TARCÍSIO GENARO RODRIGUES

SOBRE OS FUNDAMENTOS DA PROGRAMAÇÃO LÓGICA PARACONSISTENTE (Versão Preliminar)

Dissertação de Mestrado a ser apresentada ao Departamento de Filosofia do Instituto de Filosofia e Ciências Humanas da Universidade Estadual de Campinas sob a orientação do Prof. Dr. Marcelo Coniglio.

Este exemplar corresponde à redação parcial da dissertação a ser defendida em 29/09/2010.

BANCA
Prof. Dr. Marcelo Esteban Coniglio (Orientador)
Prof. Dra. Juliana Bueno Soler
Prof. Dr. Hércules de Araújo Feitosa
Prof. Dr. Walter Alexandre Carnielli (Suplente)
Profa. Dra. Ítala Maria Loffredo D'Ottaviano (Suplente)

SETEMBRO/2010

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DO IFCH - UNICAMP
Bibliotecária: Sandra Aparecida Pereira CRB nº 7432

Rodrigues, Tarcísio Genaro
Sobre os Fundamentos da Programação Lógica Paraconsistente / Tarcísio Genaro Rodrigues. - - Campinas, SP : [s. n.], 2010.

Orientador: Marcelo Esteban Coniglio.
Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Filosofia e Ciências Humanas.

1. Lógica matemática não clássica. 2. Lógica simbólica e matemática. 3. Inconsistência (Lógica). 4. Programação Lógica. 5. Linguagens formais - Semântica. I. Coniglio, Marcelo Esteban. II. Universidade Estadual de Campinas, Instituto de Filosofia e Ciências Humanas. III. Título.

Resumo

Inicialmente este trabalho estava focado em encontrar regras de resolução para lógicas paraconsistentes, mais especificamente para versões quantificadas de primeira ordem das Lógicas da Inconsistência Formal (**LFI**'s), tendo em vista aplicações no campo da programação lógica como ferramenta para a pesquisa em inteligência artificial e representação do conhecimento inconsistente (*e.g.* bancos de dados inconsistentes). **Tal proposta se mostrou fora de nosso alcance, principalmente por demandar uma versão do teorema de Herbrand para tais cálculos.** Terminamos, por fim, com um trabalho nos fundamentos da programação lógica paraconsistente: uma nova técnica de demonstração dos Teoremas de Completude para as **LFI**'s quantificadas e uma formulação por sequentes de **QmbC** e **QmCi** – versões de primeira ordem das **LFI**'s **mbC** e **mCi**. Também pudemos provar, para **QmbC**, o teorema da eliminação do corte e uma versão do teorema de Herbrand restrita ao fragmento Π_2.

Sumário

Capítulo 1

Da Lógica à Programação Lógica

Neste primeiro capítulo pretendemos trabalhar aspectos históricos e metodológicos. Buscaremos evidenciar como o carácter unviversal com que se apresenta a lógica matemática e, em particular, o cálculo de primeira ordem clássico determinaram o desenvolvimento dos primeiros métodos automáticos de demonstração na década de 1950 e de alguns ramos da Inteligência Artificial nas décadas de 1960 e 1970. Essas duas áreas são o solo do qual se ergue a pesquisa em programação lógica. A programação lógica é um belo exemplo de interação entre lógica e tecnologia.

Nesse primeiro capítulo nossa linha de raciocínio seguirá em torno da seguinte progressão de relações:

- Completude e teorema de Herbrand
- Teorema de Herbrand e resolução
- Resolução e programação lógica
- Programação lógica e PROLOG.

Esperamos com isso dar estofo às ponderações mais críticas que virão no Capítulo 5.

1.1 A Lógica no Século XX

A lógica matemática tem um papel universal em pelo menos dois sentidos. Por um lado, possibilita, de maneira homogênia, a descrição sintática de um tipo de linguagem comum a diversas áreas do conhecimento, bem como o entendimento de fundo necessário a certos conceitos gerais que garantem expressividade a diversas disciplinas: as noções de predicado e função de primeira ordem, bem como a possibilidade de compor expressões com conectivos e quantificadores têm uma parte essencial em várias áreas do conhecimento. Isso pode ser entendido como o **caráter descritivo da lógica**.

Por outro, através do intermédio da relação de consequência lógica, podemos expressar as consequências das diversas teorias quando elas se encontram devidamente expressas em uma linguagem apropriada: tal é o **caráter dedutivo da lógica**. Para tanto, não importa que as teorias em questão sejam do tipo formal, como por exemplo as diversas teorias algébricas, *com várias interpretações possíveis*, ou interpretadas, quando o que se busca é *caracterizar um único modelo*, como a mecânica newtoniana. A distinção entre função descritiva e dedutiva é trabalhada por Hintikka em [Hin96], onde afirma o caráter descritivo como fundamental, anterior mesmo ao dedutivo.[1]

Tais características são importantes para entender o desenvolvimento da própria programação lógica. Elas ajudam a entender porque a lógica matemática está nos fundamentos de duas áreas que poderiam ser tomadas como suas mães: o campo da *dedução automática* e o da *inteligência artificial*. Mas, para chegar ao ponto em que a encontramos, com tal grandeza expressiva e uma aceitação que estrapola a área da matemática e da filosofia, a lógica clássica passou por um grande desenvolvimento.

Kant, no século XVIII, a considerava uma disciplina acabada, dado que não havia sofrido modificações significativas desde Aristóteles. Mas as coisas mudam de figura em meados do século XIX. Os trabalhos de Boole [Boo47] e De Morgan [DM47] marcam o começo da abordagem matemática em lógica clássica. Boole desenvolveu o caráter algébrico do silogismo aristotélico, à maneira do que Descartes fez com a geometria grega. Inicia-se, então, uma tradição – cultivada por Pearce, Schröder e outros – conhecida como *álgebra da lógica*, que lançou as bases da abordagem posteriormente desenvolvida pela *teoria dos modelos*.

Ainda no século XIX, através das discuções fundacionais que perpassaram a matemática, evoluiu como pano de fundo à visão *logicista* dos fundamentos. Estava presente em meio a teorias que buscavam bases sólidas para determinadas áreas, como na *Begriffsschrift* de Frege, que elaborou pela primeira vez um *cál-*

[1] [Hin96, p.9]

culo conceitual com linguagem e regras de dedução formalmente especificadas, a ser usada para os *Grundgesetze der Arithmetik*. No trabalho de Peano, apesar de não contar com um sistema de *regras*, fazia-se presente enquanto sistema notacional. Nessa perspectiva fundacional, teve a missão de eliminar paradoxos que surgiam na teoria dos conjuntos (a primeira grande teoria que se propunha a abarcar a matemática como um todo), com a teoria Russelliana de tipos. Desse modo, constituiu-se a outra linha tradicional de investigação em lógica matemática, de carácter preponderantemente sintático.[2]

Mas foi com o programa de Hilbert para a fundamentação da matemática que certos conceitos essenciais foram pela primeira vez propostos. Em Frege ou Whitehead e Russell, por exemplo, o foco estava na lógica de alta ordem, não destacavam o fragmento de primeira ordem à parte.

Parece que foi Hilbert o primeiro a reconhecer que sistemas de prova de primeira ordem eram dignos de estudo[3]. Pelo começo do ano de 1920, em seus seminários sobre os fundamentos da matemática, Hilbert foi levado a propor o que considerava *o principal problema da lógica matemática*, o problema da decisão[4] para a lógica de primeira ordem[5]. Em seus *Princípios da Lógica Teórica* [HA28], Hilbert e Ackermann o definem precisamente: encontrar um processo "que permita decidir a validade de expressões lógicas em um número finito de operações" [6].

Martin Davis, testemunha ocular e grande contribuidor para a historiografia da dedução automática, além de co-autor de um método[7] de demonstração automática, afirma que nesse livro "*algumas das idéias centrais no trabalho em dedução automática apareceram pela primeira vez*"[8]. Entre elas, além do problema da decisão, temos a questão da completude do cálculo funcional clássico. Essas questões estão relacionadas com o cerne do programa de Hilbert: encontrar demonstrações de consistência por meios finitários.

Alguns anos mais tarde, com os teoremas de incompletude de Gödel, ficou claro que, para sistemas com uma certa expressividade, tais demonstrações de consistência não poderiam se realizar internamente aos próprios sistemas. Isso acabava com as perspectivas de algum sistema fraco da aritmética (ou seja, o que se entende formalmente por *finitário*) provar sua própria consistência. Mas o programa de Hilbert não deixou apenas resultados negativos, foi em seu contexto que surgiu a importante noção de *metamatemática*: a investigação matemática de sistemas ló-

[2] [DvH86, p.44]
[3] [Bur98, p.382]
[4] *Entscheidungsproblem*, no original em alemão, forma pela qual é normalmente referenciado.
[5] [Soa96, p.6]
[6] [HA28, pp.72-73], conforme citado em [Soa96, *ibidem*].
[7] O procedimento de Davis-Putnam [DP60].
[8] [Dav83, p.11]

gicos formalizados. Esse contexto deu sentido à questão da completude e, mais tarde, possibilitou o estudo sistemático de diversos sistemas de prova e suas inter-relações. Desse modo, o desenvolvimento dos primeiros provadores automáticos, na década de 1950, pode também entrar na conta dos "herdeiros" dessas primeiras investigações metamatemáticas.

Do ponto de vista da fundamentação da matemática, a completude é um resultado notório por si. O próprio fato de que tal problema passou a fazer sentido foi significativo do desenvolvimento da lógica como um todo, conforme a análise de van Heijenoort e Dreben em [DvH86, §1]. Na linha do *logicismo*, que se extendeu de Frege até Russel e Whitehead, tal questão não poderia ter surgido tendo em vista o caráter universal que imprimiam à lógica: à ela cabia a formalização de toda a matemática, sistemas quantificacionais *puros* não estavam no centro das investigações. O caráter absoluto da lógica os impedia de considerar seus sistemas como totalidades passíveis de estudo, a não ser pela pela derivação (interna) de teoremas. Já do ponto de vista da linha da *álgebra da lógica*, fazia falta a própria noção de *sistema formal*, tudo se dava por um viés modelo-teórico, semântico.

A completude, demonstrada por Gödel em [Gö30], é um resultado metamatemático que relaciona essas duas correntes. O teorema da completude mostrou por meios *não construtivos* que as axiomatizações (o sistema em que trabalhava era essencialmente o fragmento de primeira ordem dos *Principia*, seguindo a notação de Hilbert e Ackermann [HA28]) "detectavam" sintaticamente todas as teorias que não tinham modelos. A prova (na verdade, sua generalização para teorias enumeráveis pelo teorema da compacidade) consistia em mostrar que todo sistema axiomático de primeira-ordem ou é inconsistente (entendido como *derivando uma contradição*) ou tem um modelo. Ou seja, as teorias sintaticamente inconsistentes (*que derivam contradições*) são exatamente aquelas semanticamente inconsistentes (*que não têm modelos*). Com isso, na própria opinião de Gödel, tem-se a fundamentação teórica do método – já usual na época – de se provar que uma teoria não deduz contradições através de demonstrações de existência de modelos para a mesma.[9]

Tendo em vista que podemos ter cálculos definidos sobre linguagens não recursivamente enumeráveis, uma demonstração de caráter não construtivo é essencial. O argumento original de Gödel, apesar de dar ter se baseado em argumentos não construtivos, não se extende a teorias definidas sobre linguagens não-enumeráveis e se baseia na existência de *domínios numéricos* para teorias não contraditórias. A formulação mais usada atualmente, baseada na demonstração de Henkin [Hen49], admite teorias sobre linguagens arbitrárias e faz uso de *modelos sintáticos*, obtidos da própria linguagem da teoria, enriquecida com constantes testemunhas para

[9] [DvH86, p.48]

fórmulas existenciais[10]. Para se chegar a tais modelos também é necessário algum *princípio de escolha* (como o teorema de Teichmüller-Tukey em [Sho67, p.47]) que demonstre a existência de extensões completas para teorias consistentes.

Relacionado a esse resultado positivo para o problema da completude, temos outra negativa aos problemas levantados por Hilbert em [HA28]. A completude de Gödel pode ser entendida como uma solução não construtiva para o *Entscheidungsproblem* – uma das motivações da lógica formal na década de 1920 e 1930 – dado que para determinar a inconsistência de uma fórmula, depende de algum equivalente do axioma da escolha. Uma solução finitária veio a se mostrar impossível com o trabalho de Church [Chu36] e Turing [Tur37].

Apesar de não construtiva, a completude de Gödel demonstra que o conjunto das sentenças lógicamente válidas é recursivamente enumerável. Restringindo nossa atenção às bases da dedução automática, a demonstração de Gödel estava construída sobre o trabalho de Skolem, o qual foi fundamental para essa área:

> *The Key work for automated deduction was that of Skolem. He carried out a systematic study of the problem of the existence for an interpretation which will satisfy a given formula of the predicate calculus.*
>
> [Dav83, p.9]

Gödel adaptou e generalizou um resultado de Skolem publicado em 1920 [Sko20]. Skolem escrevera esse artigo para clarificar o teorema de Löwenheim, publicado em [Lö15]. Esse trabalho de Löwenheim foi o primeiro a estabelecer um resultado significativo para a lógica de primeira ordem[11]: prova que se uma fórmula é satisfatível, também o é em um domínio enumerável. Skolem generaliza, em sua clarificação, o resultado para um conjunto enumerável de fórmulas.

Para chegar ao resultado, Skolem prova antes que para toda fórmula de primeira ordem existe uma outra, em uma forma normal, que é satisfatível em algum domínio exatamente quando a original também é. Essas formas normais estão no que designamos hoje em dia por fragmento Π_2, isto é, são fórmulas constituídas por uma sequência de quantificadores universais seguidos por uma sequência de existênciais e, ao final, uma fórmula livre de quantificadores. Para chegar às interpretações numéricas para teorias não contraditórias, Gödel fez uso da *forma normal*

[10] Um resultado original da presente dissertação é uma técnica de prova para demonstrações de completude para lógicas onde não haja interdefinibilidade entre os dois quantificadores. Na seção 3.3, a extensão da linguagem original por constantes testemunhas para existenciais é feita também por *constantes contra-testemunhas para fórmulas universais*.

[11] Conforme afirma Burris em [Bur98, p.365]. Há quem diga mesmo que "Com esse artigo, a lógica de primeira ordem se tornou uma área de interesse especial, devido às surpreendentes propriedades metamatemáticas (...)", [WSBA09, p.17].

de Skolem e pode restringir sua atenção apenas às fórmulas nesse formato (depois de mostar que seu resultado mais geral era obtido a partir do resultado para as formas normais). O que Gödel demonstrou, de fato, é que toda fórmula ou tem um modelo enumerável, ou é *refutável*, no sentido de sua negação ser demonstrável.

Em 1928 [Sko28], Skolem introduziu a noção de *Skolemização* de uma fórmula, reduzindo o problema da satisfatibilidade ao fragmento das fórmulas com apenas quantificadores universais. Toda fórmula é satisfatível em um domínio se e somente se sua versão Skolemizada é também. Sobre esse artigo de 1928, diz Martin Davis:

> *This remarkable paper, not only has a treatment of what is usually called Herbrand's theorem in writings on automated deduction, but has a clear and complete definition of what is usually called is this field the* Herbrand universe *for a formula.*
>
> [Dav83, p.10]

A Skolemização é feita em dois passos. Primeiramente os quantificadores são postos no início, efetuando-se renomeações de variáveis, se necessário. Considere, por exemplo, as seguintes fórmulas equivalentes, com a última nesse formato intermediário:

$$\forall x\,(\forall x\,P(x) \wedge Q(x)) \Rightarrow \exists x\,(Q(x) \wedge \exists x\,R(x))$$

$$\forall x\,(\forall y\,P(y) \wedge Q(x)) \Rightarrow \exists z\,(Q(z) \wedge \exists w\,R(w))$$

$$\exists z\,\Big(\forall x\,(\forall y\,P(y) \wedge Q(x)) \Rightarrow (Q(z) \wedge \exists w\,R(w))\Big)$$

$$\exists z\,\forall x\,\Big((\forall y\,P(y) \wedge Q(x)) \Rightarrow (Q(z) \wedge \exists w\,R(w))\Big)$$

$$\exists z\,\forall x\,\Big(\forall y\,(P(y) \wedge Q(x)) \Rightarrow (Q(z) \wedge \exists w\,R(w))\Big)$$

$$\exists z\,\forall x\,\Big(\forall y\,(P(y) \wedge Q(x)) \Rightarrow \exists w\,(Q(z) \wedge R(w))\Big)$$

$$\exists z\,\forall x\,\forall y\,\Big((P(y) \wedge Q(x)) \Rightarrow \exists w\,(Q(z) \wedge R(w))\Big)$$

$$\exists z\,\forall x\,\forall y\,\exists w\,\Big((P(y) \wedge Q(x)) \Rightarrow (Q(z) \wedge R(w))\Big)$$

Então, os quantificadores existenciais são eliminados pela adição de símbolos funcionais. Com a eliminação de um existencial, adiciona-se um símbolo funcional de igual aridade ao número de quantificadores universais dos quais o existencial está no escopo:

$$\exists z\ \forall x\ \forall y\ \exists w\ \Big(P(y) \wedge Q(x)\ \Rightarrow\ Q(z) \wedge R(w)\Big)$$

$$\forall x\ \forall y\ \exists w\ \Big(P(y) \wedge Q(x)\ \Rightarrow\ Q(f_z) \wedge R(w)\Big)$$

$$\forall x\ \forall y\ \Big(P(y) \wedge Q(x)\ \Rightarrow\ Q(f_z) \wedge R(g_w(x,y))\Big)$$

A partir dessa técnica, Skolem conseguiu um procedimento de prova (na verdade, de refutação) que não dependia de um sistema dedutivo em particular. O método consistia em ir, sistematicamente, substituindo as variáveis por todos os termos formados com as constantes e símbolos funcionais da fórmula em questão (que constituem o chamado *Universo de Herbrand* da fórmula). O que chamamos de *Teorema de Herbrand* diz que a fórmula é insatisfatível se e somente se alguma conjunção dessas substituições é falsa, no sentido do cálculo proposicional. Herbrand, posteriormente, refinou esse resultado de maneira a funcionar para todas as fórmulas, não apenas para formas Skolemizadas.

Podemos ver o grande foco sobre as noções de refutação e insatisfatibilidade, o que está relacionado com o fato de que um dos métodos mais impactantes de dedução automática, a *resolução de Robinson*, é um procedimento de refutação. Na verdade, no cálculo funcional clássico, os problemas da refutação, da insatisfatibilidade e da validade universal são equivalentes[12]. Também é comum considerar equivalentes os problemas da validade e da satisfatibilidade, como afirma o próprio Hilbert:

> *It is customary to refer to the two equivalent problems of* universal validity *and* satisfiability *by the common name of the* decision problem *of the restricted predicate calculus.*
>
> [HA50, p.113]

[12]Temos também que o problema da validade universal (lógica) é um caso especial do problema da consequência lógica, isto é, uma sentença é universalmente válida exatamente quando é consequência lógica do conjunto vazio:

$$\varphi \text{ é válida} \iff \vDash \varphi$$

Algumas considerações sobre essas equivalências, na perspectiva das **LFI**'s, podem ser encontradas na Seção 6.1.

Mas, apesar de intimamente relacionados, os problemas da validade lógica e da satisfatibilidade, no que diz respeito ao seu grau de irresolubilidade, não são equivalentes. A questão da validade universal é semidecidível[13], enquanto a da existência de modelos não é sequer semidecidível. No panorama clássico, o problema da satisfatibilidade é equivalente ao da existência de contramodelos:

$$\begin{aligned} \Gamma, \varphi \textit{ é satisfatível} \quad &\Longleftrightarrow \quad \Gamma \nvDash \neg\varphi \\ \Gamma \nvDash \varphi \quad &\Longleftrightarrow \quad \Gamma, \neg\varphi \textit{ é satisfatível} \end{aligned}$$

enquanto o da insatisfatibilidade, ao da consequência lógica:

$$\begin{aligned} \Gamma, \varphi \textit{ é insatisfatível} \quad &\Longleftrightarrow \quad \Gamma \vDash \neg\varphi \\ \Gamma \vDash \varphi \quad &\Longleftrightarrow \quad \Gamma, \neg\varphi \textit{ é insatisfatível} \end{aligned}$$

1.2 Desenvolvimento da Programação Lógica

1.2.1 Primeiros Avanços na Área de Dedução Automática

A ubiquidade do cálculo clássico de primeira ordem[14] mostrou-se presente nos diversos formalismos criados na década de 1920 e 1930, à Hilbert ou à Gentzen, que tinham sentido como ferramentas de investigação metamatemática, seguindo as linhas gerais do programa de Hilbert. Com o desenvolvimento dos computadores digitais, começou a aumentar as possibilidades de serem utilizados como ferramentas para "fazer" matemática, com o início da pesquisa em dedução automática, trazendo novas perspectivas à visão Leibniziana de um *Calculus Ratiocinator* (conforme argumenta Leitsch em [Lei, p.1]).

Esta "onipresença" da lógica de primeira ordem, no tocante ao seu carácter dedutivo e interpretativo, veio também a influenciar a área de automatização de demonstrações e posteriormente da programação lógica. Possibilitou o estabelecimento de um formalismo sintático universalmente reconhecido, com suas linguagens com predicados e funções, unidos por conectivos proposicionais e quantificadores. Há também grande consenso quanto a uma semântica padrão (a concepção Tarskiana de verdade para linguagens formalizadas, cf. [Tar44]).

Além disso, as variadas espécies de cálculos permitiam deduzir a partir das informações codificadas em fórmulas de primeira ordem. Com efeito, temos o

[13] Logo, o problema de decidir se uma fórmula é consequência lógica de um conjunto recursivamente enumerável, ou mesmo recursivo, de fórmulas ($\Gamma \vDash \varphi$) é, em geral, apenas semidecidível.

[14] Que veio mesmo a ser considerado *a lógica por excelência*, posição hoje em dia desafiada por vários vieses, como atesta a enorme área de lógicas não clássicas, estando as paraconsistentes inclusas. Veja, também, [Hin96] e [BF85].

depoimento de Robert Kowalski em um artigo sobre o ínicio da pesquisa em programação lógica:

> *Logic programming shares with mechanical theorem proving the use of logic to represent knowledge and the use of deduction to solve problems by deriving logical consequences.*
>
> [Kow88, p.38]

Os métodos de Skolem e Herbrand, surgidos em meio às investigações metamatemáticas do programa de Hilbert, também foram determinantes sobre a forma como evoluíram os primeiros provadores de teorema. Sobre o uso de tais técnicas em dedução automática, Davis afirma ter sido inicialmente sugestão de Abraham Robinson (sugestão essa publicada em [Rob57]):

> *The first suggestion that methods based on "Herbrand's theorem" were appropriate for general purpose theorem-provers seems to have been made by Abraham Robinson in a brief talk delivered at the Summer Institute for Symbolic Logic at Cornell University in 1954.*
>
> [Dav83, p.16]

Além disso, outro fator fundamental é *a possibilidade de reduzir todas as sentenças a uma forma normal conjuntiva*, característica que foi um suporte ao método divisor de águas nessa área, a resolução de Robinson, introduzida em [Rob65].

A importância de tal método é enfatizada por Martin Davis (que é também co-autor do procedimento de Davis-Putnam, influencia direta do trabalho de Robinson):

> *The explosion of interest which has produced the field as we know it today can be traced to [Rob65] in which the elegance and simplicity of the resolution principle as a basis for mechanized deduction first appeared.*
>
> [Dav83, p.1]

1.2.2 A Resolução e o Fragmento das Cláusulas de Horn e dos Programas Positivos

A resolução combina, em um único procedimento, duas etapas que até então eram executadas separadamente pelos métodos de demonstração automática baseados no teorema de Herbrand: o processo de substituição de termos por variáveis e a análise vero-funcional (nos parâmetros do cálculo proposicional) do resultado:

> *A close analysis of the process of substitution (of terms for variables) and the process of truth-functional analysis of the results of such substitutions, reveals that both processes can be combined into a single new process (called* resolution*), iterating which is vastly more efficient than the older cyclic procedures consisting of substitution stages alternating with truth-functional analysis stages.*
>
> [Rob65, p.23]

A partir do desenvolvimento das pesquisas com automatização de provas no contexto da lógica clássica, Robinson foi capaz de chegar ao que veio a ser o ponto de partida da programação lógica: a regra de resolução com seu algoritmo de unificação. Com base na resolução, foi possível estabelecer um método baseado em refutação que permitisse decidir a satisfabilidade de fórmulas proposicionais e decidir (apenas positivamente) a insatisfatibilidade das de primeira-ordem. O método, vez ou outra, poderia também chegar a determinar a satisfatibilidade de certas fórmulas de primeira ordem, mas não sempre, tendo em vista a indecidibilidade desse problema, que não é sequer positivamente decidível.

O método de Robinson assume fórmulas no que se chama de notação clausal. Aceita, como entrada, um conjunto finito de *cláusulas*, cada uma consistindo em um conjunto finito de *literais*. Os literais são fórmulas atômicas, ou negações delas. Cada cláusula é interpretada como a disjunção de seus literais e o conjunto das cláusulas, como a conjunção de todas as cláusulas.

Formalmente, a resolução é um método para decidir a insatisfatibilidade de um conjunto finito de cláusulas. Na lógica clássica, isso é equivalente a provar que a negação de uma cláusula é consequência lógica das outras:

$$C \vDash \neg\varkappa \quad \Longleftrightarrow \quad C,\ \varkappa \textit{ é insatisfatível}$$

Também no panorama clássico, a negação de uma cláusula (que é universalmente quantificada) é equivalente à existência de elementos que falsifiquem simultaneamente seus literais (representamos por $\overline{L}$ o complemento do literal L, isto é, se o literal é uma negação, seu complemento é formado excluindo-se o sinal de negação e, caso contrário, adicionando-se):

$$\varkappa \equiv \forall\vec{x}\,(L_0 \vee \cdots \vee L_n)$$
$$\neg\varkappa \equiv \exists\vec{x}\,(\overline{L_0} \wedge \cdots \wedge \overline{L_n})$$

Logo, a resolução é um método de refutação que computa contra-exemplos para o conjunto das cláusulas. Isto é, se trata de um método para computar substituições para as variáveis que falsifiquem alguma das cláusulas de entrada. Os

contra-exemplos que computa estão restritos unicamente aos termos formados pelos símbolos funcionais já presentes na entrada. Pelo teorema de Herbrand, o método tem sucesso exatamente quando a entrada é *insatisfatível*.

À primeira vista, pode parecer contraditório a existência de um método que sirva para encontrar contra-exemplos ser suficiente para determinar a insatisfatibilidade. A insatisfatibilidade é a impossibilidade de que haja um modelo, significa que todas as substituições, em todos os possíveis domínios, são contra-exemplos. Logo, um método que determine a insatisfatibilidade encontrando *um único contra-exemplo* parece estranho.

Está justamente nisso a força do teorema de Herbrand. Tal teorema possibilita *reduzir o espaço de busca* pela insatisfatibilidade aos termos formados pelo material sintático já presente nas fórmulas.

A programação lógica é a materialização da resolução de Robinson em uma aplicação de inteligência artificial: o processamento de linguagem natural. Restrita sua abrangência às chamadas Cláusulas de Horn (que estavam sendo usadas para descrever a gramática do francês), uma especialização sua, o método **SLD**, introduzido em [Kow74], se torna completo e caracteriza o sistema chamado na época de **PROLOG** (de *PROgramation en LOGique*, cf. [Col93, p.331]).

A programação lógica tem sua origem imediata no desenvolvimento de duas principais linhas de pesquisa: uma ligada ao campo da inteligência artificial e a outra ao da dedução automática de teoremas[15], como afirma [Llo93, p.1]. Em meio às pesquisas em inteligência artificial, o desenvolvimento de aplicações para processamento de linguagem natural serviu como primeira aplicação e motivo imediato para o desenvolvimento do **PROLOG**.

Cláusulas de Horn são aquelas que têm, no máximo, uma literal positiva. Classicamente, os *programas lógicos positivos* são conjuntos de cláusulas de Horn com exatamente uma literal positiva, chamadas de *regras*. Cláusulas que não apresentam nenhuma literal positiva são denominadas *consultas*. O **PROLOG** clássico é um sistema que, tendo um programa lógico positivo como entrada, determina as consultas que são *refutáveis* a partir do programa. Os dois possíveis tipos de cláusulas de Horn (com uma ou nenhuma literal negativa) formam as possíveis entradas do **PROLOG** clássico.

Se representarmos as fórmulas atômicas por A ou B, com ou sem subescritos, uma cláusula de um programa positivo, classicamente, é equivalente a uma das

[15] Tais áreas não devem ser vistas de maneira dicotômica, tendo em vista que, por exemplo, o foco inicial das pesquisas em inteligência artificial foi na área de dedução automática.

seguintes formas:

$$A \Leftarrow B_1 \wedge \ldots \wedge B_n$$
$$A \vee \neg B_1 \vee \ldots \vee \neg B_n$$

com as variáveis universalmente quantificadas. No caso em que $n = 0$, a regra é chamada de *fato*. Já uma consulta é equivalente a

$$\neg B_1 \vee \ldots \vee \neg B_n$$

com as variáveis também universalmente quantificadas. No caso em que $n = 0$, a consulta representa uma contradição.

O procedimento **SLD** (abreviação de ***S**election-function **L**inear resolution for **D**efinite clauses*), tendo como entrada um programa positivo (também chamado na literatura por *programa definido*) e uma consulta, usa uma *função de escolha* para selecionar o literal da consulta e a cláusula do programa a serem unificados.

1.2.3 O Paradigma Declarativo: Algoritmo = Lógica + Controle

Duas maneiras de entender o significado de um programa, em geral, estavam dadas na época. Pelo lado da *semântica operacional*, entendia-se o signficado de um programa através de sua ação direta sobre cada entrada. Pelo da *semântica denotacional*, pela passagem ao limite das possíveis relações entre suas entradas e saídas. O viés operacional tenta dar conta dos processos pelos quais o programa chega às soluções. A vertente denotacional tenta entender a computação abstraindo seu lado dinâmico, vê a computação como um processo já completado.

A programação lógica, seguindo a idéia de Kowalski expressa em [Kow79], é uma maneira de deixar clara a diferença entre a parte declarativa e a operacional dos programas. O programa lógico, em si, é uma especificação do problema de certa maneira *independente da computação da resposta*. A parte operacional cabe ao interpretador, que vai executar a resolução usando alguma função de seleção para escolher quais átomos serão resolvidos antes de quais. É uma forma também de imprimir um caráter operacional à própria lógica.

Os artigos seminais de Kowalski ([Kow74] e [vEK76]) dialogam com essas duas perspectivas. Em [vEK76], Kowalski e van Emden vão elaborar a semântica dos programas lógicos em termos de pontos fixos de operadores de consequência direta. Tal teoria pode ser encarada como uma especialização da semântica denotacional, que começava a se delinear, a partir de trabalhos como os de Scott [Sco70].

Como declarado pelo próprios autores, a semântica proposta podia ser encarada como um caso especial do teorema de completude de Gödel. Troca-se a interpretação tarskiana pela de pontos fixos e sua contra-partida, os sistema axiomáticos, pela resolução **SLD**, que fornece uma descrição operacional do programa lógico:

> *We show that operational semantics is included in the part of syntax concerned with proof theory and that fixpoint semantics is a special case of model-theoretics semantics. With this interpretation of operational semantics as syntax and fixpoint semantics as semantics, the equivalence of operational and fixpoint semantics becomes a special case of Gödel's completeness theorem.*
>
> [vEK76, p.734]

A situação dos programas positivos, do ponto de vista semântico, é entendida fazendo-se uso da teoria dos operadores monotônicos sobre reticulados completos. O *operador de consequência imediata* (T_P) é definido sobre as partes da *base de Herbrand* (B_P) do programa P e simboliza todas as possibilidades de aplicação das regras sobre os fatos de entrada. Pelo fato de ser monotônico, T_P apresenta propriedades interessantes, como a presença de pontos fixos máximos e mínimos. Tais pontos fixos funcionam como boas semânticas para programas positivos.

1.2.4 Cálculo com Negações

Apesar do fragmento das cláusulas de Horn da lógica de primeira ordem poder ser usado, enquanto linguagem de programação, para definir qualquer função computável, do ponto de vista da representação de informações sua sintaxe é limitada.

De um programa positivo (ou de um que não admita literais negadas nas cabeças de suas regras) nenhuma literal negada pode ser deduzida logicamente, tendo em vista que a base de Herbrand completa é um contra-modelo para tais informações negativas.

Vemos surgir a necessidade de se lidar com deduções de informações negativas, fora dos parâmetros da lógica clássica, relacionada inicialmente com regras de inferência na área de bases de dados:

> *In logical systems, negative information is treated in the same way as positive information both in representation and manipulation. Yet, in some applied domains, especially in data bases (...), negative information is represented implicitly. More precisely, a negative ground literal $\neg L$ is assumed to be true if L fails to be proved.*
>
> *This implicit representation has several terminologies. It is called "convertion for negative information representation" by Nicolas and Gallaire [16], the "closed world assumption (CWA)" by Reiter [17], and*

[16] [NG78]
[17] [Rei78]

"interpreting negation as failure" as discussed in the chapter by Clark [18].

[GMN78, p.23]

Vemos então expressa a necessidade de duas maneiras de deduzir negações, uma explícita, comum nos sistemas lógicos, e a outra implícita, para o tratamento de informação negativa em bases de dados e sistemas de representação de conhecimento. Com o surgimento dessas novas formas de estrair consequências (agora, não necessariamente lógicas), o equilibrío entre as interpretações declarativa e operacional encontrado no caso clássico sofre uma perturbação para as novas abordagens:

> *According to a well-known slogan,* Algorithm *equals* Logic *plus* Control. *The Ideal of logic programming is to write programs directly in logic so as to have a clear-cut declarative interpretation, immediately available to the practical programmer, and to leave the control to the implementation of the interpreter. This ideal is satisfactorily realized for the case of positive programs and queries (...)* ***The way in which logic programming makes use of negation destroys this simple picture.***
>
> [Doe94, §8, p.165]

Para dar conta dessas novas propostas, faz-se necessário diferenciar dois tipos de negações. Por um lado, é possível considererar um fato como sendo falso se não se tem indícios de sua veracidade. Um exemplo comum são as informações sobre horários de ônibus; se não consta algum ônibus saindo em determinado horário, pode-se enteder – e acertadamente – que não sairá ônibus algum naquele horário. Não é necessário que a tabela registre todos os horários em que não há ônibus. Tal negação é conhecida como *negação fraca*, *negação por falha* ou mesmo *negação default*, por admitir a derivação de fatos como negativos caso não tenha sido possível demonstrá-los positivamente, caso tenha resultado em falha as tentativas de sua demonstração. Sua dual, que demanda uma prova explícita para sua dedução, é chamada de *negação forte* ou *negação explícita*.

Um exemplo interessante pode ser encontrado no campo do direito. Considere como primitivo o predicado monádico *é inocente*. Em regimes de excessão, não é incomum que as pessoas sejam tomadas por culpadas, isto é, *não inocentes*, até que provem o contrário. Ou seja, na falta de informações mais detalhadas, caso não seja possível a demonstração de inocência, afirma-se a *não inocência*. Já o *princípio*

[18] [Cla78]

da presunção de inocência, afirmado em nossa Constituição e na Declaração dos Direitos do Homem e do Cidadão, diz o oposto: todo acusado deve ser considerado inocente até que se prove o contrário. Logo, em ditaduras, os julgamentos de *culpa* (entendida formalmente como *não inocência*) são feitos com a interpretação fraca da negação.

Evidentemente, poderia-se inverter a análise se o predicado tomado como primitivo fosse o que afirmasse formalmente a culpa. Nesse caso, regimes de excessão fariam uso da interpretação forte da negação para afirmar a inocência (entendida formalmente como *não culpa*).

Essas mesmas técnicas e estudos que surgiram no contexto da programação lógica (que se aproveitou por sua vez dos avanços da área da prova automatizada de teoremas) mostraram-se – via considerações sobre a negação – um terreno fértil para se trabalhar uma área nova, que veio a se constituir ao longo da decada de 80: as lógicas não monotônicas. Tais lógicas foram investigadas principalmente por pesquisadores relacionados à área de Representação de Conhecimento, um ramo da Inteligência Artificial:

> *For several years default logic and logic programming have been developed by disjoint groups of researchers and there have been few interactions between these two fiels. Only quite recently, in the late 1980s, it was realized that the way in which logic programming interprets negation leads to nonmonotonic systems. Subsequently, connections between "negation as failure to prove" principle of logic programming and default and autoepistemic logics were established.*
>
> [MT93, p.141]

Quanto ao raciocínio não monotônico, ele é uma característica importante do raciocínio do senso comum:

> *Non-monotonic reasoning grew out of attempts to capture the essential aspects of commonsense reasoning. It resulted in a number of important formalisms, the most known of them being the* circumscription method *of McCarthy* [19]*, the* default theory *of Reiter* [20] *and* autoepistemic logic *of Moore* [21].
>
> [AB94, p.1]

[19] [McC80]
[20] [Rei80]
[21] [Moo83]

Na medida em que foi se tornando necessário calcular predicados negativos, como uma reação à incompletude e à impossibilidade de se realizar um tal intento do ponto de vista da lógica de primeira ordem clássica (do ponto de vista computacional e de expressividade), ampliaram-se as análises semânticas que permitissem computar negações a partir de programas positivos.

Ao se analisar o raciocínio do senso comum, normalmente usando como modelo alguma linguagem formal, tem de se ir além das inferências que levam em conta todos os modelos possíveis. Deve-se considerar que o senso comum se caracteriza justamente por focar em certas realidades vistas como mais *comuns* ou prováveis:

> *Such an approach reflects a common feature of human reasoning agents who usually distinguish between the models and do not regard all of them as equally possible. For example, when a mechanic tries to find an explanation for a problem with a car then he or she excludes some models knowing they are possible but unlikely.*
>
> [MT93, p.1]

Formularam-se algumas propostas iniciais baseadas em características do Modelo de Herbrand minimal do programa em questão, todas não monotônicas. A figura 1.1 sinaliza o alcance das diversas regras (retirada de [Llo93, p.101]).

Temos algumas relações:

- $T_P \uparrow \omega = \bigcup\{T_P^n(\emptyset) | n \in \omega\}$ é o menor ponto fixo de T_P e também o conjunto de todos os átomos de B_P que são consequência de P.

- $mpf(T_P)$ é o ponto fixo máximo de T_P e também o conjunto dos átomos de B_P que são consequência da *complementação* de P, diferindo no geral de $T_P \downarrow \omega = \bigcap\{T_p^n(B_P) | n \in \omega\}$.

É de se notar que a falta de simetria entre o comportamento de $T_P \uparrow \omega$ e de $T_P \downarrow \omega$ está relacionada à impossibilidade de se computar na parte negativa das consequências de um programa lógico positivo.

A não monotonicidade dessas regras levou a uma ampliação da análise por meio de operadores de consequência imediata; buscando correlatos nesse tipo de semântica, introduziu complexidade. Semânticas bem fundadas, de modelo estável e toda a análise com lógicas 3 valoradas estão sintonizadas com a teoria destes operadores, que não são mais monotônicos.

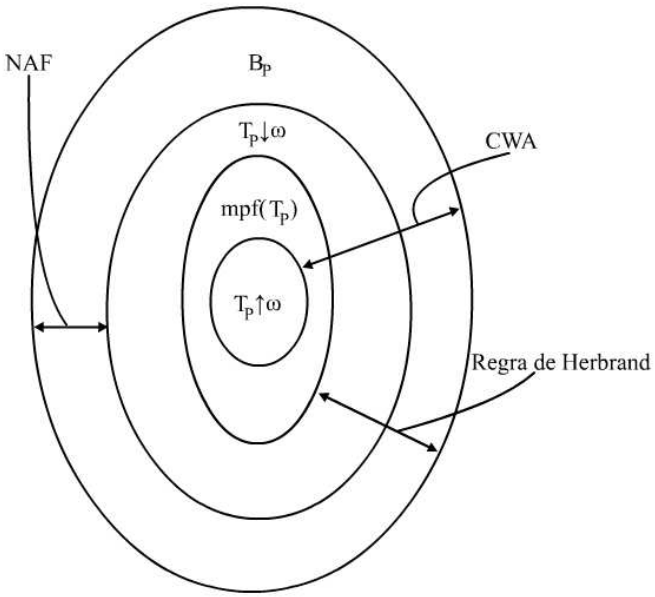

Figura 1.1: Alcance de algumas regras de negação não-monotônicas

1.3 Aspectos Formais

1.3.1 Universo de Herbrand e Base de Herbrand

Definição 1.1 (Universo de Herbrand)**.** A parte algébrica de uma assinatura é o subconjunto de suas funções e constantes. Tendo em vista uma dada linguagem L, definimos seu Universo de Herbrand (U_L) como o conjunto dos termos fechados que podem ser formados das constantes e funções de L. Caso L não tenha constante alguma, consideramos seu Universo de Herbrand como o da extensão L' obtida a partir de L pela introdução de uma constante nova "$*$", $U_L = U_{L\cup\{*\}}$.

■

Definição 1.2 (Base de Herbrand)**.** A Base de Herbrand de uma linguagem (B_L) é o conjunto das sentenças atômicas (isto é, das fórmulas atômicas fechadas) de L ou de $L \cup \{*\}$, caso a assinatura de L careça de símbolos de constantes. Para um programa lógico positivo P sobre uma linguagem L, sua base de Herbrand B_P será B_L.

■

1.3.2 Tipologia dos Programas Lógicos

As diferentes abordagens na literatura se referem aos mesmos conceitos por variadas denominações. Aqui vamos definir claramente o que entendemos pelos nomes que permeiam o campo da programação lógica.

Definição 1.3. Para dar conta dos dois tipos de negação descritas na seção 1.2.4, designaremos a negação *explícita* (*forte*, *monotônica*) por "$\neg$" e a *default* (*fraca*, *não-monotônica*) por "$\sim$".

■

Literais são fórmulas atômicas ou sua negação. No que segue, representaremos fórmulas atômicas por A e literais por L, possivelmente acompanahados de subescritos. Também representaremos a possibilidade de um símbolo colocando-o entre colchetes ("[]"). Por exemplo, podemos representar um literal "L" por "$[\neg]A$".

Definição 1.4. Uma cláula é um conjunto finito de literais e sempre é entendendida como a disjunção de seus elementos:

$$L_1 \vee \ldots \vee L_n$$

e, para o caso em que $n = 0$, uma cláusula é entendida como uma contradição, e é representada simbolicamente por:

□

■

Programas Positivos

Definição 1.5. Um programa positivo é um conjunto de fórmulas da forma:

$$A_0 \leftarrow A_1 \wedge \ldots \wedge A_n$$

sendo que tais fórmulas são denominadas regras e seu conjunto será sempre entendido como a conjunção de seus elementos. As regras estão implicitamente quantificadas sobre toda variável livre que nelas apareça:

$$\forall \vec{x} \, (A_0 \leftarrow A_1 \wedge \ldots \wedge A_n)$$

Denomina-se A_0 cabeça da regra e $A_1 \wedge \ldots \wedge A_n$, seu corpo. Permitem-se regras com corpo vazio, sendo estas chamadas de fatos:

$$A_0 \leftarrow$$

■

Negação *Default* e os Programas Lógicos Gerais

Definição 1.6. Um programa lógico geral permite a negação no corpo de suas regras, normalmente entendida em seu sentido *default*:

$$A_0 \leftarrow L_1, \ldots, L_n$$

Ou representada de outra maneira:

$$A_0 \leftarrow [\sim]A_1, \ldots, [\sim]A_n$$

■

Negação Explícita e os Programas Lógicos Estendidos Definidos

Definição 1.7. Os programas estendidos definidos permitem negações em qualquer parte, entendidas em seu sentido explícito:

$$L_0 \leftarrow L_1, \ldots, L_n$$

$$[\neg]A_0 \leftarrow [\neg]A_1, \ldots, [\neg]A_n]$$

■

Programas Lógicos Estendidos

Definição 1.8. Por último, o caso mais geral, os programs lógicos que permitem tanto o uso da negação explícita (em qualquer lugar), quanto da *default* (apenas no corpo das regras):

$$L_0 \leftarrow [\sim]L_1, \ldots, [\sim]L_n$$

Ou:

$$[\neg]A_0 \leftarrow [\sim][\neg]A_1, \ldots, [\sim][\neg]A_n$$

■

1.3.3 Considerações Semânticas sobre Programas Lógicos

Definição 1.9 (Modelo de Herbrand)**.** Um modelo de Herbrand para um programa positivo P (ou conjunto de cláusulas) é um subconjunto de sua base de Herbrand B_P que, interpretado como verdadeiro, induz uma bivaloração que satisfaça todas as regras de P.

Teorema 1.10 (Teorema de Herbrand Clássico)**.** *Uma das possíveis formulações para o teorema de Herbrand clássico, restrito aos conjuntos de cláusulas, diz que um tal conjunto tem modelo exatamente quando tem um modelo de Herbrand. Evidentemente, os programas positivos são casos particulares de conjuntos de cláusulas.*

Definição 1.11 (Modelo de Herbrand Mínimo)**.** O modelo de Herbrand mínimo de um conjunto C de cláusulas é a interseção de todos os seus modelos de Herbrand:

$$\mu_C = \bigcap \{M \subseteq B_C \mid M \vDash C\}$$

Pelo teorema de Herbrand, temos que o conjunto de sentenças atômicas consequências lógicas de um conjunto de cláusulas é, exatamente, seu modelo de Herbrand mínimo:

$$A \in \mu_c \qquad \Longleftrightarrow \qquad C \vDash A$$

■

Quando nos pomos a analisar o cálculo clássico dos programas lógicos positivos, vemos que duas espécies de *incompletude* estão agindo.

De um lado, só podemos decidir positivamente se um átomo da base de Herbrand pertence ao conjunto das consequências lógicas do programa (considerando o programa na linguagem da base em questão): existem programas positivos para os quais o conjunto $\mu_P \subseteq B_P$ é apenas recursivamente enumerável. Ou em outras palavras, qualquer proposta de cálculo que venha a lidar com esse problema, se quiser ser completa com relação a dedução de átomos (deduzir todos que são de fato consequência lógica), não pode ser finitariamente decidível (chegar sempre a uma resposta negativa em um tempo finito).

Ou seja, qualquer análise semântica que tenha pretensões de caracterizar a parte positiva da consequência clássica dos programas lógicos tem de levar em conta a problemática da indecidibilidade.

Por outro lado, esses mesmos modelos sintáticos indicam a existência de contramodelos para toda literal negativa: nenhuma fórmula atômica negada é consequência lógica de um programa positivo (ou mesmo de um programa que permita literais

negativas no corpo de suas cláusulas). Isso porque B_P é sempre um modelo de P que falsifica todas as literais negadas de sua linguagem.

Definição 1.12 (Operador de Consequência Imediata)**.** Pode-se associar, a um programa positivo P, um operador monotônico T_P definido sobre o conjunto das partes da base de Herbrand B_P de P. Tal operador, chamado de *operador de consequência imediata*, associa a cada subconjunto $C \subseteq B_P$ o subconjunto resultante da aplicação de *modus ponens* de todas as possíveis instâncias fechadas das cláusulas de P aos elementos de C.

■

Teorema 1.13 (Modelo Mínimo)**.** *O modelo de Herbrand mínimo é o menor ponto fixo de T_P e pode ser caracterizado por:*

$$\mu_P = \bigcup_{n \in \omega} T_P^n(\emptyset)$$

Capítulo 2

Abordagens Usuais à Programação Lógica Paraconsistente

No caso clássico, a relação de *consequência lógica* serviu como substrato, através de métodos da teoria da prova, para o estabelecimento de uma linguagem voltada a aplicações em inteligência artificial (processamento de linguagem natural e representação do conhecimento) e inferência para um sistema abrangente, o **PROLOG**.

A programação lógica paraconsistente, até agora, constituiu-se por uma série de resultados ancorados em métodos eminentemente semânticos, fazendo uso de variações da interpretação por pontos fixos. A regra é a despreocupação com uma lógica mais geral que sirva de ambiente para um procedimento de resolução única, mantendo o foco em métodos pautados por modelos particulares.

Consoante com a admissão da negação, outra maneira de aumentar a expressividade da programação lógica é trabalhar no contexto das lógicas multi-valoradas. E foi justamente nesse meio que surgiu a necessidade de uma programação lógica paraconsistente. Em [Bel77], introduz-se uma lógica quatro-valorada que se mostrou um caso particular de birreticulado, uma família de valores de verdade apropriados para lidar com informações conflitantes ou incompletas. Tal lógica foi usada pela primeira vez para entender semanticamente programas lógicos paraconsistentes em [BS89][1].

Melvin Fitting, em [Fit91], extendeu a interpretação por pontos fixos de forma a admitir operadores que atribuíssem valores-de-verdade de qualquer birreticulado aos átomos da base de Herbrand.

Não tardou o reconhecimento da análise paraconsistente aplicada ao campo da programação lógica:

> *Wagner has been a supporter of the introduction of explicit negation and constructive based paraconsistent logics in logic programming*[2]*. His main motivating works are Nelson's constructive Logic N with "strong negation"*[3]*, Belnap's system B*[4] *and Levesque's vivid reasoning research programme*[5]*.*
>
> [DP98, p.258]

O caráter paraconsistente da negação explícita aparece quando, por exemplo, considera-se a possibilidade de literais negativas encabeçando regras: não é mais possível construir contra-modelos do programa para todos os fatos negativos. Também não é possível retomar a lógica clássica. Por exemplo, o seguinte programa:

$$\neg A \leftarrow A$$

$$A \leftarrow$$

não tem modelos clássicos (é necessariamente explosivo), mas admite modelos paraconsistentes onde o átomo A é apenas um fato contraditório.

Recentes pesquisas no campo da programação lógica ainda a afirmam como uma boa técnica para representação de raciocínio não monotônico e, dada sua relevância para outros campos da Inteligência Artificial, argumentam sobre a necessidade de sua integração com a paraconsistência:

[1] Conforme se afirma em [DP98, p.251].
[2] [PW90], [PW91], [Wag91a], [WAG91b], [Wag93] e [Wag94]
[3] [Nel49]
[4] [Bel77]
[5] [Lev86] e [Lev88]

(...) the most important non-monotonic formalisms (...) have a counterpart semantics in the logic programming side. Moreover, logic programming has turned out to be vehicle for implementing and exploring other important aspects of Artificial Intelligence, such as as updates and belief revision. Therefore, it is not strange that a lot of work in the logic programming community has been carried out in order to understand the integration of paraconsistent reasoning with logic programming, in preparation for an applicational and implementational rôle of greate potential, now emerging.

[DP98, p.241]

Com a necessidade de dois tipos de negação cada vez mais evidente, surgiram abordagens que dessem conta, em um mesmo formalismo, de ambas. A negação fraca, notadamente não-monotônica, mantem seu carácter explosivo nas semânticas desenvolvidas. Já a explícita, monotônica, demanda modelos paraconsistentes:

The idea of introducing paraconsistent reasoning in logic programming is farly recent. (...) Their intuitions and results have been brought to the logic programming setting mainly by Blair, Pearce, Subrahmanian and Wagner[6]*. The introduction of a non-classical explicit form of negation in logic programming led other researchers to addres this issue as well, namely Przymusinski, Sakama and ourselves, with respect to extensions of both well-founded and answer sets semantics (...).*

[DP98, p.242-243]

Nesse mesmo artigo, os autores identificam uma semântica que "é o denominador comum de quase todas as outras semânticas" examinadas por eles em seu *survey*. "Em particular está relacionanda com os programas generalizados de Horn de Blair e Subrahmanian[7], com os programas lógicos com negação forte de Wagner[8], e com o sistema construtivo paraconsitente N^- de Almukdad e Nelson[9]", cf. [DP98, p.248].

Tal semântica (Definição 3, em [DP98, p.249]), está definida para os programas lógicos estendidos definidos (Definição 1.7) e se baseia na semântica usual para programas positivos (Definição 1.5), construída fazendo uso de pontos fixos para operadores de consequência imediada associados aos programas (Definição 1.12):

[6] [BS89], [PW90], [PW91], [Pea92], [Pea93], [Wag93] e [Wag94].
[7] [BS89]
[8] [Wag93] e [Wag94]
[9] [AN84]

Definição 2.1. Seja P um programa lógico estendido definido. O modelo M_P do programa é obtido da seguinte forma:

1. Transforme o programa P em um programa positivo $P^{\neg}$ renomeando as suas literais a e $\neg a$, respectivamente, por a^p e a^n.

2. Seja $M^{\neg}$ o modelo minimal usual de $P^{\neg}$, associado ao operador $T_{P^{\neg}}$.

3. Então, para obter M_P, reverta a transformação do primeiro passo, transformando $a^P \in M^{\neg}$ ($a^n \in M^{\neg}$) em $a \in M_P$ ($\neg a \in M_P$).

■

Por estar definida por meio dos operadores clássicos, tal semântica é monotônica. É como se admitissem, na base de Herbrand associada a um programa, literais negativas também. Veja discução na Seção 6.

Capítulo 3

Lógicas da Inconsistência Formal

Lógicas paraconsistentes foram inicialmente propostas por Jaśkowski [Jas48], Nelson [Nel59] e da Costa [dC63] de forma a que pudessem acomodar teorias contraditórias, embora não triviais[1].

As Lógicas da Incosistência Formal (**LFI**'s) são lógicas paraconsistentes que internalizam as noções de consistência e inconsistência por meio de conectivos. Veja [CCM07] para uma referência completa sobre o assunto.

[1] [CCM07, p.9]

3.1 Sintaxe e Cálculos *à Hilbert*

3.1.1 Sintaxe

Serão tratados sobre a égide da *sintaxe* os problemas de se definir assinatura, termo, fórmula e linguagem. Faremos uso de uma notação inspirada nas definições de linguagens formais – de uso corrente na computação e linguística.

As linguagens aqui estudadas são de primeira ordem, finitárias[2] e admitem, além de símbolos de funções e predicados com aridade zero, uma quantidade arbitrária dos mesmos e enumeráveis símbolos para variáveis, compartilhadas por todas as linguagens que tratarmos.

Definição 3.1 (Assinatura)**.** Usaremos letras gregas maiúsculas (Σ, Δ etc.) para representar as assinaturas sobre as quais se constituirão as linguagens de que trataremos. Quando alguma definição for relativa a certa assinatura, aparecerá a metavariável que a designa em sobrescrito (T_Σ, $Pred_\Sigma$).

■

Por estarmos lidando com cálculos que fazem uso de diferentes conectivos lógicos, faz-se necessário separar uma parte comum a todos, para que suas diferenças fiquem melhor explicadas.

Definição 3.2 (Assinatura Básica)**.** Usaremos o conceito de *assinatura básica* para designar uma parte comum às assinaturas aqui discutidas: seus símbolos de predicados e funções. Às metavariáveis de assinaturas, aplicaremos superescritos para designar o acréscimo de símbolos lógicos (L_{Σ^+}, Δ° etc.). Chamaremos de *parte básica de uma assinatura* a assinatura básica nela contida (ou seja, a parte básica de Σ^+ é constituída abstraindo-se todos os símbolos que não os de predicados e funções).

■

Por estarmos tratando de linguagens finitárias, predicados e funções têm necessariamente aridade finita. Logo, o conjunto que os engloba pode ser particionado em uma família enumerável de conjuntos com símbolos de mesma aridade.

Definição 3.3. Denotaremos por Var o conjunto das variáveis (que serão as mesmas para todas as linguagens). $Func_\Sigma$ e $Pred_\Sigma$ serão, respectivamente, o conjunto dos símbolos funcionais e de predicados de Σ. As letras F e P designarão as classes

[2] Isto é, todas as expressões têm comprimento finito.

de símbolos com mesma aridade (e.g. F^0_Σ é o conjunto das constantes da assinatura Σ e P^1_Σ, o de seus predicados monádicos). Quando for evidente a assinatura, omitiremos subescritos. Para que fique claro:

$$Func_\Sigma = \dot{\bigcup} \left\{ F^n_\Sigma \,\middle|\, 0 \leq n < \omega \right\}$$

$$Pred_\Sigma = \dot{\bigcup} \left\{ P^n_\Sigma \,\middle|\, 0 \leq n < \omega \right\}$$

■

Definição 3.4 (Termos de uma Assinatura)**.** Associado a uma assinatura Σ temos o conjunto de seus termos, formado a partir das variáveis e símbolos funcionais presentes em sua parte básica:

$$T_\Sigma \ ::= \ Var \mid F^0_\Sigma \mid \ldots \mid F^n_\Sigma(T_\Sigma, \ \ldots \ , \ T_\Sigma) \mid \ldots$$

■

Definição 3.5 (Assinatura Positiva)**.** Denotaremos por Σ^+ a assinatura Σ acrescida dos símbolos para os conectivos $\wedge$, $\vee$ e $\Rightarrow$.

$$\Sigma^+ \ ::= \ Var \mid Func_\Sigma \mid Pred_\Sigma \mid \{\wedge, \vee, \Rightarrow\}$$

■

Definição 3.6 (Assinatura Clássica)**.** Denotaremos por Σ^C a assinatura que extende Σ^+ pela introdução de um símbolo para negação: $\neg$.

■

Definição 3.7 (Assinatura Paraconsistente)**.** Denotaremos por Σ°

■

Definição 3.8 (Linguagem de Primeira Ordem)**.** A partir de uma assinatura Σ é possivel definir as fórmulas de uma linguagem de primeira ordem. Conforme os conectivos presentes em Σ definiremos diferentes fórmulas.

$$L_{\Sigma^+} \ ::= \ P^0_{\Sigma^+} \mid \ldots \mid P^n_{\Sigma^+}(T_{\Sigma^+}, \ \ldots \ , \ T_{\Sigma^+}) \mid \ldots$$

■

Definição 3.9 (Sentenças)**.** Para uma linguagem de primeira ordem L, o conjunto de suas sentenças, isto é, de todas as suas fórmulas sem nenhuma variável livre, será denotado por S_L.

Vamos agora retomar uma definição apresentada em 1.3, o conceito de Base de Herbrand (Definição 1.2).

Definição 3.10 (Base de Herbrand)**.** Para uma linguagem de primeira ordem L, caso não tenha nenhuma constante, tomaremos $L^\star$

Definição 3.11 (Linguagem Positiva)**.** Dizemos que uma linguagem é positiva se tem contida, ao menos, entre sua assinatura, uma parte positiva.

■

Definição 3.12 (Linguagem Clássica)**.** Dizemos que uma linguagem é clássica se tem contida, ao menos, entre sua assinatura, uma parte clássica.

■

Definição 3.13 (Linguagem paraconsistente)**.** Diremos que uma linguagem L é paraconsistente se tiver, pelo menos, em sua assinatura uma parte paraconsistente.

■

Logo, as classes dos diferentes tipos de linguagens formam uma cadeia, com as linguagens positivas contidas nas clássicas, que, por sua vez, estão contidas nas paraconsistentes.

3.1.2 Formulação Hilbertiana de QmbC e QmCi

QmbC e **QmCi** são versões quantificadas, respectivamente, de **mbC** e **mCi**, **LFI**'s fundamentais por admitirem

Essas definições são, no caso proposicional, as mesmas de [CCM07] e para as fórmulas quantificadas seguimos de perto [Pod08].

Definição 3.14 (Axiomas Lógicos)**.** Os cálculos tratrados aqui terão como esquemas axiomáticos um subconjunto dos seguintes:

Axiomas positivos:

$$\alpha \Rightarrow \beta \Rightarrow \alpha \qquad \textbf{(Mon)}$$
$$(\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \beta \Rightarrow \gamma) \Rightarrow (\alpha \Rightarrow \gamma) \qquad \textbf{(Trans)}$$
$$\alpha \Rightarrow \beta \Rightarrow (\alpha \wedge \beta) \qquad \textbf{(E0)}$$
$$(\alpha \wedge \beta) \Rightarrow \alpha \qquad \textbf{(E1)}$$
$$(\alpha \wedge \beta) \Rightarrow \beta \qquad \textbf{(E2)}$$
$$\alpha \Rightarrow (\alpha \vee \beta) \qquad \textbf{(Ou0)}$$
$$\beta \Rightarrow (\alpha \vee \beta) \qquad \textbf{(Ou1)}$$
$$(\alpha \Rightarrow \gamma) \Rightarrow (\beta \Rightarrow \gamma) \Rightarrow \Big((\alpha \vee \beta) \Rightarrow \gamma\Big) \qquad \textbf{(Ou2)}$$
$$\alpha \vee (\alpha \Rightarrow \beta) \qquad \textbf{(OuI)}$$

Axioma Clássico

$$\alpha \vee \neg\alpha \qquad \textbf{(TND)}$$

Axiomas Paraconsistentes:

$$\circ\alpha \Rightarrow \alpha \Rightarrow \neg\alpha \Rightarrow \beta \qquad \textbf{(Exp)}$$
$$\neg \circ \alpha \Rightarrow (\alpha \wedge \neg\alpha) \qquad \textbf{(Inc)}$$
$$\circ\neg^n \circ \alpha \qquad \textbf{(Con)}$$

Axiomas Quantificacionais (se t é um termo livre para x em α):

$$\alpha_x[t] \Rightarrow \exists x\, \alpha \qquad \textbf{(}\exists\textbf{-Ax)}$$
$$\forall x\, \alpha \Rightarrow \alpha_x[t] \qquad \textbf{(}\forall\textbf{-Ax)}$$

■

Definição 3.15 (Regras de inferência)**.** Todos os cálculos terão como regras de inferência

1. Modus Ponens:
$$\alpha,\ \alpha \Rightarrow \beta \ \vdash\ \beta \qquad \textbf{(MP)}$$
2. Introdução de existencial (se x não ocorre livre em β):
$$\alpha \Rightarrow \beta \ \vdash\ \exists x\, \alpha \Rightarrow \beta \qquad \textbf{(}\exists\textbf{-In)}$$

3. Introdução de Universal (se x não ocorre livre em α):

$$\alpha \Rightarrow \beta \vdash \alpha \Rightarrow \forall x\, \beta \qquad (\forall\textbf{-In})$$

■

Definição 3.16. QmbC constitui-se sobre uma assinatura paraconsistente Σ° com todos os Axiomas Positivos da Definição 3.14, os Quantificacionais e, dos Paraconsistentes, apenas **Exp**.

■

Definição 3.17. QmCi extendende **QmbC** pela adição dos Axiomas Paraconsistentes **Inc** e **Con**.

■

3.1.3 Alguns Teoremas

Nesta seção serão enunciados resultados referentes a uma teoria T qualquer em **QmbC** ou **QmCi**. Logo, quando aparecer $\vdash$, estarão sendo representados tanto $T \vdash_{\mathbf{QmbC}}$ quanto $T \vdash_{\mathbf{QmCi}}$.

Teorema 3.18. *Para qualquer fórmula α, $\vdash \alpha \Rightarrow \alpha$.*

Demonstração.

$\vdash \alpha \Rightarrow \alpha \Rightarrow \alpha$	**Mon**
$\vdash \alpha \Rightarrow (\alpha \Rightarrow \alpha) \Rightarrow \alpha$	**Mon**
$\vdash (\alpha \Rightarrow (\alpha \Rightarrow \alpha)) \Rightarrow (\alpha \Rightarrow (\alpha \Rightarrow \alpha) \Rightarrow \alpha) \Rightarrow (\alpha \Rightarrow \alpha)$	**Trans**
$\vdash \alpha \Rightarrow \alpha$	**MP** (2x)

□

Teorema 3.19 (Dedução por Casos)**.** *Podemos supor*

Demonstração. Casos (**Ou2**, **TND** e **MP**) □

Teorema 3.20 (Generalização)**.** *Podemos introduzir quantificações arbritárias em quaisquer teoremas:*

$$\frac{\vdash \phi}{\vdash \forall x\, \phi}$$

Demonstração.

$\vdash \phi$	Hipótese
$\vdash \phi \Rightarrow (\neg\forall x\ \phi \Rightarrow \phi)$	**Mon**
$\vdash \neg\forall x\ \phi \Rightarrow \phi$	**MP**
$\vdash \neg\forall x\ \phi \Rightarrow \forall x\ \phi$	**$\forall$-In**
$\vdash \forall x\ \phi \Rightarrow \forall x\ \phi$	Teorema 3.18
$\vdash \forall x\ \phi$	Teorema 3.19

□

Teorema 3.21 (Teorema da Dedução)**.**

Demonstração. □

Teorema 3.22 (Teorema das Constantes)**.**

Demonstração. □

Teorema 3.23 (Regra de Conversão α)**.** *Para uma fórmula ϕ arbitrária e z uma variável livre para x em ϕ e totalmente indicada em $\phi_x[z]$:*

$$\vdash \exists x\ \phi \Rightarrow \exists z\ \phi_x[z]$$

$$\vdash \forall x\ \phi \Rightarrow \forall z\ \phi_x[z]$$

Demonstração. □

Lema 3.24. *Agora vamos demonstrar variações das regras* ***$\exists$-In*** *e* ***$\forall$-In****. Os dois resultados são essenciais para a completude (Capítulo 3.3), mais precisamente na demonstração de que teorias não triviais podem ser extendidas* conservativamente *a teorias de Henkin não triviais (Teorema 3.45). Se x não ocorre livre em λ e ψ, pode-se levar a cabo as seguintes inferências:*

1.

$$\frac{\vdash (\phi \Rightarrow \lambda) \Rightarrow \psi}{\vdash (\forall x\ \phi \Rightarrow \lambda) \Rightarrow \psi}$$

1'

$$\frac{\vdash (\phi \Rightarrow \lambda)}{\vdash (\forall x\, \phi \Rightarrow \lambda)}$$

2.

$$\frac{\vdash (\lambda \Rightarrow \phi) \Rightarrow \psi}{\vdash (\lambda \Rightarrow \exists x\, \phi) \Rightarrow \psi}$$

2'

$$\frac{\vdash (\lambda \Rightarrow \phi)}{\vdash (\lambda \Rightarrow \exists x\, \phi)}$$

Demonstração. □

Teorema 3.25 (Teorema das Variantes)**.**

Demonstração. □

3.2 Valorações e estruturas paraconsistentes

Definição 3.26 (Estrutura Básica)**.** Diz-se que um conjunto A constitui o domínio de uma estrutura básica sobre a linguagem de primeira ordem L se não é vazio e temos funções e predicados definidos sobre A para cada símbolo da parte básica (Definição 3.2) da assinatura Σ de L. A estrutura básica $\mathfrak{A}$ de L se constituirá, então, pelo domínio A juntamente com a função que interpreta os símbolos de predicados e funções:

$$\mathfrak{A} = \langle A, I_{\mathfrak{A}} \rangle$$

$$I_{\mathfrak{A}} : Func_{\Sigma} \cup Pred_{\Sigma} \rightarrow Func(A) \cup Pred(A)\ ,$$

sendo $Func(A)$ o conjunto de todas as possíveis funções de aridade arbritária com parâmetros e contradomínio em A e $Pred(A)$ o conjunto de todos os possíveis predicados (também de aridade arbritária) sobre A.

■

Definição 3.27 (Linguagem Diagrama)**.** Para uma linguagem L e uma estrutura básica $\mathfrak{A}$ para essa linguagem, é possível extender L com constantes que façam o papel de nomes para os elementos do domínio de $\mathfrak{A}$. Para tanto, basta adicionar uma nova constante em L para cada elemento do domínio; a esta extensão damos o nome de linguagem diagrama de $\mathfrak{A}$, $L(\mathfrak{A})$. Podemos extender também, naturalmente, a função de interpretação $I_{\mathfrak{A}}$ de $\mathfrak{A}$ à nova linguagem e considerar $\mathfrak{A}$ como uma estrutura de sua própria linguagem diagrama. Ou seja, partindo da estrutura inicial

$$\mathfrak{A} = \langle A, I_{\mathfrak{A}} \rangle \quad \text{para} \quad L\,,$$

obtemos

$$\widehat{\mathfrak{A}} = \langle A, \widehat{I_{\mathfrak{A}}} \rangle \quad \text{para} \quad L(\mathfrak{A})\,.$$

■

Toda vez que estivermos discutindo uma estrutura para certa linguagem, assumiremos implicitamente que sua função de interpretação já está extendida para os elementos da linguagem diagrama.

Definição 3.28 (Estrutura)**.** Uma estrutura é qualquer par $\langle \mathfrak{A}, v \rangle$, sendo $\mathfrak{A}$ uma estrutura básica sobre alguma linguagem L de primeira ordem e v uma bivaloração $v : S_{L(\mathfrak{A})} \to \{0, 1\}$ que obedeça a seguinte condição:

$$v(P(a_1, \ldots, a_n)) = 1 \quad \Longleftrightarrow \quad \langle\, I_{\mathfrak{A}}(a_1),\ \ldots\ , I_{\mathfrak{A}}(a_n)\,\rangle \in I_{\mathfrak{A}}(P) \qquad (vPred)$$

Quando não houver risco de confusão, vamos designar a estrutura $\langle \mathfrak{A}, v \rangle$ por $\mathfrak{A}$, v por $v_{\mathfrak{A}}$ e, se quisermos nos referir a estrutura básica sobre a qual v está definida, falaremos da *pré-estrutura* de $\mathfrak{A}$.

■

Definição 3.29 (Satisfatibilidade)**.** Dizemos que uma estrutura $\mathfrak{A}$ satisfaz a sentença $\varphi \in S_{L(\mathfrak{A})}$:

$$\mathfrak{A} \vDash \varphi$$

se

$$v_{\mathfrak{A}}(\varphi) = 1$$

Podemos estender essa noção para um conjunto Γ de sentenças. Isto é, $\mathfrak{A} \vDash \Gamma$ se e somente se, para toda $\gamma \in \Gamma$, $\mathfrak{A} \vDash \gamma$.

■

Definição 3.30 (Consequência Lógica)**.** Dada uma classe $\mathfrak{C}$ de estruturas de primeira ordem, dizemos que φ é consequência lógica de um conjunto Γ, em relação a essa classe, se e somente se, para toda estrutura $\mathfrak{A}$ sobre alguma linguagem L de primeira ordem que seja linguagem de Γ e φ, toda vez que $\mathfrak{A} \vDash \Gamma$, tenhamos $\mathfrak{A} \vDash \varphi$. Em símbolos:

$$\Gamma \vDash_{\mathfrak{C}} \varphi$$

■

Definição 3.31 (Valoração Positiva)**.** Dada uma linguagem positiva L (Definição 3.11) de assinatura Σ e uma estrutura $\mathfrak{A}$ para essa linguagem, dizemos que $v_{\mathfrak{A}}$ é uma *valoração positiva* se obedecer às seguintes condições, com $P \in P^n_\Sigma$, $\alpha, \beta \in S_{L(\mathfrak{A})}$ e a_i nomes para indivíduos em $L(\mathfrak{A})$:

- Regras proposicionais:

$$\begin{aligned}
v(\alpha \vee \beta) = 1 &\iff v(\alpha) = 1 \ \textbf{ou}\ v(\beta) = 1 && (vOu)\\
v(\alpha \wedge \beta) = 1 &\iff v(\alpha) = 1 \ \textbf{e}\ v(\beta) = 1 && (vE)\\
v(\alpha \to \beta) = 1 &\iff v(\alpha) = 0 \ \textbf{ou}\ v(\beta) = 1 && (vImp)
\end{aligned}$$

- Regras quantificacionais:

$$\begin{aligned}
v(\exists x\phi) = 1 &\iff v(\phi_x[a]) = 1 \ \textbf{p\algum individuo } a \textbf{ de } L(\mathfrak{A}) && (vEx)\\
v(\forall x\phi) = 1 &\iff v(\phi_x[a]) = 1 \ \textbf{p\todo individuo } a \textbf{ de } L(\mathfrak{A}) && (vUni)
\end{aligned}$$

■

Definição 3.32 (Valoração Paraconsistente)**.** Dada uma uma valoração positiva $v : S_{L^\circ(\mathfrak{A})} \to \{0, 1\}$ sobre a estrutura $\mathfrak{A}$, para uma linguagem paraconsistente L° (Definição 3.13), dizemos que essa é uma valoração paraconsistente se, além da condições da Definição 3.31, obedecer a um subconjunto dos ítens abaixo, com $\alpha, \beta, \exists x\phi, \forall x\phi \in S_{L^\circ(\mathfrak{A})}$:

$$\begin{aligned}
v(\alpha) = 0 &\implies v(\neg\alpha) = 1 && (vNeg)\\
v(\circ\alpha) = 1 &\implies v(\alpha) = 0 \ \textbf{ou}\ v(\neg\alpha) = 0 && (vCon)\\
v(\neg \circ \alpha) = 1 &\implies v(\alpha) = 1 \ \textbf{e}\ v(\neg\alpha) = 1 && (vInc)\\
v(\circ\neg^n \circ \alpha) = 1 &\quad \textbf{(para todo } n \geq 0\textbf{)} && (vCCon)
\end{aligned}$$

■

Definição 3.33 (Estrutura Paraconsistente)**.** Uma estrutura paraconsistente é uma estrutura $\mathfrak{A}$, com $v_{\mathfrak{A}}$ uma valoração paraconsistente definida sobre $\mathfrak{A}$.

■

Definição 3.34 (Estrutura **QmbC**)**.** Uma estrutura para **QmbC** é uma estrutura paraconsistente que obedeça aos ítems *vNeg* e *vCon* da Definição 3.32.

■

Definição 3.35 (Estrutura **QmCi**)**.** As estruturas para **QmCi** são estruturas para **QmbC** que obedecem, além das condições especificadas na Definição 3.34, aos itens *vInc* e *vCCon* da Definição 3.32.

■

Teorema 3.36. *A Definição 3.32 continua definindo a mesma classe de estruturas se trocarmos:*

$$v(\alpha) = 0 \quad \Longrightarrow \quad v(\neg\alpha) = 1 \qquad (vNeg)$$

por

$$v(\neg\alpha) = 0 \quad \Longrightarrow \quad v(\alpha) = 1 \qquad (vNeg')$$

Ou:

$$v(\circ\alpha) = 1 \quad \Longrightarrow \quad v(\alpha) = 0 \ \textbf{\textit{ou}} \ v(\neg\alpha) = 0 \qquad (vCon)$$

por

$$v(\alpha) = 1 \ \textbf{\textit{e}} \ v(\neg\alpha) = 1 \quad \Longrightarrow \quad v(\circ\alpha) = 0 \qquad (vCon')$$

Ou ainda, para estruturas que obedeçam à condição vCCon da Definição 3.32:

$$v(\neg \circ \alpha) = 1 \quad \Longrightarrow \quad v(\alpha) = 1 \ \textbf{\textit{e}} \ v(\neg\alpha) = 1 \qquad (vInc)$$

por

$$v(\alpha) = 0 \ \textbf{\textit{ou}} \ v(\neg\alpha) = 0 \quad \Longrightarrow \quad v(\circ\alpha) = 1 \qquad (vInc')$$

Demonstração. Aplicação direta da contrapositiva (que vale para a metalinguagem). □

Sumarizando as definições das estruturas paraconsistentes, temos dois corolários:

Corolário 3.37. *As estruturas para* **QmbC** *são, exatamente, aquelas estruturas paraconsistentes* $\mathfrak{A}$ *que obedecem às seguintes propriedades:*

$$v_{\mathfrak{A}}(\alpha) = 0 \quad \Longrightarrow \quad v_{\mathfrak{A}}(\neg\alpha) = 1 \tag{3.1}$$

$$v_{\mathfrak{A}}(\circ\alpha) = 1 \quad \Longrightarrow \quad v_{\mathfrak{A}}(\alpha) = 0 \ \textbf{\textit{ou}} \ v_{\mathfrak{A}}(\neg\alpha) = 0 \tag{3.2}$$

Demonstração. Trivial. □

Corolário 3.38. *As estruturas para* **QmCi** *são exatamente as estruturas paraconsistentes* $\mathfrak{A}$ *que obedecem a:*

$$v_{\mathfrak{A}}(\alpha) = 0 \quad \Longrightarrow \quad v_{\mathfrak{A}}(\neg\alpha) = 1 \tag{3.3}$$

$$v_{\mathfrak{A}}(\circ\alpha) = 1 \quad \Longleftrightarrow \quad v_{\mathfrak{A}}(\alpha) = 0 \ \textbf{\textit{ou}} \ v_{\mathfrak{A}}(\neg\alpha) = 0 \tag{3.4}$$

$$v_{\mathfrak{A}}(\circ\neg^{n}\circ\alpha) = 1 \quad \textbf{\textit{(para todo}}\ n \geq 0\textbf{\textit{)}} \tag{3.5}$$

Demonstração. Consequência imediata da Definição 3.35 e do Teorema 3.36. □

Tanto nas estruturas para **QmbC** quanto para **QmCi**, a negação ($\neg$) e o conectivo de consistência ($\circ$) não apresentam uma semântica composicional. No entanto, diferentemente do que ocorre em **QmbC**, o valor de verdade de "$\circ\alpha$" está *em função* dos de "α" e "$\neg\alpha$" em **QmCi**.

Agora, vamos traçar alguns paralelos entre as estruturas no sentido clássico e as paraconsistentes.

Definição 3.39 (Valoração Clássica)**.** Uma valoração clássica é uma valoração positiva $v : S_{L(\mathfrak{A})} \to \{0, 1\}$ sobre $\mathfrak{A}$, sendo L uma linguagem clássica (Definição 3.12), que verifica também:

$$v(\alpha) = 0 \quad \Longleftrightarrow \quad v(\neg\alpha) = 1 \tag{$vTND$}$$

■

Podemos entender as estruturas paraconsistentes como generalizações das estruturas clássicas, tendo em vista que o comportamento de todos os conectivos são composicionais (o valor de verdade da expressão complexa depende do valor de suas subfórmulas) no panorâma clássico, enquanto nas paraconsistentes isso é um caso particular.

Teorema 3.40. *Dada uma estrutura $\mathfrak{A}$, sobre ela existe uma única valoração clássica.*

Demonstração. Por indução na complexidade das fórmulas. □

Logo, podemos identificar as estruturas, no sentido clássico, com nossas estruturas.

Definição 3.41 (Estrutura Clássica)**.** Uma estrutura $\mathfrak{A}$ é considerada uma estrutura clássica, dado que determina unicamente uma valoração clássica $v : S_{L(\mathfrak{A})} \to \{0, 1\}$.

■

Teorema 3.42. *Duas estruturas $\mathfrak{A}$ e $\mathfrak{B}$ clássicas, de igual assinatura, com mesmo domínio, que interpretem seus símbolos funcionais da mesma maneira e que coincidam sobre as bases de Herbrand (Definição 3.10 ou 1.2) de suas linguagens diagrama:*

$$v_{\mathfrak{A}} \restriction_{B_{L(\mathfrak{A})}} = v_{\mathfrak{B}} \restriction_{B_{L(\mathfrak{B})}}$$

são, na realidade, as mesmas estruturas:

$$\mathfrak{A} = \mathfrak{B}\,.$$

No entanto, podemos ter diferentes estruturas paraconsistentes satisfazendo essas mesmas condições.

Demonstração. A parte correspondente às duas estruturas clássicas é decorrência do Teorema 3.40.

Por outro lado, duas estruturas paraconsistentes que coincidam em uma base de Herbrand em comum às duas devem coincidir apenas em todas as fórmulas geradas a partir das fórmulas da base e das constantes lógicas que têm comportamento composicional, ou seja, $\wedge$, $\vee$, $\Rightarrow$, $\exists$ e $\forall$. Mas, se forem duas estruturas para **QmCi** e, adicionalmente, coincidirem em todas as fórmulas negadas, então coincidem em todas as fórmulas. □

3.3 Uma nova prova de completude das lógicas QmbC e QmCi

Procederemos nesta seção da maneira mais genérica possível, para facilitar a extensão do método para outras **LFI**'s. Na completude clássica, o que se demonstra é que teorias *não contraditórias* têm modelos. Para as lógicas paraconsistentes,

o foco não está na *não contradição*, mas na *não trivialidade*, dado que admitêm inconsistências sem a trivialização (que é regra na lógica clássica). O método clássico vai (com a versão generalizada pelo teorema da compacidade em [Gö30], ou diretamente com a demonstração de [Hen49]), mais ou menos, pelas seguintes linhas:

$$\begin{aligned} \Delta \nvdash \varphi \quad &\Longrightarrow \quad \Delta, \neg\varphi \textit{ não é contraditória} \\ \Delta, \neg\varphi \textit{ não é contraditória} \quad &\Longrightarrow \quad \Delta, \neg\varphi \textit{ é satisfatível} \\ \Delta, \neg\varphi \textit{ é satisfatível} \quad &\Longrightarrow \quad \Delta \nvDash \varphi \end{aligned}$$

Para a completude das **LFI**'s, tomaremos um caminho mais direto, tendo em vista que algumas dessas implicações podem não funcionar (ver a discussão na Seção 6.1). A partir de uma teoria que não deduza uma fórmula φ qualquer (não sendo trivial portanto), encontraremos uma estrutura paraconsistente (Definição 3.33) que não satisfaça φ:

$$\Delta \nvdash \varphi \qquad \Longrightarrow \qquad \Delta \nvDash \varphi$$

Primeiramente, faremos a extensão por constantes de uma teoria qualquer, garantindo a presença de constantes testemunhas e contra-testemunhas (caracterizando uma teoria de Henkin, cf. Definição 3.44). Para realizar esse passo, nada exigimos de tal teoria. Esse passo garante que as regras quantificacionais da Definição 3.31 vão funcionar.

Na sequência, assumindo que uma dada teoria não deduza uma fórmula φ, encontraremos uma extensão maximal sua sobre a relação de derivabilidade – fazendo uso fundamental do axioma da escolha, sob a forma do teorema de Teichmüller-Tukey – que não deduza φ também, conforme o método clássico de Lindenbaum-Asser. Seguimos de perto a demonstração de completude de [Sho67].

3.3.1 Teorias de Henkin

Teorema 3.43. *Qualquer teoria Δ de alguma lógica $\mathfrak{L}$, considerada sobre uma linguagem postiva L que tenha alguma constante, induz uma estrutura (Definição 3.28) $\mathfrak{D}$ tal que, para toda $\varphi \in S_L$:*

$$\mathfrak{D} \vDash \varphi \qquad \Longleftrightarrow \qquad \Delta \vdash_{\mathfrak{L}} \varphi$$

Demonstração. O domínio $D = dom(\mathfrak{D})$ da estrutura que se tem em vista é formado pelos termos fechados da linguagem:

$$D = \{\, t \in L \mid t \text{ é termo fechado de } L \,\}$$

Definem-se então as interpretações das funções e predicados de maneira que a interpretação de f aplicada aos termos "t_n" seja o próprio termo que simboliza essa aplicação "$f(t_1, \ldots, t_n)$":

$$I_{\mathfrak{D}}(\text{“}f\text{”})(\text{“}t_1\text{”}, \cdots, \text{“}t_n\text{”}) = \text{“}f(t_1, \cdots, t_n)\text{”}$$

E que os predicados sejam iterpretados válidos exatamente naqueles termos que podem ser deduzidos sintaticamente, a partir de Δ, pertencendo ao predicado:

$$(\text{“}t_1\text{”}, \cdots, \text{“}t_n\text{”}) \in I_{\mathfrak{D}}(\text{“}P\text{”}) \qquad \Longleftrightarrow \qquad \Delta \vdash_{\mathfrak{Q}} P(t_1, \cdots, t_n)$$

Para completar, basta tomar, para toda $\varphi \in S_L$:

$$v_{\mathfrak{D}}(\varphi) = 1 \Longleftrightarrow \Delta \vdash_{\mathfrak{Q}} \varphi$$

e estender $v_{\mathfrak{D}}$ a todo o conjunto $S_{L(\mathfrak{D})}$ de forma a preservar seu valor quando se substitui nomes de termos pelos termos originais.

Esse último passo garante que $\mathfrak{D}$ forma uma estrutura básica com $v_{\mathfrak{D}}$ obedecendo ao ítem *vPred* da Definição 3.28 e que $\mathfrak{D} \vDash \varphi \iff \Delta \vdash_{\mathfrak{Q}} \varphi$. □

Agora, munidos de uma estrutura, com interpretação para símbolos de predicados e função e uma valoração incipiente, daremos mais um passo rumo a uma estrutura paraconsistente para Δ.

Definição 3.44 (Teoria de Henkin)**.** Dada uma teoria Δ de uma lógica $\mathfrak{Q}$, sobre uma linguagem L de primeira ordem, diz-se que ela é *teoria de Henkin* se obedece a:

1. Para toda sentença $\exists x\, \phi \in S_L$, existe um termo $t \in L$ livre para x em ϕ, tal que:
$$\Delta \vdash_{\mathfrak{Q}} \exists x\, \phi \qquad \Longrightarrow \qquad \Delta \vdash_{\mathfrak{Q}} \phi_x[t]$$

2. Para toda sentença $\forall x\, \phi \in S_L$, existe um termo $t \in L$ livre para x em ϕ, tal que:
$$\Delta \nvdash_{\mathfrak{Q}} \forall x\, \phi \qquad \Longrightarrow \qquad \Delta \nvdash_{\mathfrak{Q}} \phi_x[t]$$

■

Teorema 3.45. *Toda teoria Δ de uma lógica $\mathfrak{Q} \in \{\mathbf{QmbC}, \mathbf{QmCi}\}$ pode ser extendida conservativamente a uma teoria de Henkin Δ^H.*

Demonstração. A idéia da prova é ir adicionando constantes ao longo de vários níveis (ω níveis) que façam as vezes de testemunhas para fórmulas quantificadas do nível anterior. Por sua vez, essas constantes novas engendram novas fórmulas em cada nível, que serão testemunhadas por constantes do próximo nível. Tais testemunhas podem ser de dois tipos: comprovações de existenciais (testemunhas propriamente ditas) ou contra-testemunhas que atestem a falsificação de universais[3].

A construção é feita em dois passos. Primeiramente extende-se Δ por meio das constantes, dando origem à teoria Δ' com os mesmos axiomas não lógicos, mas com axiomas lógicos a mais: aqueles envolvendo as constantes novas. Pelo teorema das constantes, Δ' é extensão conservativa de Δ.

Na sequência, adicionam-se outros axiomas não lógicos à Δ' envolvendo as constantes novas. Desse processo resulta a teoria Δ^H. Tais axiomas vão garantir que a teoria final seja uma teoria de Henkin. Por último, prova-se que os axiomas introduzidos podem ser eliminados de qualquer prova de uma fórmula da linguagem original, garantindo que Δ^H é extensão conservativa de Δ.

Com efeito, define-se primeiramente a linguagem L_ω de Δ', que vai ser a extensão por constantes de Δ:

$$
\begin{aligned}
C_0 &= \text{Constantes de } L(\Delta) \\
C_1 &= C_0 \bigcup \{\, c_{\exists x\,\psi},\ c_{\forall x\,\psi} \;\big|\; \text{“}\exists x\,\psi\text{” e “}\forall x\,\psi\text{” são fórmulas fechadas de } L(\Delta) \,\} \\
C_n &= C_{n-1} \bigcup \{\, c_{\exists x\,\psi},\ c_{\forall x\,\psi} \;\big|\; \text{“}\exists x\,\psi\text{” e “}\forall x\,\psi\text{” são sentenças de } L_{n-1} \setminus L_{n-2} \,\} \\
C_\omega &= \bigcup_{n\in\omega} C_n
\end{aligned}
$$

$$
\begin{aligned}
L_0 &= L(\Delta) \\
L_n &= L_{n-1} \bigcup C_n \\
L_\omega &= L(\Delta) \bigcup C_\omega
\end{aligned}
$$

[3] A relação entre os dois tipos de constantes poderia também ser analisada através da dualidade entre *exemplo* e *contra-exemplo*. Essa extensão por contra-testemunhas é a principal novidade da demonstração que estamos aqui apresentando.

$$\Delta' = \Delta$$
$$L(\Delta') = L_\omega$$

Note que a definição de L_n depende de já ter-se definido C_n. Para $n \geq 2$, C_n depende, por sua vez, da definição de L_{n-1} e de L_{n-2}. Essa definição por dupla recursão garante que em cada C_n há testemunhas e contra-testemunhas das fórmulas novas de L_{n-1}, ou seja, fórmulas que têm ao menos uma constante de C_{n-1}. Dada uma quantificação arbitrária $Qx\phi \in L_\omega$, pela finitude das fórmulas, ou existe uma ocorrência em ϕ de uma constante em C_n com n máximo, ou a fórmula é original de $L(\Delta)$. Existem então testemunhas para tal quantificação em C_{n+1} ou em C_1. Tal processo independe da cardinalidade de $L(\Delta)$, depende apenas da finitude das fórmulas que estamos tratando, motivo pelo qual não precisa-se definir recursões além de ω para obter teorias de Henkin.

$AX_0 = \{\}$
$AX_n = \{\exists x\ \phi \Rightarrow \phi_x[c_{\exists x\ \phi}],\ \phi_x[c_{\forall x\ \phi}] \Rightarrow \forall x\ \phi \mid \exists x\ \phi,\ \forall x\ \phi\ \in L_n \text{ fechadas}\}$

$$\Delta^H = \Delta' \bigcup_{n\in\omega} AX_n$$
$$L(\Delta^H) = L_\omega$$

Por fim, se tivermos para alguma $\phi \in L(\Delta)$, $\Delta^H \vdash_{\mathfrak{L}} \phi$, então:

$$\Delta',\ \exists x\ \phi \Rightarrow \phi_x[c_{\exists x\ \phi}],\ \phi_x[c_{\forall x\ \phi}] \Rightarrow \forall x\ \phi\ \vdash_{\mathfrak{L}}\ \phi$$
$$\Delta'\ \vdash_{\mathfrak{L}}\ (\exists x\ \phi \Rightarrow \phi_x[c_{\exists x\ \phi}])\ \Rightarrow\ \Big((\phi_x[c_{\forall x\ \phi}] \Rightarrow \forall x\ \phi) \Rightarrow \phi\Big) \qquad \text{(Teorema 3.21)}$$
$$\Delta\ \vdash_{\mathfrak{L}}\ (\exists x\ \phi \Rightarrow \phi_x[y])\ \Rightarrow\ \Big((\phi_x[z] \Rightarrow \forall x\ \phi) \Rightarrow \phi\Big) \qquad \text{(Teorema 3.22)}$$
$$\Delta\ \vdash_{\mathfrak{L}}\ (\exists x\ \phi \Rightarrow \exists\, y\ \phi_x[y])\ \Rightarrow\ \Big((\phi_x[z] \Rightarrow \forall x\ \phi) \Rightarrow \phi\Big) \qquad \text{(Lema 3.24)}$$
$$\Delta \vdash_{\mathfrak{L}} \exists\ x\phi \Rightarrow \exists\, y\ \phi_x[y] \qquad \text{(Teorema 3.23)}$$
$$\Delta \vdash_{\mathfrak{L}} (\phi_x[z] \Rightarrow \forall x\ \phi) \Rightarrow \phi \qquad \textbf{(MP)}$$
$$\Delta \vdash_{\mathfrak{L}} (\forall z\ \phi_x[z] \Rightarrow \forall x\ \phi) \Rightarrow \phi \qquad \text{(Lema 3.24)}$$
$$\Delta \vdash_{\mathfrak{L}} \forall z\ \phi_x[z] \Rightarrow \forall x\ \phi \qquad \text{(Teorema 3.23)}$$
$$\Delta \vdash_{\mathfrak{L}} \phi \qquad \textbf{(MP)}$$

□

Teorema 3.46. *Toda teoria* Δ *de* $\mathfrak{L} \in \{\mathbf{QmbC}, \mathbf{QmCi}\}$*, sobre uma linguagem* L *de primeira ordem, induz uma estrutura* $\mathfrak{D}$ *sobre a mesma linguagem tal que* $v_{\mathfrak{D}}$ *obedece às regras quantificacionais (vEx e vUni) da Definição 3.31. Tal estrutura satisfaz exatamente os teoremas deduzidos por* Δ*. Isto é, para toda* $\varphi \in L$*:*

$$\mathfrak{D} \vDash \varphi \qquad \Longleftrightarrow \qquad \Delta \vdash_{\mathfrak{L}} \varphi$$

Demonstração. Para obtermos $\mathfrak{D}$ primeiramente, pelo Teorema 3.45, obtemos uma extensão Δ^H de Henkin (sobre uma linguagem L', enriquecida de constantes) para a teoria Δ. Para tal extensão Δ^H, por ser teoria de Henkin, se tomarmos $\forall x\,\phi, \exists x\,\phi \in S_L$, temos que:

$$\Delta^H \vdash_{\mathfrak{L}} \exists x\,\phi \qquad \Longrightarrow \qquad \Delta^H \vdash_{\mathfrak{L}} \phi_x[t] \text{ para algum } t \in L' \text{ livre para } x \text{ em } \phi$$
$$\Delta^H \nvdash_{\mathfrak{L}} \forall x\,\phi \qquad \Longrightarrow \qquad \Delta^H \nvdash_{\mathfrak{L}} \phi_x[t] \text{ para algum } t \in L' \text{ livre para } x \text{ em } \phi$$

e as seguintes deduções são logicamente válidas em $\mathfrak{L}$:

$$\vdash_{\mathfrak{L}} \exists x\,\phi \qquad \Longleftarrow \qquad \vdash_{\mathfrak{L}} \phi_x[t] \text{ para algum } t \in L' \text{ livre para } x \text{ em } \phi$$
$$\nvdash_{\mathfrak{L}} \forall x\,\phi \qquad \Longleftarrow \qquad \nvdash_{\mathfrak{L}} \phi_x[t] \text{ para algum } t \in L' \text{ livre para } x \text{ em } \phi$$

logo, como Δ^H é extensão conservativa de Δ, temos:

$$\Delta \vdash_{\mathfrak{L}} \exists x\,\phi \qquad \Longleftrightarrow \qquad \Delta^H \vdash_{\mathfrak{L}} \phi_x[t] \text{ para algum } t \in L' \text{ livre para } x \text{ em } \phi$$
$$\Delta \nvdash_{\mathfrak{L}} \forall x\,\phi \qquad \Longleftrightarrow \qquad \Delta^H \nvdash_{\mathfrak{L}} \phi_x[t] \text{ para algum } t \in L' \text{ livre para } x \text{ em } \phi$$

Agora, pelo Teorema 3.43, podemos obter uma estrutura $\mathfrak{D}'$ para Δ^H, cujo domínio se constitui pelos termos sem variáveis de L' e satisfaz exatamente os teoremas de Δ^H. Tal estrutura também está definida sobre a linguagem L (da qual L' é extensão) e podemos obter seu reduto $\mathfrak{D}$ sobre L. Para todo elemento $t \in L'$ do domínio de $\mathfrak{D}$, temos uma constante em $L(\mathfrak{D})$ que nomeia esse termo e toda fórmula com a constante tem igual valor de verdade de sua substituição pelo termo, logo:

$$\Delta \vdash_{\mathfrak{L}} \exists x\,\phi \qquad \Longleftrightarrow \qquad v_{\mathfrak{D}'}(\phi_x[a]) = v_{\mathfrak{D}}(\phi_x[a]) = 1 \textbf{ p\algum individuo } a \textbf{ de } L(\mathfrak{D})$$
$$\Delta \nvdash_{\mathfrak{L}} \forall x\,\phi \qquad \Longleftrightarrow \qquad v_{\mathfrak{D}'}(\phi_x[a]) = v_{\mathfrak{D}}(\phi_x[a]) = 0 \textbf{ p\algum individuo } a \textbf{ de } L(\mathfrak{D})$$

Por fim, como Δ^H é extensão conservativa de Δ e $\mathfrak{D}'$ satisfaz exatamente os teoremas de Δ^H, $v_{\mathfrak{D}}$ deve valer 1 exatamente nos teoremas de Δ. Podemos então concluir que:

$$v_{\mathfrak{D}}(\exists x\phi) = 1 \qquad \Longleftrightarrow \qquad v_{\mathfrak{D}}(\phi_x[a]) = 1 \textbf{ p\algum individuo } a \textbf{ de } L(\mathfrak{D})$$
$$v_{\mathfrak{D}}(\forall x\phi) = 0 \qquad \Longleftrightarrow \qquad v_{\mathfrak{D}}(\phi_x[a]) = 0 \textbf{ p\algum individuo } a \textbf{ de } L(\mathfrak{D})$$
$$\mathfrak{D} \vDash \varphi \qquad \Longleftrightarrow \qquad \Delta \vdash_{\mathfrak{L}} \varphi$$

□

Com isso, temos uma estrutura $\mathfrak{D}$ para a toda teoria Δ. Tal estrutura satisfaz exatamente seus teoremas e sua bivaloração $v_{\mathfrak{D}}$ obedece às regras quantificacionais da Definição 3.31. Na próxima seção, partiremos de uma teoria que não deduza uma fórmula φ e obteremos uma extensão completa sua que também não satisfará φ. A função característica de tal extensão obedecerá todas as regras da Definição 3.31 e, conforme seja uma teoria sobre **QmbC** ou **QmCi**, obedecerá às regras das Definições 3.34 ou 3.35. Pelo teorema anterior, então, haverá uma estrutura para tal extensão que satisfaz Δ, mas não φ.

3.3.2 Extensões Maximais: Argumento de Lindenbaum-Asser

Definição 3.47 (Teoria γ-não-trivial)**.** Diz-se de uma teoria Δ que ela é γ-não-trivial, ou $\gamma - nt$, se $\Delta \nvdash \gamma$.

Dada uma teoria Δ $\gamma - nt$, a possibilidade de construir extensões maximais γ-não-triviais de Δ é fundamental para se estabelecer a completude da semântica proposta.

A estrutura final $\mathfrak{D}$ ($dom(\mathfrak{D}) = \overline{\Delta^H}$) que estamos buscando tem três características importantes:

(i) verifica Δ

$$\mathfrak{D} \vDash \Delta$$

(ii) não verifica γ

$$\mathfrak{D} \nvDash \gamma$$

(iii) é de natureza sintática

$$\mathfrak{D} \vDash \phi \Longleftrightarrow \text{“}\phi\text{”} \in D$$

Teorema 3.48 (Extensão Maximal γ-não-trivial)**.** *Para todo conjunto Δ $\gamma-nt$ sobre uma linguagem L, existe $\overline{\Delta}$ $\gamma - nt$ que extende (simplesmente) Δ e é maximal no seguinte sentido*

$$se \quad \overline{\Delta} \subseteq \Delta' \subset For_L^{\circ} \quad e \;\; \Delta' \; \gamma - nt$$

$$\Delta' = \overline{\Delta}$$

Demonstração. Defina-se $J(\Delta) = \{A \subset For_L^{\circ} | \quad \Delta[A] \nvdash \gamma\}$. Vamos demonstrar que $J(\Delta)$ tem carácter finito.

Seja $A \subseteq For_L^{\circ}$:

a) $A \in J(\Delta)$

$$\begin{aligned} &\Delta[A] \nvdash \gamma \\ &\text{se } A^\circ \subseteq A, \text{ com } A^\circ \text{ finito:} \\ &\Delta[A^\circ] \nvdash \gamma \qquad \text{(Monotonicidade)} \\ &A^\circ \in J(\Delta) \\ &\forall A^\circ \subseteq A : A^\circ \in J(\Delta) \end{aligned}$$

b) $A \notin J(\Delta)$

$$\begin{aligned} &\Delta[A] \vdash \gamma \\ &\Delta[A^\circ] \vdash \gamma \text{ , para algum } A^\circ \subseteq A \text{ finito} \qquad \text{(Compacidade)} \\ &\exists A^\circ \subseteq A : A^\circ \notin J(\Delta) \end{aligned}$$

Logo, pelo teorema de Teichmüller-Tukey[4], existe algum D maximal em $J(\Delta)$. Para completar a demonstração, basta tomar

$$\overline{\Delta} = \Delta[D]$$

□

Teorema 3.49 (Fechamento da Extensão Maximal)**.** *Dada uma teoria* Δ*, se* $\overline{\Delta}$ *é uma extensão maximal* $\gamma - nt$ *sua,* $\overline{\Delta} \vdash \phi \iff \phi \in \overline{\Delta}$*.*

Demonstração. Se $\overline{\Delta} \vdash \phi$,

$$\overline{\Delta},\ \phi \vdash \gamma \Longrightarrow \overline{\Delta} \vdash \gamma \qquad \text{(Corte)}$$

Logo temos que $\overline{\Delta} \cup \{\phi\}$ é $\gamma - nt$ e que $\overline{\Delta} \subseteq \overline{\Delta} \cup \{\phi\}$. Pela maximalidade de $\overline{\Delta}$

$$\overline{\Delta} = \overline{\Delta} \cup \{\phi\}$$

e finalmente:

$$\phi \in \overline{\Delta}$$

□

Teorema 3.50 (Extensões Maximais e Valorações)**.** *Seja* Δ *um conjunto* $\gamma - nt$ *maximal de fórmulas. A função* $v : S^\circ_{L(\Delta)} \to \{0, 1\}$ $v(\phi) = 1 \iff \phi \in \Delta$ *obedece todas as Regras Proposicionais da Definição 3.31 e, se for uma teoria sobre* **QmbC** *também satisfaz as regra vNeg e vCon da Definição 3.32. Caso seja sobre* **QmCi***, obedece a todas as regras da Definição 3.32.*

[4] cf. [Sho67, p.47]

Demonstração. □

Corolário 3.51. *As lógicas* **QmbC** *e* **QmCi** *são completas.*

Demonstração. □

Capítulo 4

Um Teorema de Herbrand para LFI's

4.1 Sequentes para QmbC e QmCi

4.1.1 Definição

A seguinte formulação se baseia na construção de **BC** e **CI**, conforme aparecem em [Gen10] e [Gen07]. Tais cálculos são versões quanficadas, elaboradas por meio de sequentes, para **bC** e **Ci** respectivamente. Aqui vamos reformular os cálculos apresentados na Definição 3.14 (*a la Hilbert*) através de sequentes. Note que o sentido intuitivo de se afirmar um sequente $\Gamma \longrightarrow \Delta$ é que alguma $\gamma \in \Gamma$ é falsa ou alguma $\delta \in \Delta$ é verdadeira.

Definição 4.1.1 (**MBC**)**.** Começaremos com a contrapartida por sequentes de **QmbC**, na sequência apresentaremos a extensão necessária para **QmCi** (**MCI**).

Axiomas Clássicos

$$\alpha \longrightarrow \alpha \quad \textbf{:}\ \boldsymbol{Ax}$$

Enfraquecimento

$$\frac{\Gamma \longrightarrow \Delta}{\alpha, \Gamma \longrightarrow \Delta}\ \textbf{:}\ \boldsymbol{Enf\text{-}E} \qquad \frac{\Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta, \alpha}\ \textbf{:}\ \boldsymbol{Enf\text{-}D}$$

Contração

$$\frac{\Gamma, \alpha, \alpha \longrightarrow \Delta}{\Gamma, \alpha \longrightarrow \Delta}\ \textbf{:}\ \boldsymbol{Cont\text{-}E} \qquad \frac{\Gamma \longrightarrow \Delta, \alpha, \alpha}{\Gamma \longrightarrow \Delta, \alpha}\ \textbf{:}\ \boldsymbol{Cont\text{-}D}$$

Introdução de $\Rightarrow$

$$\frac{\Gamma \longrightarrow \Delta,\ \alpha \quad \beta,\ \Gamma' \longrightarrow \Delta'}{\alpha \Rightarrow \beta,\ \Gamma,\ \Gamma' \longrightarrow \Delta,\ \Delta'}\ \textbf{:}\ \Rightarrow \textbf{E} \qquad \frac{\Gamma,\ \alpha \longrightarrow \beta,\ \Delta}{\Gamma \longrightarrow \alpha \Rightarrow \beta,\ \Delta}\ \textbf{:}\ \Rightarrow \textbf{D}$$

Introdução de $\wedge$

$$\frac{\Gamma,\ \alpha,\ \beta \longrightarrow \Delta}{\Gamma,\ \alpha \wedge \beta \longrightarrow \Delta}\ \textbf{:}\ \wedge \textbf{E} \qquad \frac{\Gamma \longrightarrow \alpha,\ \Delta \quad \Gamma' \longrightarrow \beta,\ \Delta'}{\Gamma,\ \Gamma' \longrightarrow \alpha \wedge \beta,\ \Delta,\ \Delta'}\ \textbf{:}\ \wedge \textbf{D}$$

Introdução de $\vee$

$$\frac{\alpha,\ \Gamma \longrightarrow \Delta \quad \beta,\ \Gamma' \longrightarrow \Delta'}{\alpha \vee \beta,\ \Gamma,\ \Gamma' \longrightarrow \Delta,\ \Delta'}\ \textbf{:}\ \vee \textbf{E} \qquad \frac{\Gamma \longrightarrow \Delta,\ \alpha,\ \beta}{\Gamma \longrightarrow \Delta,\ \alpha \vee \beta}\ \textbf{:}\ \vee \textbf{D}$$

Introdução de $\neg$[1]

[1] É de se notar que **LK** (o cálculo de predicados clássico) pode ser obtido se substituirmos $\neg$***-E*** por $\dfrac{\Gamma \longrightarrow \Delta,\ \alpha}{\Gamma,\ \neg\alpha \longrightarrow \Delta}$, cf. [Gen10, p.7].

$$\frac{\circ\alpha, \Gamma \longrightarrow \Delta, \alpha}{\circ\alpha, \neg\alpha, \Gamma \longrightarrow \Delta} : \neg \mathbf{E} \qquad \frac{\alpha, \Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta, \neg\alpha} : \neg \mathbf{D}$$

Regras para Quantificadores

$$\frac{\alpha_x[t],\ \Gamma \longrightarrow \Delta}{\forall x\ \alpha,\ \Gamma \longrightarrow \Delta} : \forall \mathbf{E} \qquad \frac{\Gamma \longrightarrow \Delta,\ \alpha}{\Gamma \longrightarrow \Delta,\ \forall x\ \alpha} : \forall \mathbf{D}$$

$$\frac{\alpha,\ \Gamma \longrightarrow \Delta}{\exists x\ \alpha,\ \Gamma \longrightarrow \Delta} : \exists \mathbf{E} \qquad \frac{\Gamma \longrightarrow \Delta,\ \alpha_x[t]}{\Gamma \longrightarrow \Delta,\ \exists x\ \alpha} : \exists \mathbf{D}$$

Para essas regras, t está restrito aos **termos** livres para x em α e pode aparecer ainda em $Qx\ \alpha$ (isto é, t não está necessariamente *completamente indicado*[2] em $\alpha_x[t]$). Já x é uma **variável** que não ocorre livre[3] em Γ e Δ das regras $\forall$**-D** e $\exists$**-E**, designada na literatura por autovariável (*eigenvariable*).

Regra do Corte

$$\frac{\Gamma \longrightarrow \Delta,\ \alpha \quad \alpha,\ \Gamma' \longrightarrow \Delta'}{\Gamma,\ \Gamma' \longrightarrow \Delta,\ \Delta'}$$

■

Definição 4.1.2. MCI extende **MBC** pelas seguintes regras:

$$\frac{\Gamma \longrightarrow \Delta,\ \neg^n \circ \alpha}{\neg^{n+1} \circ \alpha,\ \Gamma \longrightarrow \Delta} : \neg \circ \mathbf{E} \qquad \frac{\alpha,\ \neg\alpha,\ \Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta,\ \circ\alpha} : \circ \mathbf{D}$$

■

4.1.2 Corretude da Formulação Hilbertiana em Relação aos Sequentes

Para que fique claro que nossa formulação por sequentes é adequada, devemos demonstrar sua equivalência com relação às **LFI**'s em questão. Primeiramente, obteremos a corretude dos axiomas e regras de inferência em relação aos cálculos de sequentes:

$$\vdash_{Hilbert} \wedge\Gamma \Rightarrow \vee\Delta \qquad \Longrightarrow \qquad \vdash_{Sequentes} \Gamma \longrightarrow \Delta$$

[2] Cf. [Tak75, p.8]
[3] Conforme [GTL89, p.32]

$$\wedge\Gamma \vdash_{Hilbert} \vee\Delta \qquad \Longrightarrow \qquad \vdash_{Sequentes} \Gamma \longrightarrow \Delta$$

A corretude dos sequentes será estabelecida em relação à semântica de valorações paraconsistentes na Seção 4.1.4:

$$\vdash_{Sequentes} \Gamma \longrightarrow \Delta \qquad \Longrightarrow \qquad \models_{Paraval} \wedge\Gamma \Rightarrow \vee\Delta$$

$$\vdash_{Sequentes} \Gamma \longrightarrow \Delta \qquad \Longrightarrow \qquad \wedge\Gamma \models_{Paraval} \vee\Delta$$

Tendo em vista que tal semântica é completa para as **LFI**'s, resultado demonstrado no Capítulo 3.3, obteremos de uma só vez a *corretude* do esquema Hilbertiano em relação às bivalorações e a *completude* do cálculo de sequentes em relação ao Hilbertiano, obtendo assim a equivalência entre os três formalismos.

Teorema 4.1.3. *Os axiomas de* **QmbC** *e* **QmCi** *(Definição 3.14) são demonstráveis por sequentes (Definição 4.1.1).*

Demonstração. Note que os axiomas **Inc** e **Con** são demonstrados apenas em **MCI**.

Ax. **OuI**

Ax	$\alpha \longrightarrow \alpha$
Enf-D	$\alpha \longrightarrow \alpha, \beta$
Perm-D	$\alpha \longrightarrow \beta, \alpha$
$\Rightarrow$**-D**	$\longrightarrow \alpha \Rightarrow \beta, \alpha$
Perm-D	$\longrightarrow \alpha, \alpha \Rightarrow \beta$
$\vee$**-D**	$\longrightarrow \alpha \vee (\alpha \Rightarrow \beta)$

Ax. **??**

Ax	$\alpha \longrightarrow \alpha$
$\neg$**-D**	$\longrightarrow \alpha, \neg\alpha$
$\vee$**-D**	$\longrightarrow \alpha \vee \neg\alpha$

Ax. **Exp**

Ax	$\alpha \longrightarrow \alpha$
Enf-E	$\circ\alpha, \alpha \longrightarrow \alpha$
Enf-D	$\circ\alpha, \alpha \longrightarrow \alpha, \beta$
$\neg$**-E**	$\circ\alpha, \alpha, \neg\alpha \longrightarrow \beta$
$\Rightarrow$**-D** (3x)	$\longrightarrow \circ\alpha \Rightarrow \alpha \Rightarrow \neg\alpha \Rightarrow \beta$

Ax. **Inc**

$$\begin{array}{ll} \textbf{Ax} & \alpha \wedge \neg\alpha \longrightarrow \alpha \wedge \neg\alpha \\ \circ\textbf{-D} & \longrightarrow \circ\alpha,\ \alpha \wedge \neg\alpha \\ \neg\circ\textbf{-E} & \neg \circ \alpha \longrightarrow \alpha \wedge \neg\alpha \\ \Rightarrow\textbf{-D} & \longrightarrow \neg \circ \alpha \Rightarrow (\alpha \wedge \neg\alpha) \end{array}$$

Ax. **Con**

$$\begin{array}{ll} \textbf{Ax} & \neg^n \circ \alpha \longrightarrow \neg^n \circ \alpha \\ \neg\circ\textbf{-E} & \neg^n \circ \alpha,\ \neg^{n+1} \circ \alpha \longrightarrow \\ \circ\textbf{-D} & \longrightarrow \circ\neg^n \circ \alpha \end{array}$$

Ax. $\exists$**-Ax**

$$\begin{array}{ll} \textbf{Ax} & \alpha_x[t] \longrightarrow \alpha_x[t] \\ \exists\textbf{-D} & \alpha_x[t] \longrightarrow \exists x\, \alpha \\ \Rightarrow\textbf{-D} & \longrightarrow \alpha_x[t] \Rightarrow \exists x\, \alpha \end{array}$$

Ax. $\forall$**-Ax**

$$\begin{array}{ll} \textbf{Ax} & \alpha_x[t] \longrightarrow \alpha_x[t] \\ \forall\textbf{-E} & \forall x\, \alpha \longrightarrow \alpha_x[t] \\ \Rightarrow\textbf{-D} & \longrightarrow \forall x\, \alpha \Rightarrow \alpha_x[t] \end{array}$$

□

Para demonstrar a completude de **MBC** e **MCI**, faz-se necessário que nesses formalismos também seja possível simular as regras de inferência de **QmbC** e **QmCi** (que são as mesmas para os dois cálculos). Para tanto, faremos uso de um lema intermediário:

Lema 4.1.4. *Eliminição do conectivo "⇒":*

$$\Gamma \longrightarrow \alpha \Rightarrow \beta \quad \vdash \quad \Gamma,\ \alpha \longrightarrow \beta$$

Demonstração. Com efeito, suponha $\Gamma \longrightarrow \alpha \Rightarrow \beta$. Dos axiomas "$\alpha \longrightarrow \alpha$" e "$\beta \longrightarrow \beta$", por repetidos enfraquecimentos obtemos:

$$\Gamma,\ \alpha \longrightarrow \beta,\ \alpha$$

$$\beta,\ \Gamma,\ \alpha \longrightarrow \beta$$

Podemos então usar a regra ⇒**-E**:

$$\frac{\Gamma,\ \alpha\ \longrightarrow\ \beta,\ \alpha \quad \beta,\ \Gamma,\ \alpha\ \longrightarrow\ \beta}{\alpha \Rightarrow \beta,\ \Gamma,\ \alpha\ \longrightarrow\ \beta}$$

Por fim, com o corte desse último sequente com a hipótese, obtemos o resultado desejado:

$$\frac{\Gamma\ \longrightarrow\ \alpha \Rightarrow \beta \quad \alpha \Rightarrow \beta,\ \Gamma,\ \alpha\ \longrightarrow\ \beta}{\Gamma,\ \alpha\ \longrightarrow\ \beta}$$

□

Teorema 4.1.5. MBC *e* **MCI** *podem simular as regras de inferência de* **QmbC** *e* **QmCi** *(Definição 3.15).*

Demonstração. São apenas três as regras, comuns a **QmbC** e **QmCi**, que devemos ser capazes de simular nossos sequentes:

1. Modus Ponens

   Suponha:

   $$\longrightarrow\ \alpha$$
   $$\longrightarrow\ \alpha \Rightarrow \beta$$

   Logo:

   $$\dfrac{\begin{array}{rl} \longrightarrow\ \alpha \Rightarrow \beta & \\ \alpha\ \longrightarrow\ \beta & \text{Lema 4.1.4} \\ \longrightarrow\ \alpha & \text{Hipótese} \end{array}}{\longrightarrow\ \beta \qquad \text{Corte}}$$

2. Introdução de Existencial

   Suponha, sem que x ocorra livre em β:

   $$\longrightarrow\ \alpha \Rightarrow \beta$$

   Logo

   $$\dfrac{\begin{array}{rl} \longrightarrow\ \alpha \Rightarrow \beta & \\ \alpha\ \longrightarrow\ \beta & \text{Lema 4.1.4} \\ \exists x\,\alpha\ \longrightarrow\ \beta & \exists\text{-E} \end{array}}{\longrightarrow\ \exists x\,\alpha \Rightarrow \beta \qquad \Rightarrow\text{-D}}$$

3. Introdução de Universal

   Suponha, sem que x ocorra livre em α:

$$\longrightarrow \alpha \Rightarrow \beta$$

   Logo

$$\dfrac{\begin{array}{c}\longrightarrow \alpha \Rightarrow \beta \\ \alpha \longrightarrow \beta \\ \alpha \longrightarrow \forall x\,\beta\end{array}\qquad\begin{array}{c}\\ \text{Lema 4.1.4} \\ \forall\text{-D}\end{array}}{\longrightarrow \alpha \Rightarrow \forall x\,\beta \qquad \Rightarrow\text{-D}}$$

□

Teorema 4.1.6. *Os axiomas usados em demonstrações em* **MBC** *podem se limitar, conservativamente, a três tipos de fórmulas ("At" indica uma fórmula atômica e "φ" uma fórmula qualquer):*

1.

$$At \longrightarrow At$$

2.

$$\neg\varphi \longrightarrow \neg\varphi$$

3.

$$\circ\varphi \longrightarrow \circ\varphi$$

Para **MCI** *bastam os ítens 1 e 2 acima.*

Demonstração. A demonstração é por dupla indução no número de instâncias, em uma demonstração qualquer, de axiomas com a fórmula complexa e na complexidade dessas fórmulas. O passo indutivo principal consiste em substituir o uso de uma instância do axioma com a fórmula em questão por demonstrações que terminem nessa fórmula e que façam uso apenas de axiomas com fórmulas de complexidade menor. Temos 6 casos, conforme a fórmula φ na instância de "$\varphi \longrightarrow \varphi$" seja:

1. $\alpha \Rightarrow \beta$

$$\dfrac{\begin{array}{c}\alpha \longrightarrow \alpha \qquad \beta \longrightarrow \beta \\ \alpha \Rightarrow \beta,\ \alpha \longrightarrow \beta\end{array}\qquad\begin{array}{c}\text{Hipóteses de Indução} \\ \Rightarrow \boldsymbol{E}\end{array}}{\alpha \Rightarrow \beta \longrightarrow \alpha \Rightarrow \beta \qquad \Rightarrow \boldsymbol{D}}$$

2. $\alpha \wedge \beta$

$$\dfrac{\dfrac{\alpha \longrightarrow \alpha}{\alpha \wedge \beta \longrightarrow \alpha} \qquad \dfrac{\beta \longrightarrow \beta}{\alpha \wedge \beta \longrightarrow \beta}}{\alpha \wedge \beta \longrightarrow \alpha \wedge \beta}$$

Hipóteses de Indução
Enf-E e $\wedge\, \boldsymbol{E}$
$\wedge\, \boldsymbol{D}$ e **Cont-E**

3. $\alpha \vee \beta$

$$\dfrac{\dfrac{\alpha \longrightarrow \alpha}{\alpha \longrightarrow \alpha \vee \beta} \qquad \dfrac{\beta \longrightarrow \beta}{\beta \longrightarrow \alpha \vee \beta}}{\alpha \vee \beta \longrightarrow \alpha \vee \beta}$$

Hipóteses de Indução
Enf-D e $\vee\, \boldsymbol{E}$
$\vee\, \boldsymbol{E}$ e **Cont-D**

4. $\forall x\, \alpha$

$$\begin{array}{cc} \alpha \longrightarrow \alpha & \text{Hipótese de Indução} \\ \forall x\, \alpha \longrightarrow \alpha & \forall\, \boldsymbol{E} \\ \forall x\, \alpha \longrightarrow \forall x\, \alpha & \forall\, \boldsymbol{D} \end{array}$$

5. $\exists x\, \alpha$

$$\begin{array}{cc} \alpha \longrightarrow \alpha & \text{Hipótese de Indução} \\ \alpha \longrightarrow \exists x\, \alpha & \exists\, \boldsymbol{D} \\ \exists x\, \alpha \longrightarrow \exists x\, \alpha & \exists\, \boldsymbol{E} \end{array}$$

6. $\circ\alpha$ (apenas para **MCI**)

$$\begin{array}{cc} \alpha \longrightarrow \alpha & \text{Hipótese de Indução} \\ \circ\alpha,\ \alpha \longrightarrow \alpha & \textbf{Enf-E} \\ \circ\alpha,\ \alpha,\ \neg\alpha \longrightarrow & \neg\, \boldsymbol{E} \\ \circ\alpha \longrightarrow \circ\alpha & \circ\, \boldsymbol{D} \end{array}$$

Note que por esse processo não introduzimos nenhum corte a mais.

□

4.1.3 Eliminação do Corte

A eliminação do corte consiste em mostrar que o corte é, em princípio, *dispensável*. Isto é, em sistemas para os quais vale o teorema da eliminação do corte, pode-se excluir essa regra sem que se percam teoremas, em troca de um aumento superexponencial no comprimento das desmontrações:

> *Roughly speaking, a cut-free proof is a proof from which all formulas which are "too general" have been banished. General formulas in a proof carry the ideas of the proof: it is only because we have general formulas, that we can have short and understandable proofs; when (in the Hauptsatz) we eliminate all these general formulas, we increase the length and obscurity of the proof: for that reason, cut-free proofs are unlikely objects for mathematical practice. Their interest lies somewhere else: these proofs are very interesting to study, because their general properties are very important.* [Gir87, p.95]

Quanto às propriedades interessantes de demonstrações sem corte, faremos uso delas mais adiante, quando for demonstrado o teorema de Herbrand para **LFI**'s no Capítulo 4.2.

Podemos ainda enteder o resultado de outra maneira: que os sistemas obtidos pela supressão da regra de corte são fechados em relação a ela, ou seja, a regra de corte é *redundante*. No entanto, se admitimos axiomas não lógicos, nem sempre é possível eliminar todos os cortes (sem que percamos possíveis demonstrações feitas a partir de tais axiomas).

A demonstração começa com os casos chaves (Lema 4.1.11): demonstramos que o corte pode ser eliminado quando as duas subprovas de ambos os sequentes das premissas de algum corte terminem em uma regra lógica que introduza o conectivo principal da fórmula de corte. Todos os outros casos são tratados no Lema Principal (4.1.12), quando pelo menos uma das duas subprovas não termina em regra lógica: por exemplo, em uma contração. Esse caso em particular levou Gentzen, em [Gen35], a introduzir o que denominou de *multi-corte*. Tal regra, que permite cortes simultâneos, é equivalente à regra de corte simples. A demonstração original consistia, na verdade, em demonstrar que os multi-cortes poderiam ser eliminados (cf. [TS96, 4.1.4]).

Se fazem necessárias algumas definições, a serem usadas como variáveis de indução. A demonstração apresentada aqui foi adaptada de [GTL89] seguindo os métodos de [Gen10] para lidar com as características não clássicas.

Definição 4.1.7 (Grau de uma Fórmula)**.**

Definição 4.1.8 (Grau do Corte e Grau da Demonstração)**.**

Definição 4.1.9 (Altura da Demonstração)**.**

Notação 4.1.10. *Vamos simbolizar por* $\Gamma - \varphi$ *o conjunto obtido a partir de* Γ *suprimindo todas as fórmulas* φ*.*

Casos Chaves

Lema 4.1.11 (Casos Chaves)**.** *O objetivo é eliminar cortes:*

$$\frac{\Gamma \longrightarrow \Delta, \varphi \quad \varphi, \Gamma' \longrightarrow \Delta'}{\Gamma, \Gamma' \longrightarrow \Delta, \Delta'} \textbf{ : Corte de } \varphi$$

em que φ *seja a fórmula principal de regras lógicas nas duas subprovas das premissas. Tais cortes serão substituídos por demonstrações das mesmas conclusões que façam uso de cortes com fórmulas menos complexas.*

Demonstração. Procederemos de diferentes maneiras, conforme φ seja:

1. $\alpha \wedge \beta$

$$\dfrac{\dfrac{\dfrac{\Gamma \longrightarrow \alpha, \Delta \quad \Gamma \longrightarrow \beta, \Delta}{\Gamma \longrightarrow \alpha \wedge \beta, \Delta} : \wedge \textbf{ D} \qquad \dfrac{\Gamma, \alpha, \beta \longrightarrow \Delta}{\Gamma, \alpha \wedge \beta \longrightarrow \Delta} : \wedge \textbf{ E}}{\Gamma \longrightarrow \Delta} \textbf{ : corte de } \alpha \wedge \beta}{\dfrac{\Gamma \longrightarrow \beta, \Delta \qquad \dfrac{\Gamma \longrightarrow \alpha, \Delta \quad \Gamma, \alpha, \beta \longrightarrow \Delta}{\Gamma, \beta \longrightarrow \Delta} : \textbf{corte de } \alpha}{\Gamma \longrightarrow \Delta} \textbf{ : corte de } \beta}$$

2. $\neg\alpha$

$$\dfrac{\dfrac{\dfrac{\Gamma, \alpha \longrightarrow \Delta}{\Gamma \longrightarrow \neg\alpha, \Delta} : \neg \textbf{ D} \qquad \dfrac{\Gamma, \circ\alpha \longrightarrow \alpha, \Delta}{\Gamma, \circ\alpha, \neg\alpha \longrightarrow \Delta} : \neg \textbf{ E}}{\Gamma, \circ\alpha \longrightarrow \Delta} \textbf{ : corte de } \neg\alpha}{\dfrac{\Gamma, \circ\alpha \longrightarrow \alpha, \Delta \qquad \Gamma, \alpha \longrightarrow \Delta}{\Gamma, \circ\alpha \longrightarrow \Delta} \textbf{ : corte de } \alpha}$$

□

Lema Principal

Lema 4.1.12 (Lema Principal)**.** *Seja φ uma fórmula de grau d, e π e π' provas de $\Gamma \longrightarrow \Delta$ e de $\Gamma' \longrightarrow \Delta'$ com grau estritamente menor que d. Podemos construir uma prova ϖ de $\Gamma, \Gamma' - \varphi \longrightarrow \Delta - \varphi, \Delta'$ de grau estritamente menor que d:*

$$\dfrac{\dfrac{\begin{array}{cc}\pi & \pi' \\ \vdots & \vdots \\ \Gamma \longrightarrow \Delta & \Gamma' \longrightarrow \Delta'\end{array}}{\varpi}}{\Gamma, \Gamma' - \varphi \longrightarrow \Delta - \varphi, \Delta'}$$

Demonstração. Constrói-se ϖ por indução em $h(\pi)+h(\pi')$. As regras de sequentes podem ter uma ou duas premissas, logo, se chamarmos a última regra da demonstração π de r, sua(s) premissa(s) será(serão) designadas por $\Gamma_i \longrightarrow \Delta_i$, com $i = 1$ ou $i \in \{1, 2\}$. Paralelamente, a última regra de π' será r' com premissa(s) $\Gamma'_j \longrightarrow \Delta'_j$.

Portanto, conforme tenhamos uma ou duas premissas em π ou π', I e J serão respectivamente o número de premissas de cada demosntração:

$$\dfrac{\overbrace{\begin{array}{cc}\pi & \\ \vdots & \\ \pi_1 & \pi_I \\ \vdots & \vdots \\ \overbrace{\Gamma_1 \longrightarrow \Delta_1} & \overbrace{\Gamma_I \longrightarrow \Delta_I}\end{array}}}{\Gamma \longrightarrow \Delta}\ \mathbf{:r} \qquad \dfrac{\begin{array}{cc}\pi' & \\ \iddots & \\ \pi'_1 & \pi'_J \\ \vdots & \vdots \\ \Gamma'_1 \longrightarrow \Delta'_1 & \Gamma'_J \longrightarrow \Delta'_J\end{array}}{\Gamma' \longrightarrow \Delta'}\ \mathbf{:r'}$$

Seguem-se, então, todos os casos para encontrar a demonstração ϖ de $\Gamma, \Gamma' - \varphi \longrightarrow \Delta - \varphi, \Delta'$:

1. π é um axioma. Temos dois casos:

   - π é $\varphi \longrightarrow \varphi$. Basta tomar ϖ como a demonstração de $\varphi, \Gamma' - \varphi \longrightarrow \Delta'$ feita a partir de π' por meio de regras estruturais. Note que $g(\varpi) = g(\pi') < g(\varphi)$.
   - π é $\alpha \longrightarrow \alpha$, com $\alpha \neq \varphi$. Tomemos então ϖ como a demonstração de $\alpha, \Gamma' - \varphi \longrightarrow \alpha, \Delta'$ por meio de enfraquecimentos tendo $\alpha \longrightarrow \alpha$ como axioma. Obviamente $g(\varpi) = 0$.

2. π' é uma axioma. A construção de ϖ é similar ao item 1.

3. r é regra estrutural.

4. r' é regra estrutural.

5. r é alguma regra lógica que não introduza φ à direita. Pela hipótese de indução, para π_i e π' temos demonstração(ões) ϖ_i de "$\Gamma_i,\ \Gamma' - \varphi\ \longrightarrow\ \Delta_i - \varphi,\ \Delta'$" com grau estritamente menor que d. Se aplicarmos apropriadamente a mesma regra r à(s) ϖ_i (obtida(s) pela(s) hipótese(s) de indução) e possivelmente algumas regras estruturais, conforme dois casos principais:

$I = 1$

$$\dfrac{\dfrac{\Gamma_1\ \longrightarrow\ \Delta_1}{\Gamma\ \longrightarrow\ \Delta}\ :\boldsymbol{R} \qquad \Gamma'\ \longrightarrow\ \Delta'}{\dfrac{\Gamma_1,\ \Gamma' - \varphi\ \longrightarrow\ \Delta_1 - \varphi,\ \Delta'}{\Gamma,\ \Gamma' - \varphi\ \longrightarrow\ \Delta - \varphi,\ \Delta'}\ :\boldsymbol{R}}$$

Estando $R \in \{\wedge\ \boldsymbol{E}, \vee\ \boldsymbol{D}, \Rightarrow\ \boldsymbol{D}, \neg\ \boldsymbol{E}, \neg\ \boldsymbol{D}, \forall\ \boldsymbol{E}, \forall\ \boldsymbol{D}, \exists\ \boldsymbol{E}, \exists\ \boldsymbol{D}\}$

$I = 2$

$$\dfrac{\dfrac{\Gamma_1\ \longrightarrow\ \Delta_1 \quad \Gamma_2\ \longrightarrow\ \Delta_2}{\Gamma\ \longrightarrow\ \Delta}\ :\ \wedge\,\mathbf{D},\ \vee\,\mathbf{E},\ \Rightarrow\ \mathbf{E} \qquad \Gamma'\ \longrightarrow\ \Delta'}{\dfrac{\Gamma_1,\ \Gamma' - \varphi\ \longrightarrow\ \Delta_1 - \varphi,\ \Delta' \quad \Gamma_2,\ \Gamma' - \varphi\ \longrightarrow\ \Delta_2 - \varphi,\ \Delta'}{\Gamma,\ \Gamma' - \varphi\ \longrightarrow\ \Delta - \varphi,\ \Delta'}\ :\ \wedge\,\mathbf{D},\ \vee\,\mathbf{E},\ \Rightarrow\ \mathbf{E}}$$

obteremos ϖ sem maiores problemas, pois todas as regras estão previstas na demonstração do caso clássico à excessão de $\neg\ \boldsymbol{E}$:

$$\dfrac{\dfrac{\Gamma_1',\ \circ\alpha\ \longrightarrow\ \Delta,\ \alpha}{\Gamma_1',\ \circ\alpha,\ \neg\alpha\ \longrightarrow\ \Delta}\ :\ \neg\,\mathbf{E} \qquad \Gamma'\ \longrightarrow\ \Delta'}{\dfrac{\Gamma_1',\ \circ\alpha,\ \Gamma' - \varphi\ \longrightarrow\ (\Delta,\ \alpha) - \varphi,\ \Delta'}{\Gamma_1',\ \circ\alpha,\ \neg\alpha,\ \Gamma' - \varphi\ \longrightarrow\ \Delta - \varphi,\ \Delta'}\ :\ \neg\,\mathbf{E}}$$

Para essa regra, pode-se facilmente ver que a fórmula de restrição $\circ\alpha \in \Gamma_1$ ainda se mantem em "$\Gamma_1,\ \Gamma' - \varphi$" (que, nesse caso, é identico a "$\Gamma_1',\ \circ\alpha,\ \neg\alpha,\ \Gamma' - \varphi$"), permitindo aplicar a regra novamente ao sequente final de ϖ_1 caso a fórmula principal de $\neg\ \boldsymbol{E}$ não seja $\neg\varphi$ (isto é, não seja o caso de que $\alpha \equiv \varphi$). Caso seja, basta ***Enf-E*** para introduzir $\neg\varphi$ à esquerda ($\neg\varphi \in \Gamma$).

6. r' é alguma regra lógica que não introduza φ à esquerda. Pela hipótese de indução, para π e π'_j temos demonstração(ões) ϖ'_j de "$\Gamma,\ \Gamma'_j - \varphi\ \longrightarrow\ \Delta - \varphi,\ \Delta'_j$". O único caso que nos impede de proceder como no item anterior é quando r' é $\neg\, \boldsymbol{E}$ com fórmula principal $\neg\phi$ e $\varphi \equiv \circ\phi$:

$$\begin{array}{c} \pi' \\ \vdots \\ \dfrac{\Gamma''_1,\ \circ\phi\ \longrightarrow\ \Delta', \phi}{\Gamma''_1,\ \circ\phi,\ \neg\phi\ \longrightarrow\ \Delta'} \quad \mathbf{:}\, r' = \neg\, \mathbf{E} \end{array}$$

pois teríamos que demonstrar:

$$\frac{\Gamma,\ \Gamma''_1 - \circ\phi\ \longrightarrow\ \Delta - \circ\phi,\ \Delta', \phi}{\Gamma,\ \Gamma''_1 - \circ\phi,\ \neg\phi\ \longrightarrow\ \Delta - \circ\phi,\ \Delta'} \equiv \frac{\text{HI}}{\text{QED}} \qquad (\star)$$

usando apenas $\neg\, \boldsymbol{E}$ e regras estruturais. Pode ser que não haja uma $\circ\phi$ em Γ que sirva de fórmula de restrição para a aplicação de $\neg\, \boldsymbol{E}$.

Logo, para chegar em uma demonstração ϖ que termine em $(\star)$, o caso em que $\circ\phi \notin \Delta$ é trivial, pois QED pode ser obtido a partir de π (que termina em $\Gamma\ \longrightarrow\ \Delta$) através de repetidos enfraquecimentos.

Suponha $\circ\phi \in \Delta$. Tendo em vista que em **MBC** não há regras para a introdução de $\circ$, os ancestrais de $\circ\phi$ estão todos ao lado direito dos sequentes de π e a eles se aplicaram apenas regras estruturais.

(a) Se entre tais ancestrais não encontramos nenhuma fórmula introduzida por axioma, elimine-se de π todas as regras estruturais, obtendo então uma demonstração de $\Gamma\ \longrightarrow\ \Delta - \circ\phi$. Para chegar à ϖ bastam regras estruturais e temos que $g(\varpi) = g(\pi)$.

(b) Se encontrarem-se ancestrais das $\circ\phi$ que ocorrem em Δ entre axiomas de π, usando a hipótese de indução encontramos uma demonstração ϖ_0 de $\circ\phi,\ \Gamma''_1 - \circ\phi,\ \neg\phi\ \longrightarrow\ \Delta'$ a partir do axioma e de π'. Substituindo-se tais demonstrações pelos axiomas $\circ\phi\ \longrightarrow\ \circ\phi$ em π e apagando todos herdeiros da $\circ\phi$ à direita, bem como os enfraquecimentos ancestrais a qualquer $\circ\phi \in \Delta$ que não tenha se originado em axioma obtemos a demonstração ϖ desejada, com $g(\varpi) = g(\pi)$.

7. Ambas r e r' são regras lógicas, r introduzindo φ à direita e r' introduzindo φ à esquerda. Pela hipótese de indução aplicada respectivamente a

   - π_i e π', obtemos a(s) demonstração(ões) ϖ_i de $\Gamma_i,\ \Gamma'-\varphi\ \longrightarrow\ \Delta_i-\varphi,\ \Delta'$
   - π e π'_j, obtemos a(s) demosntração(ões) ϖ'_j de $\Gamma,\ \Gamma'_j-\varphi\ \longrightarrow\ \Delta-\varphi,\ \Delta'_j$

□

A Eliminação do Corte

Lema 4.1.13. *Se π é uma demonstração com grau $d > 0$ para um sequente, então é possível construir uma nova prova ϖ para o mesmo sequente com grau estritamente menor.*

Demonstração. Por indução em $h(\pi)$. Seja r a última regra de π e π_i a(s) subprova(s) correspondente(s) à(s) premissa(s) de r. Temos dois casos:

1. r não é um corte de grau d. Pela hipótese de indução, temos ϖ_i de grau $< d$. Obtemos ϖ aplicando r à(s) ϖ_i.

2. r é um corte de grau d:

$$\frac{\Gamma\ \longrightarrow\ \varphi,\ \Delta \qquad \Gamma',\ \varphi\ \longrightarrow\ \Delta'}{\Gamma,\ \Gamma'\ \longrightarrow\ \Delta,\ \Delta'}\ :r$$

   A hipótese de indução nos dá ϖ_i de grau $< d$ para as premissas de r. Aplicando o Lema Principal (4.1.12) obtemos uma demonstração ϖ de $\Gamma,\ \Gamma'\ \longrightarrow\ \Delta,\ \Delta'$ com grau $< d$.

□

Teorema 4.1.14 (Eliminação do Corte)**.** *A regra de corte é redundante em* **MBC**.

Demonstração. Pela iteração do Lema 4.1.13. □

4.1.4 Corretude em Relação às Valorações Paraconsistentes

4.2 Teorema de Herbrand para QmbC

A demonstração apresentada aqui é de uma versão restrita do teorema, no sentido de que estamos tratando apenas fórmulas Π_2 que sejam consequências lógicas de conjuntos de fórmulas não quantificadas. Em sua tese de doutorado ([Her30]),

Herbrand apresentou a forma geral, aplicável a quaisquer sentenças logicamente válidas. Note que, no entanto, Herbrand tratava como primitivos apenas "$\neg$", "$\vee$" e "$\exists$", e também "$\forall$" para fórmulas prenexas (cf. [WSBA09, p.20]), o que não pode ser feito no contexto das **LFI**'s; nessas lógicas não é sempre possível definir todos os conectivos em termos de conjuntos que para as lógicas clássicas são suficientes (cf. [CCM07]).

A presente demonstração foi baseada em [Bus98], onde está provado também o caso mais geral, dadas as devidas correções de [Mck]. Em tais artigos também, quando tratam da versão geral, assumem sem perda de generalidade fórmulas restritas à forma normal negativa (em que a negação aplica-se apenas às fórmulas atômicas) e ao fragmento gerado a partir de "$\neg$", "$\vee$", "$\wedge$" e os quantificadores. Tal coisa igualmente não é possível para as **LFI**'s.

Teorema 4.2.1 (Teorema de Herbrand)**.**

Demonstração. □

Capítulo 5

Uma Nova Perspectiva em Programação Lógica Paraconsistente

Aqui tentaremos sublinhar a relevância do entendimento (crítico) de negação proporcionado pelas lógicas paraconsistentes (e **LFI**'s) para as diversas negações nas extensões da programação lógica clássica (incluindo a paraconsistente).

Como pudemos ver na Seção 1.2.4, a programação lógica, levada pela necessidade de aumentar sua expressividade através da admissão de consultas por literais negadas – demanda inicialmente ligada às pesquisas em bases de dados – teve um impulso ainda maior quando se percebeu que poderia naturalmente suportar deduções não-monotônicas com a introdução de regras para derivar negações.

Hoje em dia, há inúmeras semânticas paraconsistentes para programas lógicos que admitem dois tipos de negação: uma não monotônica explosiva e outra monotônica paraconsistente.

Um teorema de Herbrand será útil se nos fornecer um ponto de partida para um método de resolução geral, abrangendo vários raciocínios paraconsistentes. Com isso, modelar-se-á, pela relação de consequência lógica, deduções monotônicas de informações inconsistentes. É como se o caminho fosse pela contra-mão do **PROLOG** clássico. Para o **PROLOG**, os métodos sintáticos da teoria da prova foram preponderantes, vieram primeiro e abriram caminho para as análises semânticas, de viés denotacional.

Um teorema de Herbrand possibilita, de uma maneira concisa, lidar com universos potencialmente infinitos, como é o caso quando se tem símbolos de função e variáveis nos programas. Na literatura em programação lógica paraconsistente, muitas vezes assume-se que o programa não tem variáveis e constitui-se de (possivelmente infinitas) regras (como em [DP98]). Algumas vezes (como em [dAP07]), admite-se apenas universos finitos e a resolução fica restrita ao caso proposicional.

O teorema de Herbrand é um ponto em comum para se chegar, antes de qualquer especialização que diga qual a resolução correta ou mesmo qual a forma lógica precisa das cláusulas de *Horn* ou das *queries*. É também importante porque pode ser estabelecido para fórmulas quaisquer, não está preso a alguma forma normal.

Esperamos ter avançado algo em direção a um teorema de Herbrand geral com nossa formulação por sequentes e o teorema da eliminação do corte para **QmbC** do Capítulo 4.

Capítulo 6

Considerações Finais

Uma conjectura que acreditamos ser possível demonstrar relaciona as **LFI**'s com várias abordagens de programação lógica paraconsistente:

Conjectura 6.1. *A semântica monotônica definida por Damásio e Pereira em [DP98] (Definição 2.1) para programas estendidos definidos, que fazem uso da negação forte, é correta e completa para a consequência lógica em* **QmbC** *e* **QmCi***, quando as consideramos restritas ao fragmento em questão. Ou seja, para um programa P estendido definido, e* $\Omega \in \{\mathbf{QmbC}, \mathbf{QmCi}\}$*:*

$$L \in M_P \qquad \Longleftrightarrow \qquad P \vdash_\Omega L$$

Parece, então, que seria possível caracterizar tais modelos minimais fazendo uso da mesma definição clássica de operador de consequência imediata. A única diferença estaria em admitir, nas bases e modelos de Herbrand (Definições 1.2 e 1.9), também literais negadas, com a ressalva de que uma sentença atômica ou sua negação deveriam estar sempre presentes em qualquer modelo.

Logo, é como se tais lógicas fossem *a lógica* por trás da parte monotônica de uma ampla gama de abordagens à programação lógica paraconsistente.

Pretendemos, futuramente, analisar com detalhe as semânticas atuais que dão conta das regras não monotônicas da negação (paraconsistente e clássica), como a bem fundada e de modelo estável.

Pretedemos também desenvolver uma teoria da resolução de forma a abranger as **LFI**'s como um todo. Desta maneira estaríamos lançando as bases para um ambiente de programação lógica que desse conta de uma ampla gama de maneiras de se lidar logicamente com inconsistências. Também pretendemos estender os resultados de completude para teorias com igualdade.

Com isso, teríamos condição de realizar uma análise conceitual sobre a paraconsistência em bases de conhecimento e nas diferentes possibilidades de negações

da programação lógica clássica.

Na sequência, algumas considerações sobre assuntos que, possivelmente, terão relevância em futuras pesquisas na área de programação lógica, teoria de modelos para primeira ordem e dedução automática paraconsistentes.

6.1 Problemas com a dualidade entre satisfatibilidade e validade: considerações semânticas

Está claro que, pela própria definição de consequência e validade lógicas, a validade universal é um subproblema do problema mais geral da consequência lógica. No referencial clássico, graças à lei da dupla negação, tem-se que a validade está reduzida à insatisfatibilidade e, graças também à completude e à compacidade, a consequência lógica pode ser estabelecida por métodos sintáticos de refutação:

$$\begin{aligned} \varphi \textit{ é válida} &\iff \emptyset \vDash \varphi \\ \varphi \textit{ é válida} &\iff \neg\varphi \textit{ é insatisfatível} \\ \Gamma \vDash \varphi &\iff \Gamma \vdash \neg(\neg\varphi) \end{aligned}$$

Conforme vimos em 1.1, para determinar a consequência lógica em primeira ordem clássica, a insatisfatibilidade é suficiente e necessária. Para essa, basta e são necessários procedimentos sintáticos de refutação:

$$\begin{aligned} \Gamma \vDash \varphi &\iff \Gamma, \neg\varphi \textit{ é insatisfatível} \\ \Gamma, \varphi \textit{ é insatisfatível} &\iff \Gamma \vdash \neg\varphi \end{aligned}$$

As coisas mudam de figura quando estamos considerando semânticas paraconsistentes. Em geral, podemos ter:

$$\Gamma \vDash_{\mathbf{P}} \neg\varphi \ \textbf{ e } \ \Gamma, \varphi \textit{ satisfatível}$$

ou mesmo:

$$\Gamma \vDash_{\mathbf{P}} \varphi \ \textbf{ e } \ \Gamma, \neg\varphi \textit{ satisfatível}.$$

Logo, em geral, temos apenas:

$$\begin{aligned} \Gamma \vDash_{\mathbf{P}} \varphi &\impliedby \Gamma, \neg\varphi \textit{ insatisfatível} \\ \Gamma \vDash_{\mathbf{P}} \neg\varphi &\impliedby \Gamma, \varphi \textit{ insatisfatível} \end{aligned}$$

Mas, pela eliminação do corte, temos:

$$\vdash_{\mathbf{QmbC}} \neg\varphi \iff \varphi \text{ instatisfatível.}$$

Apesar de ser possível:

$$\vDash_{\mathbf{P}} \varphi \quad \textbf{com} \quad \neg\varphi \text{ satisfatível.}$$

Como ficou claro com o Teorema 3.42, a busca pela base de Herbrand não é suficiente para determinar consequencia lógica.

6.2 Métodos de Resolução

Como visto na Seção 1.2, a resolução faz uso essencial da forma normal conjuntiva. Nas lógicas paraconsistentes, nem sempre é possível estabelecer formas normais para as fórmulas.

Outro problema é que o silogismo disjuntivo $[\alpha,\ \neg\alpha \vee \beta \vdash \beta]$ não pode valer em nenhuma extensão paraconsistente da lógica positiva clássica ou intuicionista, conforme o Teorema 3.19 de [CM02, p.40]. Mas, se partirmos de fórmulas já na forma normal conjuntiva, é possível uma espécie de regra de resolução para algumas **LFI**'s.

Teorema 6.2 (Resolução Restrita)**.** *Podemos definir, para* **QmbC** *e* **QmCi***, uma regra de resolução restrita, que não elimine totalmente a fórmula resolvente:*

$$(\alpha \vee \beta) \ \wedge \ (\neg\alpha \vee \gamma) \ \wedge \ \Phi \quad \vdash \quad (\neg \circ \alpha \vee \beta \vee \gamma) \ \wedge \ \Phi$$

Demonstração. Pode-se verificar checando todas as valorações paraconsistentes possíveis para as duas lógicas (ver Corolários 3.37 e 3.38). □

Referências Bibliográficas

[AB94] *Krzysztof R. Apt and Roland Bol. Logic programming and negation: A survey.* JOURNAL OF LOGIC PROGRAMMING*, 19:9–71, 1994.*

[AN84] *Ahmad Almukdad and David Nelson. Constructible falsity and inexact predicates.* The Journal of Symbolic Logic*, 49(1):pp. 231–233, 1984.*

[Bel77] *Nuel D. Belnap. A useful four-valued logic. In J. Michael Dunn and George Epstein, editors,* Modern Uses of Multiple-Valued Logics*, pages 8–37. Reidel, Dordrecht, 1977.*

[BF85] *J. Barwise and S. Feferman, editors.* Model-theoretic logics*. Perspectives in Mathematical Logic. Springer-Verlag, New York, 1985.*

[Boo47] *George Boole.* The Mathematical Analysis of Logic: Being an Essay towards a Calculus of Deductive Reasoning*. Macmillan, Barclay and Macmillan, Cambridge, 1847.*

[BS89] *H. A. Blair and V. S. Subrahmanian. Paraconsistent logic programming.* Theoretical Computer Science*, 68(2):135–154, 1989.*

[Bur98] *S. Burris.* Logic for Mathematics and Computer Science*. Prentice Hall, 1998.*

[Bus98] *Samuel R. Buss. An introduction to proof theory. In Samuel R. Buss, editor,* Handbook of Proof Theory*, pages 1–78, Amsterdam, 1998. Elsevier.*

[CCM07] *Walter Carnielli, Marcelo E. Coniglio, and João Marcos. Logics of formal inconsistency. In* Handbook of Philosophical Logic*, volume 14 of* V.*, pages 1–93. Springer, 2007.*

[Chu36] *Alonzo Church. An unsolvable problem of elementary number theory.* American Journal of Mathematics*, 58(2):pp. 345–363, 1936.*

[Cla78] *Keith L. Clark.* Negations as Failure*, chapter Negative Information and Data Bases, pages 293–322. In Gallaire and Minker [GM78], 1978.*

[CM02] *Walter A. Carnielli and João Marcos. A taxonomy of c-systems. In Walter A. Carnielli, Marcelo E. Coniglio, and Itala M. Loffredo D'Ottaviano, editors,* Paraconsistency - the Logical Way to the Inconsistent*, volume 228 of* Lecture Notes in Pure and Applied Mathematics*, pages 01–94, New York, 2002. Marcel Dekker.*

[Col93] *Alain Colmerauer. The birth of prolog. In* III, CACM Vol.33, No7*, pages 37–52, 1993.*

[dAP07] *Sandra de Amo and Mônica Sakuray Pais. A paraconsistent logic programming approach for querying inconsistent databases.* Int. J. Approx. Reasoning*, 46(2):366–386, 2007.*

[Dav83] *Martin Davis.* The Prehistory and Early History of Automated Deduction*, pages 1–28. Volume 1 of Siekmann and Wrightson [SW83], 1983.*

[dC63] *Newton Affonso Carneiro da Costa.* Inconsistem Formal Systems*. Tese de livre docência, Universidade Federal do Paraná, 1963.*

[Dec05] *Hendrik Decker. A case for paraconsistent logic as foundation of future information systems. In* Information Systems', CAiSE Workshops*, pages 451–461, 2005.*

[DM47] *Augustus De Morgan.* Formal Logic, or, The Calculus of Inference, Necessary and Probable*. Taylor and Walton, Booksellers and Publishers to University College, Upper Gower Street, 28, London, 1847.*

[Doe94] *Kees Doets.* From Logic to Logic Programming*. MIT Press, Cambridge, MA, USA, 1994.*

[DP60] *Martin Davis and Hilary Putnam. A computing procedure for quantification theory.* J. ACM*, 7(3):201–215, 1960.*

[DP98] *Carlos Viegas Damásio and Luís Moniz Pereira. A survey of paraconsistent semantics for logic programs.* Handbook of defeasible reasoning and uncertainty management systems: volume 2: reasoning with actual and potential contradictions*, pages 241–320, 1998.*

[DvH86] *Burton Dreben and Jean van Heijenoort. Introductory note to [Gö30], 1986. Em [Fef86].*

[Fef86] *Solomon Feferman, editor.* Kurt Gödel Collected Works - Volume I - Publications 1929-1936*. Oxford University Press, New York, 1986.*

[Fit91] *Melvin Fitting. Bilattices and the semantics of logic programming.* The Journal of Logic Programming*, 11(2):91 – 116, 1991.*

[Gen35] *Gerhard Gentzen. Untersuchungen über das logische schliessen.* Mathematische Zeitschrift*, 39:176–210, 405–431, 1935.*

[Gen07] *Paolo Gentilini. Paraconsistent arithmetic with a local consistency operator and global selfreference.* Electron. Notes Theor. Comput. Sci.*, 169:73–86, 2007.*

[Gen10] *Paolo Gentilini. Proof-theory and Mathematical Meaning of Paraconsistent C-Systems.* Forthcoming in the Journal of Applied Logic*, 2010.*

[Gir87] *Jean-Yves Girard.* Proof Theory and Logical Complexity*, volume 1. Bibliopolis, 1987.*

[GM78] *Hervé Gallaire and Jack Minker, editors.* Logic and Databases*. Plenum Press, New York, N.Y., 1978.*

[GMN78] *Hervé Gallaire, Jack Minker, and Jean Marie Nicolas.* An Overview and Introduction to Logic and Data Bases*, chapter Introduction, pages 3–30. In Gallaire and Minker [GM78], 1978.*

[GTL89] *Jean-Yves Girard, Paul Taylor, and Yves Lafont.* Proofs and types*. Cambridge University Press, New York, NY, USA, 1989.*

[Gö30] *Kurt Gödel. Die vollständigkeit des axiome des logischen funktionenkalküls.* Monatshefte für Mathematik und Physik*, 37:349–360, 1930. Encontra-se traduzido em [Fef86].*

[HA28] *David Hilbert and Wilhelm Ackermann.* Grundzüge der Theoretischen Logik*. Julius Springer, 1928.*

[HA50] *David Hilbert and Wilhelm Ackermann.* Principles of Mathematical Logic*. Chelsea Publishing Company, 1950. Tradução da segunda edição (publicada em 1938) de [HA28].*

[Hen49] *Leon Henkin. The completeness of the first-order functional calculus.* The Journal of Symbolic Logic*, 14(3):pp. 159–166, 1949.*

[Her30] *Jackes Herbrand.* Recherches sur la théorie de la démonstration. *PhD thesis, Université de Paris, 1930.*

[Hin96] *Jaakko Hintikka.* The principles of mathematics revisited. *Cambridge University Press, 1996.*

[Jas48] *Stanisław Jaskowski. Rachunek zdań dla systemów dedukcyjnych sprzecznych.* Studia Societatis Scientiarum Torunensis, *Sectio A, I(5):57–77, 1948. Traduzido em [Jas99].*

[Jas99] *Stanisław Jaskowski. A propositional calculus for inconsistent deductive systems.* Logic and Logic Philosophy, *7:35–56, 1999.*

[Kam06] *Norihiro Kamide. Foundations of paraconsistent resolution.* Fundam. Inf., *71(4):419–441, 2006.*

[Kow74] *Robert A. Kowalski. Predicate logic as programming language. In* Proceedings IFIP'74*, pages 569–574. North-Holland, 1974.*

[Kow79] *Robert Kowalski. Algorithm = logic + control.* Commun. ACM, *22(7):424–436, 1979.*

[Kow88] *Robert A. Kowalski. The early years of logic programming.* Commun. ACM, *31(1):38–43, 1988.*

[Lei] *Alexander Leitsch. Resolution theorem proving: A logical point of view.*

[Lev86] *H J Levesque. Making believers out of computers.* Artif. Intell., *30(1):81–108, 1986.*

[Lev88] *Hector J. Levesque. Logic and the complexity of reasoning.* Journal of Philosophical Logic, *17:355–389, 1988. 10.1007/BF00297511.*

[Llo93] *John Wylie Lloyd.* Foundations of Logic Programming. *Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1993.*

[Lö15] *Leopold Löwenheim. über möglichkeiten im relativkalkül.* Mathematische Annalen, *76:447–470, 1915.*

[McC80] *John McCarthy. Circumscription—a form of non-monotonic reasoning.* Artificial Intelligence, *13:27–39, 1980.*

[Mck] *Richard Mckinley. A sequent calculus demonstration of herbrand's theorem.*

[Moo83] *Robert C. Moore. Semantical considerations on nonmonotonic logic. In* IJCAI'83: Proceedings of the Eighth international joint conference on Artificial intelligence*, pages 272–279, San Francisco, CA, USA, 1983. Morgan Kaufmann Publishers Inc.*

[MT93] *V. W. Marek and M. Truszczyński.* Nonmonotonic logics; context-dependent reasoning. *Spring-Verlag, Berlin, 1993.*

[Nel49] *David Nelson. Constructible falsity.* The Journal of Symbolic Logic*, 14(1):pp. 16–26, 1949.*

[Nel59] *David Nelson. Negation and separation of concepts in constructive systems. In Arend Heyting, editor,* Constructivity in Mathematics*, Studies in Logic and the Foundations of Mathematics, pages 208–225, Amsterdan, 1959. North-Holland. Proceeding of the Colloquium held in Amsterdan, NL, 1957.*

[NG78] *J. M. Nicolas and H. Gallaire.* Data Base: Theory vs. Interpretation (Nicolas, Gallaire - 1978)*, chapter Data Bases Viewed Through Formal Logic, pages 33–54. In Gallaire and Minker [GM78], 1978.*

[Pea92] *David Pearce. Reasoning with negative information, ii: Hard negation, strong negation and logic programs. In David Pearce and Heinrich Wansing, editors,* Nonclassical Logics and Information Processing*, volume 619 of* Lecture Notes in Computer Science*, pages 63–79. Springer Berlin / Heidelberg, 1992. 10.1007/BFb0031924.*

[Pea93] *David Pearce. Answer sets and constructive logic, ii: extended logic programs and related nonmonotonic formalisms. In* Proceedings of the second international workshop on Logic programming and nonmonotonic reasoning*, pages 457–475, Cambridge, MA, USA, 1993. MIT Press.*

[Pod08] *Rodrigo Podiacki. Logicas da inconsistência formal quantificadas. Dissertação de mestrado, Universidade Estadual de Campinas, 2008.*

[PW90] *David Pearce and Gerd Wagner. Reasoning with negative information i: Strong negation in logic programs. In L. Haaparanta, M. Kusch, and I. Niiniluoto, editors,* Language, Knowledge and Intentionality*, volume 49 of* Acta Philosophica Fennica*, pages 430–453, 1990.*

[PW91] *David Pearce and Gerd Wagner. Logic programming with strong negation. In Peter Schroeder-Heister, editor,* Extensions of Logic Pro-

gramming, *volume 475 of* Lecture Notes in Computer Science*, pages 311–326. Springer Berlin / Heidelberg, 1991. 10.1007/BFb0038700.*

[Rei78] *Raymond Reiter.* On Closed World Data Bases*, chapter Data Bases Viewed Through Formal Logic, pages 55–76. In Gallaire and Minker [GM78], 1978.*

[Rei80] *Raymond Reiter. A logic for default theory.* Artificial Intelligence*, 13:81–132, 1980.*

[Rob57] *A. Robinson. Proving a theorem (as done by man, logician, or machine). In* Summaries of Talks Presented at the Summer Institute for Symbolic Logic*, 1957. Second edition published by the Institute for Defense Analysis, 1960. Reprinted in [SW83], pp. 74–76.*

[Rob65] *J. A. Robinson. A machine-oriented logic based on the resolution principle.* J. ACM*, 12(1):23–41, 1965.*

[Sco70] *Dana S. Scott. Outline of a mathematical theory of computation. Technical Monograph PRG–2, Oxford University Computing Laboratory, Oxford, England, November 1970.*

[Sho67] *Joseph R. Shoenfield.* Mathematical Logic*. Addison-Wesley, Reading, MA, 1967.*

[Sko20] *Thoralf Skolem. Logisch-kombinatorische untersuchungen über die erfüllbarkeit oder beweisbarkeit mathematischer satze nebst einem theoreme über dichte mengen.* Videnskapsselskapets skrifter,I. Matematisk-naturvidenskabeligklasse,no.4.*, 1920. Traduzido como* "Logico-comhinatorial investigations in the satisfiability or provability of mathematical propositions: A simplified proof of a theorem by L. Lowenheim and generalizations of the theorem*" em [vH02].*

[Sko28] *Thoralf Skolem. Über die matematische logik.* Norsk matematisk tidsskrift*, 10:125–142, 1928. Traduzido como "On Mathematical Logic" em [vH02].*

[Soa96] *Robert I. Soare. Computability and recursion.* The Bulletin of Symbolic Logic*, 2(3):284–321, 1996.*

[SW83] *J. Siekmann and G. Wrightson, editors.* Automation of Reasoning: Classical Papers on Computational Logic 1957-1966*, volume 1. Springer, Berlin, Heidelberg, 1983.*

[Tak75] *Gaisi Takeuti.* Proof Theory. *Number 81 in Studies in Logic and the Foundations of Mathematics. North-Holland, 1975. Second edition, 1987.*

[Tar44] *Alfred Tarski. The semantic conception of truth: and the foundations of semantics.* Philosophy and Phenomenological Research*, 4:341–376, 1944.*

[TS96] *A. S. Troelstra and H. Schwichtenberg.* Basic proof theory. *Cambridge University Press, New York, NY, USA, 1996.*

[Tur37] *A. M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem.* Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1937.*

[vEK76] *M. H. van Emden and R. A. Kowalski. The semantics of predicate logic as a programming language.* Journal of the ACM*, 23:569–574, 1976.*

[vH02] *Jean van Heijenoort, editor.* From Frege to Gödel: A Source Book in Mathematical Logic, 1879-1931. *Source Books in History of Sciences. Harvard University Press, January 2002.*

[Wag91a] *Gerd Wagner. A database needs two kinds of negation. In Bernhard Thalheim, János Demetrovics, and H. Gerhardt, editors,* MFDBS 91*, volume 495 of* Lecture Notes in Computer Science*, pages 357–371. Springer Berlin / Heidelberg, 1991. 10.1007/3-540-54009-1*$_2$5.

[WAG91b] *GERD WAGNER. Logic Programming with Strong Negation and Inexact Predicates.* Journal of Logic and Computation*, 1(6):835–859, 1991.*

[Wag93] *Gerd Wagner. Reasoning with inconsistency in extended deductive databases. In* Proceedings of the second international workshop on Logic programming and nonmonotonic reasoning*, pages 300–315, Cambridge, MA, USA, 1993. MIT Press.*

[Wag94] *Gerd Wagner.* Vivid Logic: Knowledge-Based Reasoning with Two Kinds of Negation*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1994.*

[WSBA09] *Claus-Peter Wirth, Jörg Siekmann, Christoph Benzmüller, and Serge Autexier. Lectures on jacques herbrand as a logician. SEKI Report SR-2009-01 (ISSN 1437-4447), SEKI Publications, DFKI Bremen GmbH, Safe and Secure Cognitive Systems, Cartesium, Enrique Schmidt Str.5, D-28359 Bremen, Germany, 2009.*