Universidade Federal do Ceará
Departamento de Computação
Programa de Pós-Graduação em Ciência da Computação

Sistemas Formais Avançados e a Estratificação Relevante
Uma defesa da predicatividade das definições

Autor
Wladimir Araújo Tavares

Orientador
Prof. Dr. Marcelino Cavalcante Pequeno

Dissertação de Mestrado apresentada à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal do Ceará como parte dos requisitos para obtenção do grau de ***Mestre em Ciência da Computação****.*

Fortaleza – Ceará
Agosto 2008

Sumário

Capítulo 1

Introdução

Na matemática e na computação, definições indutivas são largamente utilizadas para a definição de conjuntos. Intuitivamente, podemos dizer que uma definição indutiva é capaz de condensar a definição de um conjunto que não necessariamente são finitos através de um conjunto finito de regras de inferências. As regras de inferências são usadas para a obtenção de novos elementos a partir da presença ou ausência de outros elementos no conjunto definido. Intuitivamente, as regras de inferências podem ser entendidas como um tipo de receita para construção de novos elementos do conjunto pretendido. Tecnicamente, os elementos cuja presença é exigida na regra de inferência são chamados de premissas positivas e os elementos cuja ausência é exigida são chamados de premissas negativas.

Smullyan (SMULLYAN, 1961) foi o pioneiro em tentar capturar a noção de definibilidade indutiva para isso ele propôs um formalismo denominado *sistemas formais elementares*. Nesse formalismo, Smullyan propõe um método de representação para as definições indutivas através de regras e uma "lógica"para extrair o conjunto gerado através das regras de inferências positivas. O conjunto gerado pelas regras de inferências é o menor conjunto obtido pela aplicação das regras. Ele formalizou o conjunto gerado através da noção de derivação.

Moschovakis estudou a definibilidade indutiva em estruturas abstratas (MOSCHOVAKIS, 1974). Neste trabalho, Moschovakis codifica a definição indutiva de um predicado P em uma fórmula de primeira ordem φ. Quando o predicado P tem ocorrência positiva em φ então o predicado é dito indutivo e o conjunto gerado pode ser obtido através do menor ponto fixo de I_φ. Neste trabalho, ele estuda diversas

propriedades da classe dos predicados indutivos.

Aczel(ACZEL, 1977) estudou as definições indutivas positivas com uma óbvia correspondência com os programas lógicos positivos. Aczel propôs diversas maneiras de descrever a semântica de uma definição indutiva positiva através dos *sistemas formais elementares*. Todas elas são equivalentes com a maneira de como a semântica dos programas lógicos positivos é definida em (KOWALSKI; EMDEN, 1976) através do menor ponto fixo de um operador monótono associado à definição indutiva.

Mas nem sempre um SFE é a escolha natural e suficiente para todas as aplicações. Algumas aplicações precisam de regras com premissas positivas e negativas, ou seja, de *sistemas formais avançados* (**SFA**). Os SFA foram introduzidos em (LANGE; GRIESER; JANTKE, 2003) durante o seu estudo sobre o impacto das regras do tipo $A \Leftarrow \textbf{not}\ B_1$, onde A e B_1 são átomos e **not** representando um tipo de negação conceitualmente próxima da negação por falha na definição de linguagens formais. Lange e seus colaboradores perceberam que as definições de linguagens formais podiam ser drasticamente simplificadas com a utilização de regras deste tipo. Recentemente, este tipo de inferência tem encontrado bastantes aplicações em inteligência artificial, devido a sua capacidade de suportar inferências mais especulativas, principalmente em:

- Programação lógica com a negação por falha(CLARK, 1978).
- Lógica Default(REITER, 1980).

As investigações em programação lógica com a negação resultaram em duas principais direções de pesquisas: a abordagem estratificada e abordagem não-monotônica (motivada pela pesquisa em lógica não-monotônicas). A abordagem estratificada resultou em diversos conceitos de estratificação cada vez mais refinados de programas normais[1]. Esta direção culminou na semântica bem-fundada que pode ser aplicada a classe de todos os programas normais (até mesmo para os programas não estratificados). A abordagem não-monotônica propôs diversas maneiras de representar a negação por falha em formalismo não-monotônico (lógica autoepistêmica(MOORE, 1985) e lógica default(REITER, 1980)). Esta abordagem culminou na inauguração de um novo paradigma de programação lógico denominado

[1]Programas lógicos com a negação

Answer Set Programming (**ASP**)(MAREK; TRUSZCZYNSKI, 1999; NIEMELã, 1999) que pode ser aplicada a todos os programas normais.

Em programação lógica com a negação, a abordagem estratificada através da semântica bem-fundada obteve certo consenso entre a comunidade de programação lógica e diversas caracterizações alternativas foram obtidas para esta semântica. Mas utilização do ASP vem ganhando impulso nos últimos anos devido a sua utilização em ambientes mais ricos do que da programação lógica usual.

Temos certas ressalvas a serem feitas em relação as duas abordagens que são aplicadas aos programas não estratificados. Acreditamos que alguns programas são realmente mal formados e não deveriam nem ser considerado. Veremos no capítulo de apresentação das abordagens não estratificadas são divergentes. Mostrando que os programas estratificados são o último ponto seguro para a programação lógica. Na verdade, os programas não estratificados são *impredicativos*.

Vamos propor para os *sistemas formais avançados*(SFA) o conceito de estratificação relevante. Os SFA que admitem a estratificação relevante serão denominados SFA relevantemente estratificados. Mostraremos que o conceito de estratificação relevante coincide com todas as abordagens estratificadas e não estratificadas em programação lógica. A nossa abordagem visa restaurar a predicatividade da definição do conjunto gerado para um SFA. A predicatividade é uma característica das definições que visa evitar o problema do *círculo vicioso*. O *círculo vicioso* acontece quando indiretamente um termo é definido em função dele mesmo, este problema foi extensivamente estudado por Poincaré. Ele apontou que a impredicatividade era a origem de diversos paradoxos matemáticos e as definições predicativas deveriam ser sempre buscadas. Há precedentes para esta busca, nos trabalhos de Russel em teorias dos conjuntos que resultou na formalização das *teorias dos tipos simples*. Entendemos que apenas a simples consideração das circularidades negativas introduzem um *círculo vicioso* na definição do SFA. Logo, tais circularidades devem ser evitadas para não contaminar o conjunto gerado para um SFA.

Na lógica default(REITER, 1980), dois problemas principais têm sido objetos de investigação desde o seu início em 1980 . A falta de extensão (o equivalente ao conjunto gerado), chamado de *problema da coerência*, e o surgimento de extensões indesejadas para algumas teorias, chamado de *problema das extensões anômalas* (HANKS; MCDERMOTT, 1987). Na dissertação sugerimos que os dois problemas

têm origem na *circularidades das teorias*, isto é, em algumas teorias defaults há uma dependência recíproca entre as premissas negativas de regras default. Argumentamos na dissertação que estas circularidades são indesejáveis do ponto de vista técnico, causando os problemas apontados acima, e mais, profundamente, que estas circularidades revela uma má formação das teorias defaults. Nosso conceito de estratificação para teorias defaults (generalizando o conceito de estratificação em programação lógica intensamente estudada no final da década de 80) (APT; BLAIR; WALKER, 1988; PRZYMUSINSKI, 1988; PRZYMUSINSKA; PRZYMUSINSKI, 1988, 1990; BIDOIT; FROIDEVAUX, 1991b; GELDER; ROSS; SCHLIPF, 1991), ao evitar a ocorrência de circularidade, e a proposição da *lógica default estratificada*, ineditamente proposta nesta dissertação, atacam simultaneamente os problemas da coerência e extensões anômalas em teorias defaults.

Há precedentes para esta abordagem que parcialmente propuseram e utilizaram os conceitos que lançamos mão para construir nossa proposta. Embora David Etherington (ETHERINGTON, 1987) tenha proposto o conceito de teorias ordenadas (evitando a circularidade) para resolver o problema da coerência - ele propõe um critério suficiente para garantir a existência de extensões - ele nada propôs para o problema das extensões anômalas. Vale ressaltar que o problema das extensões anômalas se manifesta mesmo em teorias que são ordenadas segundo os critérios de Etherington. O problema aqui não é a falta, mas o excesso de extensões. Por outro lado, Cholewinski (CHOLEWINSKI, 1995a, 1995b, 1996) propõe o que ele chama de *Stratified Default Logic*. Novamente a intenção não foi resolver o problema das extensões anômalas, mas propor métodos eficientes para a computação de extensões. Aliás, a proposta dele de estratificação é bastante precária do ponto de vista do conjunto a ser gerado por uma teoria default uma vez que deixa de fora teorias com conjuntos gerados perfeitamente bem definidos. Podemos dizer que nos beneficiamos diretamente dos trabalhos dos que nos precederam. Utilizamos a estratificação nos moldes propostos pelo Etherington e calculamos a extensão conforme proposto por Marek e Truczszsinski.

Interessante observar que 20 anos após o problema das extensões anômalas ter sido detectado (HANKS; MCDERMOTT, 1987) o problema ainda persista até hoje em lógica default. Parte disto explica-se porque encontrou-se uma solução para o problema em programação lógica através do Answer Set Paradigm, proposto pioneiramente em (GELFOND; LIFSCHITZ, 1988), mas que ganhou um grande impulso

no final da década de 90 (MAREK; TRUSZCZYNSKI, 1999; NIEMELã, 1999), e que hoje ocupa a cena da área de programação lógica(LIFSCHITZ, 2002; COSTANTINI; WATSON, 2007; FABER; LEE, 2008).

Argumentamos na dissertação que esta solução não é satisfatória, o problema só é resolvido graças a limitação da linguagem da programação lógica (apenas cláusulas de Horn, estendidas com a negação por falha), e mostramos como resolver o problema em teorias que utilizam a linguagem da lógica de primeira ordem sem restrições. A solução é a Lógica Default Predicativa que propomos na dissertação, que foi desenvolvida em parceria com o orientador da dissertação, Marcelino Pequeno.

No capítulo 2, vamos apresentar algumas definições importantes sobre ordens , boa ordens e operadores e seus pontos fixos.

No capítulo 3, vamos apresentar os diversos estudos sobre as definições indutivas positivas que culminou nos sistemas formais elementares e em diversas caracterizações do conjunto gerado por eles.

No capítulo 4, vamos apresentar os SFA's, definir Os critérios para a boa formação de SFA e apresentar uma metodologia de construção para o conjunto gerado para um SFA.

No capítulo 5, vamos apresentar várias propostas de estratificação em programação lógica com a negação e mostraremos que o conjunto gerado por nossa abordagem em SFA quando aplicada em programas normais coincide com o conjunto gerado em todas as abordagens estratificadas.

No capítulo 6, vamos apresentar a *Lógica Default Predicativa* que foi obtida através da generalização do conceito de estratificação do SFA's. Esta lógica default ataca simultaneamente os dois principais problemas da lógica default: problema da coerência e problema das extensões anômalas. Faremos uma breve comparação com as abordagens anteriores que atacaram os problemas da coerência e o problema das extensões anômalas.

Capítulo 2

Preliminares

Neste capítulo, nós vamos rever a definição de vários conceitos importantes que serão utilizados ao longo do trabalho. Na primeira seção, veremos as principais definições relacionadas com os diversos tipos de relações de ordens. As relações de ordens serão importantes para definir a estratificação. Na seção 2, veremos as definições relacionadas com as boas ordens e os números ordinais. A boa ordem será utilizada para definir o conjunto gerado para SFA e a extensão da lógica default predicativa . Na seção 3, veremos as definições dos operadores e seus pontos fixos. Apresentaremos alguns resultados importantes sobre os operadores monótonos.

2.1 Relações e Ordens

Nesta seção, vamos ver a definição de alguns conceitos de relações de ordem. A notação utilizada é a padrão. Os símbolos $\cup$, $\cap$ denotam as operações de união e intersecção de conjuntos. O símbolo $\setminus$ denota a operação de complemento de conjunto. O conjunto das partes de X, isto é, o conjunto de todos os subconjuntos, será denotado por 2^X

Definição 2.1.1. *(Relação) Uma relação binária R em A é um subconjunto de A $\times$ A. Se o par (a,b) $\in$ R podemos denotar por a R b.*

Definição 2.1.2. *(Tipos de relações) Seja R uma relação binária em X.*

i. *A relação R em X é reflexiva se (x,x) $\in$ R , para todo x $\in$ X.*

ii. *A relação R em X é irreflexiva se (x,x) $\notin$ R, para todo x $\in$ X.*

iii. *A relação R em X é anti-simétrica se para todo x,y* $\in X$ *, (x,y)* $\in R$ *e (y,x)* $\in$ *R implica que x=y.*

iv. *A relação R em X é transitiva se para todo x,y,z* $\in X$ *, (x,y)* $\in R$ *e (y,z)* $\in R$ *implica que (x,z)* $\in R$*.*

v. *A relação R em X é total se para todo x,y* $\in X$*, (x,y)* $\in R$ *ou (y,x)* $\in R$*.*

Definição 2.1.3. *(Relações de ordem e conjuntos ordenados)*

i. *Seja* $\preceq$ *uma relação binária em X é uma relação de ordem parcial se* $\preceq$ *é reflexiva,transitiva e anti-simétrica.*

ii. *Seja* $\prec$ *uma relação binária em X é uma relação de ordem parcial estrita se* $\prec$ *é irreflexiva,transitiva e anti-simétrica.*

iii. *Um relação de ordem parcial* $\preceq$ *em X é uma ordem total se* $\preceq$ *é total em X.*

iv. *Um relação de ordem parcial estrita* $\prec$ *em X é uma ordem total estrita se* $\prec$ *é total.*

v. *Seja X um conjunto e* $\preceq$ *uma relação binária em X. O par* $<X,\preceq>$ *é denominado conjunto parcialmente ordenado se* $\preceq$ *é uma relação de ordem parcial.*

vi. *Seja X um conjunto e* $\preceq$ *uma relação binária em X. O par* $<X,\preceq>$ *é denominado conjunto totalmente ordenado se* $\preceq$ *é uma relação de ordem total.*

vii. *Seja* $<X,\preceq>$ *um conjunto parcialmente ordenado. O conjunto* $Y \subseteq X$ *é uma cadeia se* $\preceq$ *é total em Y.*

Definição 2.1.4. *(Elementos de um conjunto ordenado)*

i. *Seja* $<X,\preceq>$ *um conjunto parcialmente ordenado. Seja* $Y \subseteq X$ *e* $x \in X$ *então x é dito um limite inferior(superior) de Y se para todo* $y \in Y$*,* $x \preceq y (y \preceq x)$*.*

ii. *Seja* $<X,\preceq>$ *um conjunto parcialmente ordenado. Seja* $Y \subseteq X$ *e x é limite superior(inferior) de Y, então x é dito ínfimo(supremo) de Y se para todo limite superior(inferior) x' de Y,* $x \preceq x' (x' \prec x)$*.*

iii. *Seja $<X,\preceq>$ um conjunto parcialmente ordenado. Seja $C \subseteq X$ e $x \in C$, x é denominado elemento $\preceq$-minimal(maximal) de C se para todo $y \in C$, $y \preceq x$($x \preceq y$) então $x=y$.*

iv. *Seja $<X,\preceq>$ um conjunto parcialmente ordenado. Seja $C \subseteq X$ e $x \in C$ é x é denominado $\preceq$-menor(maior) elemento de C se para todo $y \in C$, $x \prec y$ ($y \prec x$).*

v. *Uma relação de ordem parcial estrita $\preceq$ em X é dita bem-fundada em X se para todo subconjunto Y de X tiver um elemento $\preceq$-minimal*

O ínfimo de X é único, se ele existe, e denotaremos por inf(X). Semelhantemente, o supremo de X é único, se ele existe, e denotaremos por sup(X).

Definição 2.1.5. *Um conjunto parcialmente ordenado $<X,\prec>$ é denominado reticulado completo se para todo $Y \subseteq X$, existe inf(Y) e sup(Y).*

Definição 2.1.6. *Seja $<X,\prec>$ um reticulado completo e $X \subseteq L$. X é direcionado se para todo subconjunto finito de X tem um limite superior em X.*

Seja $<$X,$\preceq>$ um reticulado completo. sup(X) é representado por $\top$ e denominado elemento topo. inf(X) é representado por $\perp$ e denominado elemento base.

Exemplo 2.1.1. *$<2^S,\subseteq>$ é um reticulado completo. O ínfimo de $X \subseteq 2^S$ é dado por $\cap X$ e supremo de X é dado por $\cup X$. O elemento topo é S e o elemento base $\emptyset$.*

2.2 Boa Ordem e números ordinais

Nesta seção, vamos rever a definição de boa ordem e números ordinais.

Definição 2.2.1. *Seja $\preceq$ um relação binária em conjunto X. Dizemos que $\preceq$ é uma boa ordem se:*

i. *$\preceq$ é uma ordem total.*

ii. *todo subconjunto não vazio Y de X possui um menor elemento.*

Se $\preceq$ é uma boa ordem em um conjunto X então o par $<X,\preceq>$ é denominado um conjunto bem ordenado.

Agora vamos definir uma classe particular de conjuntos bem ordenados denominado números ordinais. Existe uma associação entre um conjunto bem ordenado e um ordinal da seguinte maneira: todo conjunto bem ordenado é isomórfico a um número ordinal. A abordagem que vamos utilizar leva em consideração os ordinais que são definidos através da relação de inclusão entre seus elementos. Freqüentemente, os ordinais definidos dessa maneira são denominados ordinais de Von Neumann.

Definição 2.2.2. *(Números Ordinais)*

i. *Um conjunto x é chamado transitivo se $y \in x$, então $y \subseteq x$.*

ii. *Um conjunto x é chamado um número ordinal(de Von Neumann), ou apenas ordinal, se x é transitivo e para todo y,z $\in$ x, $y \subseteq z$ ou $z \subseteq y$.*

O conjunto $\emptyset$ é um ordinal. Ele corresponde ao número ordinal 0. O conjunto $\{\emptyset\}$ é um ordinal. Ele corresponde ao número ordinal 1. O conjunto consistindo de todos os ordinais finitos, ou seja, $\{\emptyset, \{\ \emptyset\ \}, \{\ \emptyset, \{\emptyset\}\}, \ldots\}$ é um ordinal. Ele é o menor ordinal infinito e é denotado por ω.

Teorema 2.2.1. *(Propriedades básicas dos ordinais)*

i. *Todo ordinal é um conjunto bem ordenado com a relação de inclusão.*

ii. *Todo ordinal α, $\alpha = \{\beta : \beta \subset \alpha$ e β é um ordinal$\}$.*

iii. *Todo conjunto bem ordenado $<X,\preceq>$, existe um único ordinal α tal que $<X,\preceq>$ é isomórfico a $<\alpha,\subseteq>$. Ainda mais, tal isomorfismo é único.*

Nós vamos usar $\alpha \leq \beta$ representando $\alpha \subseteq \beta$, e $\alpha < \beta$ representando $\alpha \subseteq \beta$. Para quaisquer dois ordinais α e β, $\alpha < \beta$ ou $\beta < \alpha$.

Definição 2.2.3. *Se α é número ordinal, o ordinal sucessor de α é o ordinal $\alpha + 1 = \alpha \cup \{\alpha\}$, que é o menor ordinal maior que α. Um ordinal é dito um ordinal limite se ele não é o sucessor de nenhum ordinal. O menor ordinal limite infinito é ω. 0 é um ordinal limite finito. O ordinal sucessor de ω é $\omega + 1 = \omega \cup \{\omega\}$. O próximo sucessor ordinal limite $\omega.2 = \omega + \omega$.*

Para todo ordinal n $\in \omega$, nós temos $\omega.n$. Tomando a união de todos $\omega.n$ (o supremo de qualquer conjunto de ordinais), nós temos o ordinal $\omega.\omega = \omega^2$.

Teorema 2.2.2. *(Princípio da Indução Transfinita) Seja α um ordinal e seja $Z \subseteq \alpha$ um subconjunto de α possuindo estas duas propriedades:*

i. *$0 \in Z$.*

ii. *Para todo $\beta < \alpha$, se $\{\gamma : \gamma < \beta\} \subseteq Z$ então $\beta \in Z$.*

Então $Z = \alpha$.

Todo ordinal é um ordinal sucessor ou um ordinal limite, a indução transfinita geralmente é aplicada de uma maneira diferente(mas equivalente) levando em conta este fato.

Teorema 2.2.3. *(Princípio da Indução Transfinita) Seja α um ordinal e seja $Z \subseteq \alpha$ um subconjunto de α possuindo estas três propriedades:*

i. *$0 \in Z$.*

ii. *Para todo $\beta < \alpha$, tal que $\beta + 1 < \alpha$, se $\beta \in Z$ então $\beta + 1 \in Z$.*

iii. *Para todo $\lambda < \alpha$ ordinal limite, $\{\xi : \xi < \lambda\} \subseteq Z$ então $\lambda \in Z$.*

Então $Z = \alpha$.

Os teoremas 2.2.2 e 2.2.3 são aplicados da seguinte maneira. Assuma que uma propriedade P vale para algum ordinal α. Nós vamos denotar P(β) se um ordinal $\beta < \alpha$ tem a propriedade P. Defina Z = $\{\beta : \beta < \alpha\}$. Aplicando o príncipio da indução transfinita ao conjunto Z temos um método para provar que para todo ordinal menor ou igual a α tem a propriedade P.

Teorema 2.2.4. *Seja P uma propriedade qualquer, e seja $\alpha > 0$ um número ordinal. Assuma que:*

i. *P(0)*

ii. *Para todo $\beta < \alpha$, $\beta + 1 < \alpha$ e P(β), então P($\beta + 1$).*

iii. *Para todo $\lambda < \alpha$ ordinal limite, e para todo $\xi < \lambda$ nós temos P(ξ), então nós temos P(λ).*

Então, para todo $\beta < \alpha$, P(β) vale.

2.3 Operadores e Ponto Fixo

Nesta seção vamos discutir alguns elementos da teoria de operadores em X, ou seja, um mapeamento de 2^X em 2^X.

Definição 2.3.1. *(Tipos de Operadores)*

i. *Um operador em X é um mapeamento* $F : 2^X \rightarrow 2^X$.

ii. *Um operador F é chamado monótono se para todo* $X_1, X_2 \subseteq X$,$X_1 \subseteq X_2$ *implica* $F(X_1) \subseteq F(X_2)$.

iii. *Um operador F é chamado anti-monótono se para todo* $X_1, X_2 \subseteq X$,$X_1 \subseteq X_2$ *implica* $F(X_2) \subseteq F(X_1)$.

iv. *Um operador F é compacto se para todo* $X_1 \subseteq X$,

$$F(X_1) = \bigcup\{F(X_2) : X_2 \subseteq X_1, X_2 \text{ é finito } \}$$

Definição 2.3.2. *(Ponto Fixo) Seja F um operador em X então um conjunto* $Y \subseteq X$ *é denominada um ponto fixo de F se F(Y)=Y e um pré-ponto fixo se F(Y)* $\subseteq$ *Y.*

Teorema 2.3.1. *(Teorema Knaster-Tarski)(TARSKI, 1955) Seja* $F : 2^X \leftarrow 2^X$ *um operador monótono. O operador F possui o menor ponto fixo denotado por* ***lfp****(F). O menor ponto fixo é dado por:*

$$\textbf{\textit{lfp}}(F) = \bigcap\{Y : F(Y) \subseteq Y\}.$$

Demonstração. Seja G = $\{Y|F(Y) \subseteq Y\}$ e g=$\bigcap\{Y : F(Y) \subseteq Y\}$. Para mostrar que g $\in$ G. Temos que g $\subseteq$ x, para todo x $\in$ G, então pela monotonicidade de F, temos que F(g) $\subseteq$ F(x), para todo x$\in$ G. Assim F(g) $\subseteq$ x, para todo x $\in$ G, e então F(g) $\subseteq$ g. Portanto g $\in$ G.

Para mostrar que g é um ponto fixo de F. Temos que mostrar g $\subseteq$ F(g). Temos que $F(g) \subseteq g$ implica em $F(F(g)) \subseteq F(g)$ que implica em $F(g) \in G$. Portanto, g $\subseteq$ F(g), então g é um ponto fixo de F.

Seja G' = {Y | F(Y)=Y } e g'= $\bigcup$G'. Como g é ponto fixo, logo g' $\subseteq$ g. Por outro lado, G' $\subseteq$ G e então g $\subseteq$ g'. Assim nós temos que g=g' e portanto **lfp**(F) = $\bigcap\{Y : F(Y) \subseteq Y\}$. □

Teorema 2.3.2. *(Teorema Knaster-Tarski)(TARSKI, 1955) Seja $F : 2^X \leftarrow 2^X$ um operador monótono. A seqüência F^α, para um ordinal α:*

$$F^0 = \emptyset$$
$$F^{\alpha+1} = F(F^\alpha).$$
$$F^\lambda = \cup\{F^\alpha : \alpha < \lambda\}, \text{ para um ordinal limite } \lambda$$

atinge um ponto fixo em algum ordinal. Seja cl(F) o menor ordinal tal que $F^\alpha = F^{\alpha+1}$ denominado ordinal de fechamento. Então

$$F^{cl(F)} = \boldsymbol{lfp}(F)$$

A prova de $\boldsymbol{lfp}(F)$ segue-se de (a) e (e). A prova de (a),(b) e (c) usa indução transfinita.

i. *Para todo $\alpha + 1$, $F^{\alpha+1} \subseteq \boldsymbol{lfp}(F)$:*
Se α é um ordinal sucessor, então pela hipótese de indução temos que $F^\alpha \subseteq \boldsymbol{lfp}(F)$, e pela monotonicidade de F temos que $F(F^\alpha) \subseteq F(\boldsymbol{lfp}(F))$e a propriedade do ponto fixo temos que $F(\boldsymbol{lfp}(F)) = \boldsymbol{lfp}(F)$. Logo,

$$F^{\alpha+1} = F(F^\alpha) \subseteq F(\boldsymbol{lfp}(F)) = \boldsymbol{lfp}(F).$$

Se $\alpha + 1$ é ordinal limite, então $F^\alpha = \bigcup\{F^\beta : \beta < \alpha + 1\} \subseteq \boldsymbol{lfp}(F)$, pela hipótese de indução.

ii. *Para todo α, $F^\alpha \subseteq F^{\alpha+1}$:*
Se α é um ordinal sucessor, então pela hipótese de indução temos que $F^{\alpha-1} \subseteq F^\alpha$, e pela monotonicidade de F temos que $F(F^{\alpha-1}) \subseteq F(F^\alpha)$. Logo,

$$F^\alpha = F(F^{\alpha-1}) \subseteq F^\alpha = F(F^\alpha) = F^{\alpha+1}.$$

Se $\alpha + 1$ é ordinal limite, pela hipótese de indução temos que $\bigcup\{F^\beta : \beta < \alpha\} \subseteq \bigcup\{F^\beta : \beta < \alpha + 1\}$. Logo,

$$F^\alpha = \bigcup\{F^\beta : \beta < \alpha\} \subseteq \bigcup\{F^\beta : \beta < \alpha + 1\} = F^{\alpha+1}.$$

iii. *Para todo α, β, $\alpha < \beta$ implica que $F^\alpha \subseteq F^\beta$:*
Se β é um ordinal sucessor, então $\alpha \leq \beta - 1$ e então pela hipótese de indução temos que $F^\alpha \subseteq F^{\beta-1} \subseteq F^\beta$. Se $\alpha+1$ é ordinal limite, pela hipótese de indução temos que
$F^\alpha \subseteq \bigcup\{F^\gamma : \gamma < \beta\} = F^\beta$.

iv. *Para todo α, β, $\alpha < \beta$ e $F^\alpha = F^\beta$, então $F^\alpha = F^\beta$:*
Temos que $F^\alpha \subseteq F^{\alpha+1} \subseteq F^\beta$, por (c). Portanto $F^\alpha = F^{\alpha+1} = F(F^\alpha)$ e então F^α é um ponto fixo. Assim, $F^\alpha \subseteq \mathbf{lfp}(F)$, por (a).

v. *Existe um β tal que $\beta \leq \gamma$ implica que $F^\gamma = \mathbf{lfp}(F)$:*
Seja α o menor ordinal de cardinalidade maior que a cardinalidade de X. Suponha que $F^\delta \neq \mathbf{lfp}(F)$, para todo $\delta < \alpha$. Defina $h{:}\alpha \to L$ por $h(\delta){=}F^\delta$. Então, por (d), h é injetiva, o que contradiz a escolha de α. Assim $F^\beta = \mathbf{lfp}(F)$, para algum $\beta < \alpha$, e o resultado segue a partir (a) e (c).

Capítulo 3

Sistemas Formais Elementares

3.1 Definições Indutivas

Na matemática e na computação, definições indutivas são largamente utilizadas para a definição de conjuntos através da recursão. Intuitivamente, podemos dizer que uma definição indutiva é capaz de condensar a definição de um conjunto através de regras de inferências. As regras de inferências são usadas para a obtenção de novos elementos a partir da presença ou ausência de outros elementos no conjunto definido. Intuitivamente, as regras de inferências podem ser entendidas como um tipo de receita para construção de novos elementos do conjunto pretendido. Muitos trabalhos foram desenvolvidos para obter uma representação natural da definição indutiva positiva. Primeiramente, veremos os trabalhos realizados por Smullyan(SMULLYAN, 1961) na proposição dos sistemas formais elementares. Seguido do trabalho de Moschovakis(MOSCHOVAKIS, 1974) sobre a representação da indução elementar sobre estruturas abstratas. Finalizando com o trabalho realizado por Aczel(ACZEL, 1977), uma extensão dos sistemas formais elementares. Encerrando com um breve estudo sobre as definições indutivas não monótonas.

3.1.1 Sistemas Formais Elementares - Smullyan

Na época de Smullyan, a noção de "‘formal"’ aplicada a sistemas na matemática, e a noção de "‘mecânico"’ aplicada às operações (procedimentos) precisavam ser mais bem compreendidas. Intuitivamente, a noção de operação mecânica era definida como uma operação realizada por algum modelo de computação(máquina de computação). Nesse ponto, a noção de sistema formal e operação mecânica

se interconectavam. Se a noção de operação mecânica fosse definida em função de algum modelo de computação (por exemplo, máquina de Turing), a noção de sistema "'formal"' seria definida como o conjunto de teoremas gerados pelo modelo de computação em questão. Alternativamente, Post define diretamente um sistema formal, para capturar a noção de formal, e a definição de operação mecânica passa a ser obtida através do sistema de Post, ou seja, uma operação mecânica é uma operação realizada pelo sistema de Post. Seguindo essa abordagem Smullyan define os sistemas formais elementares para servir de base para o estudo de sistemas "'formais"' na matemática.

Os sistemas formais elementares (SFE) foram introduzidos em (SMULLYAN, 1961) para explicar a noção de definibilidade indutiva. Freqüentemente, na matemática um conjunto W é definido através de regras:

- As regras que definem os elementos iniciais de um conjunto W denominadas axiomas.
- As regras da forma "'se tais e tais elementos estão em W então tal e tal combinação estão em W"' também chamadas de regras indutivas.
- Uma regra final (também chamada de "'regra de fechamento"') "'nada mais está em W, a não ser o que pode ser obtido através das regras"'.

A regra final precisava ser mais bem definida. Os sistemas formais elementares proviam uma "'lógica"' do que é ser realmente obtido através das regras.

Considere o simples exemplo da definição um conjunto de string $S \subseteq \Sigma = \{a, b\}^*$, que alternam a's e b's, ou seja, as strings de S não tem nenhuma ocorrência de dois a's e b's.

Exemplo 3.1.1. *Seja* $\Sigma=\{a, b\}^*$. *As regras que definem S REGRA 1.* $a \in S$
REGRA 2. $b \in S$
REGRA 3. $ab \in S$
REGRA 4. $ba \in S$
REGRA 5. Se $xa \in S$ $(x \in U)$ *, então* $xab \in S$
REGRA 6. Se $xb \in S$ $(x \in U)$ *, então* $xba \in S$
REGRA 7. Nenhum outro elemento está em S, a não ser que ele possa ser obtido através das regras 1-6.

Podemos converter o sistema de regras informais para um sistema formal da seguinte maneira: Primeiramente, vamos abreviar x $\in$ S para S(x). Vamos abreviar a regra Se *condição1* então *condição2* para *condição2* $\leftarrow$ *condição1*.

Vamos substituir a regra 7 por uma definição precisa do que é ser obtido através das regras. Um elemento X é dito ser obtido pelas regras se e somente se pode ser obtido através de um número finito de aplicações das seguintes regras:
R_1. Substituir a variável X por qualquer string de U.
R_2. Se S(X_1) é derivado então S(X_2) é derivado pela regra S(X_2) $\leftarrow$ S(X_1).

O sistema formal obtido para a definição de S é:

(1) S(a)

(2). S(b)

(3). S(ab)

(4). S(ba)

(5). S(xab) $\leftarrow$ S(xa), onde x $\in \Sigma$

(6). S(xba) $\leftarrow$ S(xb), onde x $\in \Sigma$

O conjunto gerado pelas regras pode ser obtido como o menor conjunto fechado sobre as regras ou como o conjunto obtido por todos os elementos que são derivados pela aplicação das regras R_1 e R_2.

Basicamente, o SFE proposto por Smullyan é um sistema de regras positivas onde os termos (as premissas) são padrões em $(\Sigma \cup X)^*$, onde Σ é um alfabeto e X é um símbolo para representar o conjunto definido.

Apesar da estrutura simples do SFE proposto por Smullyan, a semântica proposta por ele pode ser estendida e aplicada em sistemas baseados em regras positivas. Na verdade, um SFE pode ser entendido como um sistema de representação das inferências indutivas positivas.

3.1.2 Definições Indutivas Positivas

A representação e a caracterização das definições indutivas positivas em estruturas abstratas foi objeto de estudo em (MOSCHOVAKIS, 1974). Ele mostrou que a fórmula φ associada a uma definição indutiva positiva de um predicado P tem um comportamento monótono e poderia ser utilizada para definir uma seqüência de conjuntos definidos indutivamente e a união destes conjuntos descrevia a definição

do predicado P. Isso representou um avanço na teoria das definições uma vez que algumas relações que são geradas pelas definições indutivas positivas não podem ser definidas na linguagem de primeira ordem.

Considere o seguinte exemplo. Suponha que nós temos um banco de dados de uma empresa aérea com a relação binária R (representando as rotas) tal que se um par (A,B) está em R indica que existe um vôo entre A e B. Agora vamos supor que desejamos definir todos os pares A,B tal que existe um vôo direto entre eles. Podemos representar este predicado pela seguinte fórmula em primeira ordem com duas variáveis livres.

$$t_0(x, y) \equiv R(x, y)$$

Podemos definir também um predicado que contenha o par (x,y) representando que existe um vôo entre x e y com exatamente uma escala.

$$t_1(x, y) \equiv \exists z R(x, z) \wedge R(z, y)$$

O predicado representando no máximo uma escala pode ser representando pela disjunção dos predicados anteriores

$$T_1(x, y) \equiv t_0(x, y) \vee t_1(x, y)$$

Para um k fixo podemos escrever um predicado que represente um vôo com no máximo k escalas:

$$t_k(x, y) \equiv \exists z_1, \ldots, \exists z_k R(x, z_1) \wedge R(z_1, z_2) \wedge \ldots \wedge R(z_k, y),$$

da mesma maneira podemos definir $T_k = \bigvee_{j<k} t_j$ para representar um vôo com no máximo k escalas.

Mas para computar o fecho transitivo de R será necessário representar todos os vôos com todos o número de escalas possíveis.

$$\bigvee_{k \in N} t_k$$

Mas claro que essa fórmula não é uma fórmula de primeira ordem devido ao uso de disjunções infinitas. O argumento anterior não é uma prova formal da não expressibilidade em primeira ordem, mas com ele podemos ter uma idéia da limitação da linguagem de primeira ordem.

Podemos formalizar o banco de dados aéreo como um grafo G=(V,E). O conjunto de vértices V são as cidades do banco de dados e o conjunto de arestas E são as listas de vôos entre as cidades.

O fecho transitivo de um grafo não pode ser representado na linguagem de primeira ordem, mas podemos definir como o fecho transitivo como menor predicado $T_G \subseteq V^2$ que satisfaz as seguintes propriedades:

- (x,y) $\in T_G$ se (x,y) $\in$ E.
- (x,y) $\in T_G$ se para algum vértice z,(x,z) $\in$ E e (z,y) $\in T_G$.

podemos codificou as seguintes regras em uma fórmula de primeira ordem da seguinte maneira:

$$\varphi((x,y),T_G) \equiv E(x,y) \wedge \exists z(E(x,z) \vee T_G(z,y))$$

podemos definir o fecho transitivo explicitamente através da seguinte fórmula:

$$\vec{x} \in T_G \leftrightarrow \forall S\{\forall \vec{y}[\varphi(\vec{y},S) \Rightarrow \vec{y} \in S] \Rightarrow \vec{x} \in S\}$$

O único problema dessa definição explícita é que ela recorre a linguagem de segunda ordem. O ideal seria uma abordagem que se torna claro a existência da indução na definição de T_G e não recorre-se a linguagem de segunda ordem. O predicado T_G pode ser obtido por uma seqüência de predicados indutivamente gerado pela formula φ. Os predicados indutivamente gerados I^n_φ serão definidos da seguinte maneira:

$$\vec{x} \in I^n_\varphi \leftrightarrow \varphi(\vec{x}, \bigcup_{j<n} I^j_\varphi)$$

e através deles definir predicado T_G da seguinte maneira:

$$T_G = \bigcup_n I^n_\varphi$$

Moschovakis(MOSCHOVAKIS, 1974) mostrou que todas as definições indutivas positivas poderiam ser caracterizadas da maneira descrita acima. Ele mostrou que os predicados definidos por indução positiva têm ocorrência positiva na fórmula de primeira ordem que codifica as regras da definição indutiva $\varphi(\vec{x}, P)$. As fórmula positivas têm um comportamento monótono que garante que processo descrito acima possa ser utilizado.

Uma definição indutiva de um predicado P é dita positiva quando a fórmula de primeira ordem associada a definição $\varphi(\vec{x}, P)$ é positiva, ou seja, nenhuma ocorrência de P deve aparecer no escopo de um número ímpar de ocorrência do símbolo de negação $\neg$. Por exemplo, o predicado T_G ocorre positivamente em $\varphi((x, y), T_G)$:

$$\varphi((x, y), T_G) \equiv E(x, y) \wedge \exists z(E(x, z) \vee T_G(z, y))$$

Definição 3.1.1. *(Propriedade Monótona das Fórmulas Positivas) Seja S uma variável de segunda ordem que ocorre positivamente em $\varphi(S)$. Seja P,P' são relações sobre A, então*

$$Se\ P \subseteq P'\ e\ \varphi(P)\ então\ \varphi(P').$$

Podemos associar a cada fórmula positiva $\varphi(S)$ um operador Γ_φ, que é definido da seguinte maneira:

$$\Gamma_\varphi(S) = \{\overline{x} | \varphi(\overline{x}, S)\}.$$

Para cada ordinal α, podemos definir o conjunto I_φ^α por recursão transfinita da seguinte maneira:

$$I_\varphi^\alpha = \Gamma_\varphi(\bigcup_{\beta<\alpha} I_\varphi^\beta)$$

e assim definir o conjunto gerado por Γ denotado por I_Γ por :

$$I_\Gamma = \bigcup_\beta I_\varphi^\beta$$

vamos adotar as seguintes notações:

$$I_\varphi^\xi = I_\Gamma^\xi;\ I_\varphi^{<\xi} = \bigcup_{\eta<\xi} I_\varphi^\eta\ ;\ I_\varphi = I_\Gamma$$

Pela propriedade monótona das fórmulas positivas φ temos que o operador Γ_φ é monótono. O Teorema de Knaster-Tarski garante que Γ_φ tem o menor ponto fixo. O menor ponto fixo Γ_φ equivale a intersecção de todos os conjunto fechados com relação a Γ_φ(isto é, $\Gamma_\varphi(S) \subseteq S$),

$$I_\Gamma = \cap\{S|\Gamma(S) \subseteq S\} = \mathbf{lfp}(\Gamma)$$

A intersecção de todos os conjuntos fechados com relação a Γ_φ equivale ao menor conjunto fechado que é a caracterização de uma definição indutiva.

O Teorema de Knaster-Tarski, garante também que $\mathbf{lfp}(\Gamma_\varphi) = \Gamma_\varphi^\beta(\emptyset)$, para o menor ordinal β(ordinal de fechamento). Os conjunto Γ_φ^α para $\alpha < \beta$ são chamados de camadas. O ordinal β é chamado de ordinal de fechamento.

A seguir, vamos considerar a aplicação desta abordagem na definição de fecho transitivo de P relativo a Q.

Exemplo 3.1.2. *Seja $P \subseteq A, Q \subseteq AxA$ dados. Considere o fecho transitivo de P com relação a Q,*

$$R(x) \Leftrightarrow \textit{existe uma seqüência } y_1, y_2, \ldots, y_n \textit{ tal que } P(y_1) \textit{ e}$$
$$Q(y_1, y_2), Q(y_2, y_3), \ldots, Q(y_{n-1}, y_n) \textit{ e } x = y_1 \textit{ ou } x = y_n.$$

codificado pela seguinte fórmula:

$$\varphi(x, S) \equiv P(x) \wedge (\exists y)[S(y) \wedge Q(y, x)]$$

É fácil verificar que $R = I_\varphi$ por indução em ξ que

$$x \in I^{\xi_\varphi} \Rightarrow R(x)$$

e então por indução em $n \geq 1$ que

$$P(y_1) \wedge Q(y_1, y_2) \wedge \ldots \wedge Q(y_{n-1}, y_n) \Rightarrow y_n \in I_\varphi^{<\omega}.$$

Assim R é indutivo em P, Q. Perceba que está indução não tem parâmetro, e que ela termina em ω passos.

A seguir, vamos considerar uma definição indutiva que não termina em ω passos.

Exemplo 3.1.3. *Seja $\leq$ um ordem total em A e queremos tomar um segmento inicial bem ordenado de $\leq$, representado por*

$$W(x) \Leftrightarrow \text{não existe seqüencia infinita } x > x_1 > x_2 > x_3 > \ldots$$

codificado pela seguinte fórmula:

$$\psi(x, S) \equiv (\forall u)[u < x \Rightarrow u \in S]$$

facilmente podemos mostrar que $W = I_\varphi$ por indução em ξ que

$$x \in I_\psi^\xi \Rightarrow W(x)$$

se $x \in I_\psi^\xi$, então pela definição $(\forall u)[u < x \Rightarrow u \in I_\psi^{<\xi}]$, então pela hipótese de indução $(\forall u)[u < x \Rightarrow u \in W(u)]$, que imediatamente que W(x). Por outro lado,

$$x \notin I_\psi \Rightarrow (\exists u_1)[u_1 < x \vee u_1 \notin I_\psi]$$

e usando a mesma implicação em u_1 nós podemos encontrar $u_2 < u_1$ tal que $u_2 \notin I_\psi$, então podemos construir uma infinita seqüência descendente começando em x e $\neg W(x)$. Assim,

$$W(x) \Leftrightarrow x \in I_\psi$$

Não é difícil verificar que o ordinal de fechamento dessa indução é precisamente o ordinal do maior segmento inicial bem ordenado de $\leq$.

O teorema de Knaster-Tarski garante a nós que quando estamos tratando com uma definição indutiva positiva então uma definição altamente não construtiva como a do menor ponto fixo(definida como a intersecção de uma grande, possivelmente não enumerável, família de conjuntos de fechados) pode ser convertida em uma definição construtiva(iterar sobre um operador até o ponto fixo ser alcançado; quando o universo for enumerável, o ponto fixo será alcançado em um ordinal enumerável). Conseqüentemente, o estudo das definições indutivas positivas mostrou-se intimamente relacionado com o estudo de operadores e seus pontos fixos.

No estudo de Moschovakis, um predicado P definido por uma definição indutiva positiva φ pode ser obtido das seguintes maneiras:

- o menor conjunto que sastifaz φ(equivale a definição de P em segunda ordem).
- através da intersecção dos predicados que são fechados em relação ao operador Γ_φ.
- através do menor ponto fixo do operador Γ_φ.
- através da aplicação sucessiva do operador Γ_φ a partir do $\emptyset$.

3.1.3 Sistemas Formais Elementares - Aczel

Aczel(ACZEL, 1977) propôs a representação das definições indutivas positivas através *sistemas formais elementares*(SFE). Aczel consegue diversas maneiras de descrever a semântica de uma definição indutiva positiva através dos SFE's. A seguir, veremos diversas caracterizações do conjunto indutivamente gerado $I(\Phi)$ para um Φ(SFE).

Definição 3.1.2. ***i.*** *Uma regra é um par (X,x), onde X é um conjunto chamado de premissas e x é uma conclusão. A regra pode ser escrita como $x \Leftarrow X$. A conclusão de uma regra r será denotada por c(r) e o conjunto de premissas será denotado por p(r). A regra tem a seguinte leitura "'A partir da geração de todas as expressões de X podemos gerar a expressão x"'.*

ii. *Se Φ é um conjunto de regras, então um conjunto A é fechado se todas as regras em Φ cujas premissas estão em A então suas conclusões também estão em A. Em outras palavras, A é fechado se para toda regra $x \Leftarrow X$ e $X \subseteq A$ então $x \in A$.*

iii. *Se Φ é um conjunto de regras, então I(ϕ), o conjunto indutivamente gerado por Φ:*

$$I(\Phi) = \bigcap\{A : A \text{ é fechado}\}$$

Preposição 3.1.1. *$I(\Phi)$ é o menor conjunto fechado.*

Demonstração. Seja $x \Leftarrow X \in \Phi$ e $X \subseteq I(\Phi)$, temos que mostrar que $x \in I(\Phi)$. Se $X \subseteq I(\Phi)$, então para todo conjunto fechado A, $X \subseteq A$. Logo, para todo A fechado, x ∈ A. Assim, $x \in I(\Phi)$. □

Os conjuntos indutivamente gerados podem ser obtidos através da noção de "'derivação"'.

Definição 3.1.3. *(Derivação) Uma seqüência $a_1, \ldots, a_n$ é uma derivação de a em Φ. Se:*

i. $a_n = a$

ii. *para todo $1 \leq i \leq n$, existe uma regra $a_i \leftarrow X$ em Φ, tal que $X \subset \{a_1, \ldots, a_{i-1}\}$.*

Definição 3.1.4. *Para um Φ finitário, temos que*

$$Cn^{\Phi} = \{b : \text{ se existe uma derivação de } b \text{ em } \Phi\}$$

Preposição 3.1.2. *Para um Φ finitário,*

$$Cn^{\Phi} = I(\Phi)$$

Demonstração. Para mostrar que $Cn^{\Phi} \subseteq I(\Phi)$. Considere um $a_n \in Cn^{D}$. Seja $a_1, \ldots, a_n$ uma derivação de a_n em Φ. Por hipótese de indução, para todo i, $1 \leq$ i $\leq$ n-1, $a_i \in I(\phi)$. Seja a_n obtido por uma regra $a_n \Leftarrow X$, onde X $\subseteq \{a_1, \ldots, a_{n-1}\}$. Logo, $a_n \in I(\Phi)$ ($I(\Phi)$ é fechado). Para mostrar $I(\Phi) \subseteq Cn^{\Phi}$. Basta mostrar que Cn^{Φ} é fechado. Temos que mostrar que se $a \Leftarrow x_1, \ldots, x_n \in \Phi$ e $x_1, \ldots, x_n \in Cn^{\Phi}$ então a $\in Cn^{D}$. Se $x_1, \ldots, x_n \in Cn^{\phi}$ implica que cada x_i tem uma derivação. A concatenação da derivação de cada x_i concatenado com a é uma derivação de a em Φ. Logo, a $\in Cn^{\Phi}$. □

A seguir, vamos considerar um exemplo de uma definição indutivamente definida em Φ.

Exemplo 3.1.4. *Seja $>$ um relação binária em um conjunto A. Um segmento inicial bem-fundado de $\prec$ é o conjunto W($\prec$) de $a \in A$ tal que não existe uma seqüencia infinita descendente de $a_o > a_1 > \ldots$. A relação é bem-fundada se A=W($<$). W($<$) pode ser indutivamente definido da seguinte maneira:*

Seja Φ_v o conjunto de regras $a \Leftarrow (< a)$, para todo $a \in A$, onde ($<$a) = $\{x \in A : x < a\}$.

Para mostrar que W($<$) = $I(\Phi)$. Temos que mostrar $I(\Phi_v) \subseteq W(<)$. Basta mostrar que W($<$) é fechado. Por indução, assuma que para todo $a_0 < a$, $(< a_0) \in I(\Phi_v)$ então $(< a_0) \in W(<)$. se $(< a) \in I(\Phi_v)$ então temos a regra $a \Leftarrow (< a_0)$ em Φ_v e $(< a_0) \in I(\Phi_v)$. Por hipótese de indução, $(< a_0) \in W(<)$. Portanto ($<$a) $\in$ W($<$).

Por outro lado, para mostrar $(W(<)) \subseteq I(\Phi_v)$. *Se* $a \notin I(\Phi_v)$. *Temos que encontrar uma seqüência* $a > a_0 > a_1 > \ldots$ *para mostrar que* $a \in A$. *Se* $a \in I(\Phi_v)$ *então* $(<a) \subseteq I(\Phi_v)$. *Portanto,* $a_0 < a$ *tal que* $a_0 \notin I(\Phi_v)$. *Repetindo podemos encontrar que* $a_1 < a_0$ *tal que* $a_1 \notin I(\Phi_v)$.

Aczel mostrou que as definições indutivas também poderiam ser associados com operadores da seguinte maneira:

Definição 3.1.5. ***i.*** *Seja* Φ *um conjunto de regras em* A *(isto é,* $X \cup \{x\} \subseteq A$ *para toda regra* $x \Leftarrow X \in \Phi$*) podemos definir um operador monótono* $\Gamma : 2^A \to 2^A$ *da seguinte maneira:*

$$\Gamma(S) = \{x | x \Leftarrow X \in \phi \text{ e } X \subseteq S\}$$

ii. *Para o operador monótono* Γ, *se* $X \subseteq A$ *é* Φ*-fechado se* $\Gamma(X) \subseteq X$.

iii. *Para* Γ *o operador monótono. O conjunto indutivamente gerado por* Γ *é:*

$$I(\Gamma) = \bigcap\{X \subseteq A : \Gamma(X) \subseteq X\}.$$

Então $Y \subseteq A$ *é fechado se* $\Gamma(Y) \subseteq Y$. *Logo, I(*Φ*)=I(*Γ*).*

Para um operador monótono Γ, existe uma caracterização de $\mathrm{I}(\Phi)$ usando iterações transfinita Γ^λ de Γ para todos os ordinais λ da seguinte maneira:

$$\Gamma^\lambda = \bigcup_{\mu<\lambda} \Gamma^\mu \cup \Gamma(\bigcup_{\mu<\lambda} \Gamma^\mu)$$

podemos definir $\Gamma^\infty = \bigcup_\lambda \Gamma^\lambda$, onde λ é um ordinal qualquer.

Se denotarmos $\bigcup_{\mu<\lambda} \Gamma^\mu$ por Γ^λ então

$$\Gamma^\lambda = \Gamma^\lambda \cup \Gamma(\Gamma^{<\lambda})$$

O conjunto Γ^λ podem ser definidos diretamente por transfinita recursão por

$$\Gamma^{<\lambda} = \bigcup_{\mu<\lambda} \Gamma(\Gamma^\mu)$$

ou alternativamente por

$$\Gamma^{<0} = \emptyset$$

$$\Gamma^{<\lambda+1} = \Gamma^{\lambda} \cup \Gamma(\Gamma^{<\lambda})$$

$$\Gamma^{\lambda} = \bigcup_{\mu<\lambda} \Gamma^{<\mu}, \text{ para um ordinal limite } \lambda$$

A seqüência de conjuntos Γ^{λ} forma uma cadeia crescente e atinge um ponto fixo em algum ordinal de fechamento $\mathrm{cl}(\Gamma)$ (isto é, $\mathrm{cl}(\Gamma)$ é o menor ordinal α tal que $\Gamma^{\alpha} = \Gamma^{\alpha+1}$). O ponto fixo alcançado por essa seqüência de conjuntos é denominado ponto fixo indutivo de Γ. O ponto fixo indutivo de Γ é o menor ponto fixo de Γ.

Preposição 3.1.3. *Para um operador monótono* $\Gamma : 2^A \to 2^A$*:*

i. *I(*Γ*)=*Γ^{∞}

ii. *I(*Γ*) é o menor ponto fixo de* Γ*.*

Note que a definição de Γ^{∞} não exige que o operador Γ seja monótono. Portanto a mesma construção pode ser estendida para definições indutivas não monótona. Vamos analisar separadamente algumas abordagens para estes tipos de definições.

Aczel caracterizou alternativamente o conjunto $I(\Phi)$ através de árvores bem-fundadas. Essa caracterização pode ser aplicadas a conjunto de regras arbitrário Φ.

Definição 3.1.6. *Uma árvore bem-fundada* T *é um conjunto de finitas seqüências de tamanho* > 0 *tal que:*

i. *Existe exatamente uma seqüência de tamanho um em* T*. Ela é chamada de raiz (*a_T*) da árvore.*

ii. *Se* $(a_1, \ldots, a_{n+1}) \in T$ *então* $(a_1, \ldots, a_n) \in T$*.*

iii. T *é bem-fundada no sentido que não existe uma seqüência infinita* $a_1, a_2, \ldots$ *tal que* $(a_1, \ldots, a_n) \in T$ *para todos* $n > 0$*. Alternativamente, a relação* $<_T$ *é bem-fundada, onde*

$$(a_1, \ldots, a_m) <_T (b_1, \ldots, b_n) \text{ sse } n{=}m{+}1$$

e $a_i = b_i$ *para i=1...m. O tamanho da árvore* $|T|$ *será a cardinalidade do maior segmento inicial de* $<_T$ *iniciado em* a_T*.*

Definição 3.1.7. *Se* Φ *é um conjunto de regras, uma árvore* T *é árvore de derivação de* a. *Se*

i. $a{=}a_T$

ii. *para toda seqüência* $(a_1, \ldots, a_n) \in T$, *se* $a_n \leftarrow T_{(a_1,\ldots,a_n)} \in \Phi$, *onde*

$$T_{(a_1,\ldots,a_n)} = \{a : (a_1, \ldots, a_n, a) \in T\}.$$

Preposição 3.1.4. ***i.*** *I(*Φ*)={ a : a tem uma árvore de derivação}.*

ii. *Se* Φ *é um conjunto de regras em* A *com um operador monótono correspondente* $\Gamma : 2^A \to 2^A$, *então para todos os ordinais* λ,

$$\Gamma^\lambda = \{a \in A : \textit{tem árvore de derivação de tamanho} < \lambda\}.$$

O conjunto indutivamente gerado I(Φ) para um conjunto arbitrário de regras Φ pode ser caracterizado através de teoria dos jogos da seguinte maneira:

Primeiramente, vamos definir as regras de um jogo G(Φ,a) entre dois jogadores I e II que fazem movimentos alternadamente quando possível. O jogo começa por II escolhendo $a_0 = a$. Se depois de n pares de movimentos o jogador II escolher a_n então o jogador I deve responder escolhendo o conjunto X_n tal que $a_n \Leftarrow X_n \in \Phi$ e então o jogador II deve responder escolhendo $a_{n+1} \in X_n$. Se o jogador não conseguir mover-se então ele perde. Se o jogo continua indefinidamente, o jogador I perde.

Preposição 3.1.5. $a \in I(\Phi)$ *sse o jogador I tem uma estratégia vencedora para o jogo G(*Φ*,a).*

Seja W o conjunto dos elementos para os quais o jogador I possui uma estratégia vencedora. Seja $a \Leftarrow X \in \Phi$ com X $\subseteq$ W. Para cada x $\in$ X, existe uma estratégia vencedora σ_x em G(Φ,x). Logo, podemos definir uma estratégia vencedora para o jogador I em G(Φ,a). O jogador I começa escolhendo X e o jogador II escolhe x $\in$ X então o jogador I segue a estratégia vencedora σ_x. Portanto a $\in$ W. Assim, W é fechado. Por indução podemos mostrar que $I(\Phi) \subseteq W$.

Por outro lado, a $\in$ W. Seja σ uma estratégia vencedora para o jogador I em G(Φ,a). Seja T o conjunto de possíveis finitas seqüências de movimentos do jogador II quando o jogador I segue σ. Observe que T é um árvore de derivação para a. Logo, a $\in$ I(Φ).

3.1.4 Conjunto Gerado por uma Boa Ordem

Marek e Truszczynski(MAREK; TRUSZCZYNSKI, 1997) estudaram o efeito do acréscimo de regras de inferências monótonas na lógica proposicional. Ele obteve uma caracterização para o conjunto gerado para o seu sistema que combinava regras de inferências monótonas e lógica proposicional. Podemos aproveitar a caracterização obtida por eles para caracterizar $I(\Phi)$ somente desconsiderando as regras de inferência que envolvem lógica clássica.

Primeiramente, podemos observar que não é necessário todo o conjunto de regras Φ para obter Cn^{Φ}, precisamos apenas de $\Phi' \subseteq \Phi$ tal que $Cn^{\Phi} = \{x : x \leftarrow X \in \Phi'\}$. Podemos obter o conjunto de regras Φ' como o limite de uma seqüencia de uma construção do conjunto das regras que são aplicáveis. Uma boa ordem $\prec$ em Φ é usada para determinar a próxima regra a ser aplicada. Quando mais nenhuma regra puder ser aplicada, a construção termina. O resultado obtido pela construção é o conjunto Φ'.

Definição 3.1.8. *Seja Φ um conjunto de regras. Seja $\preceq$ uma boa ordem em Φ.*

Vamos construir iterativamete o conjunto de regras gerado pela boa ordem $\preceq$, $GR_{\preceq}$, da seguinte maneira:

i. *se $\alpha = 0$ então $GR_\alpha = \emptyset$*

ii. *caso contrário d_α é a $\preceq$-menor regra aplicável no conjunto $\Phi \backslash (\bigcup_{\xi<\alpha} GR_\xi)$, ou seja,*
$p(d_\alpha) \in c(\bigcup_{\xi<\alpha} GR_\xi)$ e
Então $GR_\alpha = (\bigcup_{\xi<\alpha} GR_\xi) \cup \{d_\alpha\}$

iii. *caso contrário, se não existe regra $d \in \Phi \backslash (\bigcup_{\xi<\alpha} GR_\xi)$ tal que*
$p(d) \in c(\bigcup_{\xi<\alpha} GR_\xi)$
então a construção pára,$GR_{\eta_\preceq} = (\bigcup_{\xi<\eta} GR_\xi)$

Teorema 3.1.1. *Seja um conjunto de regras Φ. Para toda boa ordem $\preceq$ em Φ nós temos:*

i. *O conjunto $GR_{\eta_\preceq}$ consiste precisamente de todas as regras $x \Leftarrow X \in \Phi$ tal que $X \subseteq Cn^{\Phi}$.*

ii. *$Cn^{\Phi} = c(GR_{\eta_\preceq})$.*

Demonstração. Por hipótese de indução, temos que para todo ordinal $\xi \leq \alpha$, para todo regra $d \in GR_\alpha$ temos que $p(d) \in Cn^\Phi$ e $c(d) \in Cn^\Phi$.

Seja $\alpha+1$ é sucessor ordinal. Seja $d_{\alpha+1}$ a menor regra aplicável $\in \Phi \backslash GR_\alpha$. Logo, $p(d_{\alpha+1}) \in GR_\alpha$. Pela hipótese de indução, p$(d_{\alpha+1}) \in Cn^\Phi$. Assim, c$(d_{\alpha+1}) \in Cn^\Phi$

Para mostrar que $Cn^\Phi \subseteq c(GR_{\eta_\preceq})$. Seja $a_1, \ldots, a_n$ uma derivação de $a_n \in Cn^\Phi$. Por hipótese de indução, $\{a_1, \ldots, a_{n-1}\} \subseteq GR_{\eta_\preceq}$.

Logo, existe uma regra $a_n \Leftarrow A$, onde A $\subseteq \{a_1, \ldots, a_{n-1}\} \subseteq GR_{\eta_\preceq}$. Logo, para certo ordinal α, a regra $a_n \Leftarrow A$ é $\preceq$-menor regra aplicável em $\Phi \backslash \bigcup_{\xi<\alpha} GR_\xi$, ou seja, $\{a_1, \ldots, a_{n-1}\} \subseteq \bigcup_{\xi<\alpha} GR_\xi$. Logo, $a_n \Leftarrow A \in GR_{\eta_\preceq}$. Assim, $a_n \in c(GR_{\eta_\preceq})$. □

Podemos definir um algoritmo para computar o conjunto gerado por SFE Φ da seguinte maneira:

Conjunto_Gerado_SFE(Φ, $\preceq$)
Entrada: Um SFE Φ e uma boa ordem $\preceq$;
Saída: O conjunto gerado pela boa ordem $\preceq$.
A := $\emptyset$
R := $\emptyset$

faça
r:= a $\preceq$-menor regra r do conjunto $\Phi \backslash R$ aplicável em A
R := R $\cup$ {r}
A := A $\cup$ {c(r)}
enquanto(r $\neq \emptyset$)

retorne A;

Exemplo 3.1.5. *Seja* Φ*:*

$$\begin{cases} (1) \quad p \Leftarrow q \\ (2) \quad q \Leftarrow r \\ (3) \quad r \Leftarrow \emptyset \\ (4) \quad a \Leftarrow b \\ (5) \quad b \Leftarrow a \end{cases}$$

Note que neste SFE Φ*, nós temos uma recursão positiva (a depende de b e b depende de a). Mas isso não vai gerar nenhum problema.*

Seja uma boa ordem $\preceq$ qualquer das regras. Por exemplo,

$(p \Leftarrow q) \preceq (r \Leftarrow \emptyset) \preceq (q \Leftarrow r) \preceq (a \Leftarrow b) \preceq (b \Leftarrow a)$

$GR_0 = \{ r \Leftarrow \emptyset \}$

$GR_1 = GR_0 \cup \{ q \Leftarrow r \}$

$GR_2 = GR_1 \cup \{ p \Leftarrow q \}$

Note que qualquer boa ordem pode ser usada.

Um fato interessante é que a construção de $GR_{\eta_\preceq}$ não depende de uma boa ordem particular, e que o resultado da construção será sempre Cn^Φ. Toda boa ordem $\preceq$ gera o mesmo conjunto $GR_{\eta_\preceq}$, mas não necessariamente o ordinal η é o mesmo.

Sumarizando os resultados, o conjunto indutivamente gerado I(Φ) por um conjunto de regras Φ finitário representando um sistema formal elementar (SFE) pode ser obtido de várias maneiras alternativas como:

i. o menor conjunto fechado.

ii. o conjunto formado pelos elementos que possuem uma derivação em Φ.

iii. o menor ponto fixo de um operador Γ associado a Φ.

iv. o ponto fixo indutivo de Γ.

v. o conjunto formado pelos elementos que possuem uma árvore de derivação em Φ.

vi. o conjunto formado pelos elementos para os quais o jogador I possui uma estratégia vencedora.

vii. o limite de uma construção obtida por uma boa ordem $\prec$ em Φ.

Interessante notar que nenhuma das caracterizações anteriores não podem ser transportadas para as definições indutivas não monótonas com resultados satisfatórios. A única caracterização que pode ser utilizada para induções indutivas não monótonas é a caracterização através de ponto fixo indutivo de Γ, mas não obtém resultados expressivos. A maioria das caracterizações estão apoiadas na monotonicidade de Φ e no Teorema de Knaster-Tarski. A abordagem proposta por Marek e Truszczynski é uma caracterização que pode potencialmente ser exportada

para um conjunto de regras não monótonas. A seguir, faremos um breve estudo sobre as definições indutivas não monótonas.

3.2 Definições Indutivas não monótonas

Na matemática, podemos encontrar alguns conjuntos que são construídos a partir de regras de inferências a partir da ausência de outros elementos no conjunto (portanto modelando um operador não-monotônico). A seguir, analisaremos alguns estudos sobre as definições indutivas não monótonas em estruturas abstratas.

3.2.1 Definições Indutivas Inflacionárias

Moschovakis(MOSCHOVAKIS, 1974) realizou também um estudo sobre as definições indutivas não monótonas. Ele considerou quando a fórmula de primeira ordem φ associada a um predicado P não eram positivas e o operador Γ_φ associada não monótono. O operador Γ_φ poderia ter nenhum ou múltiplos pontos fixos minimais. Para superar este problema, Moschovakis definiu uma seqüência de iterações transfinitas S^λ obtidas através de Γ da seguinte maneira:

$$S^0 = \emptyset$$
$$S^{\alpha+1} = S^\alpha \cup \Gamma_\varphi(S^\alpha)$$
$$S^\lambda = \cup_{\alpha<\lambda} S^\alpha \text{ para um ordinal limite } \lambda$$

A seqüência S^λ forma uma cadeia crescente e atinge um ponto fixo denominado ponto fixo inflacionário definido por φ. Note que no ponto fixo $S_\varphi = S_\varphi \cup \Gamma(S_\varphi)$, e assim $\Gamma(S_\varphi) \subseteq S_\varphi$. Em outras palavras, S_φ é um pré-ponto fixo de Γ_φ. Em geral, S_φ não é ponto fixo de Γ_φ e nem mesmo é um pré-ponto fixo minimal.

Exemplo 3.2.1. *Considere o predicado P(x) definido pela seguinte fórmula* $(x = a \wedge \neg P(b)) \vee (\wedge x = b \wedge \neg P(a))$. *Podemos codificar o predicado P da seguinte maneira:*

$$\varphi(x, P) \equiv (x = a \wedge \neg P(b)) \vee (x = b \wedge \neg P(a))$$

O ponto fixo inflacionário será:

$$S^0 = \emptyset$$
$$S^1 = S^0 \cup \Gamma(S^0) = \emptyset \cup \Gamma(\emptyset) = \{a, b\}$$
$$S^2 = S^1$$

O ponto fixo inflacionário do predicado P é {a,b}. Ele não é ponto fixo e é maior que os dois pontos fixos {a} e {b} desta fórmula

Exemplo 3.2.2. *Vamos considerar agora a seguinte definição dos números pares E através da seguinte fórmula não monótona:*

$$\varphi(x, E) \equiv (x = 0) \vee (\exists y(x = s(y) \wedge \neg E(y)))$$

O ponto fixo inflacionário de $E = \mathbb{N}$

Uma característica positiva do ponto fixo inflacionário é a sua simplicidade. Uma característica negativa é que o conjunto indutivamente definido pelo ponto fixo inflacionário, embora único, possui propriedades matemáticas fracas.

3.2.2 Definições Indutivas Interativas

É bastante comum encontrarmos definições de conjuntos indutivas sobre conjuntos bem-fundados. Neste tipo de definições, a presença de um elemento é definido em função da presença (ausência) de elementos menores no conjunto definido. Assim, checar se um elemento a pertence ao conjunto será necessário checar alguma propriedade de um predecessor a. Podemos construir o conjunto, aplicando a definição do elemento minimal e depois iterando para os próximos níveis.

Para ilustrar este princípio. Considere a definição dos números pares por indução sobre os naturais:

- n=0 é par.
- se n não é par então n+1 é par, caso contrário n+1 não é par.

Podemos considerar esta definição dos pares como uma definição iterativa se separarmos a definição em uma seqüência de definições não indutivas (livre de recursões) compatíveis com a ordem sobre os números naturais. A seguinte lista descreve a separação da definição dos números pares em pequenas definições livre de recursões:

$$
\begin{array}{llll}
(0) & \text{par}(0) & := & \mathbf{t} \\
(1) & \text{par}(1) & := & \neg\ \text{par}(0) \\
(2) & \text{par}(2) & := & \neg\ \text{par}(1) \\
 & & \ldots & \\
\text{(n+1)} & \text{par(n+1)} & := & \neg\ \text{par(n)}
\end{array}
$$

O estudo lógico da indução iterativa foi iniciado em (KREISEL, 1963) e estendido em estudos posteriores da chamada Definição Indutiva Iterativa(DII) em (FEFERMAN, 1970),(BUCHHOLZ; FEFERMAN; SIEG, 1981) e (DENECKER, 1998). As definições indutivas Iterativas é usada para definir conjuntos de números naturais através da indução interativa.

Para representar a definição indutiva iterativa de um conjunto de números naturais, cada número está associado a um índice de nível. O índice de nível pode ser entendido como índice da subdefinição que determina se um número pertence ou não no conjunto definido. Intuitivamente, a codificação terá a informação de como um elemento pode ser definido e em qual instante ele pode ser definido. Embora a intuição da definição iterativa seja simples, a maneira como a idéia é implementada na abordagem DII não é diretamente percebida. Uma definição indutiva não pode ser codificada através de uma simples fórmula. A codificação torna-se complexa porque o índice de nível de cada elemento definido devem ser explicitamente codificado. A definição indutiva iterativa é descrita por uma finita fórmula parametrizada $\varphi(n, x, P, H)$, onde n representa o índice do nível, x é um número natural, P é predicado unário variando sobre os números naturais com somente ocorrências positivas em φ, e H é relação definida representada como uma relação binária variando sobre as tuplas (x,n) de um número natural x e seu índice de nível n. A fórmula $\varphi(x, n, P, H)$ codifica que n é o índice de nível de x, e x pode ser derivado(usando a definição indutiva com índice de nível n) a partir do conjunto P e a única meta variável H(α,p) representando que o nível de p é α e p é definido verdadeiro. O conjunto H é caracterizado por dois axiomas que expressam que a cada nível α, o conjunto $\{p|H(\alpha, p)$ é verdade$\}$ satisfaz o princípio da definição indutiva positiva. O primeiro axioma expressa que H é fechado sobre φ:

$$\forall n \forall x(\varphi(P(\sigma)/H(n, \sigma)) \rightarrow H(n, x))$$

Acima, $\varphi(P(\sigma)/H(n,\sigma)$ (onde σ é um termo arbitrário) denota a fórmula obtida a partir φ pela substituição de cada expressão P(σ) por H(n,σ).

O segundo axioma é um axioma de segunda ordem expressando que para cada n, o subconjunto {x | (x,n) ∈ H} de $\mathbb{N}$ é o menor conjunto de número naturais fechado sobre φ:

$$\forall n \forall P[\forall x(\varphi \rightarrow P(x) \rightarrow \forall x(H(n,x) \rightarrow P(x))].$$

Para ilustrar como a codificação é tediosa em DII. Vamos considerar a definição dos números pares como uma definição indutiva interativa, ou seja, associamos a definição de cada número natural n um índice de nível n da seguinte maneira:

$$\begin{array}{ll} (0) & \text{par}(0) \Leftarrow \mathbf{t} \\ (1) & \text{par}(1) \Leftarrow \neg\ \text{par}(0) \\ \vdots & \\ (\text{n+1}) & \text{par(n+1)} \Leftarrow \neg\ \text{par(n)} \end{array}$$

A codificação da definição iterativa é disjunção infinita:

$$\begin{cases} \{N{=}0 \wedge X{=}even(0)\} \vee \\ \bigvee\{N = n+1 \wedge X = even(n+1) \wedge \neg H(n, even(n)) | n \in \mathbb{N}\} \end{cases}$$

Aplicando os axiomas, podemos reduzir a disjunção infinita para uma fórmula finita usando a quantificação nos naturais

$$\begin{cases} \{N{=}0 \wedge X{=}even(0)\} \vee \\ \bigvee\{N = n+1 \wedge X = even(n+1) \wedge \neg H(n, even(n)) | n \in \mathbb{N}\} \\ \Rightarrow \exists M[N = s(M) \wedge X = s(M) \wedge \neg H(M,M)] \end{cases}$$

O resultado finito da codificação em DII é:

$(n = 0 \wedge x = 0) \vee \exists y(n = s(y) \wedge x = s(y) \wedge \neg H(y,y))$.

A fórmula expressa que (x,n) pode ser derivado se x e seu índice de nível n são iguais e se x = 0 ou se o predecessor de x não é par.

Existe uma correspondência entre o formalismo de DII e a semântica dos modelos perfeitos para programas localmente estratificados. A estratificação da definição

indutiva e a aplicação do formalismo de DII é o tratamento correto para definições indutivas iterativas.

A nossa abordagem para as regras com premissas negativas, ou seja, nas regras que dependem da ausência de outros elementos será baseada também em estratificação denominada estratificação relevante. O conjunto gerado será obtido iterativamente utilizando o método da boa ordem. No próximo capítulo, veremos a nossa abordagem para *sistemas formais avançados*.

Capítulo 4

Sistemas Formais Avançados

Basicamente, um sistema formal avançado(SFA) é um sistema formal com regras de inferência com premissas positivas e negativas. As regras com premissas negativa são inferências a partir do que não é gerado(da ausência de informação). A inferência com premissas negativas é uma ferramenta poderosa e bastante útil para diversas aplicações. Recentemente, este tipo de inferência tem encontrado bastantes aplicações em inteligência artificial, especialmente em:

i. Lógicas não-monotônicas.

ii. Programação lógica com negação por falha.

4.1 Sistemas Formais Avançados - SFA

Os *sistemas formais avançados*(**SFA**) foram introduzidos por (LANGE; GRIESER; JANTKE, 2003) durante o seu estudo sobre o impacto das regras do tipo $A \Leftarrow$ **not** B_1, onde A e B_1 são átomos e **not** representa um tipo de negação conceitualmente próxima da negação por falha na definição de linguagens formais. Mesmo com a utilização restrita da negação na abordagem deles, as definições de linguagens formais tornavam-se drasticamente simplificadas. A nossa abordagem vai propor uma utilização menos restrita da negação que será representada através das premissas negativas.

Definição 4.1.1. *(Sistema Formal Avançado)*

i. *Um regra r é uma tripla (x,X,Y) em A (se $x \in A$ e $\{X\} \cup \{Y\} \subseteq A$), onde x é a conclusão da regra denotado por c(r). X é conjunto das premissas positivas denotado por $p^+(r)$. Y é conjunto das premissas negativas denotado por $p^-(r)$. A regra pode ser escrita como $x \Leftarrow X;Y$. Podemos omitir "';"' quando o conjunto $p^-(r)$ for vazio. A regra de um SFA tem a seguinte leitura "'A partir da geração de todas as expressões de X e da não geração de nenhuma expressão de Y podemos gerar a expressão x"'.*

ii. *Um sistema formal avançado Φ é um conjunto de regras (x,X,Y).*

Em muitas aplicações, a utilização da inferência a partir da ausência de informação é a escolha natural. A seguir, vamos mostrar alguns exemplos de utilização dos SFA para definir certos conjuntos.

Exemplo 4.1.1. *Considere a definição do conjuntos dos números pares através do SFA Φ:*

(0)

(s(X))$\Leftarrow \emptyset$; (X)

Nesta definição, um número natural pertence é par se o seu predecessor não é e 0 é par.

Exemplo 4.1.2. *Considere a definição do predicado primo*[1] *SFA Φ:*

soma(0,X,X)

soma(s(X),Y,s(Z)) $\leftarrow$ soma(X,Y,Z)

composto(s(s(0)) , s(s(0)) , s(s(s(s(0)))))

composto(s(X) , Y , Z) $\leftarrow$ composto(X,Y,W), soma(W,Y,Z)

composto(X , s(Y) , Z) $\leftarrow$ composto(X,Y,W), soma(W,X,Z)

composto(s(0) , s(0) , s(0))

primo(X) $\leftarrow \emptyset$; composto(_ , _ ,X).

Nesta definição um número natural X pertence ao predicado primo se o predicado composto(_,_,X) não é gerado. A idéia dessa definição é um número é primo se ele não é composto.

[1]Note que tomamos a liberdade de utilizar uma linguagem mais rica pra definir os primos

Smullyan (SMULLYAN, 1961) fez o estudo sobre sistemas formais elementares para mostrar como os sistemas formais poderiam ser mais uma opção para se definir conjuntos computáveis. Entretanto, a adição das premissas negativas em um sistema formal traz profundas conseqüências do ponto de vista formal e computacional. É trivial se notar três fatos:

i. Um conjunto é gerado por um SFE sse é recursivamente enumerável.

ii. O conjunto complemento de um SFE pode ser gerado por um SFA.

iii. Existem conjuntos que são gerados por SFE, mas seus complementos não.

Portanto, existem conjuntos que são gerados por SFA, mas que não são recursivamente enumeráveis.

Desta forma, SFA's formam uma entidade matemática bem mais complexa do que os SFE's, enquanto podemos dizer que os SFE são *locais* e *cumulativos*, os SFA's são *globais* e *não cumulativos*. Uma regra em um SFE depende no máximo de um número finito de outras regras para sua aplicabilidade (as regras que derivam suas premissas); e uma vez uma regra é aplicada, a aplicação subseqüente de outras regras não afeta as regras já aplicadas. Já a aplicabilidade das regras nos SFA's pode depender de todo o SFA, pois depende do que é gerado e também do que não é gerado. E aplicação de uma regra em um estágio parcial pode ser anulada posteriormente se as premissas negativas forem derivadas posteriormente ou mesmo se as premissas positivas não mais são derivadas, como dissemos, o processo não é cumulativo.

Isto traz uma ordem de complexidade e dificuldade muito grande para que se defina o conjunto gerado por um SFA.

Observe que as definições para SFE através de conjuntos fechados, pontos fixos, noção de prova, etc. não generalizam para SFA's. E como o operador de conseqüência imediata não é *monotônico*, o teorema de Knaster-Tarski não se aplica para SFA's.

Para um SFA Φ , o operador de conseqüência imediata associado Γ é dado por:

$$\Gamma(S) = \{x | x \Leftarrow X; Y \in \phi \text{ e } X \subseteq S \text{ e } Y \nsubseteq S\}$$

Dizemos que um conjunto Y pertence a um conjunto S, se para algum y $\in$ Y, y $\in$ S.

Assim, um conjunto S será fechado em relação a um SFA, se $\Gamma(S) \subseteq$ S.

Os exemplos abaixo mostram como os conceitos de conjunto fechado minimal, ponto fixo, ponto fixo minimal, nada disso funciona como definição de conjunto gerado para SFA.

Exemplo 4.1.3. Φ = *{ (1)* $b \Leftarrow \emptyset; a$*}*
{a} é fechado e minimal, pois $\Gamma(\emptyset) = \{b\}$.
{a} não é ponto fixo, pois $\Gamma(\{a\}) = \emptyset$.

Observe que o conjunto gerado que intuitivamente esperamos é {b}, que é na verdade o único ponto fixo minimal e também um conjunto fechado minimal.

Por este exemplo, pode-se chegar a conclusão que o conjunto gerado não corresponde a conjuntos fechados minimais, mas talvez corresponda a pontos fixos minimais.

O exemplo abaixo mostra que pontos fixos minimais de Γ também não são adequados.

Exemplo 4.1.4. [2] *Considere o SFA* Φ *formado pelas seguintes regras:*

$a \Leftarrow b; \emptyset$

$b \Leftarrow a; \emptyset$

$c \Leftarrow \emptyset; a$

$d \Leftarrow \emptyset; b$

{a,b} é ponto fixo minimal, pois $\Gamma(\{a, b\}) = \{a,b\}$, *e* $\Gamma(\{a\}) = \{b,d\}$, $\Gamma(\{b\}) = \{a,c\}$ *e* $\Gamma(\emptyset) = \{c,d\}$ *(a propósito, também é fechado minimal).*

O conjunto que queremos gerar é {c,d} que é ponto fixo minimal também.

Um terceiro exemplo mostra como pontos fixos minimais e conjuntos fechados minimais não correspondem entre si.

Exemplo 4.1.5. Φ = *{(1)* $a \Leftarrow \emptyset; a$ *}.*
{a} é fechado e minimal, pois $\Gamma(\{a\}) = \emptyset$ *, e* $\Gamma(\emptyset) = \{a\}$. *Este operador não tem pontos fixos.*

[2] Note que tomamos a liberdade de escrever um conjunto unitário sem as chaves.

Vale ainda observar mais um exemplo:

Exemplo 4.1.6. $\Phi = \{\ (1)a \Leftarrow \emptyset; b\ ,\ (2)b \Leftarrow \emptyset; a\ \}$
$\{a\}$ e $\{b\}$ são pontos fixos minimais de Γ.

Qual o conjunto que deveria ser gerado nestes dois últimos exemplos? Na nossa concepção ambos são SFA's mal formados, a única regra do exemplo 4.1.5 não tem sentido, "'na ausência de 'a' derive-se 'a'", qual o propósito desta regra? O mesmo vale as duas regras do exemplo 4.1.6, cada regra faz sentido em si mesmo, mas o que significam quando estão juntas, o que se pretende gerar? Vamos argumentar na dissertação que tais sistemas mal formados não são estratificados e, portanto, não geram conjuntos.

Estes exemplos já nos deixam antever que circularidade é uma questão central em SFA. A aplicabilidade de uma regra em um SFA não pode depender dela mesma. A derivação de suas premissas negativas deve se dar sem a participação desta regra. Por exemplo, há circularidades tanto no exemplo 4.1.5, onde a regra (1) deriva sua própria premissa negativa, como também no exemplo 4.1.6, onde a regra (1) ao ser aplicada barra a regra (2) e desta forma garante a não derivação de sua premissa negativa. A circularidade se dá no fato da regra (1) ser relevante para a não derivação de sua própria premissa negativa.

Embora poucos argumentem que mesmo SFA's como o do exemplo 4.1.5 deva gerar algum conjunto, muitos argumentam que os dois pontos fixos são os conjuntos gerados pelo SFA do exemplo 4.1.6. Nossa posição, via sistemas estratificados, é contrária a este ponto de vista que encontra muitos defensores em programação lógica. Analisaremos com mais detalhes as propostas em programação lógica no último capítulo da dissertação.

Depois de escrever um SFA bem formado, um que não apresente ciclos, como descobrir o conjunto gerado por ele? A nossa investigação tem mostrado que a construção do conjunto gerado não está relacionado com o senso comum, mas está diretamente relacionado com a capacidade de estratificação das regras com premissas negativas. O papel da estratificação é atrasar a aplicação de uma regra até que as informações suficientes estejam disponíveis para a sua aplicação segura. Por aplicação segura entendemos que as regras que serão aplicadas em estágios posteriores não são capazes de derivar nenhuma premissa negativa da regra. Então precisamos definir quais são as informações suficientes para a aplicação segura de

uma regra e, a partir disto, definir o conjunto gerado por SFA como o conjunto obtido por uma seqüencia de aplicações seguras das regras.

Chamaremos de relevância a relação que se estabelece quando uma regra pode potencialmente derivar a premissa negativa de uma outra regra. A partir da relação de relevância vamos definir uma estratificação das regras. Proporemos também um método para encontrar uma seqüência de aplicações seguras das regras baseado no método de Marek e Truszczynski para caracterizar o conjunto gerado por elas.

Entendemos que a estratificação relevante vai nos permitir definir o conjunto gerado para um SFA eliminando as circularidades que causam o *círculo vicioso* na definição do conjunto gerado e tornando a sua definição *impredicativa*. A nossa abordagem visa restaurar a *predicatividade* da definição do conjunto gerado através da estratificação relevante a seguir a definição da *estratificação relevante*.

4.1.1 Estratificação relevante

A estratificação dos SFA's será obtida através da relação de relevância entre as regras de um SFA. Um regra é relevante para outra se ela pode derivar potencialmente as premissas negativas da outra regra. Todas as regras relevantes para uma regra de um estrato, deve aparecer nos estratos anteriores. Uma regra de um SFA Φ pode derivar potencialmente um x $\in \Phi$, se a regra é relevante para x em SFE trans(Φ). A relação de relevância em um SFE também será definida. Uma regra é relevante para outra em SFE, se a regra é usada em uma derivação minimal de uma das premissas positivas da outra. A seguir, definiremos formalmente cada um destes conceitos.

Definição 4.1.2. *Seja Φ um SFE. Uma derivação $a_1, ..., a_n$ para a_n é minimal em Φ sse para todo $1 \leq i \leq n$, $a_1, ..., a_{i-1}, a_{i+1}, ..., an$ não é uma derivação de a_n em Φ .*

Definição 4.1.3. *(Relevância em um SFE) Seja Φ um SFE. Dizemos que uma regra $r \in D$ é relevante para um átomo $a \in U$ sse r é usada em uma derivação minimal de a em Φ.*

Definição 4.1.4. *(Relação de Relevância em um SFE) Seja Φ um SFE. Seja $<_R$ relação binária de ordem entre as regras. Seja $r_1 = x_1 \leftarrow X_1$ e $r_2 = x_2 \leftarrow X_2$ regras em Φ.*

$$r_1 \leq_R r_2, \text{ se } r_1 \text{ é relevante para algum } y_2 \in X_2,$$

$<_R$ *é uma relação de relevância entre as regras.*

Definição 4.1.5. *(Transformação de SFA em SFE) Seja um SFA* Φ *um SFE correspondente denotado por trans(*Φ*), tal que:*

$$x \leftarrow X \in trans(\ \Phi\ ) \text{ sse } x \leftarrow X\ ;\ Y \in \Phi.$$

Se uma regra $r \in \Phi$ *denotaremos por r' correspondente em trans(*Φ*).*

Definição 4.1.6. *(Relevância em um SFA) Seja* Φ *um SFA, dizemos que uma regra* $r \in \Phi$ *é relevante para um átomo a em* Φ *sse a regra correspondente r'* $\in$ *trans(*Φ*) é relevante para a em trans(*Φ*).*

Definição 4.1.7. *(Relação de relevância em um SFA) Seja* Φ *um SFA. Sejam r1 = x1* $\leftarrow$ *X1 ; Y1 e r2 = x2* $\leftarrow$ *X2 ; Y2 regras em* Φ *.*

Dizemos que r1 $<_R$ *r2 sse r1 é relevante para y2* $\in$ *Y2 em trans(*Φ*).*

Exemplo 4.1.7. *Considere o seguinte SFA* Φ_1 *formado pelas seguintes regras:*

$$\begin{array}{lll} (0) & par(0) & \\ (1) & par(1) & \leftarrow \emptyset\ ;\ par(0) \\ (2) & par(2) & \leftarrow \emptyset\ ;\ par(1) \\ & \vdots & \\ (n+1) & par(n+1) & \leftarrow \emptyset\ ;\ par(n) \end{array}$$

A regra (0) é relevante para par(0) em trans(Φ_1*). Logo, (0)* $<_R$ *(1). As regras (0) e (1) são relevantes para par(1) em trans(*Φ_1*). Logo, (0)* $<_R$ *(2) e (1)* $<_R$ *(2). No caso geral, temos que* $\{(0), \ldots, (n)\}$ *são relevantes para par(n) em trans(*Φ_1*). Logo, para todo 1*$\leq$ *i* $\leq$ *n,(i)* $<_R$ *(n+1).*

Exemplo 4.1.8. *Considere o seguinte SFA* Φ_2 *formado pelas seguintes regras:*

$$\begin{array}{ll} (1) & b \leftarrow a \\ (2) & a \leftarrow b \\ (3) & c \leftarrow \emptyset\ ;\ a \\ (4) & a \leftarrow b\ ;\ c \end{array}$$

Não temos nenhuma regra é relevante para a em trans(Φ_2). Logo, nenhuma regra é menor que a regra (3). A regra (3) é relevante para c. Logo, (3) $<_R$ (4).

Exemplo 4.1.9. *Considere o seguinte SFA Φ_3 formado pelas seguintes regras:*

$$(1) \quad a \leftarrow ; b$$

$$(2) \quad b \leftarrow ; c$$

$$(3) \quad c \leftarrow ; a$$

A regra (2) é relevante para b em trans(Φ_3). Logo, (2) $<_R$ (1). A regra (1) é relevante para a em trans(Φ_3). Logo, (1) $<_R$ (3). A regra (3) é relevante para c em trans(Φ_3). Logo, (3) $<_R$ (2). Temos que (2) $<_r$ (1) $<_R$ (3) $<_R$ (2).

Exemplo 4.1.10. *Considere o seguinte SFA Φ_4 formado pelas seguintes regras:*

$$(1) \quad a \leftarrow ; b$$

$$(2) \quad b \leftarrow ; a$$

A regra (2) é relevante para b em trans(Φ_4). Logo, (2) $<_R$ (1). A regra (1) é relevante para a trans(Φ_4). Logo, (1) $<_R$ (2). Temos que (1) $<_R$ (2) $<_R$ (1).

Definição 4.1.8. *Um SFA é relevantemente estratificado se somente se a relação de ordem $<_R$ é uma ordem parcial estrita bem-fundada.*

Exemplo 4.1.11. *Considere os seguintes SFA's:*

Φ_1 *é relevantemente estratificado.*

Φ_2 *é relevantemente estratificado.*

Φ_3 *é não relevantemente estratificado.*

Φ_4 *é não relevantemente estratificado.*

4.1.2 Conjunto Gerado para um SFA relevantemente estratificado

O conjunto gerado, I(Φ), pode ser obtido através de uma seqüência de aplicações seguras das regras de Φ. A seqüencia segura será construída através de uma boa ordem $\preceq$ compatível com a relação ordem de $<_R$, ou seja, uma extensão linear da

ordem $<_R$. A boa ordem será usada para determinar a próxima regra segura que pode ser aplicada. Quando nenhuma regra puder ser aplicada, a construção pára.

Vamos adotar a seguinte notação, dizemos que um conjunto Y pertence a um conjunto S, se para algum y ∈ Y, y ∈ S.

Definição 4.1.9. *(Conjunto Gerado por SFA relevantemente estratificado) Seja* Φ=<U,D> *um SFA relevantemente estratificado. Seja* $\preceq$ *uma boa ordem em* Φ *compatível com* $<_R$*, ou seja, uma extensão linear da ordem* $<_R$*.*

Vamos construir iterativamente o conjunto de regras gerado pela boa ordem $\preceq$*,* $GR_{\preceq}$*, da seguinte maneira:*

i. *se* $\alpha = 0$ *então* $GR_\alpha = \emptyset$

ii. *caso contrário* d_α *é a* $\preceq$*-menor regra aplicável no conjunto* $\Phi\backslash(\bigcup_{\xi<\alpha} GR_\xi)$*, ou seja,*
$p^+(d_\alpha) \in c(\bigcup_{\xi<\alpha} GR_\xi)$
$p^-(d_\alpha) \notin c(\bigcup_{\xi<\alpha} GR_\xi)$
Então $GR_\alpha = (\bigcup_{\xi<\alpha} GR_\xi) \cup \{d_\alpha\}$

iii. *caso contrário, se não existe regra* $d \in \Phi\backslash(\bigcup_{\xi<\alpha} GR_\xi)$ *tal que*
$p^+(d) \in c(\bigcup_{\xi<\alpha} GR_\xi)$
$p^-(d) \notin c(\bigcup_{\xi<\alpha} GR_\xi)$
então a construção pára,$GR_{\eta_\preceq} = (\bigcup_{\xi<\eta} GR_\xi)$

O conjunto $c(GR_\preceq)$ *é chamado conjunto gerado pela boa ordem* $\preceq$*.*

O seguinte lema vai garantir que a ordem de escolha das regras preserva a relação de relevância entre as regras, ou seja, se uma regra r_1 é relevante para uma regra r_2 e ambas foram escolhidas então a regra r_1 deverá ser escolhida antes da regra r_2.

Lema 4.1.1. *Seja* Φ *uma SFA relevantemente estratificado, Seja* $\preceq$ *uma extensão linear de* $<_R$ *para* Φ *e* r_ξ *e* $r_\eta \in GR_\preceq$ *são regras escolhidas no* ξ*-ésimo e* η*-ésimo passo então*

$$se\ r_\xi < r_\eta\ então\ \xi < \eta.$$

O seguinte teorema estabelece que se uma regra é escolhida em um estágio ξ, ela continua sendo aplicada em $GR_\preceq$.

Teorema 4.1.1. *Seja Φ um SFA relevantemente estratificado e $\preceq$ uma extensão linear de $<_R$ para Φ então a seguinte condição é válida*

$$\textit{para toda regra } r = x \Leftarrow X; Y \in GR_{\preceq} \textit{ temos que } Y \notin c(GR_{\preceq}).$$

Podemos definir um algoritmo para computar o conjunto gerado por SFA da seguinte maneira:

Conjunto_Gerado_SFA_Relevantemente_Estratificado(Φ, $\preceq$)
Entrada: Um SFA Φ relevantemente estratificado e $\preceq$ compatível com a ordem $<_R$, ou seja, uma extensão linear de $<_R$;
Saída: O conjunto gerado pela boa ordem $\preceq$.
A := $\emptyset$
R := $\emptyset$

faça
r:= a $\preceq$-menor regra r do conjunto $\Phi \backslash R$ aplicável em A
R := R $\cup$ {r}
A := A $\cup$ {c(r)}
enquanto(r $\neq \emptyset$)

retorne A;

Exemplo 4.1.12. *Considere o SFA Φ_1 e $\preceq$ uma boa ordem compatível com $<_R$ de Φ_1 temos que:*

$$GR_{\eta_{\preceq}} = \{par(0), par(2) \leftarrow \neg par(1), par(4) \leftarrow \neg par(3), \ldots\}$$
$$c(GR_{\eta_{\preceq}}) = \{par(0), par(2), par(4), \ldots\}.$$

Considere o SFA Φ_2 e $\preceq$ uma boa ordem compatível com $<_R$ de Φ_2 temos que:

$$GR_{\eta_{\preceq}} = \{c \leftarrow \emptyset; a\}$$
$$c(GR_{\eta_{\preceq}}) = \{c\}$$

No próximo capítulo, vamos fazer uma comparação com o nossa abordagem com as principais abordagens para a semântica para programas normais:

i. Semântica Estratificada

ii. Semântica Estável

iii. Semântica Bem-Fundada

No capítulo seguinte, apresentaremos cada semântica para programas normais através dos sistemas formais avançados. Mostraremos que a nossa abordagem coincide com todas as semânticas para programas normais estratificadas e com as semânticas para programas não estratificados. A nossa abordagem é consensual entre todas as abordagens para programas normais.

Capítulo 5

Programas Lógicos Normais

5.1 Programação Lógica com a negação

Nesta seção, vamos apresentar diversas abordagens estratificadas para a programação lógica com a negação. Vamos mostrar que a nossa abordagem coincide com todas as abordagens estratificadas para programas normais. A seguir, vamos fazer uma breve apresentação dos programas normais, mostraremos como é uma regra de um programa normal e como ela pode ser mapeada em uma regra de um SFA.

Os programas normais foram introduzidos por Clark(CLARK, 1978). Um programa normal P é um programa lógico contendo regras do tipo $A \leftarrow \neg B$, onde A e B são átomos e $\neg$ representa certo tipo de negação, que tem natureza não-monotônica, denominada negação por falha.

Definição 5.1.1. *Um programa normal é um conjunto de regras normais. Uma regra normal é uma regra da forma*

$$A \Leftarrow A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m,$$

onde A, $A_1, \ldots, A_n, B_1, \ldots, B_m$ *são átomos de*B_H*(Base Herbrand). O átomo* A *é conclusão da regra e será denotado por c(r).* $A_1, \ldots, A_n$ *são as premissas positivas e será denotado por* $p^+(r)$. $B_1, \ldots, B_m$ *serão as premissas negativas e será denotado por* $p^-(r)$. *O conjunto das premissas será denotado por* $p(r) = p^+(r) \cup p^-(r)$. *Uma regra é chamada fato se o conjunto da premissas é vazio.*

Note que o mapeamento para um SFA é quase direto. As premissas positivas de uma regra normal equivalem as premissas positivas de uma regra de um SFA. Semelhantemente, as premissas negativas de uma regra normal equivalem as premissas negativas de uma regra de um SFA.

Seja P um programa normal, nós denotaremos por fecho(P), a instanciação de todas as regras sem nenhum variável livre.

Definição 5.1.2. *(Operador de Conseqüência Imediata) Seja P um programa normal. O operador de conseqüência imediata denotado por $T_P(I)$ será:*

$$T_P(I) = \{ A : A \Leftarrow A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m \in P, \\ A_1, \ldots, A_n, \in I \ e \ B_1, \ldots, \neg B_m \notin I\}$$

A introdução da negação por falha(NF) trouxe dois dilemas para a comunidade de programação lógica:

i. A interpretação dos programas positivos proposta por Kowalski e Van Emden(KOWALSKI; EMDEN, 1976) como o menor ponto fixo do operador de conseqüencia imediata T_P não poderia ser aplicada aos programas normais. Nos programas normais, T_P tem um comportamento não-monotônico e o menor ponto fixo do operador T_P pode não existir.

ii. A capacidade de fazer inferências a partir da ausência de informações é uma ferramenta poderosa, útil e natural para diversas aplicações práticas e não deveria ser desconsiderada.

Clark introduziu a primeira semântica para os programas normais denominada *completion* e com ela introduziu também o primeiro grande erro da programação lógica com a negação. A partir dele, os programas normais passaram a ser associados com lógica e uma visão errônea foi sendo perpetuada.

Clark propôs interpretar um programa lógico como uma teoria de primeira ordem, chamada completion de um programa. A sua abordagem é matematicamente elegante e fundamentada na idéia natural que no discurso comum nós usamos implicações, quando nós realmente queremos dizer são equivalências.

Infelizmente, a semântica de Clark possui sérios problemas. A *completion* consegue apenas ser adequada para programas livres de recursão (APT; BEZEM, 1990) e falha em prover uma semântica para os programas que envolve recursão mesmo quando se trata de programas positivos.

Recursão Positiva: pode ser ilustrada pelo programa p $\Leftarrow$ p, no qual o predicado p depende *positivamente* dele mesmo.

Recursão Negativa: pode ser ilustrada pelo programa p $\Leftarrow \neg$ p, no qual o predicado p depende *negativamente* dele mesmo.

Note que a visão limitada dos programas normais através de uma teoria de primeira ordem não levou uma reflexão profunda sobre estes problemas. A recursão negativa deveria ser rejeitada por causa da sua impredicatividade, mas a recursão positiva não deve causar nenhum problema. O problema das recursões positivas na semântica completion foi apontada em (PRZYMUSINSKI, 1989; GELDER; ROSS; SCHLIPF, 1991).

Exemplo 5.1.1. *Vamos considerar o seguinte programa:*

aresta(a,b)

aresta(c,d)

aresta(d,c)

encontrado(a)

encontrado(X) $\Leftarrow$ *encontrado(Y)* , *aresta(Y,X)*

A completion do predicado "'encontrado"' é:

$$encontrado(X) \leftrightarrow (X = a \vee \exists.Y(encontrado(Y) \wedge aresta(Y, X)))$$

O resultado esperado seria obter que $\neg$ encontrado(c) e $\neg$ encontrado(d) mas este resultado não pode ser obtido pela completion do predicado encontrado. O problema é causado pela presença da cláusula simétricas aresta(c,d) e aresta(d,c). Para obtermos $\neg$ encontrado(c) temos que mostrar

$$\neg\exists.Y(encontrado(Y) \wedge aresta(Y, X))$$

e para isso temos que mostrar

¬ encontrado(d).

Interessante notar que a semântica não consegue sair dessa armadilha criada pela definição que precisa de recursões positivas.

Denecker e seus colaboradores (DENECKER; BRUYNOOGHE; MAREK, 2001) defendem a tese que os programas lógicos devem ser entendidos como uma representação natural de uma definição indutiva não através de ferramentas lógicas. Nós compactuarmos com essa visão e propormos a nossa visão particular de um programa lógico através dos SFA relevantemente estratificados. A seguir, vamos apresentar as diversas semânticas de programação lógica de uma maneira particular.

5.2 Abordagens Estratificadas

5.2.1 Programas Estratificados

Apt, Blair e Walker (APT; BLAIR; WALKER, 1988) introduziu a classe dos programas normais estratificados. Um programa normal é estratificado se ele satisfaz alguns tipos restrições sintáticas que tem por objetivo evitar as recursões negativas, ou recursões pela negação, na definição dos predicados. Por exemplo, o programa P = $\{p \Leftarrow \neg p\}$ não é estratificado, porque o predicado p é definido pela negação do predicado p. O programa P = $\{p \Leftarrow \neg q, q \Leftarrow p\}$ não é estratificado, porque o predicado p é definido pela negação de q que por sua vez é definido pela negação de p.

Definição 5.2.1. *(Definição de um predicado) Seja p um predicado de um programa lógico. Então:*
def(p) o conjunto de regras que definem o predicado p, ou seja, um conjunto de regras com o predicado p como conclusão. Um predicado p depende (negativamente)positivamente de um predicado q, quando q é premissa (negativa)positiva em def(p).

Definição 5.2.2. *(Programa normal estratificado) Um programa normal é estratificado se def(p) não depende da negativamente do predicado p, ou seja, tem como premissa negativa o próprio predicado p.*

Em outras palavras, um programa normal é estratificado sse não existe nenhuma recursão negativa envolvendo a definição de um predicado. A classe de programa

estratificado é muito restrita. Alguns programas que tem a recursão negativa na definição de um predicado pode não ter uma recursão negativa quando considerado a instanciação fechada de todas as regras (fecho(P)). Por exemplo,

Exemplo 5.2.1. *Considere o seguinte programa P_1 que define o predicado par:*

$$\begin{aligned} &(1) \quad par(0) \\ &(2) \quad par(s(X)) \quad \Leftarrow \neg\, par(X) \end{aligned}$$

O programa P_1 não é estratificado, uma vez que na regra(2) o predicado par depende negativamente do próprio predicado par. Tal ciclo não existe quando consideramos o fecho(P_1).

Um programa normal estratificado pode ser definido de uma maneira alternativa. Um programa normal é estratificado se ele admite uma estratificação. A seguir vamos definir uma estratificação:

Definição 5.2.3. *(Estratificação) Considere um programa $P = P_1 \cup \ldots \cup P_n$ é chamada uma estratificação de P se para todo i, $1 \leq i \leq n$, cada estrato P_i depende*

- *depende positivamente dos predicados definidos em $\bigcup_{j=1}^{i} P_j$,*
- *depende negativamente dos predicados definidos em $\bigcup_{j=1}^{i-1} P_j$.*

P_1 pode ser considerado vazio. Por conveniência, quando um predicado usado em P não é definido, pode ser assumido que eles são definidos(por um conjunto vazio de regras) em P_1.

A seguir, vamos definir o conjunto gerado por cada estrato P_i em função de um conjunto de átomos I.

Definição 5.2.4. *Seja P_i um estrato e I um conjunto de átomos, o conjunto gerado pelo estrato P_i e pela interpretação I será denotado por $T(P_i, I)$, e será definido da seguinte maneira:*

$$\begin{aligned} T(P_i, I) = I \cup \{\, A : A \Leftarrow A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m \in fecho(P_i), \\ A_1, \ldots, A_n, \in T(P_i, I) \text{ e } B_1, \ldots, \neg B_m \notin I\} \end{aligned}$$

Agora podemos definir o conjunto gerado por um programa normal iterativamente utilizando $T(P_i, I)$.

Definição 5.2.5. *Considere um programa estratificado P e seja $P_1, \ldots, P_n$ um estratificação de P.*

$$\begin{aligned} I_0 &= \emptyset, \\ I_i &= T(P_i, I_{i-1}), 0 < i \leq n \end{aligned}$$

O conjunto gerado é $I_P = I_n$.

Exemplo 5.2.2. *Considere o seguinte programa P:*

$$\begin{aligned} &(1) \quad q \Leftarrow r \\ &(2) \quad p \Leftarrow \neg\, q \end{aligned}$$

Seja $P = \{q \Leftarrow r\} \cup \{p \Leftarrow \neg q\}$ uma estratificação maximal de P. Então $I_1 = \emptyset$ e $I_2 = I_P = \{p\}$.

5.2.2 Programas Localmente Estratificado

Przmunsisnski(PRZYMUSINSKI, 1988) introduziu a classe dos programas localmente estratificados. Um programa P é localmente estratificado quando não existe recursões negativas na definição dos átomos de P, ou seja, a definição de um átomo é independente do seu complemento. Por exemplo, o programa do exemplo 5.2.1 que define o predicado par não é estratificado, porque a definição do predicado par depende da negação do predicado par. Mas o mesmo programa é localmente estratificado, porque cada átomo depende da negação de um outro átomo. Por exemplo, considere par(s(X)) $\Leftarrow \neg$ par(X), apesar da regra depende negativamente dela pela negação, quando consideramos o fecho do programa a dependência negativa desaparece, por exemplo par(1) $\Leftarrow \neg$ par(0).

Definição 5.2.6. *(Definição de um átomo) Seja a um átomo.Então:*
def(a) o conjunto de regras que definem o átomo a, ou seja, o conjunto das regras r com o átomo a como conclusão. Um átomo depende (negativamente)positivamente de um átomo b, quando b é premissa (negativa) positiva em def(a).

Definição 5.2.7. *(Programa localmente estratificado) Um programa normal é localmente estratificado se def(a) não depende da negativamente do átomo a, ou seja, tem como premissa negativa o próprio átomo a. Ou equivalentemente, um programa é localmente estratificado quando fecho(P) é estratificado.*

Podemos definir um programa localmente estratificado de maneira alternativa. Um programa é dito localmente estratificado se ele admite uma estratificação local . A seguir vamos definir uma estratificação local:

Definição 5.2.8. *(Estratificação Local)*

- *a estratificação local para um programa P é uma função* ***estrato*** *que associa a cada átomo de B_P em um ordinal contável.*
- *a função de estratificação local* ***estrato****, é definida também para a negação dos átomos de B_P da seguinte maneira estrato($\neg a$) = estrato(a) + 1*
- *uma regra de P é localmente estratificada com relação a função de estratificação local* ***estrato*** *se para todo $A \Leftarrow A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m \in fecho(P)$,*

$$\textbf{\textit{estrato}}(A) \geq \textbf{\textit{estrato}}(A_i), 1 \leq i \leq n,$$
$$\textbf{\textit{estrato}}(A) \geq \textbf{\textit{estrato}}(\neg B_i), 1 \leq i \leq m,$$

- *um programa é localmente estratificado com relação a uma função de estratificação local* ***estrato****, se todas as suas regras também são.*
- *cada regra $A \Leftarrow A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m \in P_{estrato(A)}$.*

Definição 5.2.9. *Seja P um programa localmente estratificado. Seja $P = P_1, \ldots, P_n$ uma estratificação local de P. Então o conjunto gerado pode ser obtido da seguinte maneira:*

$$E_0 = T(P_0, \emptyset)$$
$$E_i = T(P_i, E_{i-1}), 0 < i \leq n$$

E_n é o conjunto gerado.

Exemplo 5.2.3. *Considere o seguinte programa P que define o predicado par:*

$$(1) \quad par(0)$$
$$(2) \quad par(s(X)) \quad \Leftarrow \neg\, par(X)$$

No seguinte programa, a estratificação local estrato pode ser definida da seguinte maneira:

$$estrato(\ par(X)\ ) = 2*X$$
$$estrato(\ \neg\, par(X)\ ) = 2*X + 1$$

Note que essa estratificação local torna todas as regras localmente estratificada e que

$$par(0) < par(1) < par(2) < \ldots$$

O conjunto gerado é {par(0),par(2),...}

Apesar da classe dos programas localmente estratificado ser mais abrangente que a classe dos programas estratificados. Alguns transformações sintáticas simples podem fazer um programa localmente estratificado deixar de ser localmente estratificado. Considere o seguinte programa:

Exemplo 5.2.4. *Considere o seguinte programa P:*

$$(1) \quad par(0)$$
$$(2) \quad par(Y)) \quad \Leftarrow sucessor(X,Y)\ ,\ \neg\, par(X)$$
$$(3) \quad zero(0) \quad \Leftarrow$$
$$(4) \quad sucessor(X,s(X)) \quad \Leftarrow$$

Este programa é obtido através de transformações sintáticas simples do programa do exemplo 5.2.1. O programa não é localmente estratificado. Observe uma instância fechada da regra (2)

$$par(0) \Leftarrow sucessor(0,0)\ ,\ \neg\, par(0)$$

Não podemos definir uma função de estratificação local para esse programa, uma vez que pela estratificação local temos que

$$estrato(\ \neg\ par(0)\ ) = estrato\ (\ par(0)\ ) + 1$$

e para a regra tornar localmente estratificada temos que

$$estrato(\ par(0)\ ) \geq estrato(\ \neg\ par(0)\ )$$

Logo,

$$estrato(\ par(0)\ ) \geq estrato(\ par(0)\ ) + 1$$

Não podemos definir uma função de estratificação local.

A seguir vamos analisar duas propostas de adaptação da estratificação local : estratificação fraca introduzida por Przymunsinska e Przymunsinski (PRZYMUSINSKA; PRZYMUSINSKI, 1988, 1990) e estratificação efetiva introduzida por Bidoit e Froidevaux (BIDOIT; FROIDEVAUX, 1991b).

A idéia central de uma extensão da estratificação local é desconsiderar as regras com premissas falsas(Como descobrir-las ?). A resposta para essa pergunta é dada de maneira diferente nos dois conceitos de estratificação : estratificação fraca e estratificação efetiva.

5.2.3 Programas Fracamente Estratificado

Przymunsinska e Przymunsinski (PRZYMUSINSKA; PRZYMUSINSKI, 1988, 1990) introduziu a classe dos programas fracamente estratificado como uma extensão da classe dos programas localmente estratificados. A idéia central desta abordagem é separar os átomos em componentes. Cada componente agrupa os átomos com dependência mútua pela negação. Podemos definir uma relação de ordem entre as componentes através da negação. A componente minimal representa os átomos que não dependem negativamente de nenhum outro. A união da componente minimal serão usados para identificar as regras com premissas falsas, ou seja, as regras não usáveis. A seguir, as definições que caracterizam a estratificação fraca.

Definição 5.2.10. *(PRZYMUSINSKA; PRZYMUSINSKI, 1988)*

i. *Seja P um programa. O grafo de precedência G_P associado ao programa P será definido da seguinte maneira: Os vértices são os átomos de B_P e existe uma aresta positiva(negativa) de A para B em G_P sse existe uma regra r em fecho(P) tal que $c(r) = A$ e B está em $p^+(r)$(B está em $p^-(r)$).*

ii. *A relação de dependência (ou prioridade)* $<$ *e* $\leq$ *entre os átomos de* B_P *da seguinte maneira. Para toda regra* $A \leftarrow A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m$*, temos que*

(a) $A \leq A_i$*, para todo* $1 \leq i \leq n$*.*

(b) $A < B_j$ *, para todo* $1 \leq i \leq m$*.*

iii. *O relacionamento transitivo para* $<$ *e* $\leq$ *será:*

(a) se $A \leq B$ *e* $B \leq C$*, então* $A \leq C$*.*

(b) se $A < B$ *e* $B < C$*, então* $A < C$*.*

(c) se $A \leq B$ *e* $B < C$ *ou se* $A < B$ *e* $B \leq C$*, então* $A < C$*.*

iv. *A relação de equivalência* $\simeq$ *entre os átomos de* B_P *é definido da seguinte maneira:*

$$A \simeq B \equiv (A = B) \vee (A < B \wedge B < A).$$

As classes de equivalência são denominadas componentes do grafo G_P *e uma componente é trivial se ela consiste apenas de um único átomo de* B_P*.*

v. *A relação* $<$ *é introduzida entre as componentes do grafo de dependência* G_P *da seguinte maneira:*

$$C_1 < C_2 \text{ sse } C_1 \neq C_2, \exists A \in C_1, \exists B \in C_2 (A < B).$$

A relação $<$ *induz uma ordem parcial. Uma componente* C_1 *é minimal se não existe nenhuma componente* C_2 *tal que* $C_2 < C_1$*.*

vi. *O menor estrato S(P) de P é a união dos componentes minimais de* G_P*.*

vii. *A menor camada L(P) de P é um subprograma consistindo das regras que a conclusão pertence ao menor estrato S(P) de P.*

Exemplo 5.2.5. *(GELFOND; LIFSCHITZ, 1988) Considere o seguinte programa P:*

$$p(1,2) \Leftarrow$$
$$q(X) \Leftarrow p(X,Y), \neg\ q(Y).$$

O fecho de P é:

$$\begin{array}{llll} (1) & p(1,2) & \Leftarrow & \\ (2) & q(1) & \Leftarrow & p(1,2), \neg\, q(2) \\ (3) & q(1) & \Leftarrow & p(1,1), \neg\, q(1) \\ (4) & q(2) & \Leftarrow & p(2,2), \neg\, q(2) \\ (5) & q(2) & \Leftarrow & p(2,1), \neg\, q(1) \end{array}$$

Note que o programa P não é localmente estratificado. A relação de dependência $<$ *e* $\leq$ *entre os átomos de* B_P *são:*

$$q(1) < q(2), q(2) < q(1),$$

$$q(1) \leq p(1,2), q(1) \leq p(1,1), q(2) \leq p(2,2), q(2) \leq p(2,1).$$

O programa P tem 5 componentes:

$$C_1 = \{q(1), q(2)\},\ C_2 = \{p(1,2)\},\ C_3 = \{p(1,1)\},\ C_4 = \{p(2,2)\},\ C_5 = \{p(2,1)\}$$

O menor estrato S(P) de P é dado pela união das componentes minimais de P:

$$S(P) = \{p(1,2), p(1,1), p(2,2), p(2,1)\},$$

A menor camada L(P) de P é um conjunto de regras R tal que para toda $r \in R$, $c(r) \in S(P)$, *e será dada por :*

$$L(P) = \{p(1,2) \Leftarrow\}.$$

Intuitivamente, dada um programa P, a interpretação de um programa fracamente estratificado é obtido da seguinte maneira. Primeiramente, o menor estrato S(P) e a menor camada L(P) de P são identificadas. O conjunto dos átomos de S(P) que são gerado por L(P) e os que não são gerados são usado para simplificar o programa inicial. O processo é iterado considerando um novo menor estrato e um nova menor camada de um programa transformado.

A transformação do programa P vai levar em consideração dois conjuntos:

i. Um conjunto dos átomos M de S(P) que são gerados pela menor camada L(P).

ii. Um conjunto dos átomos N de S(P) que não são gerados pela menor camada L(P).

A seguir vamos definir a transformação de um programa que correspondem as transformações Davis-Putman(DAVIS; PUTNAM, 1960) para conjunto de regras.

Definição 5.2.11. *Seja P um programa normal e M,N subconjuntos de átomos. A transformação do programa será dada por duas operações:*

i. *A operação de redução(P , M , N) remove todas as premissas positivas que estão em M de cada regra e as premissas negativas que estão em N de cada regra.(Em outras palavras, deleta as premissas que são desnecessárias considerando M gerado e N não gerado)*

ii. *A operação de simplificação(P , M , N) remove as regras que tem alguma premissa positiva em N ou alguma premissa negativa em M.(Em outras palavras, deleta as regras que não serão usadas considerando M gerado e N não gerado).*

Exemplo 5.2.6. *Considere o seguinte programa P:*

$$
\begin{array}{llll}
(1) & p(1,2) & \Leftarrow & \\
(2) & q(1) & \Leftarrow & p(1,2),\ \neg\, q(2) \\
(3) & q(1) & \Leftarrow & p(1,1),\ \neg\, q(1) \\
(4) & q(2) & \Leftarrow & p(2,2),\ \neg\, q(2) \\
(5) & q(2) & \Leftarrow & p(2,1),\ \neg\, q(1)
\end{array}
$$

Seja P_1 o programa obtido por Redução(P , p(1,2) , $\emptyset$) será:

$$
\begin{array}{llll}
(1) & q(1) & \Leftarrow & \neg\, q(2) \\
(2) & q(1) & \Leftarrow & p(1,1),\ \neg\, q(1) \\
(3) & q(2) & \Leftarrow & p(2,2),\ \neg\, q(2) \\
(4) & q(2) & \Leftarrow & p(2,1),\ \neg\, q(1)
\end{array}
$$

Seja P_2 o programa obtido por Simplificação(P_1 , $\emptyset$, {p(1,1),p(2,2),p(2,1)}) será:

$$(1) \quad q(1) \quad \Leftarrow \quad \neg\, q(2)$$

Este é o programa obtido pela seguinte transformação

Simplificação (Redução(P , p(1,2) , $\emptyset$) , $\emptyset$, {p(1,1),p(2,2),p(2,1)}).

Seja P um programa. O conjunto M de átomos de S(P) que são gerado por L(P) e o conjunto N de átomos de S(P) que não são gerados por L(P). O programa transformado por M e N será dado pela seguinte transformação:

Simplificação(Redução(P, M , N) , M , N)

No exemplo anterior, o conjunto M de átomos de S(P) que são gerados por L(P) é {p(1,2)} e o conjunto de átomos de S(P) que não gerados em L(P) é {p(1,1),p(2,2),p(2,1)}. O programa Simplificação(Redução(P, M , N) , M , N) será:

$$\text{q(1)} \Leftarrow \neg \text{q(2)}.$$

Observe que a redução do programa P retirou as regras irrelevantes (3),(4) e (5) de P.

A construção do conjunto gerado pela interpretação fraca é obtida da seguinte maneira. Seja o programa P = P_0 e $M_0 = \emptyset$ e N_0 o conjunto de átomos que não são definidos em P(ou seja, que não ocorrem em P). Seja P_1 = Simplificação(Redução (P_0 , M_0 , N_0) , M_0 , N_0), encontre o menor estrato S(P_1) e a menor camada L(P_1) de P_1. Seja M_1 o conjunto de átomos de S(P_1) que são gerado por L(P_1) e N_1 é o conjunto de átomos de S(P_1) que não são gerados em L(P_1) . Obtenha o novo programa P_2 = Simplificação(Redução (P_1 , M_1 , N_1) , M_1 , N_1). Continue o processo até o programa P_k seja vazio. Neste caso, $M_0 \cup \ldots \cup M_{k-1}$ é o conjunto gerado pela estratificação fraca, ou, caso contrário, se S(P_k) for vazio ou o conjunto gerado por $L(P_k)$ for vazio então o programa não é fracamente estratificado.

A seguir, as definições que caracterizam a estratificação fraca.

Definição 5.2.12. ***i.*** *Seja P é um programa normal e seja $P_0 = P, M_0 = \emptyset, N_0$ é o conjunto de átomos não definidos em P. M_P é o conjunto gerado pela estratificação fraca de P. Vamos definir o ordinal η. Para todo $\alpha < \eta$, temos que:*

$M^\alpha = \bigcup_{\delta<\alpha} M_\delta$

$N^\alpha = \bigcup_{\delta<\alpha} N_\delta$

$P_{\alpha+1} = Simplificação(Redução(P_0, M^\alpha, N^\alpha), M^\alpha, N^\alpha)$

$S_{\alpha+1} = S(P_{\alpha+1}), L_{\alpha+1} = L(P_{\alpha+1})$

$M_{\alpha+1}$ é o conjunto de átomos de $S_{\alpha+1}$ que são gerado por $L_{\alpha+1}$.

$N_{\alpha+1}$ é o conjunto de átomos de $S_{\alpha+1}$ que não são gerados por $L_{\alpha+1}$.

- *Se o programa $P_{\alpha+1}$ é vazio, então M^α é conjunto gerado pela estratificação fraca de P e $\eta = \alpha + 1$.*
- *Caso contrário, se S_α é vazio ou M_α é vazio então o programa não tem um conjunto gerado e $\eta = \alpha$.*

O ordinal η é chamado de largura de P e é denotado por $\delta(P)$. Para $0 < \alpha < \delta(P)$, o conjunto S_α é chamado de α-ésimo estrato de P e o programa L_α é chamado de α-ésima camada de P.

ii. *P é fracamente estratificado sse*

(a) todos os estratos $S_\alpha(\alpha < \delta(P))$ consiste apenas de componentes triviais ou, equivalentemente,

(b) todos as camadas $L_\alpha(\alpha < \delta(P))$ são programas lógicos positivos.

Nota: As duas definições de programas fracamente estratificados não são equivalentes.
Seja P=$\{A \Leftarrow \neg A\}$. O menor estrato de S(P) de P é $\{A\}$ é consiste de apenas componentes triviais. Pela definição 1, P é fracamente estratificado. Mas a menor camada de L(P) de P é $\{A \Leftarrow \neg A\}$ não é um programa positivo. Pela definição 2, P não é fracamente estratificado.

Exemplo 5.2.7. *Considere o programa do exemplo 5.2.5:*

$$M_0 = \emptyset$$

$$N_0 = \{p(1,1), p(2,1), p(2,2)\}$$

P_1 = Simplificação (Redução(P,M_0,N_0), M_0 , N_0) será o seguinte programa:

(1) $p(1,2) \Leftarrow$

(2) $q(1) \Leftarrow p(1,2), \neg\, q(2)$

$$S_1 = \{p(1,2)\}$$

$$L_1 = \{p(1,2) \leftarrow\}$$

$$M_1 = \{p(1,2)\}$$

$$N_1 = \emptyset$$

P_2 = Simplificação (Redução(P_1,M_1,N_1), M_1 , N_1) será o seguinte programa:

(1) $q(1) \leftarrow \neg q(2)$

$$S_2 = \{q(2)\}$$

$$L_2 = \emptyset$$

$$M_2 = \emptyset$$

$$N_2 = \{q(2)\}$$

P_3 = Simplificação (Redução(P_2,M_2,N_2), M_2 , N_2) = $\emptyset$

(1) $q(1) \leftarrow$

$$S_3 = \{q(1)\}$$

$$L_3 = \{q(1)\}$$

$$M_3 = \{q(1)\}$$

$$N_3 = \emptyset$$

P_4 = Simplificação (Redução(P_3,M_3,N_3), M_3 , N_3) = ∅
A construção do conjunto gerado por P pára, então o conjunto gerado será:

$$M_P = M_0 \cup M_1 \cup M_2 \cup M_3 = \{p(1,2),q(1)\}.$$

O programa P é fracamente estratificado, {S_1,S_2,S_3} é a estratificação fraca para P, porque S_1, S_2 são componentes triviais e L_1, L_2 são programas positivos.

Alguns programas simples, sem funções, podem ser exibidos que nem são localmente e nem fracamente estratificado mas fazem sentido ou em outras palavras exibem um "'bom comportamento"'. Por exemplo,

Exemplo 5.2.8. *Considere o seguinte programa P:*

$$B \Leftarrow A \qquad (1)$$
$$A \Leftarrow B \qquad (2)$$
$$C \Leftarrow \neg A \qquad (3)$$
$$A \Leftarrow B, \neg C \qquad (4)$$

Neste exemplo, P tem apenas uma componente minimal {A,B,C}. A camada inicial L(P) de P é o próprio programa P e ele não é um programa positivo. Logo, P não é fracamente estratificado. Podemos dizer que este programa tem uma interpretação bem definida, os átomos A e B não serão gerados porque um é definido em função do outro. Pela regra (3), podemos gerar C. A interpretação deste programa deveria ser {C}.

5.2.4 Programas Efetivamente Estratificados

Bidoit e Froidevaux(BIDOIT; FROIDEVAUX, 1991a) definiram uma classe de programas estratificados mais abrangente que a classe de programas fracamente estratificados.

Na estratificação fraca de um programa P, o menor estrato S(P) de P é usado para transformar o programa (remover algumas dependências entre átomos de B_P e as regras de P). O processo é iterado até programa resultante ficar vazio. Se todas as camadas de P (conjuntos de regras transformadas de P) são programas

lógicos positivos ou cada estrato consiste apenas de componentes triviais então o programa é fracamente estratificado e tem uma interpretação fraca. A idéia chave da estratificação fraca é remoção de dependências desnecessárias. A remoção das dependências desnecessárias é possível quando substituímos os átomos de B_P pelas componentes do grafo de dependência G_P de P.

Na estratificação efetiva, estamos interessados em descobrir quais são as regras "'efetivas"' do fecho (P). As regras "'efetivas"'de fecho(P) são um subconjunto das regras de fecho(P) sem possíveis dependências. Se as regras "'efetivas"' do fecho(P) formam um programa estratificado então o programa é dito efetivamente estratificado. As transformações do programa são obtidas considerando o conjunto de átomos definidos em fecho(P) denotado por Def(P) e o conjunto dos átomos indefinidos no fecho(P) denotado por Undef(P). Dados os dois conjuntos podemos transformar o programa. O processo de transformação do programa é definido pelo operador EFT. O operador pode ser aplicado sucessivamente até que nenhuma transformação seja possível, ou seja, o programa final tenha se tornado efetivo.

A seguir vamos definir o conjunto Def(P), PotDef(P) e UnDef(P) que serão essenciais para a definição da transformação EFT:

Denotaremos por P^+ um subprograma de P com regras sem premissas negativas. Intuitivamente, um átomo A é definido em P se A pode ser gerado pelo programa P^+. Um átomo A é potencialmente definido se A pode ser gerado a partir das regras de P ignorando as premissas negativas das regras em P.

Definição 5.2.13. *Seja P um programa. O conjunto de átomos Definidos denotado por Def(P). Formalmente definido como* $Def(P) = \bigcup_{i=1,\ldots,\infty} Def_P^i$ *,onde*
$Def_P^1 = \emptyset$
$Def_P^{i+1} = Def_P^i \cup \{c(r) : \exists r \in P^+, \forall B \in p(r), B \in Def_P^i\}$.

Definição 5.2.14. *O conjunto de átomos potencialmente definidos denotado por PotDef(P). Formalmente definido como* $PotDef(P) = \bigcup_{i=1,\ldots,\infty} PotDef_P^i$ *,onde*
$PotDef_P^1 = Def(P)$
$PotDef_P^{i+1} = PotDef_P^i \cup \{c(r) : \forall B \in Prem(r)^+, B \in Def_P^i\}$.

Definição 5.2.15. *O conjunto de átomos indefinidos denotado por UnDef(P).*
UnDef(P) = B_P *- PotDef(P).*

A seguir vamos definir o operador EFT como a composição de duas transformações (Simplificação e Redução) da seguinte maneira:

Definição 5.2.16. *Seja P um programa. O operador efetivo denotado por EFT é definido por :*

EFT(P) = Simplificação(Redução(P, Def(P) , UnDef(P)) , Def(P) , UnDef(P)).

O operador EFT é iterado até que não seja possível nenhuma transformação, ou seja, o programa final é formado somente por regras efetivas. A seguir uma seqüência de aplicações sucessivas do operador EFT da seguinte maneira:

Definição 5.2.17. *A seqüência $EFT^i(P)$ é definido por:*

$EFT(P)^0(P) = P$

$EFT(P)^{i+1}(P) = EFT(EFT^i(P))$, *para todo* $i \geq 0$

Nós denotaremos por EFT^{∞}(P) o programa EFT^i(P) tal que EFT^i(P) = EFT^{i+1}(P).

Definição 5.2.18. *Seja P um programa. Então, P é efetivamente estratificado se somente se existe um i tal que $EFT^i(P)$ é estratificado.*

O conjunto gerado pela estratificação efetiva será dado por EFT^{∞}(P).

Exemplo 5.2.9. *Considere o seguinte programa P:*

$$B \Leftarrow A \qquad (1)$$

$$A \Leftarrow B \qquad (2)$$

$$C \Leftarrow \neg A \qquad (3)$$

$$A \Leftarrow B, \neg\, C \qquad (4)$$

(1) Def(P) = ∅ ; PotDef(P) = {C} ; UnDef(P) = {A,B}. Assim EFT(P) = {C} é um programa estratificado e o programa P é estratificado. O conjunto gerado é {C}.

5.3 Abordagens Não Estratificadas

Temos duas abordagens principais que podem ser aplicadas aos programas normais não estratificados: a semântica estável e a semântica bem-fundada. As duas abordagens são divergentes em relação aos programas não estratificados. Este

fato corrobora com a nossa visão que a estratificação elimina as circularidades indesejáveis que causam o círculo vicioso na definição do conjunto gerado. Reforçando a nossa tese que a definição do conjunto gerado através da estratificação é uma definição predicativa.

5.3.1 Semântica Estável

A semântica estáveis, introduzida em(GELFOND; LIFSCHITZ, 1988), é uma extensão da semântica estratificada para a classe dos programas normais não estratificados. A semântica estável tem uma caracterização por ponto fixo simples e elegante e é intimamente relacionada com a lógica autoepistêmica(MOORE, 1985) como foi apontado em(GELFOND, 1987; MAREK; TRUSZCZYNSKI, 1991). Uma definição equivalente da semântica estável foi também obtido baseado na lógica default(REITER, 1980) em (BIDOIT; FROIDEVAUX, 1991a) e (**??**). Mais recentemente, a semântica estável foi utilizada para a fundação de um paradigma de programação denominado ASP(Answer Set Programming).

A interpretação de um programa normal dada pela semântica estável é denominada interpretação estável. A interpretação estável pode ser obtida como a extensão de uma teoria default(obtida pelo mapeamento das regras do programa normal em regras default). Entendemos que o motivo dessa correspondência é que as duas semânticas são obtidas por operadores idênticos. A seguir, mostraremos a equivalência entre os operadores Gelfond-Lifschitz e operador de Reiter da lógica default.

Um conjunto de átomos I é o conjunto estável de um programa P, quando I é o conjunto gerado pelo programa P^I. O programa P^I é obtido redução Gelfond-Lifshitz no programa P e o conjunto de átomos I. A seguir vamos definir a redução Gelfond-Lifshitz.

Definição 5.3.1. *Seja P um programa e I um conjunto de átomos de B_P. A redução Gelfond-Lifschitz P^I é aplicada no programa fecho(P) em dois passos:*

i. *Elimine todas as regras $r = A \Leftarrow A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m$ tal que $B_i \in I$, para todo $1 \leq i \leq m$.*

ii. *Elimine das regras restantes todos os átomos que aparecem com a negação.*

Agora, o programa Π^I é um programa positivo, o conjunto gerado pelo programa Π^I pode ser obtido de várias maneiras alternativas. Vamos considerar que o programa gerado pelo programa Π^I será obtido pelo $\mathbf{lfp}(\Pi^I)$.

A seguir vamos definir o operador Gelfond-Lifschitz que definir o conjunto gerado pela redução de Gelfond-Lifschitz e um conjunto de átomos I.

Definição 5.3.2. *Seja P um programa e I um conjunto de átomos. O operador Gelfond-Lifschitz GL_P é definido a seguir:*

$GL_P(I) = \mathbf{lfp}(P^I)$.

Definição 5.3.3. *Seja P um programa e I um conjunto de átomos. I é um conjunto estável sse $GL_P(I) = I$.*

Exemplo 5.3.1. *Considere o seguinte programa P:*

$$p(1,2) \Leftarrow$$
$$q(X) \Leftarrow p(X,Y),\ \neg\ q(Y).$$

Vamos mostrar que o conjunto I = {p(1,2),q(1)} é um conjunto estável. O programa P^I é:

$$\begin{array}{llll} (1) & p(1,2) & \Leftarrow & \\ (2) & q(1) & \Leftarrow & p(1,2) \\ (3) & q(2) & \Leftarrow & p(2,2) \end{array}$$

O menor ponto fixo de P^I é {p(1,2),q(1)}. Logo I é um conjunto estável.

A semântica estável para um programa normal P pode gerar mais de um conjunto estável, ou seja, temos mais de um conjunto gerado. Considere o seguinte exemplo:

Exemplo 5.3.2. *Considere o seguinte programa P:*

$$a \Leftarrow \neg\ b$$
$$b \Leftarrow \neg\ a.$$

O conjunto $I_1 = \{a\}$ e $I_2 = \{b\}$ são conjuntos estáveis de P.

A semântica estável gera uma interpretação estável única para todos os programas estratificados concordando com as interpretações obtidas pelas semânticas estratificadas. Mas não pode ser aplicada para todos os programas normais. Por exemplo:

Exemplo 5.3.3. *Consideremos o seguinte exemplo:*

$$b \Leftarrow \neg\ a$$

$$a \Leftarrow \neg\ b$$

$$p \Leftarrow \neg\ p$$

$$p \Leftarrow \neg\ a$$

Este programa tem um conjunto estável único I = {p,b}. Mas este conjunto estável é totalmente inesperado. Primeiramente, as duas primeiras cláusulas não dão nenhuma preferência para a ou b, mas b é gerado. Para a geração de p, temos que garantir que a não é gerado, e apoiado nisso, temos b é gerado. A semântica estável permite que uma regra contribua para a sua própria aplicação. Esse conjunto estável fere a visão construtiva da interpretação de um programa lógico a partir do $\emptyset$. Acreditamos que este problema está relacionado com o círculo vicioso presente na definição do conjunto estável que torna a definição impredicativa.

Outra fato interessante da semântica estável é que o operador definido pela semântica estável GL_P é equivalente ao operador definido por Reiter para a lógica default. A seguir vamos mostrar definir o operador de Reiter para programas normais e depois vamos mostrar a equivalência entre os operadores.

Seja P um programa normal, o operador Γ_P é definido a seguir: Γ_P é o menor conjunto de átomos tal que:

i. para toda $A \Leftarrow A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m \in$ fecho(P), se $A_1, \ldots, A_n \in \Gamma_P$ e $B_1, \ldots, B_m \notin I$, então A $\in \Gamma_P$.

I é uma extensão sse $\Gamma_P(I) = I$.

Esta é "'a semântica de Reiter"' para programas normais.

Teorema 5.3.1. *(PEQUENO; VERAS; TAVARES, 2007) Seja* Π *um programa normal e para todo conjunto de átomos I, então* $GL_P(I) = \Gamma_P(I)$

Demonstração. Para mostrar que $\Gamma_P(I) \subseteq GL_P(I)$ é suficiente mostrar que $GL_P(I)$ satisfaz a condição (1) da definição de $\Gamma_P(I)$.

Suponha $A \Leftarrow A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m \in$ fecho(P). Assuma que $A_1, \ldots, A_n \in GL_P(I)$ e $B_1, \ldots, B_m \notin I$. Portanto, $A \Leftarrow A_1, \ldots, A_n \in P^I$. Logo, A $\in$ $lfp(P^I)$ então A $\in$ $GL_P(I)$. Para mostrar o contrário, é suficiente mostrar que $GL_P(\Gamma_P(I)) \subseteq \Gamma_P(I)$,$GL_P(I) = lfp(P^I)$. Seja $A \Leftarrow A_1 \ldots, A_n \in P^I$ e $A_1, \ldots, A_n \in \Gamma_P(I)$. Temos dois casos:

i. $A \Leftarrow A_1 \ldots, A_n \in P$, então pela condição (1) de $\Gamma_P(I)$, então A $\in \Gamma_P(I)$.

ii. $A \Leftarrow A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m \in$ P, portanto, pela construção de P^I, temos que $B_1, \ldots, B_m \notin I$. Portanto, pela condição (1) da definição de $\Gamma_P(I)$, $A \in \Gamma_P(I)$. □

Podemos dizer que a semântica estável gera os mesmos conjuntos da semântica default de Reiter para programas normais. Por um lado, temos que lógica default gera extensões anômalas mesmo para teorias estratificadas. Por outro lado, a semântica estável não gera conjunto estáveis anômalos para a classe de programas estratificados. Acreditamos que o bom comportamento da semântica estável na classe dos programas estratificados não se dá pela própria semântica, mas sim pela restrição de linguagem. Realmente, Vladimir Lifschitz argumentou o seguinte durante a análise do "'Yale Shooting Problem"':

"'Paradoxicalmente, a limitação da linguagem de programação lógica desempenha um papel positivo no caso do "'Yale Shooting Problem"' eliminando algumas más escolhas representacionais disponíveis na lógica default quando certas ações são descritas."'

Entendemos que o problema não está nas escolhas representacionais (representação das regras), mas em como a semântica default de Reiter manipula tais escolhas representacionais.

5.3.2 Semântica Bem-fundada

A semântica bem-fundada, introduzida por (GELDER; ROSS; SCHLIPF, 1991), é uma alternativa de extensão da semântica estratificada para toda a classe de todos os programas normais. A semântica bem-fundada tem um comportamento regular com relação a recursão negativa. Assim, os conjuntos gerados pela

semântica bem-fundada são considerados naturais e intuitivos para a classe de todos os programas normais. A semântica bem-fundada tem diversas caracterizações equivalentes.

Os conjuntos gerados pela semântica bem-fundada serão denominados interpretação bem-fundada. Na semântica bem-fundada, alguns átomos são considerados indefinidos(não podem ser gerados e nem não gerados). Para os programas estratificados, os conjuntos bem-fundados serão totais, ou seja, nenhum átomo é indefinido.

A interpretação bem-fundada de um programa normal P é definido iterativamente em cada passo, encontramos um conjunto de átomos bem-fundados e um conjunto de átomos infundados com relação a conjunto de átomos fundados anteriores e o conjunto de átomos infundados anteriores. O conjunto dos átomos bem-fundados representam os átomos que são gerados por P. O conjunto de átomos infundados representam os átomos que não são gerados por P.

A seguir vamos definir um conjunto infundado A com relação a um conjunto de átomos T e um conjunto de átomos F.

Definição 5.3.4. *Seja P um programa normal, T , F subseteqB_P. Um conjunto $A \subseteq B_P$ é um conjunto infundado de P com relação a T e F se cada átomo $p \in A$ satisfaz umas das seguintes condições:*

Para cada regra $r = a \Leftarrow a_1, \ldots, a_n, \neg b_1, \ldots, \neg b_m \in fecho(P)$ cuja conclusão c(r) é p umas dessas condições valem:

i. *Algum $a_i \in F$, $1 \leq i \leq n$ ou algum $b_i \in T$, $1 \leq i \leq m$.*

ii. *Algum $a_i \in A$, $1 \leq i \leq n$*

Um átomo que torna as condições (1) ou (2) verdadeiras é chamada testemunha de descarte para a cláusula r (com relação a S e T).

Seja P um programa, T um conjunto de átomos bem-fundados e F um conjunto de átomos infundados. Intuitivamente, o conjunto formado pelas condições (1) e (2) é um conjunto de átomos que não podem ser gerados. Note que:

A condição(1) garante que a regra que gera p ∈ A não será aplicada uma vez que uma das suas premissas positivas não são geradas e uma das suas premissas

negativas são geradas. A condição(2) garante que a regra que gera p ∈ A não será aplicada uma vez que uma das suas premissas positivas estão em A.

Exemplo 5.3.4. *Considere o seguinte programa:*

$$p(a) \Leftarrow p(c), \neg p(b)$$
$$p(b) \Leftarrow \neg p(a)$$
$$p(e) \Leftarrow \neg p(d)$$
$$p(c)$$
$$p(d) \Leftarrow q(a), \neg q(b)$$
$$p(d) \Leftarrow q(b), \neg q(c)$$
$$q(a) \Leftarrow p(d)$$
$$q(b) \Leftarrow q(a)$$

Os átomos {p(d),q(a),q(b),q(c)} é um conjunto infundado com relação S = ∅ e T = ∅. Em particular, {q(c)} é infundado devido a condição (1) porque não existe regra que possa gerar q(c). O conjunto {p(d),q(a),q(b)} é conjunto infundado considerando condição (2) porque p(d) não pode ser gerado antes que q(a) e q(b) sejam gerados. Nem podemos gerar q(a) sem antes gerarmos p(d) e q(b) não pode ser gerado sem antes gerarmos q(a). Neste exemplo, q(c) nunca poderá ser gerado e em {p(d),q(a),q(b)} nenhum pode ser gerado primeiro.

Em contrate, o conjunto {p(a) , p(b)} não é um conjunto infundado mesmo que exista uma dependência mútua entre eles, porque a dependência é através da negação.

Definição 5.3.5. *O maior conjunto infundado de P com relação a T e F, denotado por $U_P(T, F)$, é a união de todos os conjuntos infundados com relação a T e F.*

Vamos definir uma seqüência possivelmente transfinita resultante da combinação de duas transformações. O limite dessa seqüência será a interpretação bem-fundado.

Definição 5.3.6. *Seja P um programa e T e F conjuntos de átomos de B_P. As transformações T_P, U_P e W_P são definidas da seguinte maneira:*

i. $T_P(T, F) = \{a : r = a \Leftarrow a_1, \ldots, a_n, \neg b_1, \ldots, \neg b_m \in fecho(P), a_1, \ldots, a_n \in T, b_1, \ldots, b_m \in F\}$.

ii. *$U_P(T, F)$ é o maior conjunto infundado de P com relação a T e F.*

Note que T_P trata as premissas positivas e negativas simetricamente, ou seja, para a aplicação da regra com a premissa negativa ¬ p é exigido que p ∈ F, não apenas que p $\notin T$.

Definição 5.3.7. *Seja α um ordinal. O conjunto T_α,F_α,T^∞ e F^∞ subconjuntos de B_P são definidos recursivamente como:*

i. *Inicialmente, temos:*

$$T_0 = \emptyset.$$
$$F_0 = \emptyset.$$

ii. *Para um ordinal sucessor $\alpha = \gamma + 1$,*

$$T_{\gamma+1} = T_P(T_\gamma, F_\gamma).$$
$$F_{\gamma+1} = U_P(T_\gamma, F_\gamma).$$

iii. *Para um ordinal limite α infinito,*

$$T_\alpha = \bigcup_{\beta<\alpha} T_\beta.$$
$$F_\alpha = \bigcup_{\beta<\alpha} F_\beta.$$

iv. *Finalmente, temos:*

$$T^\infty = \bigcup_\alpha T_\alpha.$$
$$F^\infty = \bigcup_\alpha F_\alpha.$$

A seqüência de T^α e T^α são monótonas. Pelo teorema Knaster-Tarski temos que T^∞ é o menor ponto fixo do operador T_P e F^∞ é o menor ponto fixo do operador U_P. A base Herbrand é contável, então para algum ordinal contável temos que $T^\infty = T_\alpha$ e $F^\infty = F_\alpha$.

Definição 5.3.8. *O conjunto bem-fundada de um programa P é dada pelo menor ponto fixo do operador T_P, ou pelo limite T^∞ da seqüência descrita acima; todo átomo gerado por P pertence ao conjunto bem-fundado.*

A semântica bem-fundada gera um interpretação bem-fundada total que concorda com todas as semânticas estratificadas para a classe dos programas estratificados.

A seguir vamos considerar alguns exemplos motivacionais para a semântica bem-fundada:

Exemplo 5.3.5. *Considere o seguinte programa P que representa uma versão abstrata do "'Yale Shooting Problem"':*

$$barulho(T) \Leftarrow carregada(T), atirou(T)$$

$$carregada(T) \Leftarrow succ(S,T),\ carregada(S),\ \neg\ atirou(S)$$

$$atirou(T) \Leftarrow engatilhada(T)$$

$$vivo(T) \Leftarrow \neg\ barulho(T)$$

$$engatilhada(1)$$

$$succ(0,1)$$

$$carregada(0)$$

Considere o programa dado pelo fecho(P):

(1) $barulho(0) \Leftarrow carregada(0), atirou(0)$

(2) $barulho(1) \Leftarrow carregada(1), atirou(1)$

(3) $carregada(1) \Leftarrow succ(0,1),\ carregada(0),\ \neg\ atirou(0)$

(4) $carregada(1) \Leftarrow succ(1,1),\ carregada(1),\ \neg\ atirou(1)$

(5) $carregada(0) \Leftarrow succ(0,0),\ carregada(0),\ \neg\ atirou(0)$

(6) $carregada(0) \Leftarrow succ(1,0),\ carregada(1),\ \neg\ atirou(1)$

(7) $atirou(1) \Leftarrow engatilhada(1)$

(8) $atirou(0) \Leftarrow engatilhada(0)$

(9) $vivo(0) \Leftarrow \neg\ barulho(0)$

(10) $vivo(1) \Leftarrow \neg\ barulho(1)$

(11) $engatilhada(1)$

(12) $succ(0,1)$

(13) $carregada(0)$

Este programa é estratificado e admite a seguinte estratificação:

$$S_1 = \{(11), (12), (13)\}$$
$$S_2 = \{(7), (8)\}$$
$$S_3 = \{(3), (4), (5), (6)\}$$
$$S_4 = \{(1), (2)\}$$
$$S_5 = \{(9), (10)\}$$

O conjunto padrão pela estratificação é o conjunto $M = \{carregada(0), engatilhada(1), atirou(1), carregada(1), barulho(1), vivo(0), succ(0,1)\}$.

M é o único conjunto estável para o programa P.

A semântica bem-fundada obtém o seguinte resultado:

$T_0 = \emptyset$ *e* $F_0 = \emptyset$

$T_1 = \{carregada(0), engatilhada(1), succ(0,1)\}$

$F_1 = \{succ(0,0), succ(1,0), succ(1,1), atirou(0), engatilhada(0), barulho(0)\}$

$T_2 = \{atirou(1), carregada(1), vivo(0)\}$

$F_2 = \emptyset$

$T_3 = \{barulho(1)\}$

$F_3 = \emptyset$

$T_4 = \emptyset$

$F_4 = \{vivo(1)\}$

O conjunto bem-fundado dado $\bigcup_{0 \leq i \leq 4} T_i$ *é igual a M.*

Exemplo 5.3.6. *Considere o seguinte programa P:*

$$(1) \quad B \Leftarrow A$$

$$(2) \quad A \Leftarrow B$$

$$(3) \quad C \Leftarrow \neg A$$

$$(4) \quad A \Leftarrow B, \neg\ C$$

A semântica bem-fundada obtém o seguinte resultado:

$T_0 = \emptyset$ *e* $F_0 = \emptyset$

$T_1 = \emptyset$

$F_1 = \{A, B\}$

$T_2 = \{C\}$

$F_2 = \emptyset$

O conjunto bem-fundado dado $\bigcup_{0 \leq i \leq 2} T_i$ *concorda com a estratificação fraca.*

A seguir veremos como a semântica bem-fundada comporta-se com programas que envolvem a recursão negativa através do seguinte exemplo:

Exemplo 5.3.7. *Consideremos o seguinte programa P:*

$$b \Leftarrow \neg\ a$$

$$a \Leftarrow \neg\ b$$

$$p \Leftarrow \neg\ p$$

$$p \Leftarrow \neg\ a$$

A semântica bem-fundada do programa P temos que $T^{\infty} = \emptyset$ *e* $F^{\infty} = \emptyset$. *O conjunto bem-fundado é* $\emptyset$ *e todos os átomos estão indefinidos.*

A nossa ressalva sobre a semântica bem-fundada é sobre a definição impredicativa dos conjuntos infundados. Apesar da semântica bem-fundada ter outras caracterizações construtivas para programas normais não estratificados (PRZYMUSINSKI, 1989; GELDER, 1989). Entendemos que essas caracterizações tiram proveito da limitação da linguagem e não podem ser generalizados para ambientes mais ricos como da lógica default com resultados satisfatórios.

5.4 Comparação com as outras abordagens

A nossa estratificação quando aplicada em programação lógica será denominada estratificação relevante e o programa com essa estratificação será denominado relevantemente estratificado. A seguir, vamos apresentar os seguintes resultados relacionando as abordagens estratificadas.

Teorema 5.4.1. *Todo programa fracamente estratificado é relevantemente estratificado.*

Teorema 5.4.2. *Seja P um programa fracamente estratificado. Então a conjunto gerado pela estratificação fraca e a conjunto gerado pela estratificação relevante coincidem.*

Teorema 5.4.3. *(PRZYMUSINSKA; PRZYMUSINSKI, 1990) Todo programa (localmente) estratificado é fracamente estratificado.*

Teorema 5.4.4. *(PRZYMUSINSKA; PRZYMUSINSKI, 1990) Seja P um programa (localmente) estratificado. Então a conjunto gerado pela estratificação (local) e a conjunto gerado pela estratificação fraca coincidem.*

Como corolário, temos que a nossa abordagem concorda também com a estratificação (local).

A classe dos programas relevantemente estratificados é estritamente maior que a classe dos programas fracamente estratificados.

Exemplo 5.4.1. *Considere o seguinte programa P:*

$$B \Leftarrow A \qquad (1)$$
$$A \Leftarrow B \qquad (2)$$
$$C \Leftarrow \neg A \qquad (3)$$
$$A \Leftarrow B, \neg\, C \qquad (4)$$

P não é fracamente estratificado, mas é relevantemente estratificado.

Teorema 5.4.5. *Todo programa relevantemente estratificado é efetivamente estratificado.*

Teorema 5.4.6. *Seja P um programa relevantemente estratificado. Então a conjunto gerado pela estratificação efetiva e a conjunto gerado pela estratificação relevante coincidem.*

Teorema 5.4.7. *(BIDOIT; FROIDEVAUX, 1991b) Seja P um programa efetivamente estratificado. Então a conjunto gerado pela estratificação efetiva e a conjunto gerado pela semântica estável e pela semântica bem-fundada coincidem.*

Como corolário, temos que a nossa abordagem também concorda com a semântica estável e com a semântica bem-fundada.

A classe dos programas relevantemente estratificados é estritamente menor que a classe dos programas efetivamente estratificados

Exemplo 5.4.2. *Considere o seguinte programa P:*

$$A$$
$$A \Leftarrow \neg A$$

O programa P é efetivamente estratificado, mas não é relevantemente.

Capítulo 6

Lógica Default Localmente Estratificada

Nesta capítulo, introduziremos a *Lógica Default Localmente Estratificada* que foi obtida através da generalização do conceito de estratificação dos SFA's para um ambiente mais rico como da lógica default.

A *Lógica Default Localmente Estratificada* ataca simultaneamente os dois problemas principais da lógica default: o problema da coerência e o problema das extensões anômalas em teorias defaults.

Acreditamos que os dois problemas citados têm a sua origem na *circularidade das teorias*. A utilização do conceito de estratificação é crucial para evitar a *circularidade* e com isso resolver os dois problemas.

Há dois importantes precedentes para esta abordagem que parcialmente propuseram e utilizaram os conceitos que lançamos mão para construir nossa proposta. Etherington(ETHERINGTON, 1987) desenvolve um método sintático para evitá-la circularidade de teorias default para garantir a existência de extensões. Vale ressaltar que ele nada propõe para o problema das extensões anômalas que continuam se manifestando em teorias ordenadas segundo os seus critérios. A nossa estratificação é construída seguindo os moldes propostos de Etherington. Choleswinski (CHOLEWINSKI, 1995a, 1995b, 1996) propõe o que ele chama de *Stratified Default Logic*. Novamente a intenção não foi resolver o problema das extensões anômalas, mas propor métodos eficientes para a computação de extensões de Reiter. Aliás, a proposta deles de estratificação é bastante precária do ponto

de vista do conjunto a ser gerado por uma teoria default uma vez que deixa de fora teorias com conjuntos gerados perfeitamente bem definidos (teorias ordenadas). Choleswinski propõe um método de geração de extensões através de uma ordenação dos default que concorde com a sua estratificação, baseado no método de Marek e Truszczynski. Vamos utilizar o método proposto por ele, mas utilizando a estratificação proposta por Etherington. Podemos dizer que nos beneficiamos diretamente dos trabalhos dos que nos procederam. Finalmente, mostraremos que a nossa abordagem resolve o problema das extensões anômalas.

Acreditamos também que a eliminação das circularidades das teorias default através de um critério sintático vai resolver o problema do círculo vicioso da definição da extensão. A construção da extensão pautada na ordem definida pelo critério sintático tornará a definição da extensão construtiva e predicativa.

O problema da coerência e das extensões anômalas foi atacado na lógica default anteriormente através da *Well-Behaved IDL*(*Lógica Default Inconsistente Bem Comportada*) (PEQUENO; MARTINS; PEQUENO, 1996; MARTINS, 1997). Nesta abordagem o problema das extensões anômalas é resolvido através da diferenciação da parte normal e não normal da justificativa de um default e uma a lógica paraconsistente chamada LEI*Lógica da Inconsistência Epistêmica*)(PEQUENO; BUCHSBAUM, 1991) e para resolver o problema da coerência de IDL(PEQUENO, 1990) foi utilizado um critério sintático baseado no método de Etherington para identificação de ciclos indesejáveis. Uma comparação mais efetiva é uma tarefa difícil de ser realizada devido o fato das duas lógicas estarem baseadas em lógicas subjacentes diferentes mas a medida do possível vamos apontar as principais diferenças e semelhanças entre as duas abordagens.

6.1 Preliminares

A lógica default introduzida por Reiter provou ser o formalismo não-monotônico mais proeminente. Nele, a inferência não-monotônica é representada por um conjunto de regras inconclusivas denominadas regras default. As regras default têm um padrão de inferência baseados na presença e na ausência de informações. Nós vamos estudar as teorias default sobre uma linguagem de proposicional $\mathcal{L}$. Uma teoria default é formada por um conjunto de fórmulas proposicionais e um conjunto de regras default.

Um regra default seminormal **d** é uma expressão da seguinte forma

$$\mathbf{d} = \frac{\alpha:\gamma\wedge\beta}{\gamma},$$

onde α, β e γ são fórmulas proposicionais arbitrárias. α é denominado pré-requisito e será denotada por **Pre**(d), β é denominado justificativa e será denotada por **Jus**(d) e γ é denominada a conclusão e será denotada por **Cons**(d). Um default tem a seguinte interpretação "'se você acredita em α, e é consistente acreditar em β e γ, ou equivalentemente, $\neg(\beta \wedge \gamma)$ não é derivado então acredite em γ"'.Um default seminormal $\frac{\alpha:M\gamma\wedge\beta}{\gamma}$ pode ser entendido como $\gamma \Leftarrow \alpha; \neg(\beta \wedge \gamma)$

Uma teoria default Δ é um par (W,D), onde W é um conjunto de sentenças de primeira ordem e D é um conjunto de regras default. O conjunto W representa o conjunto de fatos assumidos verdadeiros e D representa regras que contribuem com informações plausíveis mas não necessariamente aplicáveis.

Uma extensão E de uma teoria default E é o conjunto gerado através de W e dos defaults.

6.2 Teorias Estratificadas

Nesta seção, vamos apresentar o método sintático desenvolvido por Etherington para evitar a circularidade das teorias default. O método de Etherington será importante para a nossa abordagem porque através dele vamos identificar as teorias estratificadas e também garantir a existência de extensões.

Etherintgon fez um estudo bem aprofundado sobre a existência de extensões em uma teoria default em (ETHERINGTON, 1987). Nesse estudo, ele identifica os critérios para existência de extensão. O resultado de Etherington torna-se especial por caracterizar a classe de teorias bem formadas em teoria default.

A existência de mais de uma extensão em teoria default é considerada natural porque algumas crenças não podem ser assumidas ao mesmo tempo, por razões de inconsistência na extensão obtida. Consideramos este motivo para a existência de mais de uma extensão. Considere o seguinte exemplo,

Exemplo 6.2.1. *A teoria default* $\Delta = (W, D)$*:*
$W = \emptyset$
$D = \{\frac{:A}{A}, \frac{:\neg A}{\neg A}\}$
A teoria default tem duas extensões
$E_1 = \{A\}$ *e* $E_2 = \{\neg A\}$.

Mas algumas teorias não tinham nenhuma extensão, mesmo quando W era consistente. Por exemplo,

Exemplo 6.2.2. *A teoria default* $\Delta = (W, D)$*:*
$W = \emptyset$
$D = \{\frac{:A}{\neg A}\}$

E outras teorias geravam mais de uma extensão por não por motivo de consistência, mas sim pela impossibilidade de resolver o conflito entre as regras. Por exemplo,

Exemplo 6.2.3. *A teoria default* $\Delta = (W, D)$*:*
$W = \emptyset$
$D = \{\frac{:A \wedge \neg B}{A}, \frac{:B \wedge \neg A}{B}\}$
A teoria default tem duas extensões
$E_1 = \{A\}$ *e* $E_2 = \{B\}$.

Note que no exemplo 6.2.2 e 6.2.3, podemos ver que existe um problema dependência circular entre as regras. No exemplo 6.2.2, ¬ A depende da ausência ¬A. No exemplo 6.2.3, A depende da ausência de B e B depende da ausência de A.

Etherington estava preocupado em entender o porquê de algumas teorias não terem nenhuma extensão e mais de uma extensão sem ser por motivos de consistência. Mais além de apontar a existência de teorias incoerentes (ou seja, teorias sem extensões) e teoria com extensões sem ser por motivos de consistência. Ele queria descobrir as condições suficientes para garantir um bom comportamento das teorias default.

Ele percebeu que o problema da não existência de extensões não estava restrito apenas as teorias não-normais, mas o problema ocorria também em teorias seminormais:

Exemplo 6.2.4. *A teoria default* $\Delta = (W, D)$*:*
$W = \emptyset$
$D = \{\frac{:A \wedge \neg B}{A}, \frac{:B \wedge \neg C}{B}, \frac{:C \wedge \neg A}{C}\}$
A teoria default não tem extensões.

Ele identificou que no caso das teorias default (seminormais) sem extensão a aplicação de uma regra default contribui a sua não aplicação através de uma dependência circular através da ausência. Neste casos, a aplicação de uma regra default permite a aplicação de um outra regra default. Contudo, a aplicação de quaisquer duas regras resulta derivação da parte não normal da justificativa de pelo menos uma das regras. No exemplo 6.2.4, A depende da ausência de B, B depende da ausência de C e C depende da ausência de A. Portanto, se inferirmos A bloquearmos C que permite inferir B que invalida A, e similarmente para B e C. No problema da extensões "'irreais"', a aplicação de uma regra contribui para a sua própria aplicação por causa de uma dependência circular não resolvida. O conflito é resolvido gerando mais de uma extensão, mas não pela inconsistência, mas sim pela impossibilidade de resolver o conflito. No exemplo 6.2.3, A depende da ausência de B e B depende da ausência de A. Se inferirmos A bloquearmos B e se inferirmos B bloquearmos A. Logo, não é possível dar nenhuma preferência para A ou B.

O método de Etherington será importante para a nossa abordagem porque através dele vamos identificar as teorias estratificadas e também garantir a existência de extensões. As seguintes definições caracterizam o método sintático de Etherington.

Definição 6.2.1. *(ETHERINGTON, 1987)*

Seja $\Delta = (D, W)$ uma teoria default seminormal fechada. Sem perda de generalidade, assumimos que todas as fórmulas estão na forma clausal. A relação parcial, $\leq$ e $<$, entre os literais, são definidos da seguinte maneira:

i. *Se $\alpha \in W$, então $\alpha = \alpha_1, \ldots, \alpha_n$, para algum $n \geq 1$. Para todos $\alpha_i, \alpha_j \in \{\alpha_1, \ldots, \alpha_n\}$, se $\alpha_i \neq \alpha_j$, então $\neg\alpha_i \leq \alpha_j$.*

ii. *Se $\delta \in D$, então $\delta = \frac{\alpha : \beta \wedge \gamma}{\beta}$. Seja $\alpha_1, \ldots, \alpha_r$, $\beta_1, \ldots, \beta_s$ e $\gamma_1, \ldots, \gamma_t$ literais da forma clausal de α, β, e γ. Então:*

- *(a) Se $\alpha_i \in \{\alpha_1, \ldots, \alpha_r\}$ e $\beta_j \in \{\beta_1, \ldots, \beta_s\}$, então $\alpha_i \leq \beta_j$.*
- *(b) Se $\gamma_i \in \{\gamma_1, \ldots, \gamma_t\}$,$\beta_j \in \{\beta_1, \ldots, \beta_s\}$ e $\gamma_i \notin \{\beta_1, \ldots, \beta_s\}$, então $\neg\gamma_i < \beta_j$.*
- *(c) $\beta = \beta_1 \wedge \ldots \wedge \beta_m$, para algum $m \geq 1$. Para cada $i \leq m$, $\beta_i = (\beta_{i.1}, \ldots, \beta_{i.m_i})$, onde $m_i \geq 1$. Assim se $\beta_{i.j}, \beta_{i.k} \in \{\beta_{i.1}, \ldots, \beta_{m.m_m}\}$ e $\beta_{i.j} \neq \beta_{i.k}$, então $\neg\beta_{i.j} \leq \beta_{i.k}$*

iii. *O relacionamento transitivo para* $<$ *and* $\leq$*:*

(a) *se* $\alpha \leq \beta$ *e* $\beta \leq \gamma$*, então* $\alpha \leq \gamma$*.*

(b) *se* $\alpha < \beta$ *e* $\beta < \gamma$*, então* $\alpha < \gamma$*.*

(c) *se* $\alpha \leq \beta$ *e* $\beta < \gamma$ *ou se* $\alpha < \beta$ *e* $\beta \leq \gamma$*, então* $\alpha < \gamma$*.*

Embora a definição seja complexa, a intuição para $\alpha \leq \beta$ e $\alpha < \beta$ é se existe um maneira de α participar de alguma inferência de β na teoria default. A intuição por trás da parte (1) e (2c) é que uma disjunção de n literais pode ser interpretado como uma implicação em um dos literais. Por exemplo,

$$(\alpha_1, \ldots, \alpha_n) \equiv [(\neg\alpha_1 \wedge \ldots \wedge \neg\alpha_j \wedge \neg\alpha_{j+1} \wedge \ldots \wedge \neg\alpha_n) \rightarrow \alpha_j].$$

Na parte(2b), a conjunção de uma justificativa não deve ser implicada pela sua conclusão isto é refletido pela relação $<$. Na parte (2b), a negação $\neg\gamma_i$ aparece uma vez que não acreditar γ_i tornar γ_i consistente.

Note que aqui ele está estabelecendo uma ordem entre os literais que serão importante para definir uma relação de relevância entre as regras.

Nas definições seguintes, nós vamos assumir que as fórmulas estão na forma clausal, isto é, como um conjunção de disjunções de literais. Nós vamos definir as funções CLAUSULAS(.) e LITERAIS(.) da seguinte maneira:

$$\beta = (\beta_{1.1} \vee \ldots \vee \beta_{1.m_1}) \wedge \ldots \wedge (\beta_{m.1} \vee \ldots \vee \beta_{m.m_1}),$$

então
CLAUSULA(β) = $\{(\beta_{i.1} \vee \ldots \vee \beta_{i.m_i}) | 1 \leq i \leq m\}$,
LITERAIS(β) = $\{\beta_{i.j} | 1 \leq i \leq m, 1 \leq j \leq m_i\}$.

Definição 6.2.2. *Para uma teoria default seminormal,* $\Delta = (D, W)$ *e para quaisquer dois defaults* $d_1, d_2 \in D$*,*

$$d_1 = \frac{\alpha_1 : \gamma_1 \wedge \beta_1}{\gamma_1} < d_2 = \frac{\alpha_2 : \gamma_2 \wedge \beta_2}{\gamma_2} \; sse$$

existe $\gamma \in LITERAIS(\gamma_1)$ *e* $\gamma' \in LITERAIS(\gamma_2)$ *tal que*

$$\gamma <_d \gamma'.$$

Uma regra default que deriva a negação de uma justificativa de uma outra regra default, é estritamente menor do que ela. Note que a extensão que será calculada seguindo a ordem $<$ será uma seqüência de aplicações seguras das regras, ou seja, uma regra que pode ser aplicada não irá derivar a exceção (a negação da justificativa) de uma outra regra já aplicada.

Definição 6.2.3. *Uma teoria default seminormal, $\Delta = (D, W)$, é localmente estratificada sse $<_d$ é uma relação de ordem parcial estrita bem-fundada em D.*

Exemplo 6.2.5. *A teoria default $\Delta = (W, D)$:*
$W = \emptyset$
$D = \{d_1 = \frac{:A \wedge \neg B}{A}, d_2 = \frac{:B \wedge \neg A}{B}\}$
Os literais têm os seguintes relacionamentos:
$\{B < A\}, \{A < B\}$.
Logo, temos que $d_1 < d_2 < d_1$.
A relação de ordem entre os default $<$ não é estrita.

Exemplo 6.2.6. *A teoria default $\Delta = (W, D)$:*
$W = \emptyset$
$D = \{d_1 = \frac{:A \wedge \neg B}{A}, d_2 = \frac{:B \wedge \neg C}{B}, d_3 = \frac{:C \wedge \neg A}{C}\}$
Os literais têm os seguintes relacionamentos:
$\{B < A\}, \{C < B\}, \{A < C\}$.
Logo, temos que $d_1 < d_3 < d_2 < d_1$.
A relação de ordem entre os default $<$ não é estrita.

Exemplo 6.2.7. *A teoria default $\Delta = (W, D)$:*
$W = \emptyset$
$D = \{d_1 = \frac{:A \wedge \neg B}{A}, d_2 = \frac{:B \wedge \neg D}{B}, \frac{d_3 =: C \rightarrow D \wedge \neg A}{C \rightarrow D}\}$
não é estratificada. Os literais têm os seguintes relacionamentos:
$\{B < A\}, \{D < B\}, \{C \leq D, \neg D \leq \neg C, A < \neg C, A < D\}$.
Então temos que $d_1 < d_3 < d_2 < d_1$.

Importante notar o fato que a teoria ser estratificada é somente uma condição suficiente para a existência de extensões. Teorias não estratificadas podem ter circularidade potencialmente não resolvidas mas, por alguma razão, estas

circularidade nem sempre interferem. A teoria do exemplo 6.2.7 não é estratificada, mas tem um extensão $Th(\{B, C \rightarrow D\})$. A circularidade poderia causar problemas, se C fosse adicionado em W e assim teoria resultante não teria extensões. Acreditamos que existam condições mais fortes para identificação de circularidade potencialmente não resolvida para teorias seminormais mas possivelmente não serão decidíveis

Embora David Etherington (ETHERINGTON, 1987) tenha proposto o conceito de teorias estratificadas(evitando a circularidade) para resolver o problema da coerência - ele nada propôs para o problema das extensões anômalas. Vale ressaltar que o problema das extensões anômalas se manifesta mesmo em teorias que são estratificadas segundo os critérios de Etherington. O problema aqui não é a falta, mas o excesso de extensões.

A *Well-Behaved IDL*(PEQUENO; MARTINS; PEQUENO, 1996) define um critério de boa formação para as teorias IDL através de uma adaptação do método sintático proposto por Etherington para identificar as circularidade indesejáveis.

Na teoria do exemplo 6.2.1 traduzida para IDL gera apenas uma extensão que comporta as duas crenças mas é não inconsistente. Na teoria do exemplo 6.2.3 traduzido para IDL gera duas extensões, cada extensão suportando uma das crenças. Na Well-Behaved IDL, a teoria do exemplo 6.2.3 será considerada cíclica e será desconsiderada. Um resultado importante pode ser estabelecido em Well-behaved IDL, se a teoria é ordenada (ou seja, sem ciclos) então ela tem uma única extensão IDL. Em (MARTINS, 1997), uma caracterização construtiva equivalente para a extensão Well-Behaved IDL é apresentada baseada na ordem entre os defaults de IDL. A well-behaved IDL ataca os dois problemas da lógica default : problema da coerência através da adaptação do método de Etherington de identificação de teorias cíclicas para IDL e o problema das extensões anômalas através da diferenciação da parte normal e não normal da justificativa do default associado com a lógica paraconsistente LEI.

A nossa abordagem vai resolver os dois problemas principais da lógica default por um outro caminho sem lançar mão de uma lógica paraconsistente e continuando utilizando a linguagem original da lógica default. Vamos definir as extensões da lógica default predicativa através de uma boa ordem $\preceq$ entre os default. Método proposto por Marek e Truszczsinki(MAREK; TRUSZCZYNSKI, 1997) e utilizado por Choleswinski (CHOLEWINSKI, 1995a, 1995b, 1996).

6.3 Lógica Default Estratificada

Choleswinski (CHOLEWINSKI, 1995a, 1995b, 1996) propõe o que ele chama de *Stratified Default Logic*. Novamente a intenção não foi resolver o problema das extensões anômalas, mas propor métodos eficientes para a computação de extensões de Reiter. Aliás, a proposta deles de estratificação é bastante precária do ponto de vista do conjunto a ser gerado por uma teoria default uma vez que deixa de fora teorias com conjuntos gerados perfeitamente bem definidos (teorias ordenadas).

Nesta seção, vamos adotar as seguintes notações: para uma fórmula fechada φ , o conjunto de átomos de $\mathcal{L}$ presentes em φ denotaremos por Var(φ).

A seguir, a definição de estratificação proposta por Choleswinski:

Definição 6.3.1. *Seja D um conjunto de default seminormais. A função rank que associa um ordinal a todo default de D é uma função estratificação para D se para todo d,d' ∈ D, onde*

$$d = \frac{\alpha : \beta \wedge \gamma}{\gamma} \text{ e } d' = \frac{\alpha' : \beta' \wedge \gamma'}{\gamma'}$$

as seguintes três condições devem ser satisfeitas:

i. *Se Var(γ) ∩ Var(γ') ≠ ∅ então rank(d) = rank(d'),e*

ii. *Se Var(β) ∩ Var(γ') ≠ ∅ então rank(d) ≥ rank(d'),e*

iii. *Se Var(α) ∩ Var(γ') ≠ ∅ então rank(d) ≥ rank(d').*

A estratificação é chamada forte se rank(d) > rank(d'), se Var(β) ∩ Var(γ') ≠ ∅.

Definição 6.3.2. *Uma teoria default seminormal Δ=(D,W) é (fortemente) estratificada se*

i. *W é consistente, e*

ii. *Var(W) ∩ Var(c(D)) = ∅, e*

iii. *existe uma função de estratificação (forte) para D.*

Exemplo 6.3.1. *Seja Δ=(D,W) uma teoria default, onde $W = \emptyset$ e $D = \{d_1, d_2\}$, onde*

$$d_1 = \frac{: q \wedge \neg p}{q} \text{ e } d_2 = \frac{: \neg p \wedge q}{\neg p}.$$

Não existe uma função de estratificação forte para D, pela condição (ii) da definição 6.3.1 implica que rank(d_1) < rank(d_2) e rank(d_2) < rank(d_1). Note que a teoria é ordenada, mas não admite uma estratificação forte.

A extensão para uma teoria fortemente estratificada é definida através da boa ordem $\preceq$ compatível com a função de estratificação rank, ou seja, uma extensão linear da ordem definida pela função rank entre os default. O conjunto será definido utilizando o método proposto por Marek e Truczszsinski em (MAREK; TRUSZCZYNSKI, 1997).

Definição 6.3.3. *Seja Δ=(W,D) uma teoria default seminormal fortemente estratificada e $\preceq$ uma extensão linear da ordem dos defaults definida pela função rank*

Vamos construir iterativamente o conjunto de defaults gerado pela boa ordem $\preceq$, $GR_{\preceq}$, da seguinte maneira:

Para todo ordinal $0 \leq \alpha \leq \eta$:

i. *Se α=0 então $GR_0 = \emptyset$*

ii. *caso contrário, seja $d_\alpha = \frac{\alpha : \gamma \wedge \beta}{\gamma}$ o $\preceq$-menor default do conjunto $D \backslash (\bigcup_{\xi<\alpha} GR_\xi)$ aplicável, ou seja,*
$W \cup \boldsymbol{Cons}(\bigcup_{\xi<\alpha} GR_\xi) \vdash \alpha$ e
$W \cup \boldsymbol{Cons}(\bigcup_{\xi<\alpha} GR_\xi) \nvdash \neg(\gamma \wedge \beta)$
então $GR_\alpha = \bigcup_{\xi<\alpha} GR_\xi \cup \{d_\alpha\}$

iii. *caso contrário, não existe default $d_\alpha = \frac{\alpha : \gamma \wedge \beta}{\gamma}$ no conjunto $D \backslash (\bigcup_{\xi<\alpha} GR_\xi)$ aplicável, ou seja,*
$W \cup \boldsymbol{Cons}(\bigcup_{\xi<\alpha} GR_\xi) \vdash \alpha$ e
$W \cup \boldsymbol{Cons}(\bigcup_{\xi<\alpha} GR_\xi) \nvdash \neg(\gamma \wedge \beta)$,
então a construção pára e $\eta = \alpha$ e $GR_{\preceq} = \bigcup_{\xi<\alpha} GR_\xi$.

A teoria $Th(W \cup \boldsymbol{Cons}(GR_\eta))$ *é chamada de teoria gerada pela boa ordem* $\preceq$

Um resultado importante é estabelecido por Choleswinski para as teorias default seminormais fortemente estratificadas através do seguinte teorema:

Teorema 6.3.1. *Seja (D,W) uma teoria default seminormal fortemente estratificada e* $\preceq$ *uma boa ordem de D compatível com a estratificação forte rank então a seguinte condição*

$$\textit{para todo } d = \frac{\alpha : \gamma \wedge \gamma}{\gamma} \in GR_\preceq \textit{ temos que } W \cup \boldsymbol{Cons}(GR_\preceq) \nvdash \neg(\beta \wedge \gamma).$$

é valida e $Th(W \cup \boldsymbol{Cons}(GR_\preceq)$ *é uma extensão para (D,W).*

A existência de extensões para uma teoria default seminormal fortemente estratificada segue diretamente do teorema anterior. Outra conseqüência imediata do teorema anterior é a seguinte:

Corolário 6.3.1. *Seja (D,W) uma teoria default seminormal fortemente estratificada então (D,W) tem uma extensão. Para cada* $\preceq$ *uma boa ordem de D compatível com a estratificação forte rank temos que*

$$S = Th(W \cup \boldsymbol{Cons}(GR_\preceq))$$

é uma extensão para (D,W).

Ele demonstrou ainda que toda extensão de uma teoria seminormal fortemente estratificada pode ser gerada por boa ordem $\preceq$ compatível com função de estratificação forte rank. Realmente, Choleswinski não define uma nova lógica default mas sim métodos eficazes de cálculo de extensões para certa classe de teorias default.

6.4 Lógica Default Localmente Estratificada

A *Lógica Default Seminormal Localmente Estratificada* é uma restrição da lógica default de Reiter que ataca simultaneamente os dois problemas principais da lógica default: o problema da coerência e o problema das extensões anômalas em teorias defaults.

Acreditamos que os dois problemas citados têm a sua origem na *circularidade das teorias*. Podemos resolver os dois problemas utilizando a ordem parcial $<_d$ entre os default D de uma teoria (W,D) localmente estratificada para definir as extensões dessa lógica default.

A extensão para uma teoria default localmente estratificada é definida através da boa ordem $\preceq$ compatível com a ordem parcial $<_d$, ou seja, uma extensão linear dessa ordem. O conjunto será definido utilizando o método proposto por Marek e Truczszsinski em (MAREK; TRUSZCZYNSKI, 1997) e utilizado por Choleswinski.

Definição 6.4.1. *Seja Δ=(W,D) uma teoria default seminormal localmente estratificada e $\preceq$ é uma boa ordem obtida através de uma extensão linear da ordem parcial $<_d$ entre os default.*

Vamos construir iterativamente o conjunto de defaults gerado pela boa ordem $\preceq$, $GR_{\preceq}$, da seguinte maneira:

Para todo ordinal $0 \leq \alpha \leq \eta$:

i. *Se α=0 então $GR_0 = \emptyset$*

ii. *caso contrário, seja $d_\alpha = \dfrac{\alpha : \gamma \wedge \beta}{\gamma}$ o $\preceq$-menor default do conjunto $D \backslash (\bigcup_{\xi<\alpha} GR_\xi)$ aplicável, ou seja,*
$W \cup \boldsymbol{Cons}(\bigcup_{\xi<\alpha} GR_\xi) \vdash \alpha$ e
$W \cup \boldsymbol{Cons}(\bigcup_{\xi<\alpha} GR_\xi) \nvdash \neg(\gamma \wedge \beta)$
então $GR_\alpha = \bigcup_{\xi<\alpha} GR_\xi \cup \{d_\alpha\}$

iii. *caso contrário, não existe default $d_\alpha = \dfrac{\alpha : \gamma \wedge \beta}{\gamma}$ no conjunto $D \backslash (\bigcup_{\xi<\alpha} GR_\xi)$ aplicável, ou seja,*
$W \cup \boldsymbol{Cons}(\bigcup_{\xi<\alpha} GR_\xi) \vdash \alpha$ e
$W \cup \boldsymbol{Cons}(\bigcup_{\xi<\alpha} GR_\xi) \nvdash \neg(\gamma \wedge \beta)$,
então a construção pára e $\eta = \alpha$ e $GR_{\preceq} = \bigcup_{\xi<\alpha} GR_\xi$.

A teoria $Th(W \cup \boldsymbol{Cons}(GR_\eta))$ é chamada de teoria gerada pela boa ordem $\preceq$

Um resultado importante será estabelecido nesta dissertação para as teorias default seminormais localmente estratificadas através do seguinte teorema:

Teorema 6.4.1. *Seja (D,W) uma teoria default seminormal fortemente estratificada e $\preceq$ uma boa ordem de D compatível com a ordem parcial $<_d$ então a seguinte condição*

$$\textit{para todo } d = \frac{\alpha : \gamma \wedge \gamma}{\gamma} \in GR_{\preceq} \textit{ temos que } W \cup \boldsymbol{Cons}(GR_{\preceq}) \nvdash \neg(\beta \wedge \gamma).$$

é valida e Th($W \cup \boldsymbol{Cons}(GR_{\preceq})$ é uma extensão de Reiter para (D,W).

A existência de extensões para uma teoria default seminormal fortemente estratificada segue diretamente do teorema anterior. Obtermos também como conseqüência imediata da nossa abordagem o seguinte corolário.

Corolário 6.4.1. *Seja (D,W) uma teoria default seminormal fortemente estratificada então (D,W) tem uma extensão. Para cada $\preceq$ uma boa ordem de D compatível com a estratificação forte rank temos que*

$$S = Th(W \cup \boldsymbol{Cons}(GR_{\preceq}))$$

é uma extensão para (D,W).

A "'volta"' do teorema não será válida na lógica default localmente estratificada, ou seja, nem **todas** as extensões de Reiter serão geradas por uma boa ordem $\preceq$ compatível com a ordem parcial $<_d$ entre os default. As extensões de Reiter que não são geradas por uma boa ordem $\preceq$ compatível com $<_d$ são as extensões anômalas. Com isso, estamos propondo uma variante da lógica default que não tem o problema das extensões anômalas.

6.5 Problemas das Extensões Anômalas

O problema das extensões anômalas foi primeiramente apontado por Hanks e McDermott (HANKS; MCDERMOTT, 1987) através do famoso *Yale Shooting Problem*. Com este problema, eles queriam mostrar o problema no uso de lógicas não-monotônicas na formalização de problemas em domínios temporais. Ele verificou o surgimento de uma seqüência de eventos considerada anômala.

Morris(MORRIS, 1988) e Marcelino Pequeno(PEQUENO, 1994) apresentam problemas que não envolvem domínios temporais, mas que tem o mesmo padrão do Yale Shooting Problem e geram resultados indesejados. Vale ressaltar que o

Computa_Extensão_$\Delta(\Delta, \preceq)$
Entrada: Uma teoria default seminormal Δ localmente estratificada e uma boa ordem $\preceq$ obtida através de uma extensão linear da ordem parcial $<_d$;
Saída: A extensão gerado pela boa ordem $\preceq$.
A := W
R := $\emptyset$

faça
r:= o $\preceq$-menor default d do conjunto $D \backslash R$ aplicável em A
R := R $\cup$ {r}
enquanto(r $\neq \emptyset$)

retorne Th(W $\cup$ **Cons**(R));

problema das extensões anômalas se manisfeta mesmo em teorias que são ordenadas segundo o critério de Etherington. Marcelino Pequeno(PEQUENO, 1994) afirma que a razão para o comportamento anômalo está na não priorização da derivação da exceção (negação das justificativas) das regras default. Na dissertação, sugerimos que este problema pode ser resolvido para teorias estratificadas modificando a maneira como as extensões são calculadas. A seguir, alguns exemplos de teorias default localmente estratificadas com o problema das extensões anômalas.

Exemplo 6.5.1. *Considere a seguinte exemplo de teoria default(MORRIS, 1988):*

i. $BIRD(Tweedy)$

ii. $BIRD(Tweety) \rightarrow ANIMAL(Tweety)$

iii. $WING(Tweety) \rightarrow FLY(Tweety)$

iv. $\dfrac{ANIMAL(Tweety) : \neg FLY(Tweety) \wedge \neg WING(Tweety)}{\neg FLY(Tweety)}$

v. $\dfrac{BIRD(Tweety) : WING(Tweety)}{WING(Tweety)}$

tem duas extensões, mas uma delas é anômala:
$E_1 = Th(W \cup \{WING(Tweety)\})$
$E_2 = Th(W \cup \{\neg FLY(Tweety)\}$
Na primeira extensão, a partir (1) e (5), nós temos WING(Tweety) e com (3) nós

temos FLY(Tweety) que bloqueia (4). Portanto, temos uma extensão que Tweety é um pássaro com asas e voa. Na segunda extensão, a partir (1) e (2), nós temos ANIMAL(Tweety). Aplicando (4), nós temos ¬ FLY(Tweety) e pela contrapositiva de (3), nós temos ¬ WING(Tweety). Isto bloqueia a aplicação de (5). Portanto, esta extensão é anômala onde Tweety é um pássaro sem asas e não voa.

Os seguintes relacionamentos entre os literais desta teoria:

- $BIRD(Tweety) \leq ANIMAL(Tweety)$
- $\neg ANIMAL(Tweety) \leq \neg BIRD(Tweety)$
- $WING(Tweety) \leq FLY(Tweety)$
- $\neg FLY(Tweety) \leq \neg WING(Tweety)$
- $ANIMAL(Tweety) \leq \neg FLY(Tweety)$
- $WING(Tweety) < \neg FLY(Tweety)$
- $BIRD(Tweety) \leq WING(Tweety)$

Pela transitividade:

- $WING(Tweety) < \neg WING(Tweety)$

Temos o seguinte relacionamentos entre os default:

$$\frac{BIRD(Tweety):WING(Tweety)}{WING(Tweety)} <_d \frac{ANIMAL(Tweety):\neg FLY(Tweety)\wedge\neg WING(Tweety)}{\neg FLY(Tweety)}$$

A teoria é localmente estratificada e temos apenas uma boa ordem $\preceq$ compatível com a ordem $<_d$ entre os defaults:

$$\frac{BIRD(Tweety):WING(Tweety)}{WING(Tweety)} \preceq \frac{ANIMAL(Tweety):\neg FLY(Tweety)\wedge\neg WING(Tweety)}{\neg FLY(Tweety)}$$

$$GR_{\preceq} = \left\{ \frac{BIRD(Tweety) : WING(Tweety)}{WING(Tweety)} \right\}$$

A única extensão é: $E_1 = Th(W \cup cons(GR_{\eta_{\preceq}}))$

Note que a extensão anômala é eliminada.

Exemplo 6.5.2. *Considere o seguinte exemplo de teoria default* Δ*=(W,D)(PEQUENO; VERAS; TAVARES, 2007):*
O símbolo proposicional P representa pingüim e V representa voa.
$W = \emptyset$
$D = \{\frac{:V \wedge \neg P}{V}, \frac{:P}{P}, \frac{:P \rightarrow \neg V}{P \rightarrow \neg V}\}$
tem duas extensões, mas uma delas é anômala:
$E_1 = \{P, P \leftarrow \neg V\}$
$E_2 = \{V, P \leftarrow \neg V\}$
Na primeira extensão, concluímos que o animal é um pingüim e não voa. Na segunda extensão, concluirmos que o animal voa e pela contrapositiva que ele não é pingüim. A segunda extensão é considerada anômala.

Os literais têm os seguintes relacionamentos:
$\{P < V\}, \{P \leq \neg V, V \leq \neg P\}$.
e pela transitividade $\{P < \neg P\}$.

Temos o seguinte relacionamentos entre os default:

- $\frac{:P}{P} <_d \frac{:P \rightarrow \neg V}{P \rightarrow \neg V}$
- $\frac{:P}{P} <_d \frac{:V \wedge \neg P}{V}$

A teoria Δ *é localmente estratificada.*

Qualquer boa ordem $\preceq$ *compatível com* $<_d$ *pode ser utilizada para calcular a extensão:*

$\frac{:P}{P} \preceq \frac{:P \rightarrow \neg V}{P \rightarrow \neg V} \preceq \frac{:V \wedge \neg P}{V}$

$GR_0 = \emptyset$

$GR_1 = GR_0 \cup \{\frac{:P}{P}\}$

$GR_2 = GR_1 \cup \{\frac{:P \rightarrow \neg V}{P \rightarrow \neg V}\}$

A construção pára e temos:

$$GR_{\eta_{\preceq}} = \left\{ \frac{: P}{P}, \frac{: P \rightarrow \neg V}{P \rightarrow \neg V} \right\}$$

A única extensão é: $E_1 = Th(W \cup cons(GR_{\eta_{\preceq}}))$

Note que a extensão anômala é eliminada.

Considere a seguinte base de conhecimento:

- Animais (An) não voam(Vo) a não ser que sejam Pássaros (Pa);
- Pássaros voam a não ser que sejam pingüins (Pg);
- Animais de bicos (Bi) são pássaros a não ser que sejam ornitorrincos (On);
- Animais de bico são animais.

A questão é: dado que temos um animal de bico, o que esperamos deste animal, que ele voe ou não?

Exemplo 6.5.3. *Considere a seguinte teoria default representando a base de conhecimento:*

$W = \{Bi, Bi \rightarrow An\}$

$$D = \left\{ \frac{: An \rightarrow \neg Vo \wedge \neg Pa}{An \rightarrow \neg Vo}, \frac{: Pa \rightarrow Vo \wedge \neg Pg}{Pa \rightarrow Vo}, \frac{: Bi \rightarrow Pa \wedge \neg On}{Bi \rightarrow Pa} \right\}$$

A teoria default tem duas extensões, mas uma delas é anômala:

$E_1 = Th(W \cup \{Pa \rightarrow Vo, Bi \rightarrow Pa\})$

$E_2 = Th(W \cup \{An \rightarrow \neg Vo, Pa \rightarrow Vo\})$

Na primeira extensão, temos que o animal tem bico, é um pássaro e voa. Na segunda extensão, temos que animal tem bico, mas não voa e não é um pássaro. Esta extensão é anômala.

Os literais têm os seguintes relacionamentos:

$\{An \leq \neg Vo, Vo \leq \neg An, Pa \leq Vo, \neg Vo \leq \neg Pa, Bi \leq Pa, \neg Pa \leq \neg Bi\}$,

$\{Pa < \neg An, Pa < \neg Vo, Pg < \neg Pa, Pg < Vo, On < \neg Bi, On < Pa, Pa < \neg Pa, \neg Pa < \neg Bi\}$

Temos o seguinte relacionamentos entre os default:

- $\frac{:Bi \rightarrow Pa \wedge \neg On}{Bi \rightarrow Pa} <_d \frac{:An \rightarrow \neg Vo \wedge \neg Pa}{An \rightarrow \neg Vo}$

A teoria é localmente estratificada.

Qualquer boa ordem $\preceq$ compatível com $<_d$ pode ser utilizada para calcular a extensão:

$\frac{:Bi \rightarrow Pa \wedge \neg On}{Bi \rightarrow Pa} \preceq \frac{:An \rightarrow \neg Vo \wedge \neg Pa}{An \rightarrow \neg Vo} \preceq \frac{:Pa \rightarrow Vo \wedge \neg Pg}{Pa \rightarrow Vo}$

$GR_0 = \emptyset$

$GR_1 = GR_0 \cup \{\frac{:Bi \rightarrow Pa \wedge \neg On}{Bi \rightarrow Pa}\}$

$GR_2 = GR_1 \cup \{\frac{:Pa \rightarrow Vo \wedge \neg Pg}{Pa \rightarrow Vo}\}$

A construção pára e temos:

$$GR_{\eta_\preceq} = \left\{ \frac{: Bi \rightarrow Pa \wedge \neg On}{Bi \rightarrow \neg Pa}, \frac{: Pa \rightarrow Vo \wedge \neg Pg}{Pa \rightarrow Vo} \right\}$$

A única extensão é: $E_1 = Th(W \cup cons(GR_{\eta_\preceq}))$

Note que a extensão anômala é eliminada.

Exemplo 6.5.4. *Considere a seguinte exemplo de teoria default(MORRIS, 1988):*

i. *Holds(Alive, S_0)*

ii. *Holds(Loaded , res(Load, s))*

iii. *Holds(Loaded, s) $\rightarrow \neg$ Holds(Alive,res(Shoot,s))*

iv. *Holds(Loaded, s) $\rightarrow$ Ab(Alive,Shoot,s)*

v. $\frac{Holds(f,s) : Holds(f, res(e,s)) \wedge \neg Ab(f,e,s)}{Holds(F, res(e,s))}$

$S_1 = res(Load, S_0)$,$S_2 = res(Wait, S_1)$ *e* $S_3 = res(Shoot, S_2)$

Considere a instanciação de alguns fatos e default:

(1) $Holds(Alive, S_0)$

(2') $Holds(Loaded, S_1)$

(3') $Holds(Loaded, S_2) \rightarrow \neg Holds(Alive, res(Shoot, S_2))$

(4') $Holds(Loaded, S_2) \rightarrow Ab(Alive, Shoot, S_2)$

(5) $$\frac{Holds(Alive, S_0) : Holds(Alive, S_1) \wedge \neg Ab(Alive, Load, S_0)}{Holds(Alive, S_1)}$$

(5') $$\frac{Holds(Loaded, S_1) : Holds(Loaded, S_2) \wedge \neg Ab(Loaded, Wait, S_1)}{Holds(Loaded, S_2)}$$

(5") $$\frac{Holds(Alive, S_2) : Holds(Alive, S_3) \wedge \neg Ab(Alive, Shoot, S_2)}{Holds(Alive, S_3)}$$

(5"') $$\frac{Holds(Alive, S_1) : Holds(Alive, S_2) \wedge \neg Ab(Alive, Wait, S_1)}{Holds(Alive, S_2)}$$

Note que esta teoria é ordenada e tem duas extensões, mas uma delas é anômala:

A extensão intuitiva começa a partir (1) aplicando (5) obtendo
(6) $Holds(Alive, S_1)$. *A partir (2') e (5') nós temos*
(7) $Holds(Loaded, S_2)$ *e a partir (6) e (5"') nós temos*
(8) $Holds(Alive, S_2)$. *A partir (7) e (4') nós temos*
(9) $Ab(Alive, Shoot, S_2)$ *que bloqueia a aplicação de (5"), e a partir (7) e (3') nós temos (10)* $\neg Holds(Alive, S_3)$.

A extensão anômala: nós similarmente obtemos (6) e (2'). A partir de (6) aplicando (5"') nós obtemos também (8); a partir de (8) e (5") nós obtemos (11) $Holds(Alive, S_3)$. *A partir (11) e (3') pela contrapositiva obtemos (12)* $\neg Holds(Loaded, res(Wait, S_1))$. *Então a arma torna-se misteriosamente descarregada durante o evento Wait. Então agora (5') não pode ser mais aplicada.*

Temos os seguintes default:

- $d_1 = \frac{Holds(Alive,S_0):Holds(Alive,S_1)\wedge\neg Ab(Alive,Load,S_0)}{Holds(Alive,S_1)}$
- $d_2 = \frac{Holds(Loaded,S_1):Holds(Loaded,S_2)\wedge\neg Ab(Loaded,Wait,S_1)}{Holds(Loaded,S_2)}$
- $d_3 = \frac{Holds(Alive,S_1):Holds(Alive,S_2)\wedge\neg Ab(Alive,Wait,S_1)}{Holds(Alive,S_2)}$
- $d_4 = \frac{Holds(Alive,S_2):Holds(Alive,S_3)\wedge\neg Ab(Alive,Shoot,S_2)}{Holds(Alive,S_3)}$

Temos os seguinte relacionamentos entre os defaults:

- $d_2 <_d d_4$

Qualquer boa ordem $\preceq$ compatível com $<_d$ pode ser utilizada para calcular a extensão. Vamos utilizar a seguinte boa ordem $\preceq$:

$d_2 \preceq d_4 \preceq d_3 \preceq d_1$

$GR_0 = \emptyset$

$GR_1 = GR_0 \cup \{d_2\}$

$GR_2 = GR_1 \cup \{d_1\}$

$GR_3 = GR_2 \cup \{d_3\}$

A construção pará e temos:

$GR_{\eta_\preceq} = \{d_1, d_2, d_3\}$

A única extensão é: $E_1 = Th(W \cup cons(GR_{\eta_\preceq}))$

Note que a extensão anômala é eliminada.

Capítulo 7

Conclusão e Trabalhos Futuros

Na dissertação perseguimos o problema de como determinar o conjunto gerado por um sistema formal avançado (SFA). Fomos guiados por dois princípios: gerar o conjunto a partir do conjunto vazio, de preferência iterativamente, e evitar o problema da circularidade.

Estes dois princípios nos levaram à noção de estratificação que já havia sido intensamente estudada em programação lógica. Nós propomos a noção de estratificação relevante que generaliza as diversas noções de estratificação apresentadas em programação lógica (estratificação, estratificação local, e estratificação fraca).

De posse da estratificação, calculamos então, a partir do vazio e iterativamente, o conjunto gerado pelo SFA, segundo uma técnica desenvolvida em (MAREK; TRUSZCZYNSKI, 1997), utilizando-se de uma boa ordenação que respeite a estratificação. O método de calcular o conjunto gerado por boa ordenação foi escolhido por permitir se calcular o conjunto gerado tanto em sistemas formais elementares (SFE), em sistemas formais avançados (SFA), como também em lógica não monotônica, sendo assim uma técnica uniforme utilizada em toda a dissertação.

O conjunto gerado por estratificação goza de diversas propriedades comuns aos conjuntos gerados dos SFE's: é um conjunto fechado minimal; é um ponto fixo minimal; ele é calculado iterativamente e cumulativamente, etc.

A grande contribuição da dissertação está em generalizar o conceito de estratificação para o domínio das lógicas não monotônicas. Como a estratificação relevante não generaliza para este domínio de linguagem mais expressiva, lançamos

mão de uma ordenação proposta em (ETHERINGTON, 1987), que equivale a uma estratificação local. Novamente embebemos a estratificação em uma boa ordenação e calculamos as extensões em lógica não monotônica utilizando o método de Marek e Truszczynski, e criamos assim, a *Lógica Default Estratificada.*

A Lógica Default Estratificada resolve, como mostramos, dois dos principais problemas relacionados às lógicas não monotônicas: a existência de extensões para teorias defaults e a não geração de extensões anômalas.

Ao aplicarmos o conceito de estratificação para o domínio mais amplo das lógicas não monotônicas obtendo resultados significativos reforçamos a idéia de que evitar circularidade seja fundamental no uso de inferências a partir da ausência de informações. O conjunto geral por estratificação é consensualmente aceito em programação lógica, e também o é, em lógica não monotônica, as extensões geradas em *lógica default estratificada* são um subconjunto das extensões da lógica default.

Entretanto, alguns autores em programação lógica, na realidade a grande maioria, têm proposto métodos para calcular o conjunto gerado por programas não estratificados. No capítulo 4 da dissertação fizemos um apanhado das diversas semânticas propostas, as mais difundidas sendo a semântica bem-fundada e a semântica de modelos estáveis. Tais semânticas se têm a qualidade de coincidir com a semântica estratificada para programas estratificados, têm a deficiência de não generalizar para lógicas não monotônicas, ou se generalizam, não generalizam de maneira satisfatória, mostrando que estes métodos não traduzem princípios gerais para a inferência a partir da ausência de informações. Por exemplo, a semântica de modelos estáveis é uma particularização da lógica default de Reiter para a linguagem mais restrita da programação lógica. Se obtém bons resultados em programação lógica, não deixa de ser a mesma lógica default que apresenta os problemas apontados nesta dissertação.

Pretendemos realizar estudos mais avançados esta estratificação da lógica default e determinar quais são as propriedades e metas-propriedades dos sistemas lógicos não-monotônicos são válidos na lógica default estratificada.

Pretendemos ainda desenvolver um sistema de raciocínio default baseado na lógica default estratificada no estilo de outros sistemas já desenvolvidos como, por exemplo, **DeReS** (*Default Reasoning System*) desenvolvido por Cholewinski e seus colaboradores.

Exploraremos a conexão existente entre a lógica default estratificada e ASP e tentaremos mostrar que diversos resultados obtidos com a estratificação na lógica default podem ser transportados para ASP.

Realizaremos um estudo amplo sobre a complexidade e a expressibilidade da teorias default estratificadas. Mostrando em que classe de complexidade descritiva enquadra-se essa nova lógica.

Finalmente, este trabalho motiva a adoção definitiva do conceito de estratificação para as lógicas não-monotônicas e todas as implicações decorrentes deste fato. Marcando o início de uma nova visão dos sistemas formais não-monotônicos.

Apêndice A

Prova dos Teoremas

A.1 Provas do capítulo 4

Definição A.1.1. *(Sistema Formal Avançado)*

i. *Um regra r é uma tripla (x,X,Y) em A (se $x \in A$ e $\{X\} \cup \{Y\} \subseteq A$), onde x é a conclusão da regra denotado por c(r). X é conjunto das premissas positivas denotado por $p^+(r)$. Y é conjunto das premissas negativas denotado por $p^-(r)$. A regra pode ser escrita como $x \Leftarrow X;Y$. Podemos omitir "';"' quando o conjunto $p^-(r)$ for vazio. A regra de um SFA tem a seguinte leitura "'A partir da geração de todas as expressões de X e da não geração de nenhuma expressão de Y podemos gerar a expressão x"'.*

ii. *Um sistema formal avançado Φ é um conjunto de regras (x,X,Y).*

Definição A.1.2. *Seja Φ um SFE. Uma derivação $a_1, ..., a_n$ para a_n é minimal em Φ sse para todo $1 \leq i \leq n$, $a_1, ..., a_{i-1}, a_{i+1}, ..., an$ não é uma derivação de a_n em Φ .*

Definição A.1.3. *(Relevância em um SFE) Seja Φ um SFE. Dizemos que uma regra $r \in D$ é relevante para um átomo $a \in U$ sse r é usada em uma derivação minimal de a em Φ.*

Definição A.1.4. *(Relação de Relevância em um SFE) Seja Φ um SFE. Seja $\leq_R$ relação binária de ordem entre as regras. Seja $r_1 = x_1 \Leftarrow X_1$ e $r_2 = x_2 \Leftarrow X_2$ regras em Φ.*

$$r_1 \leq_R r_2, \textit{ se } r_1 \textit{ é relevante para algum } y_2 \in X_2 \textit{ no SFE } \Phi,$$

$\leq_R$ *é uma relação de relevância entre as regras de um SFE.*

Definição A.1.5. *(Transformação de SFA em SFE) Seja um SFA* Φ *um SFE correspondente denotado por trans(*Φ*), tal que:*

$$x \leftarrow X \in trans(\ \Phi\ ) \textit{ sse } x \leftarrow X\ ;\ Y \in \Phi.$$

Se uma regra $r \in \Phi$ *denotaremos por r' correspondente em trans(*Φ*).*

Definição A.1.6. *(Relevância em um SFA) Seja* Φ *um SFA, dizemos que uma regra* $r \in \Phi$ *é relevante em um SFA para um átomo a em* Φ *sse a regra correspondente r'* $\in$ *trans(*Φ*) é relevante para a no SFE trans(*Φ*).*

Definição A.1.7. *(Relação de relevância em um SFA) Seja* Φ *um SFA. Sejam r1 = x1* $\leftarrow$ *X1 ; Y1 e r2 = x2* $\leftarrow$ *X2 ; Y2 regras em* Φ *.*

$$\textit{Dizemos que r1} <_R \textit{ r2 sse r1 é relevante para y2} \in \textit{Y2.}$$

Um SFA é relevantemente estratificado se somente se a relação de ordem $<_R$ é uma ordem parcial estrita bem-fundada.

Vamos definir o conjunto gerado por SFA relevantemente estratificado atráves de um boa ordem $\preceq$ em Φ. A boa ordem $\preceq$ será uma extensão linear da ordem parcial estrita bem-fundada $<_R$. Qualquer boa ordem $\preceq$ que satisfaça essa propriedade poderá ser usada.

Vamos adotar a seguinte notação, dizemos que um conjunto Y pertence a um conjunto S, se para algum y $\in$ Y, y $\in$ S.

Definição A.1.8. *(Conjunto Gerado por SFA relevantemente estratificado) Seja* Φ=<*U,D*> *um SFA relevantemente estratificado. Seja* $\preceq$ *uma boa ordem em* Φ *compatível com* $<_R$*, ou seja, uma extensão linear da ordem* $<_R$*.*

Vamos construir iterativamente o conjunto de regras gerado pela boa ordem $\preceq$*,* $GR_{\preceq}$*, da seguinte maneira:*

i. *se* $\alpha = 0$ *então* $GR_\alpha = \emptyset$

ii. *caso contrário d_α é a $\preceq$-menor regra aplicável no conjunto $\Phi\backslash(\bigcup_{\xi<\alpha} GR_\xi)$, ou seja,*
$p^+(d_\alpha) \in c(\bigcup_{\xi<\alpha} GR_\xi)$
$p^-(d_\alpha) \notin c(\bigcup_{\xi<\alpha} GR_\xi)$
Então $GR_\alpha = (\bigcup_{\xi<\alpha} GR_\xi) \cup \{d_\alpha\}$

iii. *caso contrário, se não existe regra $d \in \Phi\backslash(\bigcup_{\xi<\alpha} GR_\xi)$ tal que*
$p^+(d) \in c(\bigcup_{\xi<\alpha} GR_\xi)$
$p^-(d) \notin c(\bigcup_{\xi<\alpha} GR_\xi)$
então a construção pára,$GR_{\eta_\preceq} = (\bigcup_{\xi<\eta} GR_\xi)$

O conjunto $c(GR_\preceq)$ é chamado conjunto gerado pela boa ordem $\preceq$.

Vamos adotar a seguinte notação $GR_{\alpha-1} = \bigcup_{\xi<\alpha} GR_\xi$

Lema A.1.1. *Seja Φ uma SFA relevantemente estratificado, Seja $\preceq$ uma extensão linear de $<_R$ para Φ e r_ξ e $r_\eta \in GR_\preceq$ são regras escolhidas no ξ-ésimo e η-ésimo passo então*

$$se\ r_\xi < r_\eta\ então\ \xi < \eta.$$

Seja $r_\xi = x_\xi \Leftarrow X_\xi; Y_\xi$ e $r_\eta = x_\eta \Leftarrow X_\eta; Y_\eta$. Suponha por absurdo que $\eta \leq \xi$. Sem perda de generalidade assuma que ξ é o primeiro ordinal maior que η tal que a regra escolhida ξ-ésimo passo é menor que a regra escolhida no η-ésimo passo. Pela definição de $GR_\preceq$, temos que

$$X_\xi \in c(GR_{\xi-1})\ e\ Y_\xi \notin c(GR_{\xi-1}).$$

e $GR_{<\eta} \subseteq GR_{<\xi}$.

Se a regra r_ξ não foi escolhida no passo η então $X_\xi \notin c(GR_{\eta-1})$ ou $Y_i \in c(GR_{\eta-1})$.

i. *Se $Y_\xi \in c(GR_{\eta-1})$. Pela definição de $GR_\preceq$, temos que $GR_{\eta-1} \subseteq GR_{\xi-1}$ então $Y_\xi \in c(GR_{\xi-1})$. Uma contradição com o fato de r_ξ ter sido escolhido no passo ξ e $Y_\xi \notin c(GR_{\xi-1})$.*

ii. *$X_\xi \notin c(GR_{\eta-1})$ e $X_\xi \in c(GR_{\xi-1})$. Por hipótese, temos que todas as regras em $GR_{\xi-1} \backslash GR_\eta$ são maiores ou iguais a r_η. Por outro lado, algumas dessas regras são relevantes para r_ξ em trans(Φ), porque participam de uma derivação minimal de X_ξ. Como $r_\xi < r_\eta$, pela transitividade as regras relevantes para r_ξ são menores que r_η. Uma contradição, temos regras r que $r > r_\eta$ e $r < r_\eta$. Logo, Φ não é relevantemente estratificado.*

O seguinte teorema estabelece que se uma regra é aplicada em um estágio i, ela continua sendo aplicada no estágio final.

Teorema A.1.1. *Seja Φ um SFA relevantemente estratificado e $\preceq$ uma extensão linear de $<_R$ para Φ então a seguinte condição é válida*

$$\textit{para toda regra } r = x \Leftarrow X; Y \in GR_\preceq \textit{ temos que } Y \notin c(GR_\preceq).$$

Seja $r_\xi = x_\xi \Leftarrow X_\xi; Y_\xi \in GR_\preceq$ a regra escolhida no ξ-ésimo passo para $GR_\preceq$.

Suponha por absurdo que $y_\eta \in Y_\xi$ tal que $y_\eta \in c(GR_\preceq)$. Seja a regra $y_\eta \Leftarrow X_\eta; Y_\eta \in GR_\preceq$ é η-ésima regra escolhida para $GR_\preceq$.

i. *Se $\eta < \xi$. Pela definição de $GR_\preceq$, $GR_\eta \subseteq GR_{\xi-1}$. Portanto, $y_\eta \in GR_\eta$ então $y_\eta \in GR_{\xi-1}$. Portanto, r_ξ não pode ser escolhida no ξ-ésimo passo. Uma contradição com o fato de r_ξ ter sido escolhida no ξ-ésimo passo.*

ii. *Se $\eta > \xi$. Pela definição de $GR_\preceq$, temos que $GR_{\xi-1} \subseteq GR_{\eta-1}$. Portanto, se $Y_\eta \notin c(GR_{\eta-1})$ então $Y_\eta \notin c(GR_{\xi-1})$. Logo, $X_\eta \notin c(GR_{\xi-1})$, caso contrário, a regra r_η seria escolhida no ξ-ésimo passo. Podemos assumir sem perda de generalidade que existe uma derivação minimal para X_η em trans(Φ). Logo, $r_\eta < r_\xi$. Pelo Lema A.1.1, temos que $\eta < \xi$. Uma contradição com o fato de $\eta > \xi$.*

A.2 Provas do capítulo 5

Teorema A.2.1. *Todo programa fracamente estratificado é relevantemente estratificado.*

Demonstração. Vamos mostrar que a estratificação fraca preserva a estratificação relevante. Vamos assumir que as diferenças entre a programação lógica com a

negação e os SFA's são apenas sintáticas. Vamos mostrar por indução na construção da estratificação fraca, que o programa P_k, $k \leq \eta$ conserva a relevância entre os estratos definidos, ou seja, todas as premissas negativas apagadas de P_k estão nos estratos $S_0, \ldots, S_{2k-1}$.

Para k=0, temos que M_0=$\emptyset$ e N_0= todos os predicados não definidos em P. A operação de **redução** remove as premissas positivas em M_0 e as premissas negativas que estão em N_0. A operação de **simplificação** remove as regras com premissas positivas em N_0 e as negativas em M_0. As regras removidas pela operação de **simplificação** formam o estrato S_0. Todas as regras em S_0 são irrelevantes para o novo programa obtido P_1. O estrato S_1 será vazio.

Hipótese de indução, para todo k $\leq$ n, o programa P_k conserva a relação de relevância entre as regras, ou seja, as premissas negativas apagadas não foram geradas pelos nos estratos anteriores $S_0, \ldots, S_{2k-1}$ e faziam parte de um estrato minimal, ou seja, não tinham nenhuma regra menor do que elas quando elas foram colocadas em um estrato.

Seja k = n+1. Seja S(P_k) o estrato minimal de P_k e seja L(P_k) o subprograma que define os átomos de S(P_k). O programa L(P_k) é positivo, então as premissas negativas que foram apagadas não foram geradas pelos estratos anteriores e nem serão geradas pelos estratos posteriores. Seja M_{k+1} são átomos que estão em S(P_k) e foram gerados pelo programa L(P_k) e N_{k+1} são os átomos que estão S(P_k) e não foram gerados pelo programa L(P_k). As regras de L(P_k) correspondente no programa P formam o estrato S_{2k}. A operação de **redução** elimina as premissas positivas que estão em M_{k+1} e elimina as premissas negativas que estão em N_{k+1}. Todas as regras relevantes para essas operações estão nos estratos anteriores $S_0, ldots, S_{2k}$. A operação de simplificação elimina as regras que têm premissas negativas em M_{k+1} ou premissas positivas em N_{k+1} formando o estrato S_{2k+1}

Teorema A.2.2. *Seja P um programa fracamente estratificado. Então a conjunto gerado pela estratificação fraca e a conjunto gerado pela estratificação relevante coincidem.*

Demonstração. Temos que mostrar que $\bigcup_\alpha M_\alpha = c(GR_{\eta_\preceq})$.

($\Rightarrow$) Por hipótese de indução, para todo ordinal $\alpha \leq \xi$, $\bigcup M_k \in c(GR_{\eta_\preceq})$ e $\bigcup N_k \notin c(GR_{\eta_\preceq})$.

Seja α um ordinal sucessor. Se $A \in M_\alpha$ então existe uma regra r tal que
r=$A \leftarrow A_1, \ldots, A_n \in L(P_\alpha)$ e $\{A_1, \ldots, A_n\} \in \bigcup_{\xi<\alpha} M_\xi$
r=$A \leftarrow A_1, \ldots, A_n, \neg B_1, \ldots, B_m \in P$ e $\{A_1, \ldots, A_n\} \in \bigcup_{\xi\leq\alpha} M_\xi$ e $\{B_1, \ldots, B_m\} \notin \bigcup_{\xi<\alpha} N_\xi$.

Por hipótese, a regra r será escolhida em algum ordinal β, $r \in GR_\beta$

($\Leftarrow$) Por hipótese de indução, para todo ordinal $\xi \leq \alpha$, $c(GR_\xi) \in \bigcup_{\beta<\delta(P)} M_\beta$ e $p^-(GR_\xi) \notin \bigcup_{\beta<\delta(P)} N_\beta$.

Seja $\alpha + 1$ um ordinal sucessor, temos que mostrar que $GR_{\alpha+1} \subseteq M_P$. Basta mostrar que a regra $d_\alpha + 1$ é a menor regra aplicável em GR_α então $c(d_\alpha + 1) \in M_P$. Seja $p^-(GR_\alpha) \subseteq \bigcup_{\xi\leq\beta} N_\xi = N^\beta$ e $p^+ \subseteq \bigcup_{\xi\leq\beta} M_\xi = M^\beta$. Seja P_β é o programa obtido pela simplificação e redução com M^β e N^β. Temos que $c(d_\alpha + 1) \in S(P_\beta)$ e regra correspondente a $d_\alpha + 1$ está em L(P_β) e $c(d_\alpha + 1) \in M_P$.

Teorema A.2.3. *Todo programa relevantemente estratificado P é efetivamente estratificado.*

Demonstração. Seja $P_1 \cup P_2 \cup \ldots \cup P_n \cup \ldots$ uma estratificação relevante minimal de P. Por hipótese de indução, temos que para todo α, Def(EFT^α(P)) = Def(P_α) e Undef(EFT^α(P)) = Undef(P_α). Para n=0, temos mostrar que Def(EFT^0(P)) $\subseteq$ Def(P_0) e UnDef(EFT^0(P)) $\subseteq$ UnDef(P_0)

- Se p $\in$ Def($EFT^0(P)$) é obtido através de um subprograma positivo de $P^+ \subseteq P_0$, então p $\in Def(P_0)$.
- Se p $\in$ UnDef($EFT^0(P)$) então p não é obtido através do programa P retirando todas as premissas negativas. Logo, as regras que geram p são irrelevantes para qualquer outra regra, pela minimalidade da estratificação então todas as regras que geram p estão em P_0. Então elas não podem ser aplicadas a partir de $\emptyset$ então p $\in$ UnDef(P_0).

De maneira análoga podemos mostrar que P_0, temos que Def(P_0) $\subseteq$ Def(EFT^0(P)) e UnDef(P_0) $\subseteq$ UnDef(EFT^0(P)).

Seja $\alpha + 1$ um ordinal sucessor.
$EFT^{\alpha+1}(P)$ = Simplificação (Redução ($EFT^\alpha(P)$, $Def(EFT^\alpha$(P)) , $UnDef(EFT^\alpha$(P))) , $Def(EFT^\alpha$(P)) , $UnDef(EFT^\alpha$(P))). Pela hipótese

de indução, temos que $Def(EFT^{\alpha}(P)) = Def(P_{\alpha})$ e $UnDef(EFT^{\alpha}(P)) = Undef(P_{\alpha})$. Temos que mostrar que $Def(EFT^{\alpha+1}(P)) \subseteq Def(P_{\alpha+1})$ e UnDef($EFT^{\alpha+1}$(P)) $\subseteq$ UnDef($P_{\alpha+1}$).

- Se p $\in$ Def($EFT^{\alpha+1}(P)$) é obtido através de um subprograma positivo de $EFT^{\alpha+1}(P)^{+} \subseteq P_{\alpha+1}$, então p $\in Def(P_{\alpha+1})$.
- Se p $\in$ UnDef($EFT^{\alpha+1}(P)$) então p não é obtido através do programa $EFT^{\alpha+1}(P)$ retirando todas as premissas negativas. Logo, as regras que geram p são irrelevantes para qualquer outra regra que está no próximo estrato, pela minimalidade da estratificação então todas as regras que geram p estão em $P_{\alpha+1}$. Então elas não podem ser aplicadas a partir de Def($P_0 \cup \ldots \cup P_{\alpha}$) então p $\in$ UnDef($P_{\alpha+1}$). □

De maneira análoga podemos mostrar que $P_{\alpha+1}$, temos que Def($P_{\alpha+1}$) $\subseteq$ Def($EFT^{\alpha+1}$(P)) e UnDef($P_{\alpha+1}$) $\subseteq$ UnDef($EFT^{\alpha+1}$(P)).

Corolário A.2.1. *Seja P um programa relevantemente estratificado. Então a conjunto gerado pela estratificação efetiva e a conjunto gerado pela estratificação relevante coincidem.*

A.3 Provas do capítulo 6

Definição A.3.1. *(ETHERINGTON, 1987)*

Seja $\Delta = (D, W)$ uma teoria default seminormal fechada. Sem perda de generalidade, assumimos que todas as fórmulas estão na forma clausal. A relação parcial, $\leq$ e $<$, entre os literais, são definidos da seguinte maneira:

i. *Se $\alpha \in W$, então $\alpha = \alpha_1, \ldots, \alpha_n$, para algum $n \geq 1$. Para todos $\alpha_i, \alpha_j \in \{\alpha_1, \ldots, \alpha_n\}$, se $\alpha_i \neq \alpha_j$, então $\neg\alpha_i \leq \alpha_j$.*

ii. *Se $\delta \in D$, então $\delta = \frac{\alpha:\beta\wedge\gamma}{\beta}$. Seja $\alpha_1, \ldots, \alpha_r$, $\beta_1, \ldots, \beta_s$ e $\gamma_1, \ldots, \gamma_t$ literais da forma clausal de α, β, e γ. Então:*

(a) Se $\alpha_i \in \{\alpha_1, \ldots, \alpha_r\}$ e $\beta_j \in \{\beta_1, \ldots, \beta_s\}$, então $\alpha_i \leq \beta_j$.

(b) Se $\gamma_i \in \{\gamma_1, \ldots, \gamma_t\}$,$\beta_j \in \{\beta_1, \ldots, \beta_s\}$ e $\gamma_i \notin \{\beta_1, \ldots, \beta_s\}$, então $\neg\gamma_i < \beta_j$.

(c) $\beta = \beta_1 \wedge \ldots \wedge \beta_m$, *para algum* $m \geq 1$. *Para cada* $i \leq m$, $\beta_i = (\beta_{i.1}, \ldots, \beta_{i.m_i})$, *onde* $m_i \geq 1$. *Assim se* $\beta_{i.j}, \beta_{i.k} \in \{\beta_{i.1}, \ldots, \beta_{m.m_m}\}$ *e* $\beta_{i.j} \neq \beta_{i.k}$, *então* $\neg\beta_{i.j} \leq \beta_{i.k}$

iii. *O relacionamento transitivo para* $<$ *and* $\leq$*:*

(a) *se* $\alpha \leq \beta$ *e* $\beta \leq \gamma$, *então* $\alpha \leq \gamma$.

(b) *se* $\alpha < \beta$ *e* $\beta < \gamma$, *então* $\alpha < \gamma$.

(c) *se* $\alpha \leq \beta$ *e* $\beta < \gamma$ *ou se* $\alpha < \beta$ *e* $\beta \leq \gamma$, *então* $\alpha < \gamma$.

Definição A.3.2. *Para uma teoria default seminormal,* $\Delta = (D, W)$ *e para quaisquer dois defaults* $d_1, d_2 \in D$,

$$d_1 = \frac{\alpha_1 : \gamma_1 \wedge \beta_1}{\gamma_1} < d_2 = \frac{\alpha_2 : \gamma_2 \wedge \beta_2}{\gamma_2} \; sse$$

existe $\gamma \in LITERAIS(\gamma_1)$ *e* $\gamma' \in LITERAIS(\gamma_2)$ *tal que*

$$\gamma <_d \gamma'.$$

Definição A.3.3. *Uma teoria default seminormal,* $\Delta = (D, W)$, *é localmente estratificada sse* $<_d$ *é uma relação de ordem parcial estrita bem-fundada em D.*

Definição A.3.4. *Seja* Δ*=(W,D) uma teoria default seminormal localmente estratificada e* $\preceq$ *é uma boa ordem obtida através de uma extensão linear da ordem parcial* $<_d$ *entre os default.*

Vamos construir iterativamente o conjunto de defaults gerado pela boa ordem $\preceq$, $GR_{\preceq}$, *da seguinte maneira:*

Para todo ordinal $0 \leq \alpha \leq \eta$*:*

i. *Se* α*=0 então* $GR_0 = \emptyset$

ii. *caso contrário, seja* $d_\alpha = \dfrac{\alpha : \gamma \wedge \beta}{\gamma}$ *o* $\preceq$*-menor default do conjunto* $D \backslash (\bigcup_{\xi<\alpha} GR_\xi)$ *aplicável, ou seja,*
$W \cup \boldsymbol{Cons}(\bigcup_{\xi<\alpha} GR_\xi) \vdash \alpha$ *e*
$W \cup \boldsymbol{Cons}(\bigcup_{\xi<\alpha} GR_\xi) \nvdash \neg(\gamma \wedge \beta)$
então $GR_\alpha = \bigcup_{\xi<\alpha} GR_\xi \cup \{d_\alpha\}$

iii. *caso contrário, não existe default* $d_\alpha = \dfrac{\alpha : \gamma \wedge \beta}{\gamma}$ *no conjunto* $D \backslash (\bigcup_{\xi<\alpha} GR_\xi)$ *aplicável, ou seja,*
$W \cup \boldsymbol{Cons}(\bigcup_{\xi<\alpha} GR_\xi) \vdash \alpha$ *e*
$W \cup \boldsymbol{Cons}(\bigcup_{\xi<\alpha} GR_\xi) \nvdash \neg(\gamma \wedge \beta)$,
então a construção pára e $\eta = \alpha$ *e* $GR_\preceq = \bigcup_{\xi<\alpha} GR_\xi$.

A teoria $Th(W \cup \boldsymbol{Cons}(GR_\eta))$ *é chamada de teoria gerada pela boa ordem* $\preceq$

Vamos adotar a seguinte notação $GR_{\alpha-1} = \bigcup_{\xi<\alpha} GR_\xi$

Lema A.3.1. *Seja* Δ*=(W,D) uma teoria default seminormal estratificada e* $\preceq$ *uma boa ordem obtida através de uma extensão linear de* $<_d$ *para D e* d_ξ *e* $d_\eta \in GR_\preceq$ *são defaults escolhidos no* ξ*-ésimo e* η*-ésimo passo então*

$$se\ d_\xi < d_\eta\ então\ \xi < \eta.$$

Seja $d_\xi = \dfrac{\alpha_\xi : \gamma_\xi \wedge \beta_\xi}{\gamma_\xi}$ *e* $d_\eta = \dfrac{\alpha_\eta : \gamma_\eta \wedge \beta_\eta}{\gamma_\eta}$. *Suponha por absurdo que* $\eta < \xi$. *Sem perda de generalidade assuma que* ξ *é o primeiro ordinal maior que* η *tal que o default escolhido no* ξ*-ésimo passo é menor do que* d_η. *Pela definição de* $GR_\preceq$, *temos que*

$$W \cup \boldsymbol{Cons}(GR_{\xi-1}) \vdash \alpha_\xi\ e\ W \cup \boldsymbol{Cons}(GR_{\xi-1}) \nvdash \neg(\gamma_\xi \wedge \beta_\xi).$$

Se o default d_ξ *não foi escolhido no* η*-ésimo passo. Então:*

$$W \cup \boldsymbol{Cons}(GR_{\eta-1}) \nvdash \alpha_\xi\ ou\ W \cup \boldsymbol{Cons}(GR_{\eta-1}) \vdash \neg(\gamma_\xi \wedge \beta_\xi)$$

i. *Se* $W \cup \boldsymbol{Cons}(GR_{\eta-1}) \vdash \neg(\gamma_\xi \wedge \beta_\xi)$ *então pela definição de* $GR_\preceq$, $GR_{\eta-1} \subseteq GR_{\xi-1}$ *que implica que* $W \cup \boldsymbol{Cons}(GR_{\xi-1}) \vdash \neg(\beta_\xi \wedge \alpha_\xi)$. *Logo,* $d_\xi \notin GR_\preceq$. *Uma contradição, com o fato de* $d_i \in GR_\preceq$.

ii. *Se* $W \cup \boldsymbol{Cons}(GR_{\eta-1}) \nvdash \alpha_\xi$ *e* $W \cup \boldsymbol{Cons}(GR_{\xi-1}) \vdash \alpha_\xi$. *Por hipótese, temos que todos os default em* $GR_{\xi-1} \backslash GR_\eta$ *são maiores ou iguais a* d_η. *Por outro lado, alguns desses default são usado em uma derivação de* α_ξ. *Portanto, estes defaults são estritamente menores ou iguais que* d_ξ *que por sua vez é estritamente menor* d_η, *pela transitividade estes defaults são estritamente menores que* d_η. *Um contradição, existe um default d tal que* $d \geq d_\eta$ *e* $d < d_\eta$, *logo* Δ *não é estratificada.*

Teorema A.3.1. *Seja Δ=(W,D) uma teoria default estratificada e $\preceq$ uma extensão linear de $<$ para D então a seguinte condição*

para todo default $= \dfrac{\alpha : \gamma \wedge \beta}{\gamma} \in GR_{\preceq}$ *temos que* $W \cup \boldsymbol{Cons}(GR_{\preceq}) \nvdash \neg(\gamma \wedge \beta)$ *é valida e Th(*$W \cup \boldsymbol{Cons}(GR_{\preceq})$ *é uma extensão de Reiter para (D,W).*

Suponha por absurdo que para algum $d_\xi = \dfrac{\alpha_\xi : \gamma_\xi \wedge \beta_\xi}{\gamma_\xi}$ *o default escolhido no ξ-ésima passo $GR_{\preceq}$. Temos que,*

$$W \cup \boldsymbol{Cons}(GR_{\preceq}) \vdash \neg(\gamma_\xi \wedge \beta_\xi).$$

ou equivaletemente

$$W \cup \boldsymbol{Cons}(GR_{\preceq}) \vdash \neg\gamma_\xi \vee \neg\beta_\xi.$$

Como $d_\xi \in GR_{\preceq}$

$$W \cup \boldsymbol{Cons}(GR_{\preceq}) \vdash \gamma_\xi,$$

então

$$W \cup \boldsymbol{Cons}(GR_{\preceq}) \vdash \neg\beta_\xi,$$

Sem perda de generalidade, podemos supor que GR_η é o menor ordinal η tal que $W \cup \boldsymbol{Cons}(GR_\eta) \vdash \neg\beta_\xi$.

- ***i.*** *Se $\eta < \xi$. Pela definição de $GR_{\preceq}$, $GR_\eta \subseteq GR_{\xi-1}$. Logo, $W \cup \boldsymbol{Cons}(GR_\eta) \vdash \neg\gamma_\xi$ então $W \cup \boldsymbol{Cons}(GR_{\xi-1}) \vdash \neg\gamma_\xi$. Portando, o default $d_\xi \notin GR_{\preceq}$. Uma contradição com o fato que $d_\xi \in GR_{\preceq}$.*
- ***ii.*** *Se $\eta > \xi$. Então a regra* $d_\eta = \dfrac{\alpha_\eta : \gamma_\eta \wedge \beta_\eta}{\gamma_\eta}$ *foi escolhida no η-ésimo passo. Por hipótese, temos um literal $\delta \in$LITERAIS(γ_η) $\cap$ LITERAIS($\neg\beta_\xi$). Como $\delta < \gamma_\xi$ então $d_\eta < d_\xi$. Pela Lema A.3.1, temos que $\eta < \xi$. Uma contradição com o fato que $\eta > \xi$.*

*Pelo Teorema 3.65, página 75, de (*MAREK; TRUSZCZYNSKI, *1997). Temos que Th(*$W \cup$ ***cons***$(GR_{\preceq})$*) é uma extensão para Δ.*

Referências Bibliográficas

ACZEL, P. *An introduction to inductive definitions.* [S.l.]: Elsevier Science Publishers B.V., 1977. 739–782 p.

APT, K.; BEZEM, M. *Acyclic programs.* Cambridge, MA, USA: Logic programming, MIT Press, 1990. 617–633 p. ISBN 0-262-73090-1.

APT, K.; BLAIR, R. H.; WALKER, A. Towards a theory of declarative knowledge. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, p. 89–148, 1988.

BIDOIT, N.; FROIDEVAUX, C. General logical databases and programs: default logic semantics and stratification. *Inf. Comput.*, Academic Press, Inc., Duluth, MN, USA, v. 91, n. 1, p. 15–54, 1991. ISSN 0890-5401.

BIDOIT, N.; FROIDEVAUX, C. Negation by default and unstratifiable logic programs. *Theoretical Computer Science*, Elsevier Science Publishers Ltd., Essex, UK, v. 78, n. 1, p. 85–112, 1991. ISSN 0304-3975.

BUCHHOLZ, W.; FEFERMAN, S.; SIEG, W. P. W. Iterarted inductive definitions and subsystems of analysis: Recent proof-theoretical studies. *Lectures Notes in Mathematics, vol. 897, Springer-Verlag*, 1981.

CHOLEWINSKI, P. Reasoning with stratified default theories. In: *LPNMR '95: Proceedings of the Third International Conference on Logic Programming and Nonmonotonic Reasoning.* London, UK: Springer-Verlag, 1995. p. 273–286. ISBN 3-540-59487-6.

CHOLEWINSKI, P. Stratified default theories. In: *CSL '94: Selected Papers from the 8th International Workshop on Computer Science Logic*. London, UK: Springer-Verlag, 1995. p. 456–470. ISBN 3-540-60017-5.

CHOLEWINSKI, P. Seminormal stratified default theories. *Annals Mathematical Artificial Intellingence*, v. 17, n. 3-4, p. 213–234, 1996.

CLARK, K. L. Negation as failure. *Logic and Data Bases*, p. 293–322, 1978.

DAVIS, M.; PUTNAM, H. A computing procedure for quantification theory. *Journal ACM*, ACM, New York, NY, USA, v. 7, n. 3, p. 201–215, 1960. ISSN 0004-5411.

DENECKER, M. The well-founded semantics is the principle of inductive definition. In: *JELIA '98: Proceedings of the European Workshop on Logics in Artificial Intelligence*. London, UK: Springer-Verlag, 1998. p. 1–16. ISBN 3-540-65141-1.

DENECKER, M.; BRUYNOOGHE, M.; MAREK, V. Logic programming revisited: Logic programs as inductive definitions. *Computational Logic*, v. 2, n. 4, p. 623–654, 2001. Disponível em: <citeseer.ist.psu.edu/denecker01logic.html>.

ETHERINGTON, D. W. Formalizing nonmonotonic reasoning systems. *Artif. Intell.*, Elsevier Science Publishers Ltd., Essex, UK, v. 31, n. 1, p. 41–85, 1987. ISSN 0004-3702.

FEFERMAN, S. In: *Formal theories for transfinite iterations of generalized inductive definitions and some subsystems of analysis*. [S.l.]: In A. Kino, J. Myhill e R.E. Vesley, eds., Intuitionism and Proof Theory, North Holland, 1970. p. 303–326.

GELDER, A. V. The alternating fixpoint of logic programs with negation. In: *PODS '89: Proceedings of the eighth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. New York, NY, USA: ACM, 1989. p. 1–10. ISBN 0-89791-308-6.

GELDER, A. V.; ROSS, K. A.; SCHLIPF, J. S. The well-founded semantics for general logic programs. *J. ACM*, ACM Press, New York, NY, USA, v. 38, n. 3, p. 619–649, 1991. ISSN 0004-5411.

GELFOND, M. On stratified autoepistemic theories. *Proceedings of AAAO87*, Morgan Kaufman, p. 207–211, 1987.

GELFOND, M.; LIFSCHITZ, V. The stable model semantics for logic programming. In: KOWALSKI, R. A.; BOWEN, K. (Ed.). *Proceedings of the Fifth International Conference on Logic Programming*. Cambridge, Massachusetts: The MIT Press, 1988. p. 1070–1080. Disponível em: <citeseer.ist.psu.edu/gelfond88stable.html>.

HANKS, S.; MCDERMOTT, D. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, Elsevier Science Publishers Ltd., Essex, UK, v. 33, n. 3, p. 379–412, 1987. ISSN 0004-3702.

COSTANTINI, S.; WATSON, R. (Ed.). *ASP2007 - 4th International Workshop on Answer Set Programming*. International Conference on Logic Programming ICLP 2007, Porto, Portugal: Journal of Logic and Computation, 2007.

FABER, W.; LEE, J. (Ed.). *Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP) 2008*. International Conference on Logic Programming ICLP 2008, Udine, Italy: Computing Research Repository (CoRR), 2008.

KOWALSKI, R. A.; EMDEN, M. H. V. The semantics of predicate logic as a programming language. *J. ACM*, ACM Press, New York, NY, USA, v. 23, n. 4, p. 733–742, 1976. ISSN 0004-5411.

KREISEL, G. Generalized inductive definitions. *Technical Report*, Standford University, 1963.

LANGE, S.; GRIESER, G.; JANTKE, K. P. Advanced elementary formal systems. *Theoretical Computer Science*, n. 298, p. 51–70, 2003.

LIFSCHITZ, V. Answer set programming and plan generation. *Journal of Artificial Intelligence*, v. 138, p. 39–54, 2002.

MAREK, W.; TRUSZCZYNSKI, M. Autoepistemic logic. *J. ACM*, ACM, New York, NY, USA, v. 38, n. 3, p. 587–618, 1991. ISSN 0004-5411.

MAREK, W.; TRUSZCZYNSKI, M. *Nonmonotonic Logic: Context-Dependent Reasoning*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1997. ISBN 0387564489.

MAREK, W.; TRUSZCZYNSKI, M. Stable models and an alternative logic programming paradigm. *In The Logic Programming Paradigm: a 25-Year Perspective*, Springer-Verlag, p. 375–398, 1999.

MARTINS, A. T. C. *A Syntactical and Semantical Uniform Treatment for the IDL and LEI Nonmonotonic System*. Tese (Doutorado em Informática) — Department of Informatics, Federal University of Pernambuco, Brazil., 1997.

MOORE, R. C. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, Elsevier Science Publishers Ltd., Essex, UK, v. 25, n. 1, p. 75–94, 1985. ISSN 0004-3702.

MORRIS, P. H. The anomalous extension problem in default reasoning. *Artif. Intell.*, Elsevier Science Publishers Ltd., Essex, UK, v. 35, n. 3, p. 383–399, 1988. ISSN 0004-3702.

MOSCHOVAKIS, Y. N. Elementary induction on abstract structures. North-Holland Publishing, 1974.

MOSCHOVAKIS, Y. N. *On non-monotone inductive definability.* [S.l.]: Fundamenta Mathematica 82, 1974. 39-83 p.

NIEMELã, I. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, Kluwer Academic Publishers, Hingham, MA, USA, v. 25, n. 3-4, p. 241–273, 1999. ISSN 1012-2443.

PEQUENO, M. C. *Defeasible Logic with Exception-First.* Tese (Doutorado) — Department of Computing Imperial College of Science, Technology and Medicine,, 1994.

PEQUENO, M. C.; MARTINS, A. T. C.; PEQUENO, T. H. C. Well-behaved idl theories. *In Lecture Notes in Artificial Intelligence, Proceedings of the 13th Brazilian Symposium on Artificial Intelligence*, Springer-Verlag, Curitiba, Oct, v. 1159, p. 11–20, 1996.

PEQUENO, M. C.; VERAS, R. M. S.; TAVARES, W. A. Handling exceptions in nonmonotonic reasoning. In: *Proceedings of the Third Latin American Workshop on Non-Monotonic Reasoning 2007 (LANMR07), Informatik V, RWTH Aachen, 2007. v. 286.* [S.l.: s.n.], 2007.

PEQUENO, T. H. C. A logic for inconsistent nonmonotonic reasoning. In: . [S.l.]: Technical Report 90/6 Department of Computing, Imperial College London, 1990.

PEQUENO, T. H. C.; BUCHSBAUM, A. R. The logic of epistemic inconsistency. *In Proceeding of the 2nd KR'91 J.A. Allen, R.Fikes and E. Sandewall editor*, Morgan Kaufamann Publisher Inc, Apr, p. 453–60, 1991.

PRZYMUSINSKA, H.; PRZYMUSINSKI, T. C. Weakly perfect model semantics for logic programs. *Proceeding of the Fifth Internacional Conference and Symposium on Logic Programming*, MIT Press, p. 1106–1120, 1988.

PRZYMUSINSKA, H.; PRZYMUSINSKI, T. C. Weakly stratified logic programs. *Fundamenta Informaticae*, XIII, p. 51–65, 1990.

PRZYMUSINSKI, T. C. *On the declarative semantics of deductive databases and logic programs with negation*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988. 193–216 p. ISBN 0-934613-40-0.

PRZYMUSINSKI, T. C. Every logic program has a natural stratification and an iterated least fixed point model. In: *PODS '89: Proceedings of the eighth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. New York, NY, USA: ACM, 1989. p. 11–21. ISBN 0-89791-308-6.

PRZYMUSINSKI, T. C. On the declarative and procedural semantics of logic programs. *Journal of Automated Reasoning*, v. 5, p. 167–205, 1989.

REITER, R. A logic of default reasoning. *Artificial Intelligence*, n. 13, p. 81–132, 1980.

SMULLYAN, R. Theory of formal systems. Princeton University Press, 1961.

TARSKI, A. A lattice-theoretical fixpoint theorem and its applications. Pacific Journal of Mathematics., v. 5., p. 285–310., 1955.