(r) = notHead(r')$. Vyrobíme množinu zamietnutých pravidiel *Rejected*, ktorá zamietne každé pravidlo r, ku ktorému existuje pravidlo prioritnejšie r', ktoré je s ním v konflikte. Priorita pravidla je daná prioritou programu, v ktorom sa pravidlo nachádza, teda poradím programu v postupnosti. Za mietať umožníme iba tým pravidlám r', ktorá sú podporené modelom.

Definícia 2.4.1 ([9]) *Nech $\mathcal{P} = (P_1, \ldots, P_n)$ je DLP a M je interpretácia, potom:*

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

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

Množiny *Rejected* a $Rejected^{+}$ sú dve verzie zamietania pravidiel. Pripravíme si ešte množinu defaultovo negovaných literálov. Máme dva spôsoby ako vyrobiť túto množinu. Prvý spôsob hovorí, že do tejto množiny patrí *not A* ak v celej postupnosti programov neexistuje pravidlo, ktorého telo je pravdivé v modeli M také, že podporí A. Druhý spôsob ako vyrobiť množinu *Defaults* je vložiť do nej negácie *not A* všetkých nepravdivých atómov A, teda atómov A nepatriacich do modelu M.

Definícia 2.4.2 ([9]) *Nech $\mathcal{P} = (P_1, \ldots, P_n)$ je DLP a M je interpretácia, potom:*

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

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

Kde $\rho(\mathcal{P})$ je multimnožina obsahujúca všetky pravidlá v $P_1, \ldots, P_n$. V ďalšom kroku zoberieme všetky pravidlá v postupnosti programov $\mathcal{P}$, odstránime zamietnuté pravidlá a pridáme defaultne negované literály v množine Defaults. *Pre výsledný program nájdeme najmenší model.*

Definícia 2.4.3 ([9]) *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* zlepšený 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* zlepšená 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 Sémantika multi-dimenzionálnych dynamických logických programov založená na stabilných modeloch

Postupnosť aktualizácií, z predchádzajúcej časti, je lineárna, ľahko sa s ňou pracuje, ale má svoje obmedzenia a preto bola rozšírená. Na postupnosť sa dá pozerať ako na organizáciu alebo hierarchiu programov v jednej dimenzii, a preto rozšírenie spočíva uvoľnení obmedzení tejto hierarchie, teda odstránenie lineárnosti hierarchie, čo sa dá nazvať aj pridaním dimenzií, čím získame *multi-dimenzionálne dynamické logické programy* [10][8]. Jednou z motivácií sú aj multiagentové systémy, kde sme mali postupnosť aktualizácií

v čase pre každého agenta, čo tvorí jednu dimenziu, pričom sme mali viac ako jedného agenta čo tvorilo ďalšiu dimenziu hierarchie programov. Teraz už nebudeme mať programy usporiadané v jednoduchej postupnosti, ale budú organizované podľa orientovaného grafu bez cyklov. Tento graf určuje aj prioritu programov a pravidiel, pre účely zamietania pravidiel v prípade konfliktu.

Formálne si zadefinujeme novú hierarchiu programov.

Definícia 2.4.4 ([10]) *Nech $D = (V, E)$ je acyklický graf. Nech $v \in V$. Graf závislostí $\mathcal{D}$ 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 závislostí *grafu $\mathcal{D}$ je podgraf grafu $\mathcal{D}$, obsahujúci všetky vrcholy a hrany vo všetkých cestách smerujúcich do* v.

Definícia 2.4.5 ([10]) Multi-dimenzionálny dynamický logický program *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.

Keďže pracujeme s aktualizáciami na rovnakom princípe ako pri *DLP* a zmenila sa iba organizácia programov, vyrábame množiny *Defaults* a *Rejected*, ktorých význam ostal rovnaký. Taktiež aj postup hľadania modelu, teda hľadanie najmenšieho modelu programu, ktorý vznikol z pôvodných pravidiel všetkých programov, okrem zamietnutých pravidiel, a taktiež pridaním množiny *Defaults*.

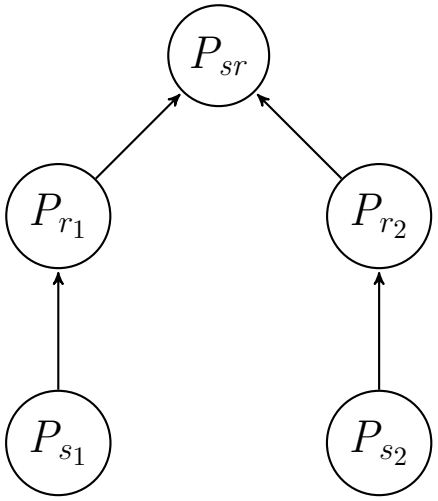
Definícia 2.4.6 ([10]) *Nech $(\mathcal{P}_D, D)$ je* Multi-dimenzionálny Dynamický Logický Program, *kde $\mathcal{P}_D = \{P_v : v \in V\}$ a $D = (V, E)$. Interpretácia M_s je stabilný model multi-dimenzionálnej aktualizácie v stave $s \in V$ práve vtedy, keď:*

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

kde

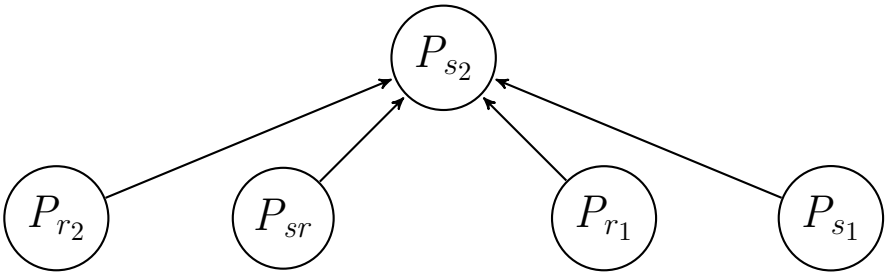
- $\mathcal{P}_s = \bigcup_{i \leq s} P_i$
- $Reject(s, M_s) = \{r \in P_i | \exists r' \in P_j, i \leq j \leq s, head(r) = not\, head(r') \wedge M_s \models body(r')\}$
- $Default(\mathcal{P}_s, M_s) = \{notA |\ \nexists r \in \mathcal{P}_s : (head(r) = A) \wedge M_s \models body(r)\}$

Príklad 2.4.2 *Ako príklad ([8]) použijeme hierarchickú štruktúru výskumného tímu. Tento tím tvoria vedúci výskumný pracovník (sr), dvaja výskumní pracovníci (r_1 a r_2) a dvaja študenti (s_1 a s_2) pod dohľadom spomínaných výskumníkov.*



Obr. 2.1: pohľad vedúceho výskumného pracovníka

Hierarchia reprezentujúca pohľad vedúceho výskumného pracovníka je znázornená na obrázku 2.1. Nie je neobvyklé, že študenti si myslia, že majú vždy pravdu a ich pohľad na komunitu je tiež odlišný. Obrázok 2.2 znázorňuje možný pohľad študenta (s_2). Môžeme použiť MDyLP*, obrázky budú predstavovať graf závislostí $\mathcal{D}$ pre jednotlivé pohľady, na porovnanie týchto rozdielnych pohľadov. Priraďme jednotlivým aktérom nasledujúce jednoduché programy:*



Obr. 2.2: pohľad študenta

$P_{sr} = \{a \leftarrow b\}$

$P_{r_1} = \{c \leftarrow\}$

$P_{r_2} = \{b \leftarrow\}$

$P_{s_1} = \{\}$

$P_{s_2} = \{not\ a \leftarrow b\}$

V situácii z pohľadu vedúceho výskumného pracovníka (2.1) máme jeden stabilný model a to $M_{sr} = \{a, b, c\}$ *a v situácii z pohľadu študenta (2.2) máme tiež práve jeden stabilný model* $M_{s_2} = \{not\, a, b, c\}$ *(not a sa obyčajne v modeli neuvádza, tentokrát je uvedené na zdôraznenie rozdielu). Takže podľa pohľadu študenta na svet je* a *nepravdivé, zatiaľ čo z pohľadu na svet vedúceho výskumného pracovníka je* a *pravdivé.*

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

Podobne ako obyčajné stabilné modely, stabilné modely pre *MDyLP* môžu byť tiež charakterizované prostredníctvom mapovacej funkcie a rovnako budú pomenované *dobre podporené modely* pre *MDyLP* [3].
Rovnako podmienka pre dobre podporený model je existencia mapovacej funkcie. Tato funkcia má mať takú vlastnosť, že atóm A patrí do modelu práve

vtedy, keď existuje pravidlo, ktorého hlavou je atóm A, jeho telo je pravdivé v danom modeli a mapovacia funkcia hlave tohto pravidla priradí vyššiu hodnotu ako jeho telu.

Majme *MDyLP* $\mathcal{P}$. Nech $\mathcal{D}$ je príslušný graf závislostí. Budeme používať označenie $\rho(\mathcal{P})$ pre multi-množinu všetkých pravidiel v $\mathcal{P}$ a podobne $\rho(\mathcal{P}^n)$ pre multi-množinu všetkých pravidiel v $\mathcal{P}^n$, kde $\mathcal{P}^n$ označuje podmnožinu prvkov $\mathcal{P}$ menších (vzhľadom na graf závislostí) ako P_n vrátane samotného P_n. budeme používať zjednodušený zápis $i \prec j$ na vyjadrenie $P_i \prec_{\mathcal{D}} P_j$ teda, že v $\mathcal{D}$ vedie cesta z P_i do P_j.

Množina *Rejected* funguje rovnako ako *MDyLP*, iba na zamietajúce pravidlo kladieme ďalšiu podmienku, že mapovacia funkcia má hlave pravidla priradiť vyššiu hodnotu ako telu pravidla.

Definícia 2.4.7 ([3]) *Nech $\mathcal{P}$ je ľubovolný* MDyLP *nad jazykom $\mathcal{L}$, nech P_n je update $\mathcal{P}$ a nech ℓ je 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:*

$$Rejected_\ell(\mathcal{P}, M, n) = \{r \mid r \in P_i, \exists r' \in P_j : i \prec j \preceq n, r \bowtie r', M \models Body(r'), \ell(Head(r')) > \ell(Body(r')\}$$

Definícia 2.4.8 ([3]) *Nech $\mathcal{P}$ je ľubovolný* MDyLP *nad nejakým jazykom $\mathcal{L}$ a nech M je interpretácia nad $\mathcal{L}$, P_n je update $\mathcal{P}$. Hovoríme, že M je dobre podporený model P_n práve vtedy, keď existuje mapovacia funkcia ℓ nad $\mathcal{L}$ taká, že i) M je model $\rho(P)\backslash Rejected(M, n)$ a ii) $\forall A \in M \exists r \in \rho(P)\backslash Rejected(M, n)$ také, že $Head(r) = A, \ell(A) > \ell(Body(r))$ a r je podporené M.*

Sémantika pevného bodu je ďalšia charakterizácia *dobre podporených modelov* pre *MDyLP*. Nepočíta sa priamo *dobre podporený model* pre *MDyLP* ale *zlepšený multi-stabilný model*, ktorý je ekvivalentný s *dobre podporeným modelom*.

Definícia 2.4.9 ([3]) *Nech $\mathcal{P}$ je ľubovolný* MDyLP *nad jazykom $\mathcal{L}$, nech P_n je update $\mathcal{P}$ a nech M je ľubovolná 2-hodnotová interpretácia nad $\mathcal{L}$. Konečná množina defaultne negovaných literálov je:*

$$Defaults(\mathcal{P}^n, M) = \{not\ A \mid \nexists A \leftarrow body \in \rho(\mathcal{P}^n) : body \subseteq M\}$$

Definícia 2.4.10 ([3]) *Nech $\mathcal{P}$ je ľubovolný* MDyLP *nad jazykom $\mathcal{L}$. Pridajme nové atómy $rej(A, i)$ nepatriace do $\mathcal{L}$, kde A je literál v $\mathcal{L}$ a* i *je v rozsahu určenom aktualizáciami $\mathcal{P}$. Množina zamietnutých pravidiel nad rozšíreným jazykom je definovaná nasledovne:*

$$Rj(\mathcal{MP}) = \{rej(not\ A, j) \leftarrow body \mid A \leftarrow body \in P_k \wedge j \prec k\}$$

Definícia 2.4.11 ([3]) *Nech $\mathcal{P}$ je ľubovolný* MDyLP *v jazyku $\mathcal{L}$. Pridajme nové atómy $safe(A, i)$ nepatriace do $\mathcal{L}$, kde A je literál v $\mathcal{L}$ a* i *je v rozsahu určenom aktualizáciami $\mathcal{P}$. Nasledujúcu množinu pravidiel označujeme $\sum(\mathcal{P})$:*

$$\sum(\mathcal{P}) = \{A \leftarrow body, safe(A, i) \mid A \leftarrow [body] \in P_i\}$$

Nech M je interpretácia nad $\mathcal{L}$. Množina podmienok aby bol literál A striktne bezpečný *v P_i je definovaná nasledovne:*

$$cond^S(\mathcal{P}, M, A, i) = \{rej(not\ A, j) \mid \exists r \in P_j, j \not\prec i, M \models Body(r) \wedge Head(r) = not\ A\}$$

Zneužitím notácie, $cond^S(\mathcal{P}, M, A, i)$ taktiež označuje konjunkciu všetkých literálov v $cond^S(\mathcal{P}, M, A, i)$. Množina striktne bezpečných pravidiel je definovaná nasledovne:

$$Safe^S(\mathcal{P}, M) = \{cond^S(\mathcal{P}, M, A, i) \leftarrow cond^S(\mathcal{P}, M, A, i) \mid \exists r \in P_i : Head(r) = A\}$$

Definícia 2.4.12 ([3]) *Nech $\mathcal{P}$ je ľubovolný* MDyLP *v jazyku $\mathcal{L}$, nech P_n je aktualizácia $\mathcal{P}$ a M je interpretácia nad $\mathcal{L}$. Definujeme operátor $\Gamma^S_{(\mathcal{P},n)}$ na interpretáciách nad $\mathcal{L}$ nasledovne:*

$$\Gamma^{S}_{(\mathcal{P},n)}(I) = least(\sum(\mathcal{P}^n) \cup Safe^{S}(\mathcal{P}^n, I) \cup Rj(\mathcal{P}^n) \cup Defaults(\mathcal{P}^n, I)) \mid_{\mathcal{L}}$$

Definícia 2.4.13 ([3]) *Nech $\mathcal{P}$ je ľubovolný* MDyLP *v jazyku $\mathcal{L}$, nech P_n je prvok $\mathcal{P}$ a nech M je interpretácia nad $\mathcal{L}$. Hovoríme, že M je zlepšený multi-stabilný model $\mathcal{P}$ v P_n práve vtedy, keď $M = \Gamma^{S}_{(\mathcal{P},n)}(M)$.*

Veta 2.4.1 ([3]) *Nech $\mathcal{P}$ je* MDyLP *nad jazykom $\mathcal{L}$,* n *je index a M je interpretácia nad $\mathcal{L}$. Potom M je zlepšený multi-stabilný model programu $\mathcal{P}$ v P_n práve vtedy, keď M je dobre podporený model $\mathcal{P}$ v P_n*

Kapitola 3

Prínos

Pozitívne programy sú veľmi obmedzujúce a umožňujú popísanie iba jednoduchých situácií. Na popísanie zložitejších situácií používame pravidlá s negáciou v tele pravidla. Avšak ani toto nestačí, nakoľko je veľa situácií, ktoré je možné popísať bez negácie v hlave pravidla. Preto ďalším prirodzeným krokom bolo umožnenie pravidlám mať negáciu aj v hlave pravidla. Takéto pravidlá síce zlepšili možnosti popísať rôzne komplikované situácie, avšak priniesli so sebou aj konflikt pravidiel. Pomocou jednoduchého príkladu si ukážeme možnosti ako pristupovať k takýmto konfliktom:

Príklad 3.0.3 *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 *oddelene, 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 už nie je jednoznačný. Zjavne potrebujeme riešiť situáciu, kedy by bol atóm A pravdivý aj nepravdivý zároveň. Používajú sa tri prístupy:*

1) *Nekonzistencia sa nerieši, výsledkom je celý jazyk* $\mathcal{L}_\mathcal{P}$*. Tento prístup sa nazýva explozívny prístup a nekonzistenia spôsobí, že nemáme žiadne relevantné výsledky*

2) *Nekonzistenciu identifikujeme, ale neovplyvní nezávislú časť znalostnej bázy. V tomto príklade sú výsledkom pravdivé atómy B, C, D a nekonzistentné A. Toto je parakonzistentný prístup.*

3) *Aby sme nekonzistencii predišli, odstránime menej preferované informácie, teda buď odstránime $A \leftarrow B$ ak program P_1 je menej prioritný ako program P_2 a dostaneme pravdivé B, C, D a nepravdivé A; alebo odstránime $notA \leftarrow B$ ak program P_2 je menej dôležitý ako program P_1 a dostaneme pravdivé A, B, C, D*

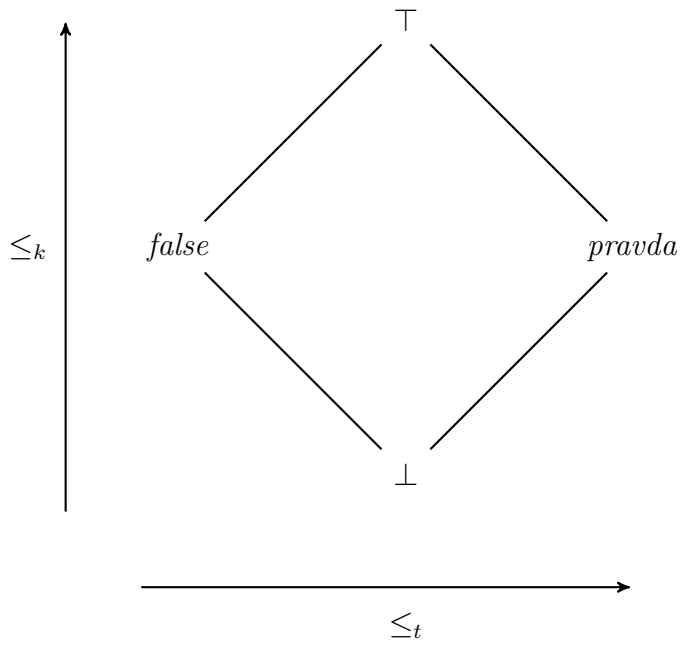
My predstavujeme prístup, ktorý je kombinácia 2. a 3. prístupu z príkladu, teda prístup, ktorý nám umožní predísť nekonzistencii, tam kde sa to dá, zamietaním menej prioritných pravidiel v prípade konfliktu, pri ktorom vieme porovnať priority pravidiel. V prípade, že nevieme porovnať prioritu pravidiel, a teda nie sme schopní určiť pravidlo, ktoré zamietne. V takom prípade dostaneme nekonzistenciu, s ktorou však budeme vedieť ďalej pracovať, čo nám umožní zachovať výsledky, ktoré nie sú nekonzistenciou ovplyvnené.

3.1 Zavedenie sémantiky

Keďže sa chceme vedieť vysporiadať s nekonzistenciou v prípade, že jej nevieme predísť, budeme potrebovať viac pravdivostných hodnôt ako iba *pravda* a *nepravda*. Použijeme dvojzväz, keďže ponúka zaujímavé vlastnosti. Výhodou použitia dvojzväzu je to, že sa nemúsíme obmedzovať na konkrétne pravdivostné hodnoty, ale môžeme použiť aj iný dvojzväz s rovnakými vlastnosťami a revízia znalostí, teda zamietanie konfliktných pravidiel, bude fungovať aj naďalej. Taktiež nevyžadujeme aby bol počet pravdivostných hodnôt konečný.

Budeme používať dvojzväzový súčin $\mathcal{L}_1 \odot \mathcal{L}_2 = (\mathbf{L}_1 \times \mathbf{L}_2, \leq_t, \leq_k)$, kde $\mathcal{L}_1 = (\mathbf{L}_1, \leq_t)$ a $\mathcal{L}_2 = (\mathbf{L}_2, \leq_k)$ sú zväzy. Pre jednoduchosť budeme najmenší prvok vzhľadom na pravdivostné usporiadanie ($\leq_t$) označovať ako *nepravda* a najväčší prvok na toto usporiadanie budeme označovať ako *pravda*. Podobne pre najmenší prvok vzhľadom na znalostné usporiadanie ($\leq_k$) budeme

používať symbol $\bot$ a pre najväčší prvok vzhľadom na toto usporiadanie budeme používať symbol $\top$. Vzťah medzi týmito extrémnymi prvkami a usporiadaniami vidno je možné vidieť na obrázku 3.1.



Obr. 3.1: extrémne prvky vzhľadom na usporiadania $\leq_t$ a $\leq_k$

Usporiadania definujeme $<_t$, $\geq_t$, $>_t$, $<_k$, $\geq_k$ a $>_k$ obvyklým spôsobom.

Definícia 3.1.1 *Nech* I *je interpretácia a nech* $\mathcal{L}_1 \odot \mathcal{L}_2$ *je dvojzväzový súčin* $\mathcal{L}_1 \odot \mathcal{L}_2 = (\mathbf{L}_1 \times \mathbf{L}_2, \leq_t, \leq_k)$*, kde* $\mathcal{L}_1 = (\mathbf{L}_1, \leq_t)$ *a* $\mathcal{L}_2 = (\mathbf{L}_2, \leq_k)$ *sú zväzy. Valuácia* val_I *je funkcia* $val_I : \mathcal{L}_\mathcal{P} \mapsto \mathcal{L}_1 \odot \mathcal{L}_2$ *taká, že pre každý literál* $A \in \mathcal{L}_\mathcal{P}$ *platí:*

- *pravda* $\leq_k val_I(A)$ a $\top \leq_t val_I(A)$ ak $A \in I$
- $val_I(A) \leq_k$ *pravda* a $\bot \leq_t val_I(A)$ ak $A \notin I$
- *nepravdaval*$_I \leq_k (A)$ a $val_I(A) \leq_t \top$ ak *not* $A \in I$
- $val_I(A) \leq_k$ *nepravda* a $val_I(A) \leq_t \bot$ ak *not* $A \notin I$

- $val_I(not\ A) = not\ val_I(A) = < 1 - b, 1 - a >$,
  kde $< a, b > = val_I(A)$ a $a, b \in \{0, 1\}$.
  Pozn.: $not\ val_I(A)$ budeme niekedy zapisovať aj $1 - val_I(A)$

- $val_I(L_1 \wedge \cdots \wedge L_n) = min_t\{val_I(L_1), \ldots, val_I(L_n)\}$,
  kde min_t je minimum vzhľadom na usporiadanie $\leq_t$

- $val_I(L \leftarrow C) = pravda\ ak\ val_I(C) \leq_t val_I(L)$

Keďže jeden program môže mať viac interpretácií, veľmi by sa nám hodila možnosť porovnať 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 3.1.2 *Nech* I *je interpretácia, potom:*

- $I^+ = \{A \mid A \in I\}$

- $I^- = \{A \mid not\ A \in I\}$

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 3.1.3 *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 3.1.4 *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.
Chceme používať program s aktualizáciami a zamietať konfliktné pravidlá na základe porovnania priority pravidiel, a preto budeme potrebovať rovnaké definície s istými obmenami, keďže používame viac pravdivostných hodnôt.

Definícia 3.1.5 *Nech I je interpretácia a nech* $\mathcal{P}$ *je dynamický logický program, teda program a jeho aktualizácie, potom:*

- $Defaults(\mathcal{P}, I) = \{not\ A \mid A \notin I,\ not\ A \in I\}$

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

$$Rejected(\mathcal{P}, I, i) = \{r \mid r \in P_i, \exists r' \in P_j,\ i < j,\ r \bowtie r',$$
$$not\ val_I(Body(r')) <_t val_I(Body(r)),\ P_i, P_j \in \mathcal{P}\}$$

Zmena v množine *Rejected* je zaujímavá a hlavne nie je na prvý pohľad jasne, čo sme ňou mysleli, tak sa na ňu pozrieme lepšie. Viac pravdivostných hodnôt spôsobilo, že situácia je trochu komplikovanejšia a 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 *MDyLP*. Myšlienka, ku ktorej sme dospeli je, že menej prioritné 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 $val_I(notBody(r')) <_t val_I(Body(r))$.
Operácia $\bowtie$ vyjadruje konflikt pravidiel presnejšie konflikt ich hláv, teda

$r \bowtie r'$ vtedy a len vtedy, keď $not\, Head(r) = Head(r')$. Pri $r \bowtie r'$ môžu nastať dve možnosti a to:

(I) nech $Head(r) = A$ a $Head(r') = not\, A$

(1) $not\, Head(r) = Head(r')$

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

(3) $val_I(not\, A) = not\, val_I(A)$

(4) $val_I(Body(r')) \leq_t not\, val_I(A)$

(5) $not\, val_I(Body(r')) \geq_t val_I(A)$

(6) $val_I(Head(r)) \leq_t not\, val_I(Body(r'))$

(7) $Head(r) \leftarrow Body(r)$
ak $val_I(Body(r)) \leq_t val_I(Head(r))$

(8) $val_I(Body(r)) \leq_t val_I(Head(r)) \leq_t not\, val_I(Body(r'))$

(9) takže $not\, val_I(Body(r'))$ a $val_I(Body(r))$ tvoria hornú a dolnú hranicu pre $val(Head(r))$ a pravidlo zamietneme ak bude horná hranica menšia ako dolná hranica, teda ak
$not\, val_I(Body(r')) <_t val_I(Body(r))$

(II) nech $Head(r) = not\, A$ a $Head(r') = A$

(1) $not\, Head(r) = Head(r')$

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

(3) $val_I(not\, A) = not\, val_I(A)$

(4) $val_I(Body(r')) \leq_t not\, val_I(not\, A)$

(5) $not\, val_I(Body(r')) \geq_t val_I(not\, A)$

(6) $val_I(Head(r)) \leq_t not\, val_I(Body(r'))$

(7) $Head(r) \leftarrow Body(r)$
ak $val_I(Body(r)) \leq_t val_I(Head(r))$

(8) $val_I(Body(r)) \leq_t val_I(Head(r)) \leq_t not\ val_I(Body(r'))$

(9) takže $not\ val_I(Body(r'))$ a $val_I(Body(r))$ tvoria hornú a dolnú hranicu pre $val(Head(r))$ a pravidlo zamietneme ak bude horná hranica menšia ako dolná hranica, teda ak
$not\ val_I(Body(r')) <_t val_I(Body(r))$

Keď poznáme pravidlá, ktoré zamietame, vyrobíme jeden program, už bez aktualizácií, ktorý už nebude obsahovať zamietnuté pravidlá. Takže sme sa zbavili nekonzistencií, ktorým sa dalo predísť.

Definícia 3.1.7 *Nech $\mathcal{P}$ je DLP nad $\mathcal{L}$ a I je interpretácia nad $\mathcal{L}$, potom:*

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

Program P' budeme volať aktualizovaný *program programu $\mathcal{P}$.*

Ďalší postup bude podobný tomu, ktorý je aplikovaný pri hľadaní stabilných modelov. Urobíme redukt vzhľadom na kandidáta na model, čím sa zbavíme negácií, následne vypočítame model tohto reduktu, ktorý porovnáme s kandidátom na model programu. Samozrejme redukt musí vyhovovať aj prípadným ohraničeniam, ktoré mohli vzniknúť pri redukcii.

Definícia 3.1.8 *Redukt P'_I programu P' vzhľadom na interpretáciu I urobíme tak, že každé pravidlo obsahujúce premenné nahradíme všetkými jeho uzemnenými inštanciami a následne:*

- *vynecháme všetky pravidlá obsahujúce $not\, L$ v tele ak $L \in I$ alebo $not\, L \notin I$*
- *vynecháme všetky $not\, L$ z tiel pravidiel také, že $L \notin I$ a $not\, L \in I$*
- *vynecháme všetky ohraničenia*
- *vynecháme všetky pravidlá obsahujúce negáciu v hlave a presunutím hlavy (bez negácie) do tela pravidla získame ohraničenia*

Definícia 3.1.9 *Nech $\mathcal{P}$ je DLP nad $\mathcal{L}$ a nech P' je z neho odvodený aktualizovaný program. Hovoríme, že M je parakonzistentný stabilný model pre DLP (*p-DJU*) vtedy a len vtedy, keď M je model P'_M, ktorý vznikol ako redukt programu P, vyhovuje všetkým ohraničeniam programu a neexistuje model N programu P'_M, taktiež vyhovujúci ohraničeniam, pre ktorý platí, že $N \prec_t M$.*

Dynamický logický program tvorí postupnosť programov, preto sémantiku skúsime rozšíriť pre multidimenzionálne dynamické logické program. V prípade *MDyLP* založených na stabilných modeloch alebo dobre podporených modeloch dostávame informáciu o priorite z orientovaného acyklického grafu, v ktorom vrcholy predstavujú jednotlivé logické programy.
To, že miesto postupnosti programov máme program a aktualizácie organizované pomocou grafu závislostí, spôsobí len menšie zmeny v definíciách, ktorých význam sa nezmenil.

Definícia 3.1.10 *Nech I je interpretácia a $\mathcal{P}$ je MDyLP, potom:*

$$Defaults(\mathcal{P}, I) = \{not\ A \mid A \notin I,\ not\ A \in I\}$$

Definícia 3.1.11 *Nech I je interpretácia a $\mathcal{P}$ je dynamický logický program, teda program a jeho aktualizácie. Nech ℓ je mapovacia funkcia a $\mathcal{D}$ je graf závislostí, potom:*

$$Rejected_\ell(\mathcal{P}_\mathcal{D}, I, i) = \{r \in P_i \mid \exists r' \in P_j,\ i <_D j,\ r \bowtie r',$$
$$not\ val_I(Body(r'))\ <_t\ val_I(Body(r)),\ \ell(Head(r'))\ >\ \ell(Body(r')),\ P_i,$$
$$P_j \in \mathcal{P}\}$$

Definícia 3.1.12 *Nech $\mathcal{P}$ je MDyLP nad $\mathcal{L}$, $\mathcal{D}$ je graf závislostí, ℓ je mapovacia funkcia a I je interpretácia nad $\mathcal{L}$, potom:*

- $Residue_\ell(\mathcal{P}_\mathcal{D}, I) = \bigcup_{i \in V} [P_i \setminus Rejected_\ell(\mathcal{P}_\mathcal{D}, I, i)]$
- $P' = Residue_\ell(\mathcal{P}_\mathcal{D}, I) \cup Defaults(\mathcal{P}_\mathcal{D}, I)$

Program P' budeme volať aktualizovaný *program programu $\mathcal{P}$.*

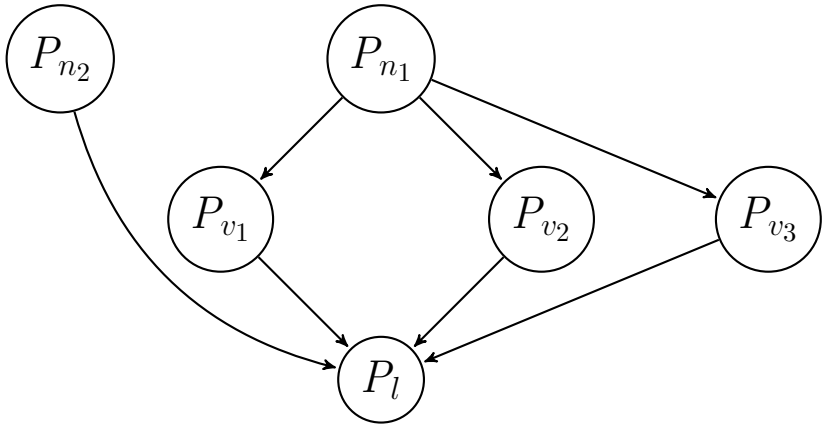
Definícia 3.1.13 *Redukt P'_M programu P' podľa kandidáta na model M urobíme tak, že každé pravidlo obsahujúce premenné nahradíme všetkými jeho uzemnenými inštanciami a následne:*

- *vynecháme všetky pravidlá obsahujúce not L v tele ak $L \in M$*
- *vynecháme všetky not L z tiel pravidiel také, že not $L \in M$*
- *vynecháme všetky ohraničenia*
- *vynecháme všetky pravidlá obsahujúce negáciu v hlave a presunutím hlavy (bez negácie) do tela pravidla získame ohraničenia*

Definícia 3.1.14 *Nech program $\mathcal{P}$ je MDyLP nad $\mathcal{L}$, M je model programu P', $\mathcal{D}$ je graf závislostí a P'_M je redukt aktualizovaného programu P'. Potom hovoríme, že M je* parakonzistentný dobre podporený model *pre MDyLP (p-WS pre MDyLP) ak vyhovuje všetkým ohraničeniam programu a existuje mapovacia funkcia ℓ taká, že pre všetky $A \in M$ $\exists r \in P'_M$ také, že $Head(r) = A$, $\ell(A) > \ell(Body(r))$ a neexistuje taký model N programu P'_M, ktorý taktiež vyhovuje všetkým ohraničeniam programu, pre ktorý platí, že $N \prec_t M$.*

Príklad 3.1.1 *Ako príklad si predstavme, že máme pacienta v nemocnici, u ktorého je podozrenie na niekoľko závažných diagnóz. Jedna z diagnóz je nádor, pri ktorom je možné použiť rôzne metódy liečby v závislosti od viacerých faktorov. Pre účely nášho príkladu sme tieto metódy a faktory zjednodušili. Metódy sú neinvazívna a invazívna, teda operácia. Pokiaľ to nie je nutné, preferujeme neinvazívnu metódu, nakoľko menej zaťažuje pacientov zdravotný stav. Preto bude použitá invazívna metóda, len v prípade, že nádor je zhubný a teda treba ho urýchlene odstrániť. Aby sme zistili, či je nádor zhubný, môžeme použiť dve rôzne vyšetrenia. Výsledky jedného z vyšetrení hovoria, že nádor je zhubný a výsledky druhého vyšetrenia hovoria, že nádor je nezhubný, čím sme dostali spor. Máme nekonzistentnú informáciu, ale je nutné aby sme aj na základe nekonzistentnej informácie urobili nejaký záver. V prípade nádoru by bola uprednostnená operácia, teda invazívna metóda, nakoľko nechceme riskovať život pacienta. U pacienta bolo podozrenie na ďalšiu*

závažnú diagnózu, vysoký krvný tlak, avšak tá sa ďalším vyšetrením nepotvrdila.
Takže máme program a aktualizácie, taktiež máme graf závislostí zobrazený na obrázku 3.2.



Obr. 3.2: vyšetrenie

jednotlivým vyšetreniam 3.2 priradíme jednoduché programy:

$P_{n_1} = \{nador \leftarrow ,\ vysoky\_krvny\_tlak \leftarrow \}$

$P_{v_1} = \{zhubny\_nador \leftarrow \}$

$P_{v_2} = \{not\ zhubny\_nador \leftarrow \}$

$P_{n_2} = \{potravinova\_intolerancia \leftarrow \}$

$P_{v_3} = \{not\ vysoky\_krvny\_tlak \leftarrow \}$

$P_l = \{operacia \leftarrow zhubny\_nador,\ dieta \leftarrow potravinova\_intolerancia\}$

model tohto multidimenzionálneho dynamického logického programu je:
$M = \{nador, zhubny\_nador, not\ zhubny\_nador, operacia, vysoky\_krvny\_tlak, potravinova\_intolerancia, dieta\}$,
teda nádor, operácia, potravinová_intolerancia *a* diéta *sú pravdivé,*
vysoký_krvný_tlak *je nepravdivý a* zhubný_nádor *je nekonzistentný.*
Je zaujímavé porovnať si náš prístup s tradičnými prístupmi na vyrovnanie sa s nekonzistenciou, konkrétne s parakonzistentným prístupom a prístup s aktualizáciami. Parakonzistentný prístup nepracuje s aktualizáciami, čím sa stráca informácia o priorite pravidiel. Parakonzistentný prístup nezamieta pravidlá, ale akceptuje prípadnú nekonzistenciu. Model teda bude vyzerať nasledovne $M = \{nador, zhubny\_nador, not\ zhubny\_nador, operacia, vysoky\_krvny\_tlak, not\ vysoky\_krvny\_tlak, potravinova\_intolerancia, dieta\}$.
Rozdiel je v tom, že vysoky_krvny_tlak *nie je nepravdivý ale nekonzistentný. Pri spracovávaní aktualizácií narazíme na konflikt dvoch neporovnateľných pravidiel v programoch* P_{v_1} *a* P_{v_2}*, čo spôsobí, že sa nepodarí predísť nekonzistencii a teda nebude existovať dvojhodnotový model.*

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 informácia odvodená z nekonzistencie použiteľná, ale sú situácie kedy nekonzistencia v odvodení nemení nič na závere, k akému sme odvodením dospeli. Keďže toto všetko je závislé od danej 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 popíšeme spôsob na zachovanie informácie o prípadnej nekonzistencii v odvodení.

Budeme používať množinu podozrivých atómov S. Atómy, ktoré sú nekonzistentné, sú automaticky podozrivé. Ďalej budeme považovať atóm za nekonzistentný, ak každé pravidlo, ktoré ho podporuje, je podozrivé pravidlo. Na predchádzajúcu podmienku sa môžeme pozerať aj tak, že atóm budeme považovať za podozrivý, ak neexistuje pravidlo, ktorého hlava je pozitívny alebo negatívny tvar daného atómu a toto pravidlo nie je podozrivé. Treba však dodať, že atómy, pre ktoré neexistuje žiadne (podozrivé alebo nepodozrivé) pravidlo, ktorého hlava je pozitívny alebo negatívny tvar tohto atómu, podozrivý nie je a takéto pre atómy vytvoríme pomocnú množinu Q. Zjavne budeme používať aj množinu podozrivých pravidiel R. Pravidlo budeme považovať za podozrivé, ak sa v jeho tele nachádza nejaký podozrivý atóm (v pozitívnom alebo negatívnom tvare).
Najskôr do množiny S vložíme všetky nekonzistentné atómy. Taktiež si hneď pripravíme aj množinu Q. Následne budeme postupne budovať obe množiny S aj R naraz, až pokiaľ nebudeme vedieť pridať prvok ani do jednej z nich.

Definícia 3.2.1 *Nech I je interpretácia, $\mathcal{P}$ je MDyLP, $\mathcal{D}$ je graf závislostí a ℓ je mapovacia funkcia. Nech $P' = Residue_\ell(\mathcal{P}_\mathcal{D}, I) \cup Defaults(\mathcal{P}_\mathcal{D}, I)$. Budeme postupne budovať dve množiny a to množinu podozrivých literálov S a množinu pravidiel ktorých telá obsahujú doteraz nájdené podozrivé literály R. Najskôr si pripravíme množinu S, keďže určite bude obsahovať všetky nekonzistentné A: $S(P', I) = \{A \mid val_I(A) = \top\}$. Taktiež si pripravíme množinu literálov, kam bude patriť každý literál, pre ktorý nie je v hlave žiadneho pravidla, teda $Q = \{A \mid \nexists r \in P', Head(r) = A \vee Head(r) = not\, A\}$. Teraz*

môžeme naplniť množiny S a R a budeme tak robiť v cykle:

(1) $R(P', I) = \{r \mid \exists A \in S(P', I),\ (A \in Body(r)) \vee (not\ A \in Body(r)),\ r \in P'\}$

(2) $S(P', I) = S(P', I) \cup \{A \mid \nexists r \in P' \setminus R(P', I), Head(r) = A \vee Head(r) = not\ A, A \notin Q\}$

(3) Ak sa zmenila množina S alebo R, tak pokračujeme bodom (1).

Tento cyklus budeme opakovať kým sa množiny S alebo R menia a preto je dobre spomenúť, že ak je program P' konečný, tak tento proces určite skončí lebo v každom kroku pridáme aspoň jeden literál alebo aspoň jedno pravidlo.

3.3 simulácia *IV* pomocou dvoj-hodnotovej logiky

Najjednoduchší rozumný príklad dvojzväzového súčinu $\mathcal{L}_1 \odot \mathcal{L}_2 = (\mathbf{L}_1 \times \mathbf{L}_2, , \leq_t, \leq_k)$, kde $\mathcal{L}_1 = (\mathbf{L}_1, \leq_t)$ a $\mathcal{L}_2 = (\mathbf{L}_2, \leq_k)$ sú dvojzväzy, je dvojzväz $(\{0,1\} \times \{0,1\}, \leq_t, \leq_k)$, ktorý budeme jednoducho označovať $\mathcal{IV}$. Pravdivostné hodnoty tohto dvojzväzu sú samozrejme interval, a teda majú hornú a dolnú hranicu. Tieto hranice môžu nadobúdať iba dve rôzne hodnoty a to 0 alebo 1, preto sa na ne môžeme pozerať ako na špeciálne literály. Preto pre každý literál $L \in \mathcal{L}$ zavedieme dva nové literály $L_l, L_h \in \mathcal{L}'$ predstavujúce dolnú a hornú hranicu intervalu. Pre tieto nové literály nám budú stačiť dve pravdivostné hodnoty nepravda a pravda reprezentované 0 a 1. Pre poriadok spomenieme, že aj tu existuje usporiadanie, ktoré je podobné pravdivostnému usporiadaniu $\leq_t$ použitému pri dvozjväza *IV*, preto ho budeme tiež označovať $\leq_t$ a bude fungovať tak ako intuitívne očakávame teda $0 \leq_t 1$.

Definícia 3.3.1 *Nech $val : \mathcal{L} \to IV$ je valuácia. Odvodíme novú valuáciu $val' : \mathcal{L}' \to \{0,1\}$ a chceme $val(A) = \langle val'(A_l), val'(A_h) \rangle$ teda:*

- *ak $val(A) = \langle 0, 0 \rangle$ potom $val'(A_l) = 0$ a $val'(A_h) = 0$*

- *ak* $val(A) = \langle 1, 1 \rangle$ *potom* $val'(A_l) = 1$ *a* $val'(A_h) = 1$
- *ak* $val(A) = \langle 0, 1 \rangle$ *potom* $val'(A_l) = 0$ *a* $val'(A_h) = 1$
- *ak* $val(A) = \langle 1, 0 \rangle$ *potom* $val'(A_l) = 1$ *a* $val'(A_h) = 0$
- *not* $val(A) = 1 - val(A) = \langle not\ val'(A_h), not\ val'(A_l) \rangle$ *kde* $not\ val'(A_x) = 1 - val'(A_x)$ *pričom* $x \in \{l, h\}$
- $val(A_1 \wedge \cdots \wedge A_m \wedge not\ B_1 \wedge \cdots \wedge not\ B_n) =$
  $\langle val'(A_{1,l} \wedge \cdots \wedge A_{m,l} \wedge not\ B_{1,h} \wedge \cdots \wedge not\ lit[B_{n,h}]),$
  $val'(A_{1,h} \wedge \cdots \wedge A_{m,h} \wedge not\ B_{1,l} \wedge \cdots \wedge not\ lit[B_{n,l}]) \rangle =$
  $\langle min(val'(A_{1,l}), \ldots, val'(A_{m,l}), 1 - val'(B_{1,h}), \ldots, 1 - val'(B_{n,h})),$
  $min(val'(A_{1,h}), \ldots, val'(A_{m,h}), 1 - val'(B_{1,l}), \ldots, 1 - val'(B_{n,l})) \rangle$
- $val(L \leftarrow C) = \langle val'(L_l \leftarrow C_l), val'(L_h \leftarrow C_h) \rangle$
  kde $val'(L_x \leftarrow C_x) = 1$
  vtedy a len vtedy, keď $val'(C_x) \leq_t val'(L_x)$
  pričom $x \in \{l, h\}$ *a nech* $C = A_1 \wedge \cdots \wedge A_m \wedge not\ B_1 \wedge \cdots \wedge not\ B_n$
  potom $C_l = A_{1,l} \wedge \cdots \wedge A_{m,l} \wedge not\ B_{1,h} \wedge \cdots \wedge not\ B_{n,h}$
  a $C_h = A_{1,h} \wedge \cdots \wedge A_{m,h} \wedge not\ B_{1,l} \wedge \cdots \wedge not\ B_{n,l}$

Tento princíp môžeme použiť aj na pravidlá a vyrobiť tak nové pravidlá obsahujúce nové literály. Pri generovaní nových pravidiel si stačí uvedomiť, že $val(L) = \langle val'(L_l), val'(L_h) \rangle$ a $val(not\ L) = \langle val'(not\ L_h), val'(not\ L_l) \rangle$. Podobne ako valuáciu, budeme chcieť odvodiť aj novú interpretáciu.

Definícia 3.3.2 *Nech I je interpretácia nad $\mathcal{L}$. Chceme odvodiť interpretáciu I' nad $\mathcal{L}'$ tak, že pre každé $A \in \mathcal{L}$ platí:*

- *ak* $A \in I \wedge not\ A \notin I$ *tak* $A_l \in I' \wedge A_h \in I'$
- *ak* $A \notin I \wedge not\ A \in I$ *tak* $A_l \notin I' \wedge A_h \notin I'$
- *ak* $A \in I \wedge not\ A \in I$ *tak* $A_l \in I' \wedge A_h \notin I'$
- *ak* $A \notin I \wedge not\ A \notin I$ *tak* $A_l \notin I' \wedge A_h \in I'$

Keďže I' je dvojhodnotová interpretácia tak ak v interpretácii nepíšeme negatívne atómy $not\, A$ a teda ak $A \in I'$ tak $val_{I'}(A) = 1$ a ak $A \notin I'$ tak $val_{I'}(A) = 0$.

Majme program P pracujúci nad štyrmi pravdivostnými hodnotami dvojzväzu IV. Ukážeme si ako vyrobiť program, ktorý bude pracovať nad dvoma pravdivostnými hodnotami. Chceme aby tento program odvodiť z pôvodného a chceme aby generoval podobné modely (pôvodné atómy budú reprezentované novými, ktoré zastávajú horné a dolné hranice ich intervalov pravdivostných hodnôt). Postup je taký, že každý literál je nahradený dvoma novými literálmi, predstavujúcimi hornú a dolnú hranicu intervalu, ktorý je pôvodnému literálu priradený ako pravdivostná hodnota. Hoci pôvodný program pracoval so štyrmi pravdivostnými hodnotami, odvodený program používa dva literály pre každý pôvodný literál a teda na simulovanie všetkých štyroch možných pravdivostných hodnôt stačia nový dve pravdivostné hodnoty. Rovnako budeme mať aj dvojice pravidiel za rovnakým účelom.

Definícia 3.3.3 *Nech $\mathcal{P}$ je MDyLP nad $\mathcal{L}$. Chceme vyrobiť nový MDyLP $\mathcal{P}'$ nad $\mathcal{L}'$ tak, že:*

- *Pre každý program $P \in \mathcal{P}$ vyrobíme program $P' \in \mathcal{P}'$ tak, aby sme zachovali graf závislostí.*
- *Pre každé pravidlo $r \in P$ vyrobíme dve nové pravidlá $r_l, r_h \in P'$:*

  Nech pravidlo r má hlavu pravidla A, potom hlavou pravidla r_l bude nový literál A_l a podobne pre pravidlo r_h teda $Head(r_l) = A_l$ a $Head(r_h) = A_h$

  Nech pravidlo r má hlavu pravidla $not\, A$, potom hlavou pravidla r_l bude nový literál $not\, A_h$ a podobne pre pravidlo r_h teda $Head(r_l) = not\, A_h$ a $Head(r_h) = not\, A_l$

  Nech telom pravidla r je konjunkcia $A_1 \wedge \cdots \wedge A_m \wedge not\, B_1 \wedge \cdots \wedge not\, B_n$, potom telo pravidla r_l bude konjunkcia $A_{1,l} \wedge \cdots \wedge A_{m,l} \wedge not\, B_{1,h} \wedge \cdots \wedge not\, B_{n,h}$ a telo pravidla r_h bude konjunkcia $A_{1,h} \wedge \cdots \wedge A_{m,h} \wedge not\, B_{1,l} \wedge \cdots \wedge not\, B_{n,l}$.

Keďže literály L_l a L_h tvoria pár mali by mať aj rovnakú hodnotu mapovacej funkcie, preto pre mapovacia funkciu ℓ zavedieme novú *rozšírenú mapovacia funkciu* $\ell' : \mathcal{L}' \to \mathbb{N}$ tak, že pre každý literál $L \in \mathcal{L}$ $\ell(L) = \ell'(L_l) = \ell'(L_h)$. pre všetky $L' \in \mathcal{L}'$ zachováme aj vlastnosť, že $\ell'(L') = \ell'(not\, L')$. 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 0. Aby sme mohli pracovať s aktualizáciami, tak ako pôvodný program, musíme kúsok upraviť definície množín *Defaults*, *Rejected* a *Residue* aby vedeli pracovať s dvojicami literálov a dvojicami pravidiel.

Definícia 3.3.4 *Nech $\mathcal{P}$ je MDyLP nad jazykom $\mathcal{L}$ a I je štvor-hodnotová interpretácia nad jazykom $\mathcal{L}$ a val k nej príslušná interpretácia. Nech $\mathcal{L}'$ je odvodený jazyk a I' je interpretácia nad $\mathcal{L}'$ odvodená z I a val$'$ je valuácia príslušná k I' potom:*

$$Defaults(\mathcal{P}, I') = \{not\, A_l, not\, A_h \mid val'(A_l) = 0 \wedge val'(A_h) = 0\}$$

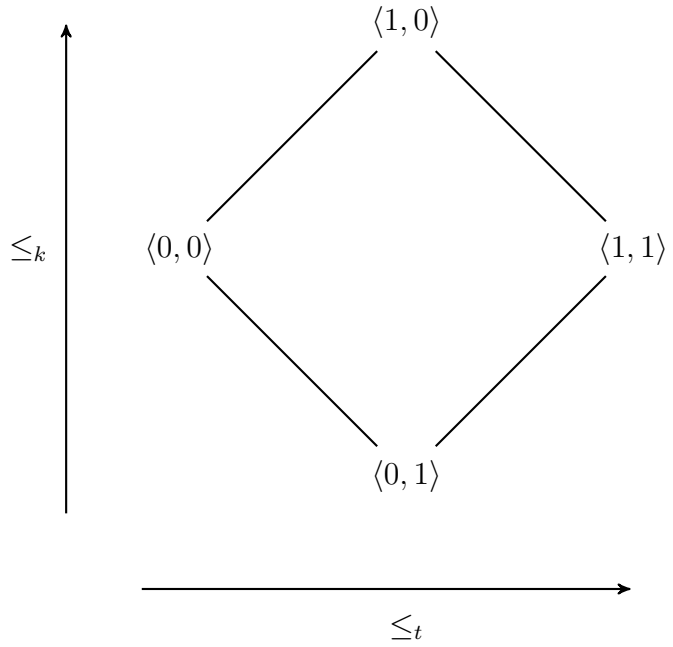
Pri zamietaní pravidiel sme potrebovali porovnávať hlavy a aj telá zamietaného a aj zamietajúceho pravidla. Keďže pri našich nových pravidlách máme na oboch stranách dve pravidlá, situácia sa stáva menej prehľadnou. Preto zavedieme funkciu, reprezentovanú symbolom $\Rightarrow\Leftarrow$, ktorá bude testovať či dané dve dvojice pravidiel spĺňajú podmienky na zamietnutie pravidiel.

Definícia 3.3.5 *Nech I je dvojhodnotová interpretácia odvodená zo štvorhodnotovej J a val valuácia príslušná k I. Pravidlá r_l a r_h zamietneme ak pravidlá r_l, r_h, r'_l a r'_h spĺňajúce nasledujúce podmienky:*

- $val(Head(r_l)) = not\, val(Head(r'_h))$
  $\wedge val(Head(r_h)) = not\, val(Head(r'_l))$
- $\Big(not\, val(Body(r'_h)) < val(Body(r_l))$
  $\wedge\, not\, val(Body(r'_l)) < val(Body(r_h))\Big)$
  $\vee\Big(not\, val(Body(r'_h)) < val(Body(r_l))$
  $\wedge\, not\, val(Body(r'_l)) = val(Body(r_h))\Big)$

$$\vee\Big(not\ val(Body(r'_h)) = val(Body(r_l)) \\ \wedge not\ val(Body(r'_l)) < val(Body(r_h))\Big)$$

To, že pravidlá r_l, r_h, r'_l *a* r'_h *spĺňajú tieto podmienky budeme zapisovať* (r_l, r_h) $\Rightarrow\Leftarrow$ (r'_l, r'_h).



Obr. 3.3: usporiadania $\leq_t$ a $\leq_k$

Prvá podmienka je hovorí, že nech pravidlá r a r' sú pravidlá, od ktorých boli r_l, r_h, r'_l a r'_h odvodené, potom platí, že $r \bowtie r'$.
Druhá podmienka je menej zjavná avšak dosahuje rovnaký efekt ako pôvodná podmienka $not val(Body(r')) <_t val(Body(r))$. Je dobré pripomenúť, že valuácia *val* pracuje nad štvorhodnotovou logikou IV a pravdivostné usporiadanie má vlastnosti ako je možné vidieť na obrázku 3.3 a teda podmienku spĺňajú iba kombinácie: $\langle 0,0\rangle$ a $\langle 1,0\rangle$, $\langle 0,0\rangle$ a $\langle 0,1\rangle$, $\langle 0,0\rangle$ a $\langle 1,1\rangle$, $\langle 1,0\rangle$ a $\langle 1,1\rangle$, $\langle 0,1\rangle$ a $\langle 1,1\rangle$. Pravidlá r_l a r'_l reprezentujú dolnú hranicu teda prvú zložku intervalu a pravidlá r_h a r'_h reprezentujú hornú hranicu intervalu. Takže aby intervaly

reprezentované týmito pravidlami spĺňali pôvodnú podmienku musia spĺňať túto novú podmienku.

Definícia 3.3.6 *Nech I je dvojhodnotová interpretácia a $\mathcal{P}$ je MDyLP. Nech $\mathcal{D}$ je graf závislostí a ℓ' je rozšírená mapovacia funkcia, potom:*

$$\begin{aligned}&Rejected_\ell(\mathcal{P}_\mathcal{D}, I, i) = \{r_l, r_h \mid r_l, r_h \in P_i, \exists r'_l, r'_h \in P_j,\ i <_D j,\\ &(r_l, r_h) \Rightarrow\Leftarrow (r'_l, r'_h),\ \ell'(Head(r'_l)) > \ell'(Body(r'_l)),\\ &\ell'(Head(r'_h)) > \ell'(Body(r'_h)),\ P_i, P_j \in \mathcal{P}\}\end{aligned}$$

Definícia 3.3.7 *Nech $\mathcal{P}$ je MDyLP nad $\mathcal{L}$, $\mathcal{D}$ je graf závislostí, ℓ je rozšírená mapovacia funkcia a I je interpretácia nad $\mathcal{L}_\mathcal{P}$, potom:*

- $Residue_\ell(\mathcal{P}_\mathcal{D}, I) = \bigcup_{i \in V} [P_i \setminus Rejected_\ell(\mathcal{P}_\mathcal{D}, I, i)]$
- $P' = Residue_\ell(\mathcal{P}_\mathcal{D}, I) \cup Defaults(\mathcal{P}_\mathcal{D}, I)$

Program P' budeme volať aktualizovaný *program programu $\mathcal{P}$.*

Upravíme aj spôsob, akým získavame redukt. Pri výpočte reduktu simulujeme výpočet reduktu pôvodného programu, čím získame program odvodený z reduktu pôvodného programu. Avšak to neznamená že tento odvodený redukt je redukt v stave ako ho my potrebujeme, preto urobíme ďalší redukt, tentokrát vzhľadom na dvojhodnotovú interpretáciu. A tento redukt použijeme na výpočet modelu.

Definícia 3.3.8 *Nech $\mathcal{P}$ je* MDyLP *a J je štvorhodnotová interpretácia, obe nad jazykom $\mathcal{L}$. Nech $\mathcal{P}'$ je program odvodený z $\mathcal{P}$ a I je dvojhodnotová interpretácia odvodená z J. Nech val_I je valuácia príslušná k I a $P = Residue_\ell(\mathcal{P}'_\mathcal{D}, I) \cup Defaults(\mathcal{P}'_\mathcal{D}, I)$.* Redukt *programu P vhľadom na I je program P', ktorý dostaneme z P nahradením všetkých pravidiel obsahujúcich premenné ich uzemnenými inštanciami a následovne aplikujeme tento postup:*

najskôr si pripravíme redukt ako by vyzeral keby pracovali nad IV*:*

- *vymazaním pravidiel r_l a r_h ak sa v tele pravidla r_l vyskytuje not A_h a ak sa v tele pravidla r_h vyskytuje not A_l pričom $val_I(A_l) = 1$ a $val_I(A_h) = 1$*
- *vymazaním A_l a A_h ak sa obe vyskytujú v telách príslušných pravidiel v podobe not A_l a not A_h ak $val_I(A_l) = 0$ a $val_I(A_h) = 0$*

a potom urobíme redukt pre klasický dvojhodnotový zovšeobecnený logický program ako je popísané v 2.2.2

Definícia 3.3.9 *Nech program $\mathcal{P}$ je MDyLP nad $\mathcal{L}'$, $\mathcal{D}$ je graf závislostí, P' je redukt programu $Residue_\ell(\mathcal{P}_\mathcal{D}, I) \cup Defaults(\mathcal{P}_\mathcal{D}, I)$ získaný postupom popísaným v predchádzajúcej definícii a M je model programu P'. Potom hovoríme, že M je* simulovaný parakonzistentný dobre podporený model *pre MDyLP (sp-WS pre MDyLP), ak existuje rozšírená mapovacia funkcia ℓ taká, že pre všetky $A_l, A_h \in M$ existujú pravidlá $r_l, r_h \in P'$ také, že $Head(r_l) = A_l$, $Head(r_h) = A_h$, $\ell(A_l) > \ell(Body(r_l))$, $\ell(A_h) > \ell(Body(r_h))$ a neexistuje taký model N programu P', že $N \subset M$.*

Rovnako ako vieme simulovať množín *Defaults*, *Rejected* a *Residue*, mali by sme vedieť simulovať aj množiny podozrivých literálov a podozrivých pravidiel. S rovnakým prístupom, teda ak pôvodný postup hovoril o pravidle, my hovoríme o dvojici rpavidiel a to isté platí pre literál a dvojicu literálov.

Definícia 3.3.10 *Nech I je interpretácia nad $\mathcal{L}'$, $\mathcal{P}$ je MDyLP nad $\mathcal{L}'$, $\mathcal{D}$ je graf závislostí a ℓ je mapovacia funkcia. Nech $P' = Residue_\ell(\mathcal{P}_\mathcal{D}, I) \cup Defaults(\mathcal{P}_\mathcal{D}, I)$. Budeme postupne budovať dve množiny a to množinu podozrivých literálov S a množinu pravidiel ktorých telá obsahujú doteraz nájdené podozrivé literály R. Najskôr si pripravíme množinu S, keďže určite bude obsahovať všetky nekonzistentné A: $S(P', I') = \{A_l, A_h \mid val_{I'}(A_l) = 1 \wedge val_{I'}(A_h) = 0\}$. Taktiež si pripravíme množinu literálov, kam bude patriť každý literál, pre ktorý nie je v hlave žiadneho pravidla, a teda $Q = \{A \mid / \exists r \in P', Head(r) = A \vee Head(r) = not\, A, A \in \mathcal{L}'\}$. Teraz môžeme naplniť množiny S a R a budeme tak robiť v cykle:*

(1) $R(P', I') = \{r_l, r_h \mid \exists A_l, A_h \in S(P', I'),$
$(A_l \in Body(r_l)) \vee (not\ A_h \in Body(r_l)),$
$(A_h \in Body(r_h)) \vee (not\ A_l \in Body(r_h)),\ r_l, r_h \in P'\}$

(2) $S(P', I') = S(P', I') \cup \{A_l, A_h \mid \nexists r \in P' \setminus R(P', I'),\ Head(r) = A_h \vee Head(r) = not\ A_h \vee Head(r) = A_l \vee Head(r) = not\ A_l,\ A_l, A_h \notin Q,$
$A_l, A_h \in \mathcal{L}'\}$

(3) Ak sa zmenila množina S alebo R, tak pokračujeme bodom (1).

Tento cyklus budeme opakovať kým sa množiny S alebo R menia a preto je dobre spomenúť, že ak je program P' konečný, tak tento proces určite skončí lebo v každom kroku pridáme aspoň jeden literál alebo aspoň jedno pravidlo.

3.4 Vlastnosti

Veta 3.4.1 *Nech $\mathcal{P}$ je DLP taký, že pri každej dvojici konfliktných pravidiel je možné určiť viac prioritné pravidlo (v opačnom prípade by neexistoval DJU model). Nech $\mathcal{D}$ je jeho graf závislostí. Ku každému DJU modelu N programu $\mathcal{P}$ (val_N je príslušná valuácia) existuje p-DJU pre DLP M programu $\mathcal{P}$ s príslušou valuáciou val_M taký, že $\forall A \in N : A \in M$, $\forall A \notin N : not\ A \in M$ a $\{A \mid A \in M\} \cap \{A \mid not\ A \in M\} = \emptyset$.*

Dôkaz 3.4.1.1 *Najskôr porovnajme vygenerované množiny* Defaults, Rejected *a* Residue. *Takže si porovnajme množiny* Defaults*:*

DJU: $Defaults(\mathcal{P}, N) = \{not A \mid A \notin N\}$

p-DJU: $Defaults(\mathcal{P}, M) = \{not\ A \mid A \notin M,\ not\ A \in M\}$

Podmienka $A \notin M \wedge not\ A \in M$ hovorí, že $val_M(A) \leq_t \perp$ a $val_M(A) \leq_k$ nepravda, čo v dvojhodnotovej logike korešponduje s f. Podmienka $A \notin N$ hovorí, že $val_N(A) = f$. Obe tieto hodnoty reprezentujú nepravdu, takže prvky oboch množín sú negatívne nepravdivé atómy a teda tieto množiny sú ekvivalentné.
Takže si porovnajme množiny Rejected*:*

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

p-DJU: $Rejected(\mathcal{P}, M, i) = \{r \mid r \in P_i, \exists r' \in P_j,\ i < j,\ r \bowtie r',\ not\ val_M(Body(r')) <_t val_M(Body(r)),\ P_i, P_j \in \mathcal{P}\}$

V týchto dvoch verziách množín Rejected *je rozdiel v podmienke na telo pravidla ktoré zamieta (resp. vzájomný vzťah tiel oboch pravidiel). Podmienka v* p-DJU, $not\ val_M(Body(r')) <_t val_M(Body(r))$, *vznikla zovšeobecnením podmienky* DJU, $N \models Body(r')$, *pre potreby 4-hodnotovej logiky.*
Podmienka $N \models Body(r')$ *hovorí, že telo pravidla* r' *je podporené modelom, teda* $val_N(Body(r')) = t$.
Podmienka v p-DJU *sa dá zapísať aj* $val_M(Body(r')) >_t not\ val_M(Body(r))$. *Vzhľadom na predpoklady pre program* $\mathcal{P}$, *ak by telo pravidla* r' *(vhľadom na* M*) nebolo pravdivé, tak telo pravidla* r *je nutne nepravdivé, čo znamená že toto pravidlo nemá vplyv na model nezávisle od toho či je zamietnutné alebo nie.*

Keď skúsime porovnať množiny Residue *javí sa jeden problém a to, že* DJU *model nepoužíva priamo množinu s menom* Residue, *avšak vhľadom na to, že* Residue *je program obsahujúci pravidlá pôvodného DLP programu, ktoré nie sú v množine* Rejected *to isté vidíme aj pri DJU modeloch konkrétne* $\rho(\mathcal{P}) \setminus Rejected(\mathcal{P}, M)$.
V prípade p-DJU *máme* $Residue(\mathcal{P}, I) = \bigcup_{1 \le i \le n} [P_i \setminus Rejected(\mathcal{P}, I, i)]$, *kde* n *je počet programov v* $\mathcal{P}$. *Takto definovaná množina obsahuje zjednotenie všetkých pravidiel všetkých programov postupnosti, ktoré neboli zamietnuté, teda nie sú v* Rejected. *Čo je presne to isté ako v prípade* DJU *modelu (okrem pravidiel s nepravdivým telom, ktoré model neovplyvnia).*

Následne k Residue *(a DJU alternatíve) pridáme príslušnú množinu* Defaults, *o čom sme si ukázali, že fungujú rovnako a hľadáme najmenší model. Takže oba postupy vlastne pracujú rovnako a teda* $\forall A \in N : A \in M$, $\forall A \notin N : not\ A \in M$ *a vzhľadom na počiatočnú podmienku na program* $\mathcal{P}$, *platí aj* $\{A \mid A \in M\} \cap \{A \mid not\ A \in M\} = \emptyset$.

Veta 3.4.2 *Nech $\mathcal{P}$ je MDyLP taký, že pri každej dvojici konfliktných pravidiel je možné určiť viac prioritné pravidlo (v opačnom prípade by neexistoval* dobre podporený model pre MDyLP*). Ku každému* dobre podporený model pre MDyLP *N program $\mathcal{P}$ (val_N je príslušná valuácia) existuje* parakonzistentný dobre podporený model pre MDyLP *M programu $\mathcal{P}$ s príslušou valuáciou val_M a mapovacou funkciou ℓ taký, že $\forall A \in N : A \in M$, $\forall A \notin N : not\ A \in M$ a $\{A \mid A \in M\} \cap \{A \mid not\ A \in M\} = \emptyset$.*

Dôkaz 3.4.2.1 *Najskôr porovnajme vygenerované množiny* Rejected *a* Residue. *Takže si porovnajme množiny* Rejected*:*

> dobre podporený model pre MDyLP*: $Rejected_\ell(\mathcal{P}, N, i) = \{r \mid r \in P_i, \exists r' \in P_j : i \leq_\mathcal{D} j, r \bowtie r', N \models Body(r'), \ell(Head(r')) > \ell(Body(r')), P_i, P_j \in \mathcal{P}\}$*
>
> parakonzistentný dobre podporený model pre MDyLP*:*
> *$Rejected_\ell(\mathcal{P}_\mathcal{D}, I, i) = \{r \mid r \in P_i, \exists r' \in P_j,\ i <_\mathcal{D} j,\ r \bowtie r',$*
> *$not\ val_I(Body(r')) <_t val_I(Body(r)),\ \ell(Head(r')) > \ell(Body(r')),\ P_i,$*
> *$P_j \in \mathcal{P}\}$*

V týchto dvoch verziách množín Rejected *je rozdiel iba v podmienke na telo pravidla, ktoré zamieta (resp. vzájomný vzťah tiel oboch pravidiel). Podmienka v* parakonzistentný dobre podporený model pre MDyLP*,*
$not\ val_M(Body(r')) <_t val_M(Body(r))$, vznikla zovšeobecnením podmienky dobre podporený model pre MDyLP*, $N \models Body(r')$, pre potreby 4-hodnotovej logiky.*
Podmienka $N \models Body(r')$ hovorí, že telo pravidla r' je podporené modelom, teda $val_N(Body(r')) = t$.
Podmienku $val_M(Body(r')) >_t not\ val_M(Body(r))$, v reči dvojhodnotovej logiky hovorí, že ak je pravidlo r' podporené modelom tak môže zamietať ak je aj pravidlo r podporené modelom (keďže na strane pravidla r je negácia). Ak pravidlo r nie je podporené modelom, tak ho pravidlo r' nezamietne, ale to nám nevadí, lebo to pravidlo nie je podporené modelom. Vzhľadom na predpoklady pre program $\mathcal{P}$, ak by telo pravidla r' (vhľadom na M) nebolo pravdivé, tak telo pravidla r je nutne nepravdivé, čo znamená že toto pravidlo nemá

vplyv na model nezávisle od toho či je zamietnutné alebo nie.

Následne v prípade parakonzistentný dobre podporený model pre MDyLP *k* Residue *pridáme množinu* Defaults. *Sémantika pre* dobre podporený model pre MDyLP *tento krok nerbí, ale keďže hľadá najmenší dvojhodnotový model, tak nepridá do modelu nič čo nie je podporené pravidlom. Ak je literál podporený pravidlom po spracovaní aktualizácií, tak to znamená, že dané pravidlo nebolo zamietnuté. Keďže sme ukázali, že zamietame rovnaké pravidlá v oboch prípadoch, tak toto pravidlo nebolo zamietnuté ani pri hľadaní* parakonzistentný dobre podporený model pre MDyLP *a teda tento literál nemohol byt v množine* Defaults*, a teda sme pridaním tejto množiny pri výpočte parakonzistetnej verzii modelu nič nepokazili oproti* dobre podporenému modelu pre MDyLP. *Takže oba postupy vlastne pracujú rovnako a teda* $\forall A \in N : A \in M$, $\forall A \notin N : not\, A \in M$ *a vzhľadom na počiatočnú podmienku na program* $\mathcal{P}$, *platí aj* $\{A \mid A \in M\} \cap \{A \mid not\, A \in M\} = \emptyset$.

Veta 3.4.3 *Nech* $\mathcal{P}$ *je MDyLP nad jazykom* $\mathcal{L}$ *a* I *je štvor-hodnotová interpretácia nad jazykom* $\mathcal{L}$ *a val k nej príslušná interpretácia. Nech* ℓ *je mapovacia funkcia a* ℓ' *z nej odvodená rozšírená mapovacia funkia. Nech* $\mathcal{L}'$ *je odvodený jazyk a* I' *je interpretácia nad* $\mathcal{L}'$ *odvodena z* I *a* val' *je valuácia príslušná k* I' *potom:*

(1) (a) $\forall A \in \{not\, A \mid A \notin I,\ not\, A \in I\}$
$\exists A_l, A_h \in \{not\, A_l, not\, A_h \mid val'(A_l) = 0 \wedge val'(A_h) = 0\}$
také, že $val(A) = \langle val'(A_l), val'(A_h)\rangle$

(b) $\forall A_l, A_h \in \{not\, A_l, not\, A_h \mid val'(A_l) = 0 \wedge val'(A_h) = 0\}$
$\exists A \in \{not\, A \mid A \notin I,\ not\, A \in I\}$
taký, že $val(A) = \langle val'(A_l), val'(A_h)\rangle$

(2) (a) $\forall r \in \{r \in P_i \mid \exists r' \in P_j,\ i <_D j,\ r \bowtie r',\ not\, val_I(Body(r')) <_t val_I(Body(r)),\ \ell(Head(r') > \ell(Body(r'),\ P_i, P_j \in \mathcal{P}\}$
$\exists r_l, r_h \in \{r_l, r_h \mid r_l, r_h \in P_i, \exists r'_l, r'_h \in P_j,\ i <_D j,\ (r_l, r_h) \Rightarrow\Leftarrow (r'_l, r'_h),\ \ell'(Head(r'_l)) > \ell'(Body(r'_l)),\ \ell'(Head(r'_h)) > \ell'(Body(r'_h)),\ P_i, P_j \in \mathcal{P}\}$
také, že $val(r) = \langle val'(r_l), val'(r_h)\rangle$

(b) $\forall r_l, r_h \in \{r_l, r_h \mid r_l, r_h \in P_i, \exists r'_l, r'_h \in P_j,\ i <_D j,\ (r_l, r_h) \Rightarrow\Leftarrow (r'_l, r'_h),$
$\ell'(Head(r'_l)) > \ell'(Body(r'_l)),\ \ell'(Head(r'_h)) > \ell'(Body(r'_h)),\ P_i, P_j \in \mathcal{P}\}$
$\exists r \in \{r \in P_i \mid \exists r' \in P_j,\ i <_D j,\ r \bowtie r',\ not\, val_I(Body(r')) <_t val_I(Body(r)),\ \ell(Head(r') > \ell(Body(r'),\ P_i, P_j \in \mathcal{P}\}$
také, že $val(r) = \langle val'(r_l), val'(r_h)\rangle$

Táto veta hovorí o ekvivalencii množín *Defaults* (1) a množín *Rejected* (2) medzi štvorhodnotovou a to isté simulujúcou dvojhodnotovou verziou.

Dôkaz 3.4.3.1 *(1) Odvodená valuácia val' bola odvodená tak, aby A_l a A_h mali presne takú hodnotu ako horná a dolná hranica intervalu, ktorý je priradený ako pravdivostná hodnota pre A. To znamená, že vždy keď je A nepravdivý budú aj oba A_l a A_h nepravdivé a teda pre každý A v jednej množine bude v druhej množina dvojica A_l a A_h a platí to aj opačne.*

(2) Keďže ℓ' priradí pri oboch pravidlách r_l, r_h hlave rovnakú hodnotu a aj telu rovnakú hodnotu a navyše ℓ' je odvodená z ℓ tieto podmienky budú alebo splnené alebo nesplnené súčasne.
Pozrime sa teda na vlastnosť $(r_l, r_h) \Rightarrow\Leftarrow (r'_l, r'_h)$. Prvá časť tejto vlastnosti hovorí:
$val'(Head(r_l)) = not\ val'(Head(r'_h))$ *a* $val'(Head(r_h)) = not\ val'(Head(r'_l))$
čo, keby sme prepísali do intervalov by bolo:
$\langle val'(Head(r_l)), val'(Head(r_h))\rangle = \langle not\ val'(Head(r'_h)), not\ val'(Head(r'_l))\rangle$
a teda:
$\langle val'(Head(r_l)), val'(Head(r_h))\rangle = not\langle val'(Head(r'_l)), val'(Head(r'_h))\rangle$
čo, keďže r_l, r_h sú hranice pre pravidlo r a r'_l, r'_h sú hranice pre pravidlo r', je to isté ako $r \bowtie r'$.
Ešte nám ostáva pozrieť podmienku $not\ val_I(Body(r')) <_t val_I(Body(r))$, ktorú sme už spomínali pri druhej vlastnosti $(r_l, r_h) \Rightarrow\Leftarrow (r'_l, r'_h)$ a teda, že jediné hodnoty pre val pravidiel r a r' sú: $\langle 0,0\rangle$ a $\langle 1,0\rangle$, $\langle 0,0\rangle$ a $\langle 0,1\rangle$, $\langle 0,0\rangle$ a $\langle 1,1\rangle$, $\langle 1,0\rangle$ a $\langle 1,1\rangle$, $\langle 0,1\rangle$ a $\langle 1,1\rangle$. Presne tieto hodnoty hraníc dosiahneme ak r_l, r_h, r'_l, r'_h spĺňajú druhú podmienku vlastnosti $\Rightarrow\Leftarrow$. Takže dostaneme, že za každé pravidlo r v prvej množine, budú v druhej množine pravidlá r_l a r_h a taktiež naopak.

Veta 3.4.4 *Nech $\mathcal{P}$ je MDyLP nad jazykom $\mathcal{L}$ a M je štvor-hodnotová interpretácia nad jazykom $\mathcal{L}$ a val k nej príslušná interpretácia. Nech ℓ je mapovacia funkcia a ℓ' z nej odvodená rozšírená mapovacia funkcia. Nech $\mathcal{L}'$ je odvodený jazyk a M' je interpretácia nad $\mathcal{L}'$ odvodená z M a val$'$ je valuácia príslušná k I' potom k modelu M existuje simulovaný parakonzistentný dobre podporený model N' pre MDyLP taký, že $N' = M'$ a podobne k modelu M' existuje parakonzistentný dobre podporený model N pre MDyLP taký, že $N = M$.*

Dôkaz 3.4.4.1 *Vieme, že generujeme vzájomne podobne množiny* Defaults *teda také, že je možné ich vzájomne odvodiť a to platí aj pre množiny* Rejected. *Takto sme si túto vlastnosť zaručili aj pre množiny* Residue *a teda dostaneme vzájomne odvoditeľné aktualizované programy P a P' (P je program nad $\mathcal{L}$ a P' je program nad $\mathcal{L}'$). Takže nám stačí skúmať vzťah medzi modelmi týchto programov. Pri programe P' pri generovaní reduktu vyrobíme medziprodukt ktorý simuluje výrobu reduktu programu P takže odstránime dvojice literálov a dvojice pravidiel, ktoré tvoria hranice pre literály a pravidlá odstránené pri výrobe reduktu programu P.*
chce to nejako utriasť a dokončiť myšlienku

3.5 Ďalšie pokračovanie

Pri hľadaní modelu vyberáme najmenší model vzhľadom na pravdivostné usporiadanie, pretože sme si ako základ vybrali stabilné modely. Keby sme si však vyberali najmenší model vzhľadom na znalostné usporiadanie, dostali by sme iný typ modelov, ktorý tiež môže mať zaujímave vlastnosti. Toto usporiadanie by pravdivostné hodnoty literálov netlačilo k nepravde ale k $\bot$. Pri pravdivostnom usporiadaní je každý literál nepravda ak nie je podporený pravidlom, ktoré je podporené modelom. Pri znalostnom usporiadaní by sa však situácia zmenila tak, že literál by bol nedefinovaný, teda jeho pravdivostná hodnota by bola $\bot$ 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 dobre postavených modelov, teda či by išlo o rozšírenie týchto modelov, ako aj ďalšie vlastnosti, je však potrebné ďalšie skúmanie.

Kapitola 4

Záver

Podarilo sa nám predstaviť sémantiku pracujúcu s programami a ich aktualizáciami, ktorá predchádza konfliktom zamietaním pravidiel s nižšou prioritou. 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 spôsob 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. Ak ako pravdivostné hodnoty využívame dvojzväz IV, máme spôsob ako hľadanie modelu simulovať za použitia dvojhodnotovej logiky. Hoci sme sa obmedzili na dvojzväz IV, nástroje pre dvojhodnotovú logiku sú rozšírenejšie ako pre štvorhodnotovú logiku.

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] J.A. Leite. *Evolving Knowledge Bases: Specification and Semantics*. Frontiers in artificial intelligence and applications. IOS Press, 2003.

[9] João Alexandre Leite. On some differences between semantics of logic program updates. pages 375–385, 2004.

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

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

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