Introdução à Lógica

José de Oliveira Guimarães
jose@dc.ufscar.br
Departamento de Computação - UFSCar
São Carlos - SP

6 de fevereiro de 2008

Sumário

Prefácio

Imagine que alguém queira comunicar como é uma certa pintura para uma outra pessoa. Uma possibilidade seria converter a pintura para o formato digital, um retângulo compostos por pontos (cada um com uma cor), e depois escrever a cor de cada ponto, começando do canto superior esquerdo e procedendo para baixo, linha por linha, até o canto inferior direito. Esta não é uma forma muito inteligente de comunicação. Contudo, é mais ou menos isso que fazemos quando escrevemos um livro de Matemática, Física ou Lógica. Nestes livros, uma seqüência de definições, teoremas e provas é descrita linearmente tentando descrever conceitos que não podem ser compreendidos desta forma. O leitor fica com a inteira responsabilidade de agrupar todos os conceitos e montar a "figura" em sua mente, com pouco ou nenhum auxílio por parte do texto.

Neste livro tentamos fazer diferente na medida de nossas possibilidades. Para ajudar o leitor a montar a imagem da lógica, recorremos a dois artifícios: a) comentários sobre os teoremas e provas e b) meta-informações no texto. Sendo este um livro de lógica, as informações sobre lógica constituem a maior parte do texto. Contudo, há informações também sobre o próprio texto: referências a páginas ("veja os axiomas da página 23") e figuras que descrevem a própria estrutura das definições, teoremas e provas. Estes dados adicionais são meta-informações. Com eles procuramos mostrar como a lógica é estruturada, quais as relações entre os conceitos estudados. Os comentários no texto ajudam visualizar o quadro geral. Eles procuram apresentar uma outra visão do texto, podemos dizer "do alto", fugindo da descrição linear de definições/teoremas do livro.

Ao final de cada Capítulo deste livro há uma seção chamada de "Complementos" que pode ter três subseções: a) Visão Alternativa; b) Conexões com a Computação e c) Suposições Implícitas. A subseção "Visão Alternativa" procura apresentar os conceitos do Capítulo ou parte dele de maneira diferente do que foi apresentado, facilitando a compreensão do texto. Aliás, isto já é feito em muitos comentários por todo o livro e não apenas nesta subseção. Qualquer texto se torna difícil de entender se apenas uma única visão é apresentada. O leitor não consegue formar uma idéia clara dos conceitos apresentados assim como quando enxergamos com um único olho não temos a idéia de profundidade da cena observada.

A subseção "Conexões com a Computação" apresenta os possíveis relacionamentos entre o Capítulo e a Computação. Não se pretende fazer uma apresentação profunda de tópicos da computação nesta seção. A subseção "Suposições Implícitas" apresenta as suposições implícitas para que o Capítulo seja válido. Frequentemente se assume que algumas coisas sejam válidas ou absolutamente necessárias quando isto nem sempre é o caso. Esta seção apresenta o que normalmente supomos ao estudar o Capítulo. Note que esta subseção de certa forma se sobrepõe com a seção "Visão Alternativa". Contudo, o objetivo de ambas é diferente. O objetivo da subseção "Visão Alternativa" é fazer o leitor compreender melhor o texto. O objetivo da seção "Suposições Implícitas" é abrir a mente, despertar a criatividade, mostrar que qualquer texto supõe um certo raciocínio do leitor implicitamente e que isto poderia ser diferente. O objetivo é despertar o leitor para a pesquisa em Lógica, embora saibamos que este é apenas um texto introdutório de lógica.

Os exercícios deste livro não possuem todos a mesma importância ou nível de dificuldade. Para facilitar a identificação pelo leitor dos exercícios mais importantes para a compreensão do texto e daqueles mais difíceis, este livro adota uma classificação, dada por uma seqüência de letras e números. O nível de importância recebe um de quatro valores:

i4 = muito importante
i3 = importante
i2 = medianamente importante
i1 = pouco importante

A dificuldade de resolução é classificada em

d5 = Difícil
d4 = Medianamente difícil
d3 = Médio
d2 = Relativamente fácil
d1 = Fácil

Um exercício i2d3 é medianamente importante e medianamente difícil.

Capítulo 1

Introdução

Lógica é a disciplina que estuda o raciocínio dedutivo, o que se pode deduzir a partir de premissas consideradas verdadeiras. Como exemplo inicial, a partir das frases "todo par é divisível por 2" e "6 é par", pode-se concluir que "6 é divisível por 2". Não interessa à lógica se as duas primeiras frases são verdadeiras no mundo real — o que interessa é que, se são verdadeiras, pode-se deduzir a terceira frase a partir delas. Mortari [5] dá a seguinte definição de lógica:

Lógica é ciência que estuda princípios de inferência, tendo o objetivo principal de determinar em que condições certas coisas se seguem (são conseqüência), ou não, de outras.

A lógica foi a criação de um único homem, Aristósteles (384 AC-322 AC). A lógica de Aristóteles era inteiramente verbal, sem o emprego de símbolos. Como exemplo, de "Todos os homens são mortais" e "Sócrates é um homem", pode-se deduzir que "Sócrates é mortal". Este é um dos tipos de raciocínio catalogados pelo sábio grego. Havia outros vinte e quatro (dos quais cinco estão implícitos em outros), todos empregando apenas palavras. Já na idade antiga surgiu o primeiro 'paradoxo' lógico quando o cretence Epimênides disse "Todos os cretences são mentirosos". Na verdade, esta frase não é paradoxal, pois ela pode ser tanto verdadeira quanto falsa sem que ocorra uma contradição.[1] Uma frase é paradoxal quando ela não pode ser nem verdadeira nem falsa, como "esta frase é falsa". Outro falso paradoxo é a frase "toda regra tem exceção".

A lógica era uma das disciplinas estudadas tanto na idade antiga quanto na idade média, chegando aos tempos modernos. Contudo, aparentemente nada faltava ao estudo da lógica e esta disciplina não foi um tópico sério de pesquisa até meados do século XIX quando recomeçou o interesse no assunto, começando com Boole, Peano, Fregue e Bertrand Russel. O interesse destes matemáticos era prover uma base sólida para a Matemática.

Há inumeráveis tipos de lógica como lógica clássica, modal, paraconsistente, multivalorada, intuicionista, fuzzy, temporal, quântica, ... A lógica que interessa à Computação é a Lógica Matemática, que abstrai os raciocínios utilizados em Matemática. Esta lógica utiliza três princípios colocados por Aristóteles:

1. princípio da não contradição: não é verdade que uma proposição A e a sua negação sejam

[1] Se tomarmos 'mentiroso' como alguém que mente sempre, então esta frase é falsa. Se fosse verdadeira, como foi dita por um cretence, ela estaria dizendo que ela mesmo é falsa. Contradição. Como esta frase é falsa, Epimenides queria dizer exatamente o contrário do que disse, que alguns cretences não são mentirosos !

verdadeiras ao mesmo tempo. Em símbolos (que estudaremos mais tarde), escrevemos
$\neg(A \wedge \neg A)$

2. princípio do terceiro excluído: ou a proposição A ou a sua negação são verdadeiras. Em símbolos:
$A \vee \neg A$

3. reflexividade da identidade: qualquer coisa é igual a si mesma. Em símbolos:
$\forall x(x = x)$

As lógicas chamadas clássicas, entre as quais se incluem a lógica Matemática, seguem estes três princípios. As lógicas não clássicas (paraconsistente, multivalorada, fuzzy, quântica, etc) não seguem um, dois ou três destes princípios.

No final do século XIX e início do século XX surgiram diversos paradoxos lógicos, classificados como sintáticos (ou lógicos) e semânticos. O paradoxo sintático mais conhecido foi descoberto por Russel em 1902 enquanto estudava teoria dos conjuntos de Cantor. Esta teoria não possui paradoxos, mas estes aparecem quando conjuntos podem ser definidos a partir de uma lei ou fórmula qualquer. Vejamos o paradoxo.

Considere o conjunto C de todos os conjuntos que não contêm a si mesmo:
$C = \{x : x \notin x\}$
Pergunta-se: $C \in C$? Se $C \in C$, então C deve obedecer a regra dada acima que todos os elementos de C devem obedecer, que é $C \notin C$. Contradição. Por outro lado, se $C \notin C$, então C satisfaz as condições necessárias para ser elemento de C e então $C \in C$. Contradição.

Há diversos paradoxos semânticos descritos na literatura. Veremos dois:

1. se a frase "esta frase é falsa" for verdadeira, então ela diz que é falsa. Contradição. Se ela for falsa, então o contrário é verdadeiro, ou seja, ela é verdadeira;

2. paradoxo de Berry (1906). Há um número finito de palavras em Português e portanto um número finito de frases com menos do que vinte palavras. Considere agora os números inteiros definidos com frases com menos do que vinte palavras, como "um", "trezentos e vinte e três", "o terceiro número primo", "o primeiro primo maior do que mil", "a milésima potência do bilhonésimo cubo do fatorial do trilhonésimo quadrado perfeito" (ufa !). Há um número finito de inteiros definidos com frases com menos do que vinte palavras e, portanto, existe um inteiro k que é o maior destes inteiros mais 1. Então "k é o menor inteiro que não pode ser definido com menos do que vinte palavras". Mas esta frase contém menos do que vinte palavras ...

3. paradoxo de Grelling (1908). Um adjetivo é chamado de autológico se a propriedade que ele denota se aplica a si mesmo, heterológico caso contrário. Por exemplo, "vermelho" é heterológico, por este adjetivo não é vermelho (adjetivos não têm cor). Já "polissilábico" é autológico, por esta palavra possui seis sílabas. "Monossilábico" é heterológico. Pergunta-se: é "heterológico" um adjetivo heterológico? Se sim, a propriedade heterológica não se aplica

à palavra "heterológica" e então esta palavra é autológica. Contradição, pois assuminos que a palavra é heterológica. Se "heterológico" é autológico, a propriedade heterológico se aplica à palavra "heterológico" e portanto esta palavra é heterológica.

Os paradoxos semânticos não interessam à lógica que estudaremos, a lógica Matemática. Eles envolvem noções da linguagem natural (Português) que não ocorrem em Matemática.

Começaremos o estudo da lógica apresentando os sistemas formais, que são sistemas capazes de produzir "sentenças", chamadas de teoremas, a partir de axiomas e regras. Esta introdução é necessária porque toda lógica é um sistema formal. No Capítulo 3 será estudado o cálculo proposicional, que é uma lógica bem simples e que faz parte da lógica que nos interessa, que é a lógica de primeira ordem, que será estudada no Capítulo 4. A lógica de primeira ordem utiliza os chamados quantificadores universal (para todo x vale a proposição P(x)) e existencial (existe x tal que P(x)).

Qualquer semelhança entre alguns trechos deste livro e "Introduction to Mathematical Logic" de Mendelson não é mera coincidência. Os teoremas, a ordem de apresentação do material e alguns exemplos são de lá. A prova dos teoremas e o texto é do autor deste livro.

Exercícios Triviais

1.1. *(i1d1) Cite três lógicas. Qual estudamos neste curso ? Por quê ?*

1.2. *(i2d3) Explique o paradoxo de Russel.*

1.3. *(i2d3) Explique o paradoxo de Berry.*

1.4. *(i3d1) Quem criou a lógica ? Em que século ?*

Exercícios de Desafio

1.5. *(i2d3) Faça uma sentença que não é verdadeira nem falsa diferente das que foram apresentadas no texto.*

1.6. *(i1d4) Construa um paradoxo com a seguinte estrutura: ha uma seqüência de frases F_1, F_2, ..., F_n onde cada frase F_i diz algo sobre a falsidade ou veracidade de F_{i+1} (F_n refere-se a F_1). Esta seqüência possui algum tipo de "formato" ou "forma" ? Por exemplo, n deve ser par ?*

Capítulo 2

Uma Introdução Informal a Sistemas Formais

Este capítulo introduz de maneira informal as principais idéias necessárias ao estudo de Lógica. O primeiro sistema formal deste capítulo foi criado por Hofstadter [3] e chama-se sistema MIU. Mas o que é um sistema formal? É um sistema composto por

1. um alfabeto de símbolos composto por quaisquer símbolos que podem ser colocados no papel;
2. seqüências[1] bem definidas de símbolos deste alfabeto chamadas de fórmulas. Deve ser possível definir precisamente o que é fórmula;
3. axiomas (um subconjunto das fórmulas) e
4. regras para produzir novas fórmulas a partir de outras.

Como exemplo, um sistema formal que chamaremos de S utiliza o alfabeto { 0, 1, a, b }. Uma seqüências de símbolos de S é qualquer concatenação destes símbolos, como 01, 000a1, ab, a01b, etc. Uma fórmula é uma seqüência que obedece um certo padrão. Em S, são fórmulas apenas as seqüências que começam com "a" ou com "0", como "a0", "0bb1a0a" e "a0aabbb111000". Não são fórmulas neste sistema formal as seguintes seqüências: "b00" e "1bab". Naturalmente, diferentes sistemas formais têm diferentes alfabetos e diferentes definições do que são fórmulas válidas.

Um axioma é uma fórmula. No sistema S, apenas 01 e ab são axiomas. Todo axioma é considerado um teorema. Uma regra toma um ou mais axiomas ou teoremas como entrada e produz como saída um teorema. Uma pergunta natural a respeito de regras é "qualquer frase que ensine a construir um teorema a partir de axiomas e outros teoremas (o que inclui os axiomas) é uma regra válida ?" Sim, se esta frase puder ser transformada em um algoritmo ou programa de computador que toma teoremas como entrada e produz um teorema como saída. Mas pode-se definir precisamente o que é um programa de computador ? Sim, existe uma definição formal que é estudada em uma área chamada Computabilidade. Para os nossos propósitos, uma regra é válida se for possível fazer um programa em qualquer linguagem de programação que tome um ou mais teoremas como entrada (em formato texto) e produza o texto de um teorema como saída. Pode-se

[1] *string* em Inglês

provar que nem todas as frases que "ensinam" como produzir um teorema a partir de outros pode efetivamente ser convertida em um programa em uma linguagem de programação.

As regras do sistema S são dadas abaixo, onde y é uma seqüência de símbolos qualquer do alfabeto de S, podendo inclusive ser vazia.

1. se 0y é um teorema, 0ay é um teorema;
2. se ay é um teorema, ayy é um teorema.

Quais são então os teoremas de S? Começando pelos axiomas, os teoremas são:

1. 01 (axioma)
2. ab (axioma)
3. 0a1 (01 com regra 1)
4. 0aa1 (0a1 com regra 1)
5. ...
6. abb (ab com regra 2)
7. abbbb (abb com regra 2)
8. ...

Estudaremos agora o sistema MIU. Este sistema utiliza apenas as letras M, I e U (o alfabeto). Algumas combinações destas letras são MUU, IMIU, UUUIMMM, M, I e UIM. Consideraremos qualquer seqüência como uma fórmula. Contudo, estas seqüências não necessariamente são teoremas. Falta definir os axiomas e as regras. O único axioma do sistema é a seqüência MI (então MI é um teorema). As regras são dadas abaixo, onde x e y são seqüências quaisquer formadas pelas letras M, I e U.

Regra 1 Se xI é um teorema, então xIU é um teorema.

Regra 2 Se Mx é um teorema, então Mxx é um teorema.

Regra 3 Se xIIIy é um teorema, então xUy é um teorema.

Regra 4 Se xUUy é um teorema, então xy é um teorema.

Exemplos: MI é um teorema, pois MI é um axioma. De MI, que é um teorema, podemos deduzir que MIU é teorema pela Regra 1. De MI, que é teorema, podemos deduzir MII pela Regra 2. De MIU, que é teorema, podemos deduzir MIUIU pela Regra 2. De MII podemos deduzir MIIII pela Regra 2 e daí deduzir que MUI é um teorema pela Regra 3. Ou deduzir que MIU é teorema pela Regra 3. Note que obtivemos MIU por duas regras diferentes.

Agora que você já sabe como deduzir teoremas, podemos mostrar uma dedução que utiliza a regra 4 sem maiores explicações:

1. MI	axioma
2. MII	Regra 2
3. MIIII	Regra 2
4. MIIIIU	Regra 1
5. MUIU	Regra 3
6. MUIUUIU	Regra 2
7. MUIIU	Regra 4

Os teoremas MI, MIU, MIUIU, MIIII e MUIIU foram obtidos do único axioma (no caso, MI) ou por aplicações das regras a partir de teoremas previamente deduzidos. Estes são teoremas *do sistema formal MI*. Existe um outro tipo de teorema utilizando MI: os meta-teoremas. Um meta-teorema é um teorema sobre o próprio sistema MI, não um teorema composto por M, I e U. Um exemplo bem simples e óbvio de meta-teorema é este:

[Meta-teorema] Todos os teoremas do sistema MIU começam pela letra M.

O teorema é tão claro que dispensa explicações. Um outro meta-teorema deste sistema é

[Meta-teorema] Mx é um teorema, onde x é uma seqüência contendo apenas um número de I´s que é potência de 2.

Em Lógica, estamos interessados quase todo o tempo nos meta-teoremas, não nos teoremas do próprio sistema. O leitor é convidado a descobrir um meta-teorema a respeito deste sistema. Um bom exercício é provar que MU não é um teorema deste sistema. Note que esta prova nunca poderia ser um teorema, pois é algo **sobre** o sistema MIU.

Estudaremos agora um outro sistema formal descrito por Hofstadter [3], o sistema pq. Este sistema emprega apenas os símbolos p, q e ⊙ (o alfabeto) e qualquer combinação destes símbolos é uma fórmula. Este sistema possui um infinito número de axiomas. Mas ... se é infinito, como podemos descrevê-los? Através do que chamamos "esquema de axioma", que é a forma geral do axioma. O único "esquema de axioma" deste sistema é

[Esquema de axioma] x p ⊙ q x ⊙ é um axioma se x é composto apenas por ⊙´s.

Note que x não pertence ao alfabeto do sistema formal pq. x é um meta-símbolo: neste caso ele representa um ou mais símbolos do sistema pq.

São axiomas de pq:

⊙ p ⊙ q ⊙⊙
⊙⊙ p ⊙ q ⊙ ⊙ ⊙
⊙ ⊙ ⊙⊙ p ⊙ q ⊙ ⊙ ⊙ ⊙ ⊙

Naturalmente, existe um infinito número de axiomas. A única regra é

[Regra] Se x, y e z são seqüências contendo apenas ⊙´s e
x p y q z
é um teorema, então
x p y ⊙ q z ⊙
é um teorema.

Por exemplo, ⊙ p ⊙ q ⊙⊙ é um teorema, pois é um axioma. Aplicando a regra, obtemos
⊙ p ⊙⊙ q ⊙ ⊙ ⊙

Os sistemas pq e MIU são muito interessantes pois nos permitem produzir inumeráveis seqüências de símbolos a partir de regras ... mas para que queremos estas seqüências? Seqüências quaisquer de símbolos sem significado não nos interessam, são inúteis. Mas e se associarmos "significado" aos símbolos? No sistema pq, podemos assumir que x p y q z significa que $\tilde{x} + \tilde{y} = \tilde{z}$, onde $\tilde{x}$ é o número de ⊙ que ocorrem em x (o mesmo para y e z). Então p está sendo interpretado como "plus" (soma) e q como "equals" (igual). Será que funciona? Verifiquemos, começando pelos axiomas. ⊙ p ⊙ q ⊙⊙ é um axioma, que pode ser interpretado como $1 + 1 = 2$, onde $\tilde{x} = 1$, $\tilde{y} = 1$ e $\tilde{z} = 2$. De forma geral, o esquema de axioma diz um número $\tilde{x}$ somado a 1 é igual a um número com um ⊙ a mais, que é $\tilde{x} + 1$.

A regra diz que se $\tilde{x} + \tilde{y} = \tilde{z}$, então $\tilde{x} + (\tilde{y} + 1) = (\tilde{z} + 1)$. Então temos um mapeamento dos símbolos deste sistema formal para os símbolos utilizados na aritmética:

p	mais (plus)
q	igual
⊙	um
⊙⊙	dois
⊙ ⊙ ⊙	três
...	...

Este mapeamento é chamado de interpretação. Estamos dando uma interpretação aos símbolos do sistema formal até então sem nenhum significado. Os símbolos são relacionados com objetos do "mundo real", algo que acreditamos que exista, como os números naturais.

Uma pergunta que se faz é se esta interpretação não está de certa forma embutida ou incorporada no sistema formal, nos axiomas e regras do sistema. Parece difícil que a interpretação não esteja, já que, neste exemplo, parece que o sistema pq foi feito especialmente para a soma de dois números. Contudo, isto não é verdade. Este mesmo sistema pode ser interpretado de mais de uma forma. Por exemplo, no sistema pq a seqüência

x p y q z

pode ser interpretado como $\tilde{x} = (-\tilde{y}) + \tilde{z}$ e p seria "equals" (igual) e q seria "plus" (soma). Concluímos que os símbolos do sistema formal não têm nenhum significado intrínsico — nós os interpretamos da maneira que nos convêm. Mas sempre respeitando os axiomas e regras de dedução do sistema. Por exemplo, estaria incorreto associar p à multiplicação e q a igual. Se assim fizéssemos, teríamos um teorema dizendo que

$1.1 = 2$

Exercícios Triviais

2.1. *Explique o que é um sistema formal, axioma, regra de dedução, teorema e meta-teorema.*

2.2. *Faça um sistema formal que utilize um alfabeto de quatro símbolos, tenha dois axiomas e pelo menos duas regras de dedução.*

2.3. *(i4d2) (r) Faça um meta-teorema para o sistema MIU.*

2.4. *(i3d2) Deduza o teorema*

$\odot\ p\ \odot\odot\odot\ q\ \odot\odot\odot\odot$

no sistema pq.

2.5. *(i4d2) Faça um meta-teorema para o sistema pq. Dica: qual o formato dos teoremas ?*

2.6. *(i1d2) O sistema pq é capaz de representar a soma de dois números quaisquer ? Ou x, em "x p y q z", precisa ser maior do que y?*

Exercícios Criativos

2.7. *(i2d3) (r) Seja S o sistema formal que utiliza o alfabeto* $\{ E, T, N, +, *, 0, 1, \ldots 9 \}$. *O único axioma é* E *e qualquer seqüência de símbolos é uma fórmula. As regras de dedução são:*

1. *Em um teorema qualquer,* E *pode ser substituído por* $E + T$ *ou* T*;*
2. *Em um teorema qualquer,* T *pode ser substituído por* $T * N$ *ou* N*;*
3. *Em um teorema qualquer,* N *pode ser substituído por* 0 *ou* 1 *ou ... ou* 9*.*

Escreva alguns teoremas deste sistema formal. Alguns teoremas nunca poderão ser utilizados para a construção de outros teoremas, pois eles não possuem as letras E, T *ou* N*. Com o que se parecem estes teoremas ?*

2.8. *(i2d4) Considere o seguinte sistema formal S:*

- *alfabeto =* $\{$ *0, 1, 2, 3, 4, 5, 6, 7, 8, 9,* $+, -$ $\}$*;*
- *qualquer seqüência de letras do alfabeto é uma fórmula;*
- *todo número é um axioma. Um número é uma seqüência de um ou mais dígitos como 0, 5, 32, 8213, etc;*
- *se* A *e* B *são teoremas, as regras de dedução são: a)* $(A+B)$ *é teorema; b)* $(A-B)$ *é teorema.*

Como são os teoremas deste sistema formal ? Podemos generalizar e construir um sistema formal onde todos os programas válidos da linguagem C/C++/Pascal/Java sejam teoremas ?

Exercícios de Desafio

2.9. *(i2d2) Acrescente a seguinte regra ao sistema pq: Se x, y e z são seqüências contendo apenas* $\odot$ *e*

$x\ p\ y\ q\ z$

é um teorema, então

$y\ p\ x\ q\ z$

é um teorema.

Faça uma prova de $\odot\odot\ p\ \odot\odot\ q\ \odot\odot\odot\odot$ *que utilize esta nova regra. Interpretando as seqüências como números, a que regra da aritmética corresponde este teorema ?*

2.10. *(i2d5) É MU um teorema do sistema MIU ?*

2.1 Complementos

2.1.1 Visão Alternativa

Um sistema formal pode ser comparado a um conjunto de peças de plástico com regras de como combiná-las (brinquedos do tipo Lego). As peças possuem saliências e depressões para se encaixarem umas nas outras. Duas peças quaisquer não necessariamente podem ser agrupadas pois elas podem não se encaixar. As regras do sistema formal estariam subentendidas nas saliências e depressões nas peças. O alfabeto seriam as próprias peças, cada peça seria um axioma e um teorema é qualquer agrupamento de peças.

Um sistema formal nada mais é do que um jogo de compor peças segundo regras. E observe que nestes brinquedos, os símbolos do alfabeto (peças) e os teoremas possuem três dimensões. Voltaremos a este tópico mais tarde.

2.1.2 Conexões com a Computação

Regras Como Algoritmos

Uma conexão com a computação foi dada na definição de uma "regra" de um sistema formal. Uma regra deve poder ser transformada em um algoritmo. E o que é exatamente um algoritmo ? É uma seqüência de instruções que pode ser colocada em um programa escrito em qualquer linguagem de programação.[2]

Como um exemplo, a regra 1 do sistema MIU pode ser codificada em linguagem Java como

```
String rule1(String formula) {
       // verifica se o último caráter da fórmula é I
   if ( formula.charAt(formula.length() - 1) == ´I´ )
      return formula + "U";
   else
      return null;
}
```

Programas Como Sistemas Formais

Podemos encontrar um sistema formal que faz a mesma coisa que um programa qualquer ? Aparementemente sim, pois afinal um programa é composto de algoritmos e as regras so sistema formal também. Veremos que a respostas é "sim". Um programa toma uma entrada, altera-a através de sucessivas instruções e no final de sua execução produz uma saída.[3] Então um programa qualquer toma uma seqüência de bits como entrada e produz uma seqüência de bits como saída feita através

[2]Rigorosamente, admite-se que o programa possa alocar uma quantidade potencialmente infinita de memória, o que não é o caso real, já que todos os computadores possuem uma memória finita.

[3]Não precisamos considerar os programas interativos que tomam entradas e produzem saídas *durante* a sua execução — simplesmente podemos considerar que estes programas começam uma nova execução após uma nova entrada e terminam a sua execução após uma saída.

de sucessivas modificações da entrada. Veremos então com converter um programa P qualquer em um sistema formal.

A entrada do programa P corresponde ao único axioma do sistema formal. Então para cada combinação P/entrada temos um sistema formal diferente. As regras correspondem às alterações feitas na entrada, pelo programa, através de suas instruções, até se obter uma saída. A saída seria um teorema do sistema formal. Pode-se assumir que um teorema é uma saída quando tiver um símbolo especial como último caráter. Em um caso extremo, teríamos uma única regra que seria aplicada uma única vez. A regra toma o axioma, que corresponde à entrada, e produz um teorema, que corresponde à saída. E fim.

Contudo, é mais didático imaginarmos que a cada instrução do programa, seja ele feito em uma linguagem de alto nível ou em assembler, corresponde a uma regra do sistema formal. Como exemplo, estudaremos um sistema formal (incompleto !) equivalente a um programa que soma uma seqüência de números. O axioma poderia ser a seqüência

$$100111 - 10101 - 11111 - 1 - 10$$

As regras manipulam os bits de tal forma que o teorema final seja

$$1011110\#$$

O # indicaria que nenhuma regra deveria ser aplicada deste ponto em diante.

O que aconteceu foi que colocamos os números de entrada, 100111, 10101, 11111, 1 e 10 no axioma em um formato apropriado, separados por $-$. Note que o alfabeto deste sistema formal é $\{0, 1, -, \#\}$. Depois de aplicar as regras do sistema várias vezes, obtemos um teorema (que é sempre terminado por # por nossa convenção). As regras produzem uma seqüência de bits que é exatamente a somatória dos números iniciais. Observe que temos que passar a entrada para o formato esperado pelo sistema formal e interpretar a saída de acordo com a convenção que utilizamos. Note que as regras nada sabem sobre esta nossa convenção — regras fazem manipulações mecânicas de símbolos sem conhecer a semântica deles. Naturalmente, as regras não violam o significado da nossa convenção, elas de alguma forma são coerentes com o que queremos do programa.

E quanto às regras deste sistema formal, com seriam elas ? Tudo o que podemos dizer é que seriam um tanto complexas para serem colocadas aqui, mas que existem. Para compreender isto, basta dizer que elas seriam equivalente às intruções do assembler de um computador como `mov`, `add`, `xor`, etc.[4]

Autômata Celular e Jogo da Vida

Um autômata celular [7] consiste de um conjunto quadriculado de células infinito, cada uma das quais pode estar em um conjunto finito de estados. O quadriculado pode ter qualquer número finito de dimensões. A Figura 2.1.2 mostra três autômatas celulares de duas dimensões onde cada célula pode estar em um de dois estados: branco ou preto. Um autômata celular se modifica em intervalos discretos de tempo. A cada passo, intervalo discreto, novos valores de estado são calculados para cada célula baseado em um número finito de células vizinhas. A célula assume este

[4] A primeira regra seria aplicada se a seqüência não começasse com #. A partir daí as regras iriam numerando as seqüências até que se chegue ao teorema final, que termina com #. Para exemplificar, se o axioma é $101 - 11$, a seqüência de teoremas poderia ser $\#0\#101 - 11$, $\#1\#...$, $\#2\#...$, ..., $1000\#$. O número no início da seqüência funciona com o contador de programa (PC) ou estado de uma máquina de estados finitos. Note que as regras podem utilizar a seqüência sendo transformada para armazenar números intermediários — seria a memória do computador.

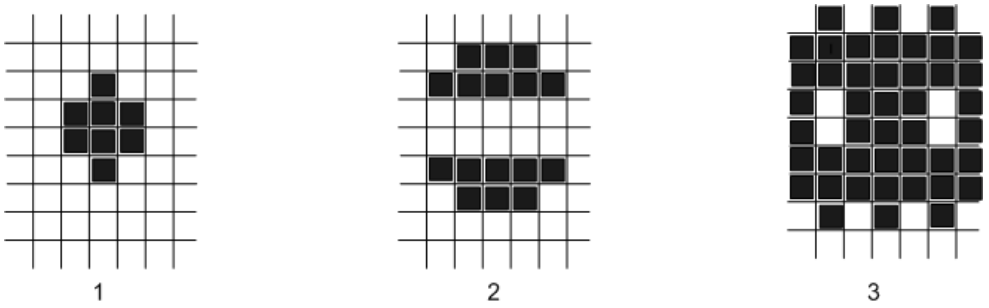

Figura 2.1: Três gerações de um autômata celular

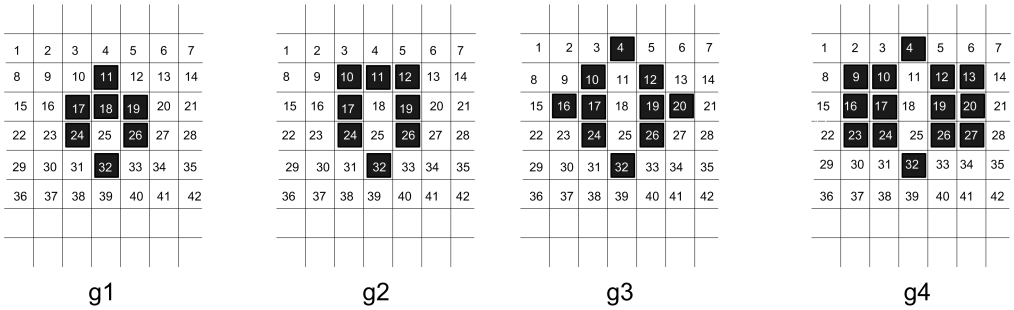

Figura 2.2: Quatro gerações de um jogo da vida

novo estado no próximo passo, também chamado de próxima geração. Por exemplo, o valor de uma célula pode ser calculado considerando-se os estados das duas células verticais e duas células horizontais adjacentes a ela. Se o autômata tem n estados, haveria n^4 diferentes possibilidades de cálculo para cada célula. Cada célula utiliza a mesma função de cálculo, independente de onde ela está. Esta função de cálculo será chamada de "regra" para cálculo do estado da célula.

A Figura 2.1.2 mostra um autômata onde a função que calcula o próximo estado de uma célula considera todos os vizinhos imediatos da célula: duas células horizontais, duas verticais e quatro diagonais. A função retorna o estado preto se o número de células vizinhas pretas (sem contar com a própria célula) é par. Caso contrário retorna o estado branco. A Figura mostra três gerações de um autômata.

O jogo da vida [2] é um autômata celular inventado por John Conway na década de 60. O autômata utiliza dois estados, branco e preto. O jogo da vida pode utilizar regras quaisquer para passar de uma geração para outra. Citamos abaixo as regras originais de Conway. Estas regras ensinam como produzir células brancas e pretas de um geração para a próxima. Cada célula possui oito vizinhos: dois horizontais, dois verticais e quatro diagonais.

Regras:

- uma célula preta com dois ou três vizinhos pretos sobrevive para a próxima geração; isto é, continua preta;
- uma célula preta com um vizinho preto ou nenhum é substituído por uma célula branca (vazia). É como se a célula morresse por solidão;
- uma célula preta com quatro ou mais vizinhos pretos é substituída por uma célula branca. É como se a célula morresse por superpopulação;

- uma célula branca que possui exatamente três células vizinhas pretas é substituída por uma célula preta na próxima geração.

A Figura 2.1.2 representa quatro gerações de um jogo. O leitor é convidado a aplicar as regras acima em uma geração para conseguir a próxima. Cada regra deve ser aplicada baseada na geração anterior, não na geração que você está montando atualmente. Veremos como a geração 2 foi construída a partir da 1. A geração 1 possui células brancas adjacentes a três células pretas nos cantos superior esquerdo e direito do "desenho" formado pelas células pretas. Estas células ficam pretas na geração 2. Na geração 1 há uma célula preta no interior do desenho que é adjacente a cinco células pretas. Esta célula desaparece na geração 2. Aplicando-se as regras para cada célula, obtem-se a geração 2 como apresentada na Figura.

O jogo da vida é um jogo incrivelmente interessante, embora extremamente simples. Chamando a configuração de células brancas e pretas de "padrão", há padrões que, depois de um certo número de gerações:

- se repetem, entrando então em um laço infinito. A configuração pode ficar enorme e depois diminuir até conter apenas poucas células pretas;
- atingem um estado estável, não mais se modificando. Como um exemplo, um quadrado formado por quatro células pretas, isoladas de quaisquer outras células, nunca se modifica;
- se tornam simétricos, apesar do padrão iniciar não o ser.

Há ainda padrões que "se movem" através do quadriculado, que criam e disparam padrões que se movem (como um canhão que atira um projétil) e outros que crescem sem parar. O leitor é convidado a pesquisar na Internet e experimentar com as dezenas de implementações do jogo da vida disponíveis na world wide web.

Gramáticas Como Sistemas Formais

Uma gramática G é uma quádrupla (N, Σ, P, S) onde

- N é o conjunto de símbolos não-terminais;
- Σ é o conjunto de símbolos terminais;
- P é um conjunto de produções;
- S é o símbolo não-terminal inicial da gramática.

Como um exemplo, considere uma gramática G que expressa algumas expressões aritméticas válidas. O conjunto N = { E, T, N }, Σ = { 0, 1, 2, ... 9 }, S é o símbolo E e o conjunto P é composto pelas "regras de produção" dadas a seguir:

1. E ::= E + T
2. E ::= T
3. T ::= T $*$ N

4. T ::= N
5. N ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 7 | 8 | 9

O símbolo | não faz parte da gramática, sendo utilizado apenas para significar "ou". Assim, na regra 5 acima, N é 0 ou 1 ou ... 9. Pode-se eliminar o símbolo | escrevendo-se dez regras para N:

N ::= 0
N ::= 1
...
N ::= 9

A gramática G é utilizada para produzir seqüências de símbolos partindo-se do símbolo inicial E e substituindo-se E por E + T ou T de acordo com a primeira "regra de produção" dada acima. A partir daí, pode-se substituir qualquer dos símbolos não terminais, do conjunto N, pelo lado direito de "::=" de qualquer regra de produção. Por exemplo, pode-se substituir T por T $*$ N ou N. Uma seqüência de substituições é chamada de *derivação*. Como exemplo, apresentamos uma derivação de E onde cada passo é seguido do símbolo $\Longrightarrow$.

$$\mathrm{E} \Longrightarrow \mathrm{E} + \mathrm{T} \Longrightarrow \mathrm{T} + \mathrm{T} \Longrightarrow \mathrm{N} + \mathrm{T} \Longrightarrow 1 + \mathrm{T} \Longrightarrow 1 + \mathrm{N} \Longrightarrow 1 + 2$$

No primeiro passo utilizamos a regra 1, no segundo a regra 2 e no terceiro a regra 4.

A cada gramática G pode-se associar um sistema formal SF_G tal que alguns dos teoremas deste sistema correspondem às sentenças que podem ser produzidas pela gramática. SF_G utiliza um alfabeto contendo os símbolos terminais e não terminais de G. O único axioma é o símbolo inicial da gramática. As regras de dedução de SF_G são baseados nas regras de produção da gramática. Uma regra de produção

A ::= α

resulta em uma regra

Em um teorema qualquer, A pode ser substituído por α

do sistema formal.

Na gramática dada acima, o sistema formal teria como alfabeto o conjunto { E, T, N, +, $*$, 0, 1, ... 9 }. O axioma seria o símbolo E e as regras de dedução seriam:

1. Em um teorema qualquer, E pode ser substituído por $E + T$ ou T;

2. Em um teorema qualquer, T pode ser substituído por $T * N$ ou N;

3. Em um teorema qualquer, N pode ser substituído por 0 ou 1 ou ... ou 9.

Um teorema em que não ocorre nenhum símbolo não terminal é chamado de sentença da gramática. Assim, são sentenças desta gramática os teoremas $0 + 1$, $1 * 2 + 3 * 7$ e $2*$. Qualquer linguagem de programação pode ser descrita por uma gramática e os programas válidos podem ser gerados por um sistema formal como descrito acima.[5]

[5]O sistema formal certamente produzirá alguns teoremas que não correspondem a programas válidos, que neste caso seriam programas sintaticamente corretos mas com erros semânticos (utilizando a terminologia de Compiladores).

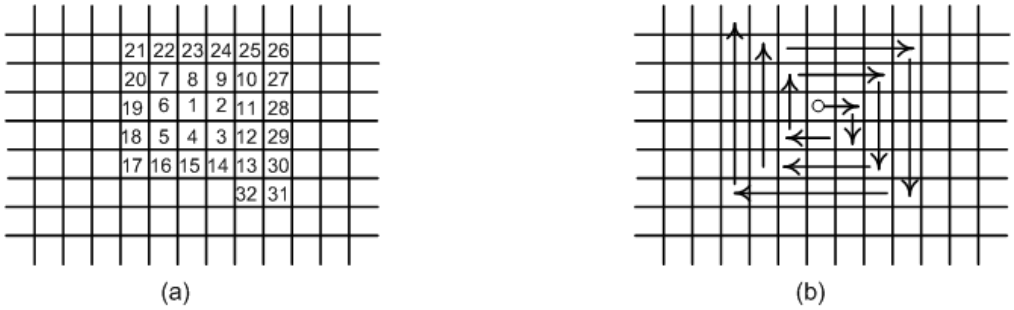

Figura 2.3: Numeração de símbolos em um sistema formal em duas dimensões

2.1.3 Suposições Implícitas

Ao apresentar um sistema formal, admitimos que o alfabeto deve ser apresentado em uma única dimensal, uma linha. Poderíamos ter um sistema formal (SF) que utiliza duas dimensões ? Claramente, sim. Mas este SF poderia ser mais poderoso do que qualquer SF convencional, unidimensional ? Isto é, haveria um SF em duas dimensões que jamais poderia ser simulado em uma única dimensão ?

Não, qualquer sistema formal de duas dimensões poderia ser simulado em uma única dimensão. Para mostrar como, observe a figura 2.1.2. Um conjunto de símbolos colocado neste quadriculado, de duas dimensões, poderia ser mapeado para uma dimensão seguindo-se os números em ordem crescente. Se um axioma fosse os números da figura (a), em duas dimensões, o axioma correspondente seria, em uma dimensão, igual a 1, 2, 3, 4, 5, 6, 7, 8, ... As setas da figura (b) indicam como seriam mapeados os símbolos. Desta forma, pode-se converter qualquer fórmula bidimensional em uma fórmula unidimensional. E mais importante, de maneira biunívoca: a cada fórmula bidimensional corresponde a uma única fórmula unidimensional e vice-versa. Possivelmente teríamos que introduzir um símbolo no sistema formal de uma dimensão para representar um quadrado vazio no sistema formal de duas dimensões. Mas pode ser feito.

E quanto às regras ? Podemos converter uma regra que manipula fórmulas em duas dimensões para uma regra que manipula fórmulas em uma dimensão ? Claramente, sim. Existe uma função que mapeia cada posição do quadriculado em uma posição unidimensional. Utilizando esta função, o mapeamento entre a regra bidimensional e a regra unidimensional torna-se trivial. Basta aplicar a função sempre que a regra bidimensional referir-se à posição um símbolo. E que função é esta ? Se ela se chama f e adotarmos que a posição 0 do exemplo (a) é (0, 0), então teríamos $f(0,\ 0) = 0$, $f(1,\ 0) = 1$, $f(1,\ -1) = 2$, $f(0,\ -1) = 3$, etc. Fazer esta função fica como exercício ao leitor. Observe que raciocínio acima pode ser generalizado para três ou mais dimensões.

Uma outra suposição implícita na apresentação dos sistemas formais foi a de que existe uma quantidade enumerável[6] de símbolos do alfabeto. Esta suposição é necessária pois pode-se provar que nenhuma regra consegue manipular uma quantidade não enumerável de símbolos. Contudo, poder-se-ia criar um sistema em que o alfabeto contivesse uma quantidade não enumerável de símbolos. Como exemplo, o alfabeto poderia ser os números reais.

Não seria possível um sistema formal onde a criação de teoremas fosse feita de maneira contínua ? Como definido neste Capítulo, um teorema é obtido depois de n passos, n inteiro. E se n pudesse

[6]O alfabeto é finito ou os símbolos podem ser colocados em uma relação bijetora com os números naturais.

ser um número real ? Uma regra deveria tomar não só as premissas como parâmetros como também n real. Haveria um contínuo entre os axiomas e teoremas. Aparentemente, nada de novo poderia ser ganho com esta idéia.

Um axioma é um conjunto bem definido de símbolos. Esta é uma suposição implícita na definição de sistema formal. Poderia ser diferente. Um axioma X poderia ser definido como todas as fórmulas B tal que a distância entre A e B fosse menor do que 1, onde A é uma fórmula fixada. A distância entre duas fórmulas poderia ser definida por um algoritmo. Então o axioma X seria composto por todas as fórmulas cuja distância de A é menor do que 1. Um axioma é agora uma *nuvem* de fórmulas, não apenas uma.

Exercícios Criativos

2.11. *Crie e estude o comportamento de um autômato celular de dois estados (branco, preto) e infinito em que o próximo estado de uma célula depende de todos os infinitos estados do autômato. Temos algumas sugestões:*

(a) *associe o valor 1 a preto e 2 a branco. Para calcular o novo estado de uma célula C, considere s_0 como a soma dos valores das oito células adjacentes à célula C. Estas células formam um anel de oito células ao redor de C. O anel mais externo a esta célula é formado por 16 células cuja soma dos valores é s_1. Calcule a soma $s = s_0 + \frac{s_1}{2} + \frac{s_2}{4} + \ldots$. A célula C será preta na próxima geração se s é ímpar e branca se s é par;*

(b) *simule um universo onde cada célula preta é um pedaço de massa. Admita que matéria atrai matéria na proporção inversa da distância que os separa. As células pretas iriam se movimentando no quadriculado se acordo com as atrações de massa. Cuidado deve ser tomado para manter contante o número de células pretas neste universo.*

2.12. *Discuta a viabilidade de um autômato celular que: a) possui dois estados (branco, preto) b) é infinito e; c) o próximo estado de uma célula depende dos estados futuros das oito células adjacentes. Como exemplo, o próximo estado da célula será preto se o número de células adjacentes pretas é par na quinta geração a partir de agora. E branco caso contrário.*

Capítulo 3

Cálculo Proposicional

Uma lógica possui uma parte composta por axiomas, teoremas, provas e regras como definido nos sistemas formais. Esta parte é chamada de sintaxe pois se interessa apenas pela forma dos axiomas e teoremas, sem se importar com o que eles significam. Um teorema é apenas uma seqüência de símbolos que não significam nada para o estudo sintático da lógica. Quando associamos significado aos símbolos, os teoremas passam a significar alguma coisa. Esse aspecto de uma lógica é chamada de semântica.

3.1 A Linguagem do Cálculo Proposicional

O cálculo proposicional (CP) é uma lógica bem simples que será incorporada à lógica de primeira ordem, a ser vista no próximo capítulo. O CP é um sistema formal (pois é uma lógica) cuja linguagem o seguinte alfabeto: $\neg$, $\longrightarrow$, (,) e as letras V_i com inteiros positivos como subescritos: V_1, V_2, ... Os símbolos $\neg$ e $\longrightarrow$ são chamados de conectivos primitivos e as letras V_i são as variáveis. O símbolo $\neg$ lê-se "negação" ou "não é verdade que". O símbolo $\longrightarrow$ é chamado de condicional ou implica. Não se confunda: estes conectivos possuem estes nomes mas são tão somente símbolos no papel sem qualquer significado.

As fórmulas do cálculo proposicional são definidas como

(a) uma variável é uma fórmula;

(b) $\neg A$ e $(A \longrightarrow B)$ são fórmulas se A e B são fórmulas;

(c) fórmulas são descritas apenas pelos itens (a) e (b).

A partir de agora, utilizaremos as letras A, B, C, ... para fórmulas do CP. Em $A \longrightarrow B$, chamamos A de antecedente e B de consequente.

3.2 Semântica do Cálculo Proposicional

O estudo semântico de uma lógica é o estudo do significado que podemos dar às fórmulas da linguagem. No cálculo proposicional, associaremos o significado esperado (ou trivial, se preferir)

às fórmulas. Diremos que estamos "interpretando" o CP. Esta interpretação é descrita abaixo.

Cada variável que aparece em uma fórmula pode receber um de dois valores: V ou F. Tanto faz o nome destes valores, poderia ser 0 e 1, T e F, qualquer coisa. A única exigência é que sejam valores distintos. Naturalmente, associamos V a verdadeiro e F a falso, o que é apenas uma associação feita em nossas mentes e que não interessa à teoria.

Associando um valor (V ou F) a cada variável de uma fórmula C, podemos calcular o valor da fórmula. Como fazer isto ? Comecemos pelas fórmulas básicas:

$\neg A$ é V se A for F e F se A for V;

$A \longrightarrow B$ é F se A for V e B for F, V em todos os outros casos.

Os conectivos derivados possuem o significado esperado:

$A \wedge B$ é V se A e B forem ambos V;

$A \vee B$ é V se A ou B for V;

$A \longleftrightarrow B$ é V se A e B forem ambos V ou se forem ambos F;

3.2.1 Tabelas Verdade

Esta descrição de verdade/falsidade é usualmente colocada em o que chamamos de "tabelas verdade". A tabela verdade da negação é

A	$\neg A$
V	F
F	V

Esta tabela diz que, se a fórmula A for V, $\neg A$ será F. E se A for F, $\neg A$ será V. Note que ao invés de colocar A como variável (V_1, V_2, ...) a colocamos como fórmula.

A tabela verdade do operador condicional é mostrada abaixo.

A	B	$A \longrightarrow B$
V	V	V
V	F	F
F	V	V
F	F	V

$A \longrightarrow B$ é F apenas quando A é V e B é F. Será que faz sentido considerar $A \longrightarrow B$ verdadeiro quando A é F e B é V ? Para responder a isto, temos que recorrer à Matemática, já que estudamos Lógica Matemática, estudamos os raciocínios válidos em Matemática. Veja a seguinte dedução, criada por Smullyan:

$$
\begin{aligned}
5 &= 5(\text{subtraia 1}) \\
4 &= 4 \\
2^2 &= 2^2(\text{substitua por algo equivalente}) \\
2^2 &= (-2)^2(\text{extraia a raiz}) \\
2 &= -2(\text{subtraia 1}) \\
1 &= -3
\end{aligned}
$$

Naturalmente, esta dedução está errada, pois se $a^2 = b^2$, não podemos deduzir que $a = b$. Mas e a dedução "Se $1 = -3$ então $5 = 5$" ? Refaça os passos ao contrário e não encontrará nenhum erro. A Matemática admite que se A é falso e B é verdadeiro, então $A \longrightarrow B$ é verdadeiro.

Um outro exemplo, citado por Mendelson [4], é a frase "se x é um inteiro ímpar, então x^2 é um inteiro ímpar" onde A é "x é um inteiro ímpar" e B é "x^2 é um inteiro ímpar". Se x for par, queremos que a frase inteira seja considerada falsa ? Claramente não. Então se x é par temos um caso onde A é F e queremos que $A \longrightarrow B$ seja V. É por causa de raciocínios como estes, da Matemática, que a tabela verdade de $\longrightarrow$ considera $A \longrightarrow B$ verdadeiro sempre que A é F.

Em língua natural, sempre se exige alguma conexão causal entre A e B quando se emprega a sentença $A \longrightarrow B$. Em Lógica Matemática, sendo uma disciplina formal, que nada conhece do mundo físico, não se exige nenhuma conexão entre o antecedente A e o conseqüente B. Então, as sentenças abaixo são verdadeiras na lógica que estudamos:

1. se 12 é primo, então o Sol existe;

2. se 6 é primo, hoje é quarta-feira.

Note que na última sentença, "hoje é quarta-feira" assume V ou F dependendo do dia em que a frase é lida. Não interessa se ela é V ou F. A sentença é verdadeira de qualquer forma.

O CP utiliza alguns conectivos derivados, que não pertencem à linguagem mas que são empregados por comodidade. Estes conectivos já foram definidos mas repetimos aqui a sua definição:

D1 $(A \wedge B)$ é $\neg(A \longrightarrow \neg B)$;

D2 $(A \vee B)$ é $(\neg A) \longrightarrow B$;

D3 $(A \longleftrightarrow B)$ é $(A \longrightarrow B) \wedge (B \longrightarrow A)$.

As tabelas verdade destes conectivos podem ser obtidas das tabelas do $\neg$ e $\longrightarrow$ e são dadas

abaixo.

A	B	$A \wedge B$
V	V	V
V	F	F
F	V	F
F	F	F

A	B	$A \vee B$
V	V	V
V	F	V
F	V	V
F	F	F

A	B	$A \longleftrightarrow B$
V	V	V
V	F	F
F	V	F
F	F	V

Em Português (ou outras línguas como Inglês), "A ou B" pode ser utilizado para duas coisas diferentes:

- ou A ou B mas não ambas as coisas ao mesmo tempo (ou exclusivo);

- A ou B ou ambos A e B (ou inclusivo).

Em lógica, como pode ser comprovado examinando-se a tabela verdade acima, utilizamos o segundo significado, ou inclusivo.

O operador bicondicional, representado por $\longleftrightarrow$ indica que duas sentenças são **logicamente equivalentes**: $A \longleftrightarrow B$ é V se e somente se A é V quando B é V e A é F quando B é F.

Cada tabela verdade define uma **função de verdade** que contém um argumento para $\neg$ e dois argumentos para todos os outros conectivos. O resultado da função é V ou F. Assim, a tabela verdade para $\neg$ define uma função

$$f_{\neg} : \{V, F\} \longrightarrow \{V, F\}$$

e, por exemplo, $\wedge$ define uma função

$$f_{\wedge} : \{V, F\}^2 \longrightarrow \{V, F\}$$

Cada uma destas funções de verdade será sempre representada por uma tabela verdade.

Pode-se construir uma tabela verdade para uma fórmula qualquer, por exemplo, para $(\neg A \wedge B) \longrightarrow A$:

A	B	$\neg A$	$\neg A \wedge B$	$(\neg A \wedge B) \longrightarrow A$
V	V	F	F	V
V	F	F	F	V
F	V	V	V	F
F	F	V	F	V

Note que colocamos os passos intermediários na tabela para facilitar o cálculo da fórmula completa. As linhas da tabela correspondem a todas as variações possíveis das entradas da função de verdade, que neste caso são A e B. Se há n entradas, temos 2^n linhas na tabela. Note que A e B não são variáveis da linguagem do CP, mas sim meta-fórmulas: representam fórmulas quaisquer.

Pode-se fazer uma forma abreviada da tabela verdade colocando-se o valor verdade de uma sub-fórmula sob o seu conectivo principal. Vejamos um exemplo:

A	B	$(A$	$\wedge$	$B)$	$\vee$	$(\neg$	A	$\wedge$	$\neg$	$B)$
V	V	V	V	V	V	F	V	F	F	V
V	F	V	F	F	F	F	V	F	V	F
F	V	F	F	V	F	V	F	F	F	V
F	F	F	F	F	V	V	F	V	V	F

O valor verdade da fórmula $(A \wedge B) \vee (\neg A \wedge \neg B)$ está sob o conectivo $\vee$.

3.2.2 Tautologias

Uma fórmula que é sempre verdadeira independente dos valores atribuídos às suas variáveis é chamada de **tautologia**. Por exemplo, $V_1 \longrightarrow V_1$ é sempre verdadeira. Utilizando meta-fórmulas, $A \longrightarrow A$ é sempre verdadeiro. A tabela verdade de uma tautologia possui apenas V na coluna que expressa o resultado:

A	$\neg A$	$A \vee \neg A$
V	F	V
V	F	V
F	V	V
F	V	V

Assim, $A \vee \neg A$ é uma tautologia.

Se uma fórmula é sempre falsa ela é chamada de **contradição**. Se uma fórmula pode ser V ou F, ela é uma **contingência**.

Definição 3.1. *Se $A \longrightarrow B$ é uma tautologia, dizemos que A implica logicamente B ou que B é uma conseqüência lógica de A.*

Definição 3.2. *Se $A \longleftrightarrow B$ é uma tautologia, dizemos que A é logicamente equivalente a B.*

Suponha que tenhamos uma fórmula A que utiliza as variáveis V_1, V_2, ... V_n. Não necessariamente uma fórmula utiliza as primeiras n variáveis, mas assumiremos sem perda de generalidade que sim.[1]

[1] Uma fórmula poderia ser $V_5 \wedge V_2 \longrightarrow V_{40}$, em que as três primeiras variáveis não seriam utilizadas.

Definição 3.3. *Dizemos que uma seqüência $s = (s_1, s_2, ...s_n)$, onde cada s_i é V ou F,* **satisfaz** *A se, quando V_i assume o valor s_i, o valor de A é V.*

Por exemplo, vejamos a fórmula $A =_{def} V_1 \vee V_2$ e a seqüência $s = (V, F)$. Assumindo que o valor de V_1 é V e o de V_2 é F, temos que $V_1 \vee V_2$ é V. Assim, $s = (V, F)$ satisfaz $V_1 \vee V_2$. Mas $s = (F, F)$ não satisfaz A, pois $F \vee F = F$. Utilizaremos letras gregas para representar fórmulas e $=_{def}$ para associar a letra à fórmula.

Uma seqüência nada mais é do que uma linha da tabela verdade. Então uma fórmula com n variáveis pode ser testada contra 2^n seqüências diferentes (uma para cada linha da tabela verdade). Claramente, uma fórmula é tautologia se qualquer seqüência s a satisfaz. Uma fórmula é contradição se nenhuma seqüência s a satisfaz.

A tabela verdade para uma fórmula A define uma função de verdade f_A que toma como parâmetros valores para as variáveis envolvidas.[2] Por exemplo, a fórmula $A =_{def} V_1 \vee V_2$ define uma função f_A tal que $f_A(V, V) = V$, $f_A(V, F) = V$, $f_A(F, V) = V$ e $f_A(F, F) = F$. Os parâmetros para esta função são precisamente as seqüências de que falamos acima. Assim, podemos escrever, em um abuso de linguagem, que se $s = (V, F)$, então $f_A(s) = V$.

Proposição 3.1. *Se A e $A \longrightarrow B$ são tautologias, então B é uma tautologia.*

Prova. Provaremos por contradição. Assuma que A e $A \longrightarrow B$ são tautologias mas B assuma o valor F para alguma atribuição de valores às variáveis de A e B. Isto é, existe uma seqüência s tal que a função de verdade de B, f_B, assume F com s: $f_B(s) = F$. Nesta atribuição de valores, A possui o valor V, $f_A(s) = V$, pois é uma tautologia. Então temos que $A \longrightarrow B$ é F pela tabela verdade de $\longrightarrow$. Contradição, pois assumimos que $A \longrightarrow B$ é uma tautologia. □

Proposição 3.2. *Se A é uma tautologia contendo as variáveis V_1, V_2, ... V_n e B é criada a partir de A pela substituição de V_i por B_i, então B é uma tautologia.*

Observação: note que B_i é uma fórmula qualquer. Por exemplo, $V_1 \longrightarrow V_1$ é uma tautologia e então o teorema nos diz que $(V_2 \wedge V_2) \longrightarrow (V_2 \wedge V_2)$ é uma tautologia, pois substituimos V_1 por uma fórmula qualquer (no caso, $V_2 \wedge V_2$). Da mesma forma, $(A \longleftrightarrow B \vee C) \longrightarrow (A \longleftrightarrow B \vee C)$ é uma tautologia. Note que uma meta-fórmula pode substituir uma variável de A. Contudo, ao provar teoremas, sempre assumimos que temos uma fórmula da linguagem do CP no lugar de B ou de qualquer meta-fórmula.

Prova. Suponha que A é tautologia. Então qualquer atribuição de valores às variáveis de A, esta fórmula terá o valor V. Considere uma atribuição de valores, dada por uma seqüência s, para as variáveis de B. Se preferir, imagine uma linha da tabela verdade de B, com valores V ou F atribuídos às variáveis de B (que não necessariamente são iguais às de A — B pode ter mais ou menos variáveis e possivelmente as variáveis são diferentes das de A). Queremos saber o valor verdade de B para esta atribuição. As fórmulas B_1, B_2, ... B_n terão os valores verdade w_1, w_2, ... w_n (cada um deles é V ou F). O valor de verdade de B é exatamente o valor de verdade de A com as variáveis de A assumindo os valores w_1, w_2, ... w_n. Mas este valor é V, pois A é uma

[2]Não se confunda: nas tabelas verdade apresentadas utilizamos A, B, etc que são meta-fórmulas. Estamos aqui falando de uma tabela que emprega fórmulas da linguagem e que, portanto, não podem ter meta-fórmulas. Obrigatoriamente a fórmula deve empregar variáveis V_1, V_2, ...

tautologia. Então o valor de verdade de B é V. Como consideramos uma atribuição de valores qualquer para B (genérica), B é sempre V, uma tautologia. Isto é, podemos pegar qualquer linha da tabela verdade de B que o raciocínio acima se aplica. □

Proposição 3.3. *Considere A uma fórmula dentro da qual há uma ou mais ocorrências de uma fórmula B. Seja A' a fórmula obtida a partir de A pela troca de uma ou mais ocorrências de B por B'. Então*

$$(B \longleftrightarrow B') \longrightarrow (A \longleftrightarrow A')$$

Observação: por exemplo, a fórmula $V_1 \longrightarrow (V_2 \wedge V_3)$ utiliza a subfórmula $(V_2 \wedge V_3)$, que seria B no teorema. B pode aparecer uma ou mais vezes em A, como em $(V_1 \vee (V_2 \wedge V_3)) \longrightarrow (V_2 \wedge V_3)$.

Prova. Deixada a cargo do leitor. □

Algumas equivalências lógicas do CP, dadas abaixo, são largamente utilizadas. Como um bom exercício, prove algumas delas utilizando tabelas verdade.

Lema 3.1. *São logicamente equivalentes:*

(a) $A \longrightarrow (B \longrightarrow C)$	*e*	$(A \wedge B) \longrightarrow C$	
(b) $A \wedge (B \vee C)$	*e*	$(A \wedge B) \vee (A \wedge C)$	*(Lei distributiva)*
(c) $A \vee (B \wedge C)$	*e*	$(A \vee B) \wedge (A \vee C)$	*(Lei distributiva)*
(d) $(A \wedge B) \vee \neg B$	*e*	$A \vee \neg B$	
(e) $(A \vee B) \wedge \neg B$	*e*	$A \wedge \neg B$	
(f) $A \longrightarrow B$	*e*	$\neg B \longrightarrow \neg A$	
(g) $A \longleftrightarrow B$	*e*	$B \longleftrightarrow A$	
(h) $(A \longleftrightarrow B) \longleftrightarrow C$	*e*	$A \longleftrightarrow (B \longleftrightarrow C)$	
(i) $A \longleftrightarrow B$	*e*	$(A \wedge B) \vee (\neg A \wedge \neg B)$	
(j) $\neg(A \vee B)$	*e*	$\neg A \wedge \neg B$	*(De Morgan)*
(k) $\neg(A \wedge B)$	*e*	$\neg A \vee \neg B$	*(De Morgan)*
(l) $A \vee (A \wedge B)$	*e*	A	
(m) $A \wedge (A \vee B)$	*e*	A	
(n) $A \wedge B$	*e*	$B \wedge A$	
(o) $A \vee B$	*e*	$B \vee A$	
(p) $(A \wedge B) \wedge C$	*e*	$A \wedge (B \wedge C)$	*(Associatividade do $\wedge$)*
(q) $(A \vee B) \vee C$	*e*	$A \vee (B \vee C)$	*(Associatividade do $\vee$)*
(r) $\neg(A \longleftrightarrow B)$	*e*	$A \longleftrightarrow \neg B$	

Como exemplo, provaremos (i) utilizando uma tabela verdade.

A	B	A	$\longleftrightarrow$	B	$(A$	$\wedge$	$B)$	$\vee$	$(\neg$	A	$\wedge$	$\neg$	$B)$
V	V	V	V	V	V	V	V	V	F	V	F	F	V
V	F	V	F	F	V	F	F	F	F	V	F	V	F
F	V	F	F	V	F	F	V	F	V	F	F	F	V
F	F	F	V	F	F	F	F	V	V	F	V	V	F

Lema 3.2. *Assumindo que $\mathcal{T}$ é uma tautologia e $\perp$ é uma contradição, os seguintes pares de fórmulas são equivalentes:*

(1) $\mathcal{T} \wedge A \quad e\ A$
(2) $\mathcal{T} \vee A \quad e\ \mathcal{T}$
(3) $\perp \wedge A \quad e\ \perp$
(4) $\perp \vee A \quad e\ A$

A veracidade destas afirmações podem ser facilmente conferidas utilizando a tabela verdade do $\wedge$ e $\vee$.

Utilizando as tabelas acima, podemos partir de uma fórmula complexa e chegar a outra mais simples e logicamente equivalente à primeira. Por exemplo, reduziremos algumas fórmula a formas mais simples.

(a) $\neg(A \wedge ((A \longrightarrow B) \longleftrightarrow (\neg B \longrightarrow \neg A))) \vee (A \wedge (B \vee A))$

Definindo $\mathcal{T}$ como $(A \longrightarrow B) \longleftrightarrow (\neg B \longrightarrow \neg A))$, a fórmula acima se transforma em
$\neg(A \wedge \mathcal{T}) \vee (A \wedge (B \vee A))$

por (n), $A \wedge \mathcal{T}$ é equivalente logicamente a $\mathcal{T} \wedge A$ e, por (1), $\mathcal{T} \wedge A$ é equivalente a A:
$\neg A \vee (A \wedge (B \vee A))$

por (o), $B \vee A$ é equivalente logicamente a $A \vee B$ e, por (m), $A \wedge (A \vee B)$ é equivalente a A. A fórmula final é
$\neg A \vee A$

(b) $(B \wedge C) \vee (A \wedge \neg B \wedge C) \vee (\neg A \wedge \neg B \wedge C)$
Substituindo $\neg B \wedge C$ por D, temos
$(B \wedge C) \vee (A \wedge D) \vee (\neg A \wedge D)$
utilizando (c) e (q), temos
$(B \wedge C) \vee ((A \wedge D) \vee \neg A) \wedge ((A \wedge D) \vee D)$
utilizando (c) e (o), temos
$(B \wedge C) \vee ((\neg A \vee A) \wedge (\neg A \vee D) \wedge ((D \vee A) \wedge (D \vee D)))$
utilizando (1) e o fato de que $D \vee D$ é logicamente equivalente a D, obtemos
$(B \wedge C) \vee ((\neg A \vee D) \wedge ((D \vee A) \wedge D))$
de (n) e (m) obtemos
$(B \wedge C) \vee ((\neg A \vee D) \wedge D)$
de (n) e (m) obtemos
$(B \wedge C) \vee D$
substituindo D por $\neg B \wedge C$,

$(B \wedge C) \vee (\neg B \wedge C)$

utilizando (c), obtemos

$((B \wedge C) \vee \neg B) \wedge ((B \wedge C) \vee C)$

utilizando (o) e (l),

$((B \wedge C) \vee \neg B) \wedge C$

utilizando (c),

$((\neg B \vee B) \wedge (\neg B \vee C)) \wedge C$

utilizando (1), já que $(\neg B \vee B)$ é uma tautologia,

$(\neg B \vee C) \wedge C$

utilizando (n) e (m),

C

Pode-se determinar se algumas frases são logicamente corretas convertendo-as para fórmulas do cálculo proposicional. Cada pedaço da frase que pode assumir os valores verdadeiro ou falso é convertido para uma letra da fórmula. Como exemplo, "Se eu estudar e o professor fizer uma prova fácil, eu vou tirar 10. Se o professor fizer uma prova fácil e eu não estudar eu também vou tirar 10" pode ser convertido para $((A \wedge B) \longrightarrow C) \wedge ((B \wedge \neg A) \longrightarrow C)$. Esta fórmula pode ser simplificada para $B \longrightarrow C$, que é "Se o professor fizer uma prova fácil, então eu vou tirar 10".

Exercícios Triviais

3.1. *(i5d1) Explique: a tabela verdade dada abaixo na verdade representa infinitas tabelas verdade.*

A	$\neg A$
V	F
F	V

3.2. *(i3d1) Escreva a tabela verdade do "ou" exclusivo.*

3.3. *(i2d1) Quais os nomes, em Português, dos operadores* $\neg$, $\longrightarrow$ *e* $\longleftrightarrow$ *?*

3.4. *(i5d1) Defina tautologia e contradição.*

Exercícios de Treinamento

3.5. *(i4d1) Podemos afirmar que, na Matemática,* $(7 < 1) \longrightarrow (0 = 0)$ *? E* $(7 < 1) \longrightarrow (0 = 1)$ *? Se sim, faça uma prova informal destes dois "teoremas".*

3.6. *(i5d3) Dada a fórmula* $A \longrightarrow B$ *podemos dizer que* A *implica logicamente* B *? Dada a fórmula* $A \longleftrightarrow B$ *podemos dizer que* A *é logicamente equivalente a* B *?*

3.7. *(i3d2) Prove: se A e $A \longrightarrow B$ são tautologias, então B é uma tautologia.*

3.8. *(i3d3) Considerando que $(A \longrightarrow B)$ é logicamente equivalente a $(\neg A \vee B)$, então a fórmula*

$$C \wedge ((A \longrightarrow B) \longleftrightarrow C)$$

é logicamente equivalente à fórmula

$$C \wedge ((\neg A \vee B) \longleftrightarrow C)$$

? Qual teorema garante isto ?

3.9. *(i4d3) Simplifique as seguintes fórmulas*

(a) $(((A \vee B) \wedge (\neg B)) \longrightarrow A)$

(b) $\neg\neg A \longleftrightarrow ((A \wedge B) \vee \neg A)$

(c) $\neg(A \wedge \neg B) \vee (A \longrightarrow B)$

(d) $\neg((A \longrightarrow \neg B) \wedge \neg(A \wedge C))$

(e) $A \vee B \longrightarrow B$

3.10. *(i4d2) Construa a tabela verdade para*

$(A \longrightarrow B) \wedge (A \longleftrightarrow B)$ *e*

$(A \longrightarrow (B \longrightarrow C)) \longrightarrow ((A \longrightarrow B) \longrightarrow (A \longrightarrow B))$.

3.11. *(i4d2) Usando tabelas verdade, prove que as fórmulas seguintes são tautologias.*

(a) $\neg\neg A \longleftrightarrow A$

(b) $(A \longrightarrow B) \longleftrightarrow (\neg A \vee B)$

(c) $\neg(A \wedge \neg A)$

(d) $(((A \vee B) \wedge (\neg B)) \longrightarrow A)$

(e) $A \wedge B \longrightarrow A$

(f) $A \longrightarrow (A \wedge A)$

3.12. *(i5d4) Represente $A \longrightarrow B$ e $A \longleftrightarrow B$ utilizando apenas os conectivos $\neg$, $\wedge$ e $\vee$.*

3.13. *(i5d4) Pode-se representar $A \wedge B$, $A \vee B$ e $\neg$ utilizando-se apenas $\longrightarrow$ e $\longleftrightarrow$?*

3.14. *(i4d3) Encontre uma fórmula correspondente à seguinte tabela verdade:*

A	B	C	$?$
V	V	V	V
V	V	F	V
V	F	V	V
V	F	F	V
F	V	V	F
F	V	F	V
F	F	V	V
F	F	F	V

3.3 Conjunto Adequado de Conectivos

A linguagem do cálculo proposicional deste manual utiliza apenas os conectivos $\neg$ e $\longrightarrow$, chamados de conectivos básicos. Todos os outros são derivados a partir deles ($\wedge$, $\vee$ e $\longleftrightarrow$). Uma pergunta então surge naturalmente: poderíamos utilizar outros conectivos como básicos ? Poderíamos utilizar $\wedge$ e $\vee$, por exemplo ? Ou $\wedge$ e $\neg$? Deixemos de lado esta questão e estudemos outra: as fórmulas que utilizam os conectivos $\neg$ e $\longrightarrow$ conseguem gerar todas as tabelas verdade possíveis ? Refinando a pergunta, as fórmulas de duas variáveis conseguem gerar todas as tabelas verdade que utilizam duas variáveis ? As fórmulas de três variáveis conseguem gerar todas as tabelas verdade de três variáveis ? Poderia acontecer que, por exemplo, uma fórmula de duas variáveis nunca poderia dar origem à seguinte tabela verdade:

A	B	$?$
V	V	F
V	F	F
F	V	V
F	F	F

Isto não seria bom, pois significaria que o CP não consegue dar conta de todos os raciocínios possíveis — ela não conseguiria expressar a realidade. Então perguntamos: $\neg$ e $\longrightarrow$ conseguem gerar todas as 2^{2^n} tabelas possíveis para fórmulas de n variáveis ?

A resposta a esta pergunta é sim. Mas o raciocínio para tanto não é óbvio. Veremos o porquê desta resposta em partes.

Definição 3.4. *Uma* **disjunção** *é uma seqüência de fórmulas separadas por $\vee$. Por exemplo, $A \vee B \vee C$.*

Definição 3.5. *Uma* **conjunção** *é uma seqüência de fórmulas separadas por $\wedge$. Por exemplo, $V_1 \wedge V_2$. Naturalmente, uma única fórmula é ambas uma disjunção e uma conjunção (a seqüência possui uma fórmula).*

Proposição 3.4. *Cada função de verdade é gerada por uma fórmula envolvendo apenas os conectivos $\neg$, $\wedge$ e $\vee$.*

Observações: a cada função de verdade corresponde a uma tabela verdade que contém uma variável para cada argumento da função e uma fórmula que corresponde ao resultado da função. Por exemplo, para a função verdade $f_\wedge$ onde $f_\wedge(V,V) = V$ e $f_\wedge(V,F) = f_\wedge(F,V) = f_\wedge(F,F) = F$, temos a tabela verdade

A	B	$A \wedge B$
V	V	V
V	F	F
F	V	F
F	F	F

onde, a cada linha, o primeiro e o segundo valores correspondem ao primeiro e segundo argumentos da função.

Antes de provar o teorema, mostraremos um exemplo. Considere a função f dada pela seguinte tabela verdade:

V_1	V_2	A
V	V	V
V	F	F
F	V	V
F	F	V

A fórmula correspondente ao resultado é A, que tentaremos encontrar. Observe que $V_1 \wedge V_2$ assume o valor V se os valores de V_1 e V_2 forem os da primeira linha da tabela. Da mesma forma, $\neg V_1 \wedge V_2$ assume V se os valores das variáveis forem os da terceira linha e $\neg V_1 \wedge \neg V_2$ assume V se os valores forem os da quarta linha. Considerando que A assume V se as variáveis assumirem os valores da primeira, terceira **ou** quarta linhas da tabela (observe o "**ou**"), podemos deduzir que A é

$$(V_1 \wedge V_2) \vee (\neg V_1 \wedge V_2) \vee (\neg V_1 \wedge \neg V_2)$$

Quando os valores assumidos forem os da primeira linha, a fórmula $(V_1 \wedge V_2)$ será verdadeira e, como temos um "ou" de expressões, o valor da fórmula toda será V. O mesmo se aplica à linhas três e quatro. Mas e se os valores forem os da segunda linha ? Então o valor de A é F. Vejamos porquê. Tome uma subfórmula qualquer de A, como $(\neg V_1 \wedge \neg V_2)$, correspondente à quarta linha. Esta subfórmula só é verdadeira se os valores de V_1 e V_2 forem exatamente F e F. Então tomando os valores de uma outra linha, como os da segunda linha, o valor desta subfórmula será F. Aplicando os valores da segunda linha, V e F, para as outras subfórmulas $(V_1 \wedge V_2)$ e $(\neg V_1 \wedge V_2)$, verificamos que o resultado é F e F. Isto sempre ocorrerá porque a segunda linha possui pelo menos um valor diferente de qualquer outra linha. Por exemplo, o segundo valor, F de V_2, correspondente à segunda linha, é diferente de V que aparece como valor de V_2 na primeira linha. Todos os valores da segunda linha diferem dos da terceira linha.

O que conseguimos ? A fórmula A é uma composta por diversas fórmulas separadas por "ou" e todas elas assumem o valor F se a valoração é aquela da segunda linha. O valor de A é F $\vee$ F $\vee$ F, que é F. Então obtemos o resultado esperado; isto é, A realmente produz os mesmos valores

que a tabela verdade.

Observe que foi fundamental para a construção de A o fato de que, se tivermos várias fórmulas separadas por "e", então o resultado é F se uma das fórmulas é F. E se tivermos várias fórmulas separadas por "ou", o resultado só será F se todas as fórmulas forem F.

A prova do teorema é baseada neste exemplo.

Prova. Assumiremos que as variáveis são V_1, V_2, ... V_n. Para cada linha da tabela verdade, cada uma destas variáveis assume o valor V ou F. Na linha i, seja C_i a fórmula $U_1^i \wedge U_2^i \wedge ... \wedge U_n^i$ onde U_j^i é V_j se a variável V_j assume o valor V e U_j^i é $\neg V_j$ se V_j assume F. Seja A a fórmula que é a disjunção de todos os C_i's onde o resultado da função é V (a última coluna é V). Ou seja, A é $C_{i_1} \vee C_{i_2} \vee ... \vee C_{i_k}$ onde C_{i_1}, C_{i_2}, ... C_{i_k} correspondem a todas as colunas onde o resultado é V. Claramente, há k linhas onde o resultado é V. Se não há nenhuma linha onde o resultado é V, então A é $V_1 \wedge \neg A_1$, uma contradição.

Afirmamos que A corresponde a esta tabela verdade. Suponha que as variáveis V_1, V_2, ... V_n assumam os valores da linha i da tabela. Se o resultado, que está na última coluna desta linha, for V, então C_i será V e todos os outros C_k serão falsos, resultando em A verdadeiro. Se as variáveis assumem os valores de uma linha cujo resultado é F, então todos os C_i serão falsos, pois, para cada C_i, pelo menos um dos valores U_j^i assumirá F, já que duas linhas da tabela diferem em pelo menos um valor verdade para as variáveis. Como A é um "ou" de C_i´s, A assume F. Conclui-se que a fórmula A tal como construída representa a tabela verdade. □

Proposição 3.5. *Cada tabela verdade corresponde a uma fórmula contendo apenas* $\neg$ *e* $\longrightarrow$.

Prova. Pelo teorema acima, pode-se representar uma tabela verdade por uma fórmula contendo apenas $\neg$, $\vee$ e $\wedge$. Mas fórmulas que utilizam $\vee$ e $\wedge$ podem substituídas por fórmulas logicamente equivalentes com $\neg$ e $\longrightarrow$. Confira que as fórmulas abaixo são tautologias:

$$(A \wedge B) \longleftrightarrow (\neg(A \longrightarrow \neg B))$$
$$(A \vee B) \longleftrightarrow (\neg A \longrightarrow B)$$

Concluimos que qualquer tabela verdade corresponde a uma fórmula contendo apenas $\neg$ e $\longrightarrow$. □

Proposição 3.6. *Cada tabela verdade corresponde a uma fórmula contendo apenas* $\neg$ *e* $\wedge$ *ou apenas* $\neg$ *e* $\vee$.

Prova. De acordo com a Proposição 3.4, cada tabela verdade pode ser gerada utilizando-se apenas $\neg$, $\wedge$ e $\vee$. Mas o "ou" lógico pode ser obtido a partir do "e" e vice-versa. As fórmulas abaixo são tautologias.

$$(A \wedge B) \longleftrightarrow \neg(\neg A \vee \neg B)$$
$$(A \vee B) \longleftrightarrow \neg(\neg A \wedge \neg B)$$

□

Mas será que conseguimos, para cada tabela verdade, uma fórmula que utilize apenas $\vee$ e $\wedge$? A resposta é não.

Proposição 3.7. *Uma fórmula com o conectivo $\neg$ não pode ser transformada em uma fórmula logicamente equivalente que utilize apenas $\wedge$ e $\vee$.*

Prova. Considere a mais simples de todas as tabelas verdade, a tabela do $\neg$:

V_1	$\neg V_1$
V	F
F	V

Suponha que seja possível representar esta tabela por uma fórmula B que utilizem apenas $\vee$ e $\wedge$. Como a única variável é V_1, B deve ser utilizar apenas conjunções e disjunções utilizando V_1, algo como

$$V_1 \wedge (((V_1 \vee V_1) \wedge (V_1 \wedge V_1)) \vee V_1)$$

Mas não é difícil de ver que esta fórmula é logicamente equivalente à fórmula "V_1", pois as fórmulas $V_1 \wedge V_1$ e $V_1 \vee V_1$ são logicamente equivalente a V_1. Portanto B não nega V_1, apenas preserva os valores verdade das variáveis. □

Quantos conectivos de uma variável podemos construir ? Dois, pois existem duas tabelas verdade com uma variável:

A	f_1
V	F
F	V

A	f_2
V	V
F	V

Há 2^4 conectivos diferentes que tomam duas variáveis como parâmetros, como $\wedge$, $\vee$, $\longrightarrow$ e $\longleftrightarrow$.

Proposição 3.8. *É possível construir qualquer tabela verdade utilizando apenas um conectivo binário.*

Prova. Considere o conectivo $\downarrow$, negação conjunta (*joint denial*), cuja tabela verdade é

A	B	$A \downarrow B$
V	V	F
V	F	F
F	V	F
F	F	V

Como pode ser facilmente conferido, $\neg A \longleftrightarrow (A \downarrow A)$ e $(A \wedge B) \longleftrightarrow ((A \downarrow A) \downarrow (B \downarrow B))$ são tautologias. Como qualquer tabela verdade pode ser construída utilizando-se apenas $\neg$ e $\wedge$, pode-se construir qualquer uma utilizando apenas $\downarrow$. □

De fato, existe um, e apenas mais um, conectivo binário que sozinho pode construir todas as tabelas verdade. É o conectivo "negação alternativa" (*alternative denial*) dado pela tabela verdade

A	B	$A\|B$
V	V	F
V	F	V
F	V	V
F	F	V

Definição 3.6. *Uma fórmula está na* **forma normal disjuntiva** *(FND) se ela é uma disjunção consistindo de uma ou mais subfórmulas, cada uma das quais é uma conjunção de uma ou mais variáveis ou negação de variáveis. Por exemplo, as fórmulas*

$(V_1 \wedge V_2 \wedge \neg V_3) \vee (\neg V_2 \wedge V_3) \vee V_1$
V_1
$\neg V_2$
$V_1 \wedge V_2$

estão na forma normal disjuntiva. Observe que os três últimos exemplos se enquadram na definição.

Definição 3.7. *Uma fórmula está na* **forma normal conjuntiva** *(FNC) se ela é uma conjunção consistindo de uma ou mais subfórmulas, cada uma das quais é uma disjunção de uma ou mais variáveis ou negação de variáveis. Por exemplo, as fórmulas*

$(V_1 \vee V_2 \vee \neg V_3) \wedge (\neg V_2 \vee V_3) \wedge V_1$
V_1
$\neg V_2$
$V_1 \vee V_2$

estão na forma normal conjuntiva. Observe que os três últimos exemplos se enquadram na definição e que estão ambas, na forma normal disjuntiva e conjuntiva. O mesmo se aplica a $V_1 \wedge V_2$.

Podemos utilizar meta-variáveis para descrever uma fórmula: $(A \wedge B) \vee (\neg B \wedge C)$ está na forma normal disjuntiva (FND). A, B e C são meta-variáveis: representam quaisquer variáveis da linguagem do cálculo proposicional, que são V_1, V_2, V_3, ... O que queremos dizer é qualquer fórmula obtida pela substituição de A, B e C por variáveis da linguagem é uma fórmula na FND.

Proposição 3.9. *Toda fórmula A é logicamente equivalente a uma fórmula na forma normal disjuntiva.*

Prova. Construa a tabela verdade de A. Por um dos teoremas acima, existe uma fórmula logicamente equivalente a A na FND. □

Proposição 3.10. *Toda fórmula A é logicamente equivalente a uma fórmula na forma normal conjuntiva.*

Prova. A prova será feita por construção da fórmula B equivalente logicamente a A e na FNC. $\neg\neg A$ é logicamente equivalente a A,

$$A \longleftrightarrow \neg\neg A$$

Pelo teorema anterior, existe uma fórmula B' na FND que é logicamente equivalente a $\neg A$:

$$A \longleftrightarrow \neg(\neg A) \longleftrightarrow \neg B'$$

Aplicando $\neg$ a B', obtemos uma fórmula B na FNC que é logicamente equivalente a A. Por quê ? É fácil provar que $\neg(A_1 \vee A_2 \vee ... \vee A_n)$ é logicamente equivalente a $\neg A_1 \wedge \neg A_2 \wedge ... \wedge \neg A_n$ e que $\neg(A_1 \wedge A_2 \wedge ... \wedge A_n)$ é logicamente equivalente a $\neg A_1 \vee \neg A_2 \vee ... \vee \neg A_n$. Isto se segue do fato que $\neg(A \wedge B) \longleftrightarrow (\neg A \vee \neg B)$ e $\neg(A \vee B) \longleftrightarrow (\neg A \wedge \neg B)$ são tautologias. Portanto, quando se aplica $\neg$ a uma fórmula na FND, obtém-se uma FNC e vice-versa. Note que uma prova rigorosa da afirmação acima pode ser feita utilizando-se indução finita.

Tomando-se A, obtemos $\neg A$ na FND (que é B') e, aplicando $\neg$ nesta fórmula, obtemos uma fórmula na FND que é equivalente a A:

$$A \longleftrightarrow \neg(\neg A) \longleftrightarrow \neg B' \longleftrightarrow B$$

□

Considere novamente a tabela verdade

V_1	V_2	A
V	V	V
V	F	F
F	V	V
F	F	V

Ao invés de construir a fórmula A na FND, podemos encontrar A na FNC. Basta tomar as linhas onde A assume o valor F e construir a FNC. Então A é $\neg V_1 \vee V_2$. De fato, podemos simplificar A encontrado anteriormente e chegar a este resultado:

$$(V_1 \wedge V_2) \vee (\neg V_1 \wedge V_2) \vee (\neg V_1 \wedge \neg V_2)$$
$$(V_1 \wedge V_2) \vee (\neg V_1 \wedge (V_2 \vee \neg V_2))$$
$$(V_1 \wedge V_2) \vee \neg V_1$$
$$(V_1 \vee \neg V_1) \wedge (V_2 \vee \neg V_1)$$
$$V_2 \vee \neg V_1$$
$$\neg V_1 \vee V_2$$

A fórmula resultante não possui nenhum $\wedge$ mas mesmo assim está na FNC.

Exercícios Triviais

3.15. *(i4d1) Quais das fórmulas abaixo estão na FNC ? E na FND ?*

(a) $(\neg A \wedge B) \vee C \wedge A$

(b) A_1

(c) $A_1 \wedge B$. *Esta é uma fórmula ?*

(d) $(\neg\neg A \wedge B) \vee (\neg B \wedge C)$

(e) $A_5 \vee (A_1 \wedge \neg A_3 \wedge \neg A_5) \vee \neg A_2$

(f) $A \vee \neg B$. *Esta é uma fórmula ?*

Exercícios de Treinamento

3.16. *(i4d2) Encontre uma fórmula na FND que seja logicamente equivalente à fórmula encontrada no exercício 3.14.*

3.17. *(i5d2) Encontre uma fórmula na FND correspondente à seguinte tabela verdade:*

A	B	C	$?$
V	V	V	V
V	V	F	F
V	F	V	F
V	F	F	F
F	V	V	V
F	V	F	V
F	F	V	V
F	F	F	F

3.18. *(i5d2) Encontre uma fórmula na FNC e outra na FND correspondente à seguinte tabela verdade:*

A	B	$?$
V	V	V
V	F	F
F	V	F
F	F	V

3.19. *(i2d3) Quantas tabelas verdade com n variáveis existem ? Justifique.*

3.20. *(i2d4) Prove que* $\neg(A_1 \vee A_2 \vee \ldots \vee A_n)$ *é logicamente equivalente a* $\neg A_1 \wedge \neg A_2 \wedge ... \wedge \neg A_n$ *utilizando indução finita em n.*

3.21. *(i2d4) Prove que* $A_1 \longrightarrow (A_2 \longrightarrow (A_3 \longrightarrow \ldots \longrightarrow A_n)) \ldots) \longrightarrow B)$ *é logicamente equivalente a* $A_1 \wedge A_2 \wedge \ldots \wedge A_n \longrightarrow B$.

3.4 Sintaxe do Cálculo Proposicional

Os esquemas de axiomas do cálculo proposicional são:

(A1) $(A \longrightarrow (B \longrightarrow A))$

(A2) $((A \longrightarrow (B \longrightarrow C)) \longrightarrow ((A \longrightarrow B) \longrightarrow (A \longrightarrow C)))$

(A3) $((\neg B \longrightarrow \neg A) \longrightarrow ((\neg B \longrightarrow A) \longrightarrow B))$

Note que estes são não axiomas e sim esquemas de axiomas. Cada um deles representa infinitos axiomas. Um axioma é uma fórmula e, por exemplo, $(A \longrightarrow (B \longrightarrow A))$ não é uma fórmula, pois não pode ser obtido pela regra dada acima para a obtenção de fórmulas. As letras A e B são meta-fórmulas: elas representam fórmulas e existem **fora** da linguagem do CP.

Como exemplo, a partir do esquema de axioma (a), podemos obter os seguintes axiomas:

1. $(V_1 \longrightarrow (V_2 \longrightarrow V_1))$, com V_1, que é uma fórmula, substituindo A e V_2 substituindo B;
2. $(V_1 \longrightarrow ((V_5 \longrightarrow \neg V_2) \longrightarrow V_1))$, com $A_5 \longrightarrow \neg V_2$ substituindo B ;
3. $(\neg V_1 \longrightarrow (V_2 \longrightarrow \neg V_1))$, com $\neg V_1$ substituindo A_1.

A única regra de inferência do cálculo proposicional é o Modus Ponens (MP): a partir de A e $A \longrightarrow B$, deduzimos B.

Definição 3.8. *Um sistema formal, como o CP, é chamado de* **teoria formal** *ou simplesmente* **teoria**.

Definição 3.9. *Uma teoria é axiomatizável se existe um algoritmo que diz se uma fórmula é um axioma ou não.*

O cálculo proposicional é claramente axiomatizável.

Definiremos agora outros conectivos lógicos a partir de $\neg$ e $\longrightarrow$:

D1 $(A \wedge B)$ é $\neg(A \longrightarrow \neg B)$;

D2 $(A \vee B)$ é $(\neg A) \longrightarrow B$;

D3 $(A \longleftrightarrow B)$ é $(A \longrightarrow B) \wedge (B \longrightarrow A)$.

Estes são chamados de conectivos derivados. Estas definições são tão somente abreviaturas: uma fórmula $V_1 \wedge V_2$, por exemplo, é apenas a abreviatura de $\neg(A_1 \longrightarrow \neg A_2)$. Os conectivos $\wedge$, $\vee$ e $\longleftrightarrow$ leêm-se "e", "ou" e "se e somente se". O conectivo $\longleftrightarrow$ é chamado de bicondicional.

Para facilitar a escrita e leitura de fórmulas, omitiremos os parênteses sempre que não houver ambigüidade. E convencionaremos que os conectivos lógicos seguem a seguinte ordem de precedência, do maior para o menor: $\neg$, $\wedge$, $\vee$, $\longrightarrow$ e $\longleftrightarrow$. Apesar de $\wedge$ ter maior precedência do que $\vee$, usualmente não escreveremos $A \wedge B \vee C$ e sim o mais legível $(A \wedge B) \vee C$. Os conectivos são associados à esquerda: se B estiver entre dois conectivos iguais, B é associado com o conectivo da esquerda. Assim, $A \longrightarrow B \longrightarrow C$ é o mesmo que $(A \longrightarrow B) \longrightarrow C$.

Com esta convenção, as fórmulas podem ser escritas de maneira mais legível:

fórmula original	fórmula resumida
$(\neg A) \longleftrightarrow (B \wedge C)$	$\neg A \longleftrightarrow B \wedge C$
$((A \longrightarrow B) \longrightarrow C) \longleftrightarrow ((B \wedge A) \wedge D)$	$A \longrightarrow B \longrightarrow C \longleftrightarrow B \wedge A \wedge D$

Definição 3.10. *Uma* **prova** *é uma seqüência A_1, A_2, ... A_n de fórmulas tal que, para cada i, A_i é um axioma ou é deduzido por MP a partir de A_j e A_k, onde $j, k < i$ e A_k é igual a $A_j \longrightarrow A_i$. Note que A_i deve ser um axioma, não um esquema de axioma. Uma prova envolve apenas elementos da linguagem, nenhum meta-elemento pode estar presente, como A e B, que são meta-fórmulas (representam fórmulas e estão fora da linguagem do CP). Contudo, como um abuso de linguagem, chamamos de "prova" seqüências que empregam méta-fórmulas.*

Definição 3.11. *A fórmula B é um* **teorema** *do CP se B aparece como último elemento de uma prova em CP: existe uma prova A_1, A_2, ... A_n e $A_n = B$. Como um abuso de linguagem, chamamos de "teorema" seqüências que empregam méta-fórmulas, como $A \longrightarrow A$. Rigorosamente falando, $A \longrightarrow A$ é um "esquema de teorema" a partir do qual podem ser produzidos teoremas substituindo-se A por qualquer fórmula. A propósito, provaremos adiante que $A \longrightarrow A$ é realmente um "teorema" do CP.*

Definição 3.12. *Se existe um algoritmo para decidir se B é um teorema de um sistema formal, então dizemos que este sistema é* **decidível**. *Caso contrário, o sistema é* **indecidível**. *O cálculo proposicional é decidível.*

Observe que, se as regras de inferência permitem que as fórmulas apenas cresçam de tamanho, então o sistema formal é claramente decidível. Se quisermos saber se uma certa seqüência T é teorema, basta, a partir dos axiomas, ir deduzindo todos os teoremas possíveis. A cada passo aplicam-se as regras a todos os teoremas já deduzidos, sempre que for possível. Quando o tamanho de todos os teoremas obtidos em um certo passo for maior do que o tamanho de T, podemos parar com as deduções. Se T não tiver sido deduzido até este ponto, nunca mais o será, pois deste passo em diante apenas se obterão teoremas maiores do que T.

Um sistema só é indecidível se houver uma regra que diminui o tamanho de algum teorema. No cálculo proposicional, utilizamos MP como única regra de inferência. E MP produz um teorema que é menor do que um dos teoremas de "entrada" (de A e $A \longrightarrow B$ obtemos B e B é menor do que $A \longrightarrow B$). Mesmo assim o CP é decidível.

Para compreender melhor este ponto, suponha que tenhamos uma seqüência de números inteiros começando por 5 e 7. A partir destes números iniciais, temos duas regras para produzir mais números:

1. soma-se dois números que já estejam na seqüência;

2. toma-se um número primo na seqüência e some 6 a ele.

Por estas regras, produziríamos 5, 7, 11, 12, 13, 17, 19, 21, ... Para saber se certo número k aparece na seqüência, basta fazer um algoritmo que produza a seqüência em ordem crescente. Se encontrarmos um número maior do que k antes de encontrar k certamente este número não está na seqüência, já que daí em diante apenas números maiores serão obtidos.

As coisas não ficam tão simples se colocarmos uma regra que diminua o número: se a soma dos dígitos de um número da seqüência for um número primo, adicione este número à seqüência.

Se surgir um número como 75146 na seqüência, teremos que acrescentar o número $7 + 5 + 1 + 4 + 6 = 23$. Mas pode ser que 23 nunca seja acrescentado. Ou que ele só apareça depois que bilhões de números tenham sido produzidos. No exemplo simples acima, é bem provável que exista um algoritmo que diga se 23 apareça na seqüência. Mas em outros casos o algoritmo não existe. Isto é equivalente a um sistema formal ser indecidível: não existe algoritmo que diga se uma fórmula é teorema; isto é, se ao produzirmos uma lista com os teoremas, algum dia a fórmula aparecerá nesta lista. Observe que não estamos afirmando que é difícil conseguir o algoritmo. Se o sistema é indecidível, o algoritmo simplesmente não existe.

Concluindo, se as regras apenas aumentam o tamanho dos teoremas, o sistema formal é decidível. Se pelo menos uma regra produz um teorema menor do que uma das entradas, então o sistema pode ser decidível ou indecidível.

Definição 3.13. *Uma fórmula A é uma* **conseqüência** *de um conjunto Γ de fórmulas se e somente se (sse) há uma seqüência A_1, A_2, ... A_n de fórmulas tal que $A = A_n$ e, para cada i, A_i é um axioma ou $A_i \in \Gamma$ ou é deduzido por MP a partir de A_j e A_k, onde $j, k < i$ e A_k é igual a $A_j \longrightarrow A_i$. Esta seqüência é uma prova de A a partir de Γ. As fórmulas de Γ são as hipóteses ou premissas de A. Escrevemos $\Gamma \vdash A$. Se Γ contiver um número pequeno de fórmulas, podemos escrevê-las sem o $\{$ e $\}$ que delimitam o conjunto. Por exemplo,*
$B, C \vdash A$
os invés de
$\{B, C\} \vdash A$
Utilizaremos $\Gamma, A \vdash B$ para indicar que B é uma conseqüência do conjunto $\Gamma \cup \{A\}$.

Podemos escrever $\vdash A$ para dizer que A é um teorema do cálculo proposicional. Se houver dúvidas quanto ao sistema formal que estamos utilizamos, podemos indicá-lo explicitamente:
$\vdash_{cp} A$.

Faremos agora uma prova utilizando o CP.

Lema 3.3. *Para qualquer fórmula A, $\vdash A \longrightarrow A$.*

Prova.

1. $(A \longrightarrow ((A \longrightarrow A) \longrightarrow A)) \longrightarrow ((A \longrightarrow (A \longrightarrow A)) \longrightarrow (A \longrightarrow A))$, fórmula obtida a partir do esquema de axioma A2;

2. $A \longrightarrow ((A \longrightarrow A) \longrightarrow A)$, instância de A1;

3. $(A \longrightarrow (A \longrightarrow A)) \longrightarrow (A \longrightarrow A)$, obtido de MP usando 1 e 2;

4. $A \longrightarrow (A \longrightarrow A)$, instância de A1;

5. $A \longrightarrow A$, obtido de MP usando 3 e 4.

□

Observe que $A \longrightarrow A$ é de fato um "esquema de teorema", já que A é uma meta-fórmula.

Veremos agora alguns meta-teoremas simples do CP. E de agora em diante chamaremos os meta-teoremas simplesmente de "teoremas".

Lema 3.4. *Se* $\Delta \subseteq \Gamma$ *e* $\Delta \vdash A$*, então* $\Gamma \vdash A$*.*

Prova. A adição de premissas não altera em nada a prova de A. □

Lema 3.5. $\Gamma \vdash A$ *sse*[3] *há um subconjunto finito* Δ *de* Γ *tal que* $\Delta \vdash A$*.*

Prova. $\Longrightarrow$ Segue do fato de que apenas um número finito Δ de fórmulas de Γ são utilizadas na prova de A.

$\Longleftarrow$ Se um número finito Δ de fórmulas já é suficiente para provar A, pode-se acrescentar outras fórmulas pelo Lema 3.4. □

Lema 3.6. *Se* $\Delta \vdash A$ *e, para cada* B *em* Δ *tivermos* $\Gamma \vdash B$*, então* $\Gamma \vdash A$*.*

Prova. Na prova de A usando as fórmulas de Δ, substitua cada fórmula de Δ pela sua prova utilizando Γ. Obtemos uma prova de A utilizando Γ e $\Gamma \vdash A$. □

Teorema 3.1. *(Teorema da Dedução) Considere as fórmulas* A *e* B *e um conjunto de fórmulas* Γ*. Se*

$$\Gamma, A \vdash B$$

então

$$\Gamma \vdash A \longrightarrow B$$

Tomando $\Gamma = \emptyset$*, temos que, se* $A \vdash B$*, então* $\vdash A \longrightarrow B$*.*

Prova. Observe que de $\Gamma, A \vdash B$ para $\Gamma \vdash A \longrightarrow B$ **retiramos** uma fórmula das premissas, o que torna as coisas mais difíceis, pois temos que provar $A \longrightarrow B$ a partir de menos fórmulas do que antes.

[3] se e somente se

Utilizaremos indução sobre o número de elementos da prova de B. Seja B_1, B_2, ... B_n, uma prova de B a partir de $\Gamma \cup \{A\}$. Naturalmente, $B = B_n$. Provaremos que $\Gamma \vdash A \longrightarrow B_i$ para $1 \leqslant i \leqslant n$. A hipótese de indução (HI) é:

HI: dada uma dedução $\Gamma, A \vdash B_k$, temos $\Gamma \vdash A \vdash B_k$ para $k < n$.

Provaremos primeiro o caso base, $\Gamma \vdash A \longrightarrow B_1$ dado que $\Gamma, A \vdash B_1$. Mas B_1 é:

1. um axioma;
2. uma fórmula de Γ;
3. A.

Nos casos 1 e 2, podemos construir uma prova para $A \longrightarrow B_1$ a partir de Γ:

1. $B_1 \longrightarrow (A \longrightarrow B_1)$, instância de A1;
2. B_1, pois B_1 ou é um axioma ou pertence a Γ;
3. $A \longrightarrow B_1$, MP utilizando 1 e 2.

No caso 3, temos $A \longrightarrow A$ (já provado) e $B_1 = A$, de onde obtemos $A \longrightarrow B_1$.

Em qualquer caso, não utilizamos A como hipótese, apenas Γ. Logo, $\Gamma \vdash A \longrightarrow B_1$.

Suponha agora que $\Gamma, A \vdash B_k$ implica que $\Gamma \vdash A \longrightarrow B_k$ para $k < n$ (hipótese de indução). Então B_n é:

- um axioma;
- uma fórmula de Γ;
- A;
- uma dedução por MP a partir de fórmulas anteriores.

Os casos 1, 2 e 3 são tratados como anteriormente. Resta o caso 4 e, neste caso, há fórmulas B_m e B_j, $1 \leqslant m, j < n$, tais que B_n é deduzido por MP a partir de B_m e B_j, onde B_j é igual a $B_m \longrightarrow B_n$. A dedução de B_n é algo como

1. B_1

2. B_2

...

m. B_m

...

j. $B_m \longrightarrow B_n$

...

n. B_n obtido por MP com m, j.

Pela HI, $\Gamma \vdash A \longrightarrow B_m$ e $\Gamma \vdash A \longrightarrow (B_m \longrightarrow B_n)$ e já podemos construir uma prova para $A \longrightarrow B_n$:

1. $(A \longrightarrow (B_m \longrightarrow B_n)) \longrightarrow ((A \longrightarrow B_m) \longrightarrow (A \longrightarrow B_n))$, instância do axioma 2;
2. $A \longrightarrow (B_m \longrightarrow B_n)$ pela HI;
3. $(A \longrightarrow B_m) \longrightarrow (A \longrightarrow B_n)$, MP com 1 e 2;
4. $A \longrightarrow B_m$ pela HI;
5. $A \longrightarrow B_n$, MP com 3 e 4.

Logo $\Gamma \vdash A \longrightarrow B_n$. □

Observe que, se $\Gamma \vdash A \longrightarrow B$, então claramente $\Gamma, A \vdash B$. Vejamos porquê: assumindo $A \longrightarrow B$ e tomando Γ e A como premissas, podemos utilizar MP para deduzir B. E então $\Gamma, A \vdash B$. Logo vale o se e somente se no teorema acima.

Mostraremos alguns teoremas importantes do Cálculo Proposicional, alguns sem a correspondente prova.

Lema 3.7. *$A \longrightarrow B$, $B \longrightarrow C \vdash A \longrightarrow C$*

Prova. Provaremos $A \longrightarrow B$, $B \longrightarrow C$, $A \vdash C$.

1. $A \longrightarrow B$ por hipótese;
2. $B \longrightarrow C$ por hipótese;
3. A por hipótese;
4. B, MP com 1 e 3;
5. C, MP com 2 e 4;

Pelo Teorema da Dedução, temos $A \longrightarrow B$, $B \longrightarrow C \vdash A \longrightarrow C$. □

Lema 3.8. *$A \longrightarrow (B \longrightarrow C)$, $B \vdash A \longrightarrow C$*

Prova. Provaremos $A \longrightarrow (B \longrightarrow C)$, B, $A \vdash C$

1. $A \longrightarrow (B \longrightarrow C)$ por hipótese;
2. A por hipótese;

3. $B \longrightarrow C$, MP 1 e 2

4. B por hipótese

5. C, MP 3 e 4

Pelo Teorema da Dedução, temos $A \longrightarrow (B \longrightarrow C),\ B \vdash A \longrightarrow C$. □

Lema 3.9. *Para quaisquer fórmulas A, B e C, as fórmulas seguintes são teoremas do CP:*

(a) $\neg\neg A \longrightarrow A$

(b) $A \longrightarrow \neg\neg A$

(c) $\neg A \longrightarrow (A \longrightarrow B)$

(d) $(\neg B \longrightarrow \neg A) \longrightarrow (A \longrightarrow B)$

(e) $(A \longrightarrow B) \longrightarrow (\neg B \longrightarrow \neg A)$

(f) $A \longrightarrow (\neg B \longrightarrow \neg(A \longrightarrow B))$

(g) $(A \longrightarrow B) \longrightarrow ((\neg A \longrightarrow B) \longrightarrow B)$

Exercícios Triviais

3.22. *(i5d1) Rigorosamente, é $A \longrightarrow B$ realmente uma fórmula do CP ? Explique o que é uma meta-fórmula.*

3.23. *(i5d1) Explique o que é um esquema de axioma. É $A \longrightarrow (B \longrightarrow A)$ realmente um axioma ?*

3.24. *(i5d1) Remova o maior número possível de parênteses das seguintes fórmulas*

$(((A \wedge B) \longrightarrow (\neg(B) \longrightarrow (B \longrightarrow A))) \wedge C)$
$(\neg A \longrightarrow (B \vee C)) \longleftrightarrow (((A \wedge B) \vee C) \longleftrightarrow A)$
$(A \wedge B) \wedge C$
$(A \vee B) \longleftrightarrow (A \longrightarrow \neg B)$

3.25. *(i5d1) Quais seqüências de símbolos abaixo são fórmulas do CP ?*

(a) $\neg\neg\neg\neg A \longrightarrow A \wedge \neg A$

(b) $\vee \wedge ABA$

(c) $((A_1 \longrightarrow A_2) \vee A_1$

3.26. *(i5d1) Os conectivos $\longleftrightarrow$, $\wedge$ e $\vee$ fazem parte da linguagem do CP ?*

3.27. *(i5d1) Explique o que é uma prova.*

3.28. *(i5d1) O que é um teorema ?*

3.29. *(i3d2) O que é uma teoria (sistema formal) decidível ?*

3.30. *(i5d2) Enuncie o Teorema da Dedução.*

Exercícios de Treinamento

3.31. *(i2d4) Se as regras de um sistema formal sempre produzem teoremas de tamanho crescente, o sistema é decidível ? Explique.*

3.32. *(i4d2) Explique o que querem dizer as notações seguintes, onde A e B são fórmulas quaisquer e Γ é um conjunto de fórmulas:*

(a) $A, B \vdash A$

(b) $\Gamma \vdash A$

(c) $A \vdash B$

(d) $A \nvdash \neg A$

(e) $\vdash_{cp} A \longrightarrow A$

(f) $\vdash_T A \wedge \neg A$, *onde T é uma teoria (sistema formal).*

3.33. *(i4d3) Considerando que Γ e Δ são conjuntos de fórmulas e A e B são fórmulas, prove:*

(a) *Se* $\Delta \subseteq \Gamma$ *e* $\Delta \vdash A$, *então* $\Gamma \vdash A$.

(b) $\Gamma \vdash A$ *sse*[4] *há um subconjunto finito* Δ *de* Γ *tal que* $\Delta \vdash A$.

(c) *Se* $\Delta \vdash A$ *e, para cada* B *em* Δ *tivermos* $\Gamma \vdash B$, *então* $\Gamma \vdash A$.

(d) $A \vdash A$

(e) *Se* $\vdash A$, *então* $\Gamma \vdash A$

(f) *Se* $A \in \Gamma$, *então* $\Gamma \vdash A$

3.34. *(i5d2) Prove utilizando os axiomas e a regra de dedução MP:*

(a) $A \longrightarrow A$

(b) $A \longrightarrow B,\ B \longrightarrow C \vdash A \longrightarrow C$

(c) $((A \longrightarrow (\neg A \longrightarrow A)) \longrightarrow ((A \longrightarrow \neg A) \longrightarrow (A \longrightarrow A)))$

[4] se e somente se

3.5 Relação entre Sintaxe e Semântica

Na Seção 3.4 vimos o que é a linguagem do cálculo proposicional, os seus axiomas e a regra de dedução deste sistema formal. A isto chamamos de sintaxe. A partir dos três axiomas e da regra Modus Ponens conseguimos produzir teoremas. Considerando que os axiomas e a regra Modus Ponens estão muito bem definidos, temos uma definição precisa do que é um teorema e o que não é. Por exemplo, $V_1 \longrightarrow V_1$ é um teorema, como já foi provado. Por um abuso de linguagem, dizemos que $A \longrightarrow A$ é um teorema, onde A é uma meta-fórmula que pode ser substituída por qualquer fórmula do CP. Na verdade, o que queremos dizer é que $A \longrightarrow A$ é um **esquema de teorema**: substituindo A por qualquer fórmula do CP, obtemos um teorema. Assim, são teoremas: $V_1 \longrightarrow V_1$, $(V_1 \wedge V_1) \longrightarrow (V_1 \wedge V_1)$ e $((V_1 \wedge V_2) \longrightarrow V_1) \longrightarrow ((V_1 \wedge V_2) \longrightarrow V_1)$. Um teorema é uma fórmula da linguagem do CP e, de acordo com a definição dada no início da Seção 3.4, pode conter apenas as variáveis V_1, V_2, ..., além de ser formada de acordo com regras bem definidas. Apesar dos conectivos lógicos básicos terem nomes bem significativos ("negação" e "implica"), para a sintaxe estes conectivos **não significam absolutamente nada**. Teoremas e mais teoremas são deduzidos sem nunca se utilizar o significado das palavras "negação" e "implica". Uma fórmula é teorema porque ela é o resultado de uma prova feita por regras bem definidas, não porque seja de alguma forma "verdadeira" ou "falsa".

Na Seção 3.2 vimos funções de verdade e suas correspondentes tabelas verdade. Para cada fórmula do CP pode-se produzir a sua tabela verdade. A semântica do cálculo proposicional associa a cada fórmula um valor que por ser V para verdadeiro ou F para falso conforme os valores que se associam às suas variáveis. Uma fórmula que assume sempre o valor V qualquer que seja a associação de valores para as suas variáveis é chamada de tautologia. Por exemplo, $V_1 \longrightarrow V_1$ é uma tautologia:

V_1	$V_1 \longrightarrow V_1$
V	V
F	V

Novamente, por um abuso de linguagem dizemos que $A \longrightarrow A$ é uma tautologia. Podemos até construir a tabela verdade deste **esquema de fórmula**. A associação de fórmulas com os valores verdade V ou F faz parte da **semântica** do CP. Os axiomas, regra de dedução e teoremas são parte da **sintaxe**. Não há teoremas na semântica e nem fórmulas que são tautologias na sintaxe.

Então $V_1 \longrightarrow V_1$ é um teorema do CP (sintaxe) e também uma tautologia (semântica). Surge então uma pergunta: qual a relação entre sintaxe e semântica ? Os teoremas do CP são obtidos a partir de axiomas e Modus Ponens e nada têm a ver, aparentemente, com tabelas verdade. Mas, estranhamente, todos os teoremas do CP apresentados na Seção 3.4 são tautologias. Por exemplo,

veja o teorema 3.9 (c):

A	B	$\neg$	A	$\longrightarrow$	$(A$	$\longrightarrow$	$B)$
V	V	F	V	V	V	V	V
V	F	F	V	V	V	F	F
F	V	V	F	V	F	V	V
F	F	V	F	V	F	V	F

Isto não é coincidência. Os axiomas e as regras de dedução de uma lógica não são feitas aleatoriamente: eles são projetados justamente para espelhar uma semântica previamente definida. Neste caso, os axiomas e a regra foram projetados justamente para fazerem com que todos os teoremas sejam tautologias. E, como veremos, qualquer tautologia é também um teorema.

No próximo capítulo veremos uma lógica mais poderosa chamada de lógica de primeira ordem. Com esta lógica, podemos expressar, por exemplo, os axiomas da Aritmética considerando apenas a soma e a subtração. Chamaremos este sistema de AS. Com estes axiomas e as regras da lógica, pode-se expressar todas as verdades que conhecemos de AS. Então o que se fez foi o seguinte, nesta ordem:

1. define-se o que é AS, qual é o vocabulário (números, os símbolos $+$, $-$, $=$, etc), a linguagem que ela utiliza ($1+4=5$ é válido, $+1+$ não é) e quais as fórmulas que devem ser consideradas válidas. Por exemplo, $1+1=2$ é válido mas $1+1=3$ não é válido. O que é e o que não é válido é expresso de maneira informal;

2. baseado no item 1, projetam-se axiomas para uma linguagem de primeira ordem de tal forma que os teoremas do sistema sejam as fórmulas válidas definidas acima. Não é necessário definir nenhuma regra nova (pode ser provado que isto não é necessário). Então utiliza-se os axiomas da lógica de primeira ordem (que incluem os axiomas do CP) mais os axiomas definidos neste item;

3. prova-se que o sistema construído do item 2 só produz verdades como definido no item 1. Se não, os axiomas devem ser refeitos para espelhar AS como o conhecemos. Isto seria equivalente a verificar se todos os teoremas são tautologias no cálculo proposicional. No CP, uma fórmula é verdade se ela é uma tautologia.

   Verifica-se também se há alguma verdade como definido no item 1 que não possa ser produzida como teorema. No cálculo proposicional, isto equivale a verificar se toda tautologia é também um teorema. É fácil verificar que uma fórmula complexa é uma tautologia, basta produzir a tabela verdade, mas provar que uma fórmula é teorema é em geral difícil.

E agora, temos um fato surpreendente: todas as verdades do sistema AS podem ser produzidos automaticamente! Utilizando as regras de inferência, podemos produzir teoremas. Mas provou-se que os teoremas são as verdades de AS. Então as verdades podem ser produzidas automaticamente. Veja o esquema na Figura 3.1.

Agora, para produzir as verdades do AS, só precisamos aplicar as regras de inferência: todos os teoremas possíveis serão produzidos.

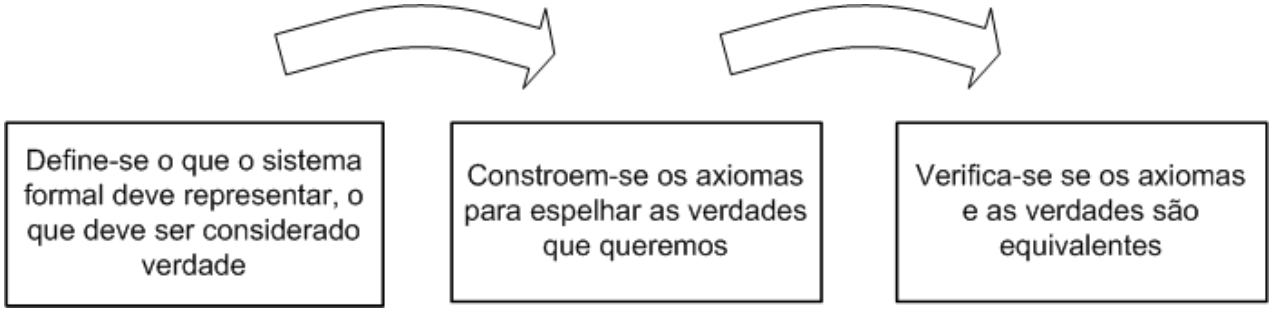

Figura 3.1: Como verdades são axiomatizadas

Este é o esquema geral seguido ao criar uma lógica: parte-se de um sistema real, como a aritmética, a geometria euclidiana, os sistemas vetoriais da álgebra, etc e constroem-se axiomas que espelham este sistema. Este axiomas são bons quando

(a) todos os teoremas que se podem obter são verdadeiros no sistema real;

(b) todas as verdades no sistema real são teoremas.

Chegamos então a duas importantes perguntas no CP:

(a) todos os teoremas do CP são tautologias ?

(b) todas as tautologias são teoremas ?

A resposta para ambas as questões é sim, no Cálculo Proposicional. Poderia ser diferente: a) poderiam existir tautologias que não são teoremas e b) algum teorema poderia não ser tautologia. Veja a Figura 3.5. Os teoremas são produzidos por axiomas e regras e é fácil de assegurar que todos os teoremas são tautologias (ou verdadeiros em um certo sentido). Contudo, verificar que todas as tautologias são teoremas é uma tarefa mais difícil.

Um quadro comparativo entre sintaxe e semântica é mostrado na Figura 3.3.

Antes de continuarmos, observe que em um programa de computador qualquer existe também a sintaxe e semântica. Para compreender isto, imagine que um programador queira fazer um programa qualquer como para:

1. somar uma seqüência de números;

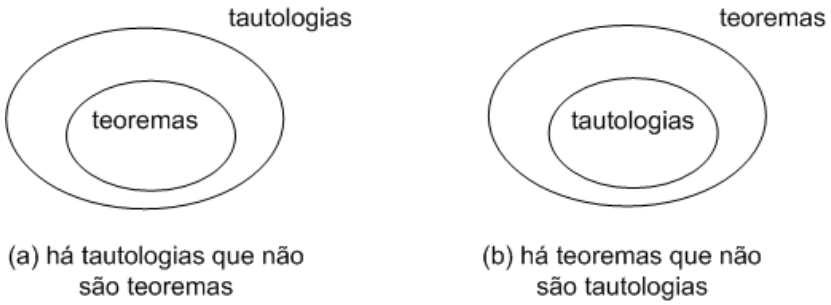

Figura 3.2: Relações possíveis entre o conjunto das tautologias e o conjunto dos teoremas

2. encontrar a menor distância entre duas cidades;

3. converter uma imagem do formato .gif para .jpg;

4. comprimir um arquivo para .zip.

Todo programa toma uma seqüência de bits como entrada e produz uma seqüência de bits como saída (mesmo se ele lê dados do mouse, acessa a Internet ou o HD, produz sons ou escreve coisas na tela — tudo isto pode ser considerado entrada ou saída). O que o programador faz ao codificar um programa ? Ele produz instruções que transformarão a entrada até que se obtenha a saída. Dada uma entrada, as instruções do programa são executadas[5] pelo computador que nada sabe sobre o significado dos bits que manipula. É como se o programa e sua entrada fosse um sistema formal onde um conjunto de regrinhas são aplicadas na entrada sistematica e precisamente até que se produza uma saída, outra seqüência de bits. O programador sabe o significado dos bits de entrada (a semântica dos bits) e faz com que o programa não danifique esta semântica durante a sua execução. Naturalmente, se o programa possui um erro esta semântica é obviamente violada. A saída possui outra semântica associada.

Ao produzir uma entrada para um programa, partimos do mundo real, que é onde as coisas possuem um significado (semântica) e produzimos uma entrada para o programa. Esta entrada é uma ligação entre o mundo e o formato que o programa espera para esta entrada. Ao executar o programa, ele produz uma saída através de sucessivas transformações desta entrada. Reforçamos que o programa nada conhece do significado dos bits que manipula, exatamente como um sistema formal (a sintaxe) nada conhece da semântica dos seus símbolos — os teoremas são produzidas por regras que independem do mundo real. Reafirmamos que **a associação entre sintaxe e semântica é feita por nós humanos, ela não existe independente de nós**.

Quando o programa pára e produz uma saída, seqüência de bits, novamente temos que interpretar esta saída e dar-lhe um significado. É exatamente isto que fazemos em lógica. A entrada corresponde aos axiomas. O programa corresponde às regras da lógica proposicional, de primeira ordem ou do sistema formal. A saída corresponde aos teoremas. Interpretando os teoremas, obtemos conhecimento do mundo real.

Então a lógica é só uma forma de **produzirmos** conhecimento do mundo real de forma automática. Cuidado deve ser tomado com a palavra **automática**: normalmente nós mesmos somos

[5]Na forma de código de máquina e não no formato original feito em uma linguagem de programação — mas se é uma coisa ou outra é irrelevante neste momento.

Sintaxe	**Semântica**
axiomas	mundo "real"
regras	verdadeiro
provas	falso
teoremas	tautologias
sistema formal	símbolos com significado
símbolos sem significado	equivalência lógica
	conseqüência lógica

Figura 3.3: Comparação entre sintaxe e semântica

os "computadores" e aplicamos as regras da lógica para produzir teoremas. Mas esta tarefa poderia ser feita por um computador. Dados os axiomas de um campo, como por exemplo da geometria euclidiana, um programa de computador poderia ir produzindo todos os teoremas possíveis desta geometria. Mas, infelizmente, este programa não poderia apontar quais são os teoremas realmente importantes e interessantes. Praticamente 100% dos teoremas seriam completamente inúteis. É preciso um humano para descobrir o que é realmente importante e interessante.

Teorema 3.2. *(Teorema da Correção) Cada teorema do CP é uma tautologia.*

Prova. Cada um dos axiomas do CP é uma tautologia. Verificaremos apenas A1:

A	B	$(A$	$\longrightarrow$	$(B$	$\longrightarrow$	$A))$
V	V	V	V	V	V	V
V	F	V	V	F	V	V
F	V	F	V	V	F	F
F	F	F	V	F	V	F F

Pelo Teorema 9, se A e $A \longrightarrow B$ são tautologias, então B é tautologia. Então MP, partindo de tautologias, só produz tautologias. Então podemos concluir que um teorema C do CP é uma tautologia.

Esta prova pode ser mais rigorosa. Suponha que a prova de C seja $B_1, B_2, ...B_n$, onde $B_n = C$. Utilizaremos indução em n com a seguinte Hipótese de Indução (HI):

HI: B_k é tautologia para $k < n$.

O caso base é quando $n = 1$ e temos que B_1 é uma instância de axioma e portanto tautologia. Considere válida a hipótese de indução. B_n pode ser uma instância de axioma (sendo portanto tautologia) ou deduzida por MP a partir de duas fórmulas B_i e B_j onde $i < n$, $j < n$ e $B_j =_{def} B_i \longrightarrow B_n$. Por HI, B_i e B_j são taulogias. Pelo Teorema 9, B_n é tautologia.

□

Teorema 3.3. *(Teorema da Completude) Se uma fórmula A do CP é uma tautologia, então A é um teorema.*[6]

Definição 3.14. *Uma teoria*[7] *é* **consistente** *se não existe uma fórmula A tal que A e* $\neg A$ *sejam ambos teoremas.*

Utilizando a notação $\vdash$, temos que, se uma teoria é consistente, então ocorre $\vdash A$ ou $\vdash \neg A$, mas não ambos. Uma outra maneira de dizer que uma teoria é consistente, no caso da lógica clássica (que engloba o CP), é dizer que existe pelo menos uma fórmula A que não é teorema. Neste caso, utilizamos a notação $\nvdash A$.

Teorema 3.4. *O Cálculo Proposicional é consistente.*

Prova. Suponha que A e $\neg A$ sejam ambos teoremas do CP. Então, pelo Teorema da Completude, ambas são tautologias. Absurdo. □

[6] Alguns autores [6] utilizam se e somente se para o teorema da completude

[7] Utilizamos a palavra **teoria** especificamente para designar um sistema formal. Logo, o CP é uma **teoria**.

A notação $\vdash A$ é utilizada para indicar que existe uma prova de A na teoria: A é teorema. A notação $\nvdash A$ é utilizada para indicar que não há prova de A na **teoria** — é impossível conseguir uma prova de A. Note que estamos discutindo o Cálculo Proposicional e todas as referências a uma teoria referem-se a ele. Assim, $\vdash A$ quer dizer, de fato, que existe uma prova de A no CP. Se for necessário ser mais específico, pode-se utilizar $\vdash_{CP} A$.

Se uma teoria é consistente, há fórmulas que não são teoremas: se $\vdash A$, então $\nvdash \neg A$ e, se $\vdash \neg A$, então $\nvdash A$. Um dos dois não é teorema e talvez nenhum dos dois o seja. Mas e se tivéssemos, no CP, uma fórmula A tal que $\vdash A$ e $\vdash \neg A$? Então todas as fórmulas do CP seriam teoremas ! Absolutamente qualquer fórmula poderia ser deduzida. É fácil de ver porquê. $\neg A \longrightarrow (A \longrightarrow B)$ é um teorema do CP (veja na página 40, item (c)). Por MP, obtemos qualquer fórmula B. Um pouco mais formalmente, temos:
queremos provar $A, \neg A \vdash B$. Então,

1. $\neg A$ (Hipótese)
2. $\neg A \longrightarrow (A \longrightarrow B)$ (teorema já provado)
3. $A \longrightarrow B$ (MP 1, 2)
4. A (Hipótese)
5. B (MP 3, 4)

Exercícios Triviais

3.35. *(i2d4) Explique o que é sintaxe e o que é semântica de uma teoria (em particular, do CP).*

3.36. *(i3d4) Podemos utilizar as palavras verdadeiro e falso quando falamos dos teoremas de um sistema formal; isto é, quando falamos exclusivamente da parte sintática de uma teoria ?*

3.37. *(i1d1) Verifique que os axiomas A2 e A3 são tautologias.*

Exercícios de Treinamento

3.38. *(i1d4) Há teorias em que existe uma verdade que nunca é alcançada pelo sistema formal. Isto é, os axiomas e as regras de dedução nunca conseguem produzir algumas fórmulas que sabemos que são verdadeiras. Deveriam ser teoremas, mas não são. O teorema da completude se aplicaria a um destes sistemas (adaptado a ele, logicamente) ?*

3.39. *(i2d4) Explique a relação entre sintaxe e semântica, em particular em como um sistema formal é construído e como se confere se o sistema é realmente o que queríamos.*

3.40. *(i3d1) Crie um sistema formal inconsistente.*

3.41. *(i3d2) Prove que o cálculo proposicional é consistente.*

3.42. *(i2d3) Explique como o teorema* $\neg A \longrightarrow (A \longrightarrow B)$ *do CP faz com que todas as fórmulas sejam teoremas se existir uma fórmula* A *tal que* $\vdash A$ *e* $\vdash \neg A$.

3.43. *(i2d4) Prove utilizando indução finita que* $(A_1 \wedge A_2 \wedge \ldots A_n) \longrightarrow B$ *é logicamente equivalente a* $(A_1 \longrightarrow A_2 \longrightarrow \ldots \longrightarrow A_n) \longrightarrow B$.

3.6 Tablôs

Tablôs são uma forma de descobrir se uma fórmula é uma tautologia ou não sem construir a tabela verdade e sem tentar demonstrar que a fórmula é teorema. É claro que demonstrar um teorema no CP é complicado pelos axiomas e MP. Mas fazer a tabela verdade não é fácil ? Não ! Uma tabela com n variáveis contém 2^n linhas. O que significa que uma fórmula com 50 variáveis tem 2^{50} linhas, aproximadamente 10^{15}. Mesmo um computador demoraria um tempo muito longo para construir uma tabela deste tamanho. E, se o objetivo é descobrir apenas se a fórmula é tautologia ou não, fazer a tabela verdade envolve uma grande quantidade de trabalho inútil (em geral), pois não estamos interessados nos casos onde a fórmula assume o valor V. Tudo o que interessa é se existe uma atribuição às variáveis onde o valor da fórmula é F. Isto acontecerá com a grande maioria das fórmulas, pois a imensa maioria delas não é tautologia. Isto pode ser deduzido examinando-se as tabelas verdade possíveis de duas variáveis. Há $2^{2^2} = 16$ tabelas verdade possíveis e apenas uma delas é tautologia. Se utilizamos três variáveis, há $2^{2^3} = 256$ tabelas verdade possíveis e apenas uma é tautologia. Pode-se concluir, de forma não rigorosa mas razoável, que apenas 1/16 das **fórmulas** de duas variáveis são tautologias e apenas 1/256 das **fórmulas** de três variáveis são tautologias. Das tabelas verdades extrapolamos para as fórmulas, daí o argumento não ser rigoroso, mas ser razoável.

Dada uma fórmula, digamos $A \wedge B \longrightarrow A$, podemos tentar falsificá-la, fazer o possível para que o resultado seja F. Se conseguirmos, então a fórmula não é tautologia. Vejamos: para que $C \longrightarrow D$ seja F, C deve ser V e D deve ser F — esta é a única possibilidade. Então assuma que $A \wedge B$ seja V e A seja F. Para $A \wedge B$ assumir V, A e B devem ser ambas V. Contradição, pois assumimos que A é F. Logo é impossível que a fórmula $A \wedge B \longrightarrow A$ seja falsa.

E quanto à fórmula $A \vee B \longrightarrow A$? Suponha que possa ser falsa (assuma o valor F para uma certa atribuição de valores às variáveis). Devemos ter $A \vee B$ V e A F (primeira hipótese). Para ter $A \vee B$ V, há duas opções: A é V **ou** B é V. Mas então se B é V e A é F (pela primeira hipótese), tem-se $A \vee B \longrightarrow A$ falso. Logo esta fórmula não é tautologia.

O método do tablô é simplesmente uma forma mais organizada de se fazer o que fizemos nos dois parágrafos anteriores. Explicaremos o método utilizando um exemplo. Suponha que se deseje verificar se a fórmula $A \wedge B \longrightarrow A$ é tautologia. Assuma que a fórmula é falsa:

$F \quad A \wedge B \longrightarrow A$

Coloque F antes da fórmula para indicar que esta fórmula deve ser falsa. Para uma implicação ser falsa, o antecedente deve ser verdadeiro e o conseqüênte, falso (se $C \longrightarrow D$ é falso, C deve ser verdadeiro e D, falso). Então, $A \wedge B$ deve ser verdadeiro e A deve ser falso:

$\checkmark F \;\; A \wedge B \longrightarrow A$
$V \;\; A \wedge B$
$F \;\; A$

Colocamos um ✓ na primeira fórmula para indicar que ela já foi analisada, o que chamaremos de *expandida*. Para que $A \wedge B$ seja verdadeiro, ambos, A e B devem ser verdadeiros.

$\checkmark F \;\; A \wedge B \longrightarrow A$
$\checkmark V \;\; A \wedge B$
$F \;\; A$
$V \;\; A$
$V \;\; B$

Neste ponto só restam fórmulas que não podem mais ser reduzidas. Podemos então analisar o que obtivemos. Assumindo inicialmente que a fórmula $A \wedge B \longrightarrow A$ é falsa, chegamos a uma contradição: A deve ser falso pela terceira linha e verdadeiro pela quarta. Portanto, a fórmula não pode nunca ser falsa.[8] É verdadeira em qualquer ocasião e portanto é uma tautologia.

Estudaremos um outro caso: $A \vee B \longrightarrow A$.

$F \;\; A \vee B \longrightarrow A$

Novamente, $A \vee B \longrightarrow A$ é falso só se $A \vee B$ é verdadeiro e A é falso:

$\checkmark F \;\; A \vee B \longrightarrow A$
$V \;\; A \vee B$
$F \;\; A$

Para $A \vee B$ ser verdadeiro, existem duas possibilidades: ou A é verdadeiro **ou** B é verdadeiro. É necessário fazer uma bifurcação, criando o que chamamos de árvore:

$\checkmark F \;\; A \vee B \longrightarrow A$
$\checkmark V \;\; A \vee B$
$F \;\; A$
$V \;\; A$ $\qquad\qquad$ $V \;\; B$

A construção do tablô terminou pois não mais fórmula a ser reduzida. Os dois ramos desta árvore "herdam" os dados do tronco principal. É como se tivéssemos duas árvores. Em um dos caminhos ou ramos (do topo até embaixo), o da esquerda, temos que A deve ser verdadeiro e falso. Dizemos, neste caso, que o caminho é **fechado** No outro, o da direita, não há contradição alguma. Isto significa que encontramos uma atribuição de valores às variáveis A e B de tal forma que a negação de $A \vee B \longrightarrow A$ não seja contraditória. Ou seja, não há problemas na fórmula ser falsa. E os valores de A e B do ramo direito, falso e verdadeiro, são um contra-exemplo de que a fórmula é uma tautologia. Se A é F e B é V, então $A \vee B$ é V. Mas, como A é F, $A \vee B \longrightarrow A$ é F e portanto não é uma tautologia. Observe que, *depois de terminada a construção do tablô* (não há mais nada a expandir):

- se a fórmula é uma tautologia, todos os ramos da árvore, começando do topo e terminando embaixo, produzem contradições (todos os caminhos ou ramos são fechados). Isto significa que tentamos provar que a fórmula pode ser falsa de todas as formas possíveis (o que envolve todas as atribuições de variáveis que poderiam fazê-la falsa[9]). Mas, em todas as possibil-

[8] Observe que utilizamos o método da redução ao absurdo.

[9] O que não envolve **todas** as atribuições de variáveis, mas apenas aquelas que, pelos conectivos utilizados na

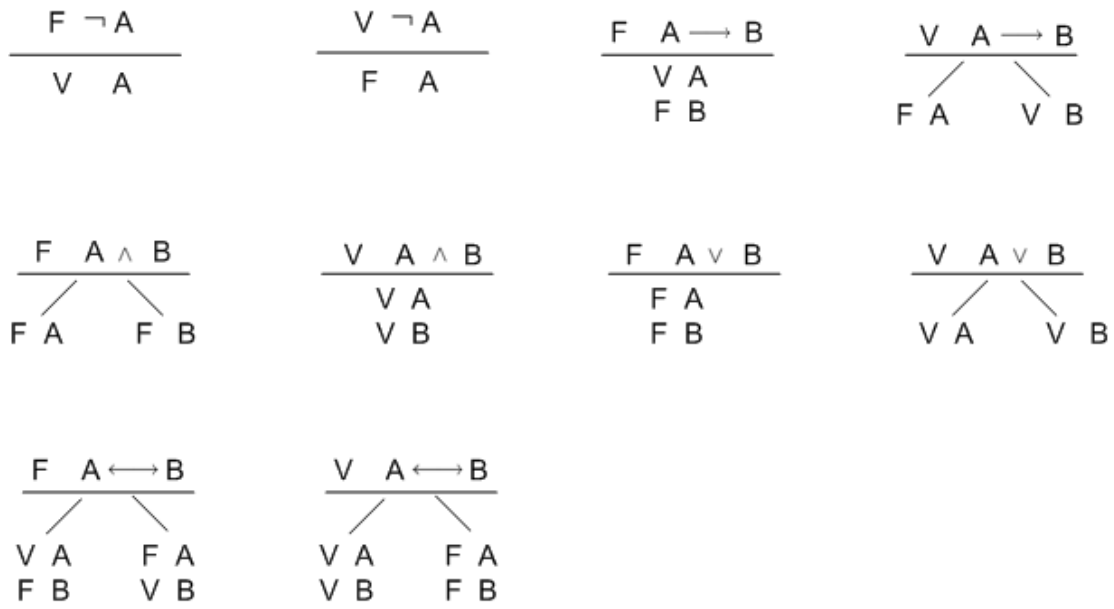

Figura 3.4: Regras para a contrução do tablô

idades, encontramos contradições. Então não é possível que a fórmula seja falsa — ela é verdadeira em todas as atribuições de variáveis que, supostamente, poderiam fazê-la falsa. Ela então é uma tautologia;

- se for encontrado um ramo que não é contraditório, então este ramo fornece exatamente os valores das variáveis que fazem com que a fórmula inicial assuma o valor F. Afinal, todo o tablô é feito exatamente para que encontremos as formas possíveis para que a fórmula inicial seja falsa. Então a fórmula não é tautologia e os valores encontrados para as variáveis são um contra-exemplo disto.

Para cada fórmula que "expandimos" durante a construção do tablô é necessário saber em que condições a fórmula é verdadeira e em que condições é falsa. A tabela 3.6 sumariza as condições necessárias para todos os conectivos.

Faremos um outro exemplo: $(A \wedge B) \longrightarrow (A \longrightarrow B)$:

$F \;\; (A \wedge B) \longrightarrow (A \longrightarrow B)$

Expandindo a fórmula inicial:

$\checkmark F \;\; (A \wedge B) \longrightarrow (A \longrightarrow B)$
$V \;\; A \wedge B$
$F \;\; A \longrightarrow B$

Expandindo $A \wedge B$:

$\checkmark F \;\; (A \wedge B) \longrightarrow (A \longrightarrow B)$
$\checkmark V \;\; A \wedge B$
$F \;\; A \longrightarrow B$
$V \;\; A$
$V \;\; B$

fórmula, poderiam de alguma forma fazê-la falsa.

Expandindo $A \longrightarrow B$:

$\checkmark F \;\; (A \wedge B) \longrightarrow (A \longrightarrow B)$
$\checkmark V \;\; A \wedge B$
$\checkmark F \;\; A \longrightarrow B$
$V \;\; A$
$V \;\; B$
$F \;\; A$

Não há mais nada a expandir e encontramos uma contradição. Mostraremos agora um exemplo com mais de um ramo, $(A \vee B) \longrightarrow (A \longrightarrow B)$:

$F \;\; (A \vee B) \longrightarrow (A \longrightarrow B)$

Expandindo a fórmula inicial:

$\checkmark F \;\; (A \vee B) \longrightarrow (A \longrightarrow B)$
$V \;\; A \vee B$
$F \;\; A \longrightarrow B$

Expandindo $A \vee B$:

$\checkmark F \;\; (A \vee B) \longrightarrow (A \longrightarrow B)$
$\checkmark V \;\; A \vee B$
$F \;\; A \longrightarrow B$

$V \;\; A$	$V \;\; B$

Expandindo $A \longrightarrow B$:

$\checkmark F \;\; (A \vee B) \longrightarrow (A \longrightarrow B)$
$\checkmark V \;\; A \vee B$
$\checkmark F \;\; A \longrightarrow B$

$V \;\; A$	$V \;\; B$
$V \;\; A$	$V \;\; A$
$F \;\; B$	$F \;\; B$

Encontramos uma contradição no ramo direito mas não no esquerdo. Então, tomando os valores da esquerda, temos que se A é V e B é F, a fórmula inicial é falsa. Confira.

Exercícios de Treinamento

3.44. *(i4d3) Faça o tablô das fórmulas seguintes e verifique quais delas são tautologias.*

(a) $\neg\neg A \longleftrightarrow A$

(b) $(A \longrightarrow B) \longleftrightarrow (\neg A \vee B)$

(c) $A \longrightarrow B \longrightarrow \neg C \longrightarrow ((A \wedge B) \vee C)$

(d) $((A \wedge B) \vee (A \longleftrightarrow B)) \longrightarrow A$

(e) $A \wedge B \wedge C \wedge D \longleftrightarrow ((A \longrightarrow B) \longrightarrow (C \longrightarrow D))$

(f) $A_2 \vee (A_1 \wedge \neg A_2) \longrightarrow \neg A_2$

3.45. *(i4d3) Faça o tablô das fórmulas do exercício 3.34.*

3.7 Complementos

3.7.1 Conexões com a Computação

Análise Sintática e Gramática da Linguagem do Cálculo Proposicional

A sintaxe de uma linguagem de programação é definida por uma gramática. Pode-se fazer o mesmo com as fórmulas do Cálculo Proposicional. Abaixo apresentamos uma gramática para fórmulas válidas do CP onde V_i é qualquer uma das variáveis V_1, V_2, ...

F ::= V_i
F ::= $\neg$ F
F ::= '(' F $\longrightarrow$ F ')'

Uma fórmula válida no CP pode ser derivada a partir do símbolo inicial F. Pode-se fazer um analisador sintático que reconheça as fórmulas válidas do CP. Um analisador sintático é a parte de um compilador que verifica se o programa a ser compilado pode ser obtido a partir da gramática da linguagem. Um programa em Java deve poder ser obtido a partir de uma gramática da linguagem Java, por exemplo.

O analisador sintático que reconhece as fórmulas válidas do CP é o método Java `analyzeFormula` apresentado abaixo. Assume-se que exista um método `nextToken` na mesma classe que pegue o próximo token e o coloque na variável de instância `token`. Há uma classe `Symbol` com uma constante para cada token. O método `error` emite uma mensagem de erro e termina o programa.

```
void analyzeFormula() {
   switch ( token ) {
      case Symbol.VARIABLE:
         nextToken();
         if ( token == Symbol.UNDERLINE )
            nextToken();
         else
            error("Underline expected");
         if ( token == Symbol.NUMBER )
            nextToken();
         else
            error("number expected");
         break;
      case Symbol.NEG :
         nextToken();
         analyzeFormula();
         break;
      case Symbol.LEFTPARENTHESIS:
         nextToken();
```

```
            analyzeFormula();
            if ( token == Symbol.LEFTARROW )
               nextToken();
            else
               error("left arrow expected");
            analyzeFormula();
            if ( token == Symbol.RIGHTARROW )
               nextToken();
            else
               error("left arrow expected");
      }
}
```

Enumeração das Fórmulas do Cálculo Proposicional

As fórmulas do CP podem ser enumeradas. Considere as regras da Seção 3.1 para produzir fórmulas válidas do CP. Há três regras para produzir fórmulas: a) V_i é fórmula; b) $\neg A$ é fórmula para A fórmula e c) $(A \longrightarrow B)$ é fórmula para A e B fórmulas. Pode-se construir um algoritmo que enumere todas as fórmulas possíveis. Este algoritmo produz a primeira fórmula pela regra a), a segunda pela fórmula b), a terceira pela fórmula c), a quarta pela fórmula a) e assim por diante. A cada aplicação da regra b), que exige uma fórmula já existente A, produz-se várias fórmulas correspondentes a todas as fórmulas A já enumeradas. O mesmo se aplica à regra c). Então a enumeração das fórmulas por este algoritmo seria

0	V_1
1	$\neg V_1$
2	$(V_1 \longrightarrow V_1)$
3	$(\neg V_1 \longrightarrow V_1)$
4	$(V_1 \longrightarrow \neg V_1)$
5	$(\neg V_1 \longrightarrow \neg V_1)$
6	V_2
7	$\neg V_2$
8	$(V_2 \longrightarrow V_1)$
...	...

Axiomatização do Cálculo Proposicional

O cálculo proposicional é axiomatizável; isto é, dada uma fórmula C, existe um algoritmo que decide se C é um axioma do CP. Isto pode ser feito de duas formas diferentes:

1. suponha que se deseja saber se C é uma instância do axioma A1; isto é, se C é da forma $(A \longrightarrow (B \longrightarrow A))$ onde A e B são fórmulas. Podemos enumerar todas as instâncias do axioma A1 tomando-se cada uma das fórmulas do CP[10] e substituindo-as por A e B em

[10] Veja a subseção anterior, Enumeração das Fórmulas do Cálculo Proposicional

$(A \longrightarrow (B \longrightarrow A))$. Este procedimento irá enumerar as instâncias do axioma A1. Neste enumeração, procura-se pela fórmula C. Quando as fórmulas desta enumeração só puderem ser maiores do que C ou só empregarem variáveis com números maiores do que aqueles utilizados em C, a busca por C pode ser encerrada. Se esta fórmula não tiver sido encontrada, nunca mais o será. Naturalmente o mesmo pode ser feito com os axiomas A2 e A3, o que resulta que existe um algoritmo que diz se certa fórmula é um axioma ou não.

2. fazendo-se uma análise sintática de C utilizando-se as gramáticas dos axiomas A1, A2 e A3. Mostraremos como isto pode ser feito utilizando-se apenas o axioma A1. A sua gramática é:

   A1 ::= '(' F $\longrightarrow$ '(' F $\longrightarrow$ F ')' ')'
   F ::= V_i
   F ::= $\neg$ F
   F ::= '(' F $\longrightarrow$ F ')'

   O analisador sintático de A1 deve conferir se a primeira e última fórmulas de A1 são iguais.

O analisador sintático de A1 é facilmente feito utilizando-se uma variação do método `analyzeFormula` apresentado na seção 3.7.1. Nesta variação, este método retorna um objeto contendo todas as informações da fórmula analisada, se a análise obteve sucesso. Ou `null` caso contrário. O método `compareFormula` retorna `false` se os seus dois parâmetros representam fórmulas diferentes.

```
boolean analyzeA1() {
   Formula formulaA, formulaB, formulaShouldBeA;
   if ( token == Symbol.LEFTPARENTHESIS )
      nextToken();
   else
      return false;
   formulaA = analyzeFormula();
   if ( formulaA == null ) return false;
   if ( token == Symbol.RIGHTARROW )
      nextToken();
   else
      return false;
   if ( token == Symbol.LEFTPARENTHESIS )
      nextToken();
   else
      return false;
   formulaB = analyzeFormula();
   if ( formulaB == null ) return false;
   if ( token == Symbol.RIGHTARROW )
      nextToken();
   else
      error("left arrow expected");
   formulaShouldBeA = analyzeFormula();
   if ( formulaShouldBeA == null ) return false;
   if ( ! compareFormula(formulaShouldBeA, formulaA) )
```

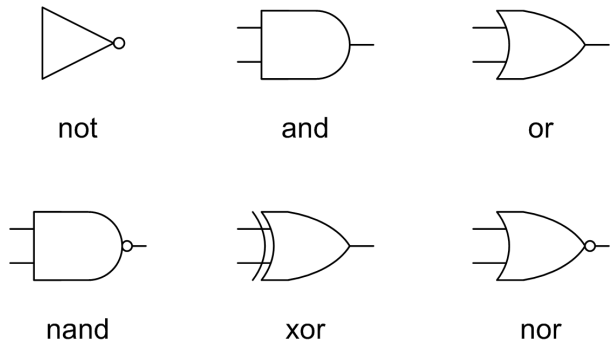

Figura 3.5: Portas lógicas

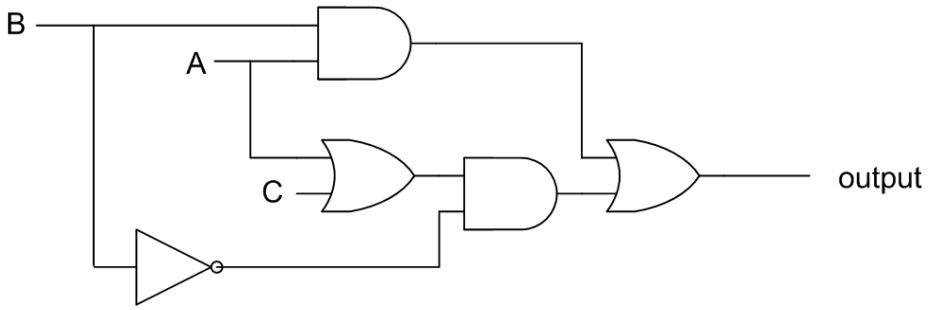

Figura 3.6: Circuito correspondente à fórmula $(A \wedge B) \vee ((C \vee A) \wedge \neg B)$

```
      return false;
   if ( token == Symbol.RIGHTPARENTHESIS )
      nextToken();
   else
      return false;
   if ( token == Symbol.RIGHTPARENTHESIS )
      nextToken();
   else
      return false;
   return true;
}
```

Circuitos Elétricos I

A Figura 3.7.1 mostra seis das principais portas lógicas utilizadas no projeto de computadores e circuitos digitais. Cada porta lógica toma uma ou mais entradas,[11] um sinal elétrico representando 0 ou 1, e produz uma saída. Cada porta se comporta como o conectivo equivalente do Cálculo Proposicional, considerando 0 como falso e 1 como verdadeiro[12] (`not` é $\neg$, `and` é $\wedge$ e `or` é $\vee$). A porta `nand` corresponde ao conectivo "negação alternativa" (*alternative denial*) visto na página 31. É a negação do `and` de dois valores. A porta `xor` é o `or` exclusivo, `A xor B` é 1 se e somente se apenas `A` ou apenas `B` é 1. A porta `nor` é a negação da porta `or`.

[11] A porta `not` é a única porta com uma única entrada.
[12] Ou o contrário.

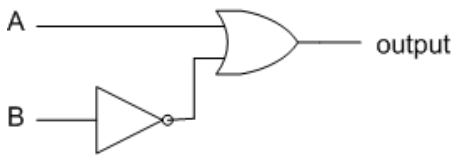

Figura 3.7: Circuito correspondente à fórmula $A \vee \neg B$

Estas portas lógicas podem ser utilizadas para montar circuitos que correspondem a fórmulas lógicas. Como exemplo, a fórmula $(A \wedge B) \vee ((C \vee A) \wedge \neg B)$ corresponde ao circuito mostrado na Figura 3.7.1. Os O fio `output` é o resultado da fórmula. O circuito é na verdade um avaliador do valor verdade da fórmula considerando-se que os valores das variáveis são dados pelos fios de entrada.

Pode-se montar um circuito para qualquer tabela verdade. Como exemplo, considere a tabela verdade

A	B	C	f
V	V	V	V
V	V	F	V
V	F	V	V
V	F	F	V
F	V	V	F
F	V	F	F
F	F	V	V
F	F	F	V

O circuito que produz um resultado equivalente a ela é dado na Figura 3.7.1 e corresponde à fórmula $(A \vee \neg B \vee \neg C) \wedge (A \vee \neg B \vee C)$ que simplificada, é $A \vee \neg B$.

As portas lógicas podem ser utilizadas para fazer um somador de números em binário. Como um exemplo, faremos um circuito que soma dois números de dois bits cada. Vejamos o que acontece em alguns exemplos de somas:

1 0	0 1	1 1
+ 0 1	+ 0 1	+ 1 1
1 1	1 0	1 1 0
(a)	(b)	(c)

No caso (a), o primeiro bit do resultado, o 1 da direita, é a somatória de 0 (do primeiro número, 1 0) com 1 (do segundo número, 0 1). O segundo bit do resultado é também 1, somatória de 1 do primeiro número com 0 do segundo. No caso (b), somando-se os bits da direita dos dois números, 1 e 1, obtêm-se 1 0 em binário. O 0 fica como resultado e 1 passa para a esquerda:

$$\begin{array}{rr} & 0^{+1}\ 1 \\ + & 0 \quad 1 \\ \hline & \quad 0 \end{array}$$

Agora temos que somar 1 com 0 do primeiro número com 0 do segundo número, que resulta

em 1, obtendo então o resultado 1 0.

A cada soma de dois bits, temos que considerar que pode vir um bit da somatória dos bits à direita, exceto na primeira soma. Este bit extra é chamado de "vai um". Então o primeiro bit do resultado, o mais à direita, depende de outros dois e cada um dos bits subsequentes depende de três bits.

Na soma (c), a soma dos dois primeiros bits à direita, $1 + 1$, resulta em 0 e um bit "vai um". Na soma dos dois bits seguintes, temos $1 + 1$ mais o bit "vai um", resultando em

$$\begin{array}{cccc} & & 1^{+1} & 1 \\ + & & 1 & 1 \\ \hline & 1 & 1 & 0 \end{array}$$

Cada bit do resultado pode ser especificado através de uma tabela verdade. Todas as possibilidades da soma de dois bits é dado pela tabela

A	B	$A + B$
0	0	0
0	1	1
1	0	1
1	1	0

Note que $1 + 1 = 0$ mas um bit "vai um"é passado para a próxima soma de bits à esquerda. Então é necessário uma outra tabela que indique se o bit "vai um"será passado ou não. De fato, consideramos que um bit é sempre passado à esquerda, mas este bit pode ser 1 ou 0. O único caso em que há um bit "vai um"é quando ambos os bits forem 1:

A	B	bit "vai um"
0	0	0
0	1	0
1	0	0
1	1	1

Cada uma destas tabelas verdade pode ser calculada por um circuito. De fato, o primeiro é o `xor`, "ou" exclusivo. O segundo é o `and`, "e" lógico.

A partir do segundo bit da soma de dois números em binário, é necessário considerar o bit "vai

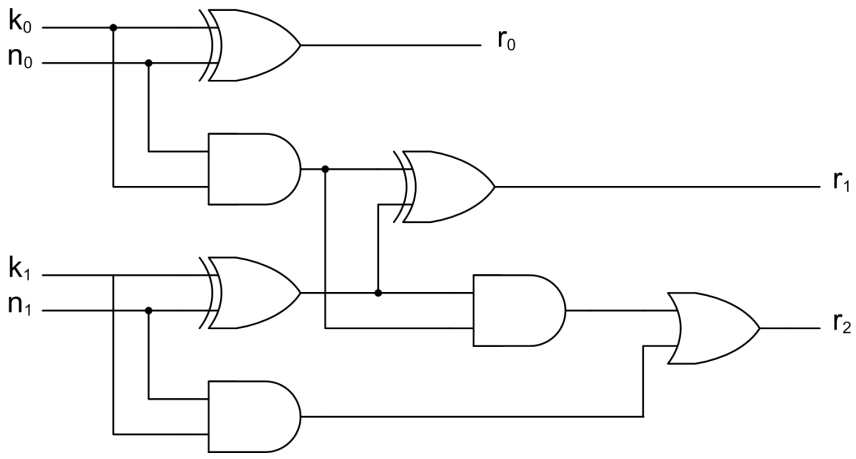

Figura 3.8: Circuito que soma os números k_0k_1 e n_0n_1 resultando em $r_0r_1r_2$

um". Portanto as tabelas verdade deve ter três entradas.

A	B	C	$A+B+C$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

A entrada C representa o bit "vai um"produzido pela soma de bits à direita.

A	B	C	bit "vai um"
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

A Figura 3.7.1 apresenta um circuito que avalia a soma dos bits de dois números binários. O primeiro número é k_0k_1 (k_0 e k_1 são os bits do número) e o segundo é n_0n_1. O resultado é o número $r_0r_1r_2$.

Como foi visto na Seção 3.6, cada tabela verdade pode ser gerada utilizando-se apenas $\neg$ e $\wedge$ ou apenas $\neg$ e $\vee$. Então qualquer circuito de um computador pode ser feito utilizando-se apenas as

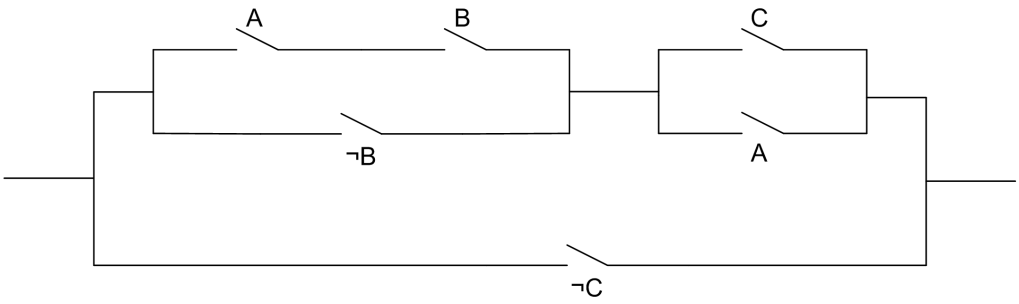

Figura 3.9: Um circuito elétrico

portas `not` e `and` ou apenas `not` e `or`. Na prática, se utiliza outras portas para diminuir o tamanho dos circuitos. Na verdade, pode-se fazer um computador utilizando-se apenas um único tipo de porta lógica, o `nand`. De acordo com a Proposição 3.8, apenas um único conectivo é necessário para construir qualquer tabela verdade. Este conectivo, $\downarrow$, corresponde à porta lógica `nand`.

Considere um programa de computador qualquer que tome k bits de entrada e produz m bits de saída. Para cada possibilidade dos bits de entrada, há uma saída possível. Esta saída pode ser expressa por um circuito composto pelas portas `and`, `or` e `not`. Para compreender melhor, considere apenas o primeiro bit da saída. O programa, conforme os valores dos k bits de entrada, deve considerar este bit 0 ou 1. Esta informação pode ser colocada em uma tabela verdade que possui uma variável para cada um dos k bits de entrada e onde o resultado é o primeiro bit de saída. De fato, devemos ter uma tabela verdade para cada um dos bits de saída, m tabelas verdade. Cada uma dessas tabelas pode ser implementada por um circuito elétrico. Então qualquer programa[13] pode ser implementado por um circuito elétrico contendo apenas as portas básicas.

Circuitos Elétricos II

Um outro tipo de circuito elétrico contém chaves que deixam ou não a passagem de corrente. A passagem de corrente representa verdadeiro e a ausência, falso. Um destes circuitos é mostrado na Figura 3.7.1. Ao lado de cada letra há um dispositivo que deixa/não deixa a correte passar. A letra representa uma condição para que a corrente passe. Dispositivos em paralelo implementão um "ou" lógico e em série um "e" lógico.

Podemos converter um circuito deste tipo em uma fórmula do cálculo proposicional e vice-versa. Como exemplo, a fórmula correspondente ao circuito da Figura 3.7.1 é $(((A \wedge B) \vee \neg B) \wedge (C \vee A)) \vee \neg C$ A simplificação desta fórmula nos conduz a $A \vee \neg B \vee \neg C$, que corresponde ao circuito da Figura 3.7.1.

Lógica Booleana

Uma lógica booleana [8] é um conjunto contendo dois valores, usualmente 0 e 1, e três operações que obedecem aos axiomas abaixo. As operações são "e" e "ou" binários e "não" unário representados

[13]Pode ser *qualquer* programa, mesmo que ele não tome um número fixo de bits de entrada, pois sempre podemos supor que o número máximo de bits da entrada está limitado pelo tamanho da memória do computador. Aqui assumimos que o programa é implementado em um computador específico onde sabemos o tamanho máximo da memória.

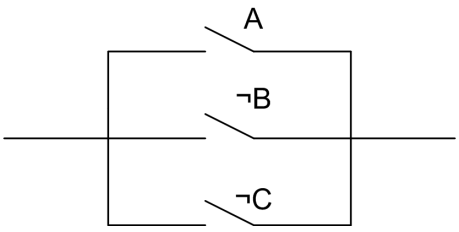

Figura 3.10: Um circuito elétrico simplificado

por $\cdot$, $+$ e $-$, respectivamente (poderia ser $\wedge$, $\vee$ e $\neg$ ou quaisquer outros símbolos). Considerando a, b e c variáveis que podem conter valores 0 e 1, os axiomas da lógica booleana são

B1 $a + (b + c) = (a + b) + c$, associatividade do $+$

B2 $a \cdot (b \cdot c) = (a \cdot b) \cdot c$, associatividade do $\cdot$

B3 $a + b = b + a$, comutatividade do $+$

B4 $a \cdot b = b \cdot a$, comutatividade do $\cdot$

B5 $a + (a \cdot b) = a$

B6 $a \cdot (a + b) = a$

B7 $a + (b \cdot c) = (a + b) \cdot (a + c)$, distributividade do $+$

B8 $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$, distributividade do $\cdot$

B9 $a + -a = 1$

B10 $a \cdot -a = 0$

Utilizando estes axiomas pode-se provar as seguintes regras:

- $a + a = a$
- $a \cdot a = a$
- $a + 0 = a$
- $a \cdot 1 = a$
- $a + 1 = 1$
- $a \cdot 0 = 0$
- $-0 = 1$ e $-1 = 0$
- $-(a + b) = -a \cdot -b$
- $-(a \cdot b) = -a + -b$

- $--a = a$

Estes axiomas podem ser utilizados para simplificar fórmulas como $(A \wedge B) \vee ((C \vee A) \wedge \neg B)$. Convertendo os operadores para os utilizados na lógica booleana, temos

$(A \,.\, B) + ((C + A) \,.\, -B)$

Agora podemos aplicar os axiomas e as regras acima para simplificar a fórmula:

$(A \,.\, B) + ((C + A) \,.\, -B) =$
$(A \,.\, B) + (-B \,.\, (C + A)) =$
$(A \,.\, B) + ((-B \,.\, C) + (-B \,.\, A)) =$
$(A \,.\, B) + ((-B \,.\, A) + (-B \,.\, C)) =$
$((A \,.\, B) + (-B \,.\, A)) + (-B \,.\, C) =$
$((A \,.\, B) + (A \,.\, -B)) + (-B \,.\, C) =$
$(A \,.\, (B + -B)) + (-B \,.\, C) =$
$(A \,.\, 1) + (-B \,.\, C) =$
$A + (-B \,.\, C)$

A fórmula simplificada é $A \vee (\neg B \wedge C)$ na notação usual. Com um pouco de prática, pode-se saltar alguns passos na simplificação:

$(A \,.\, B) + ((C + A) \,.\, -B) =$
$A \,.\, B + C \,.\, -B + A \,.\, -B =$
$A \,.\, (B + -B) + C \,.\, -B =$
$A + C \,.\, -B$

Exercícios de Treinamento

3.46. *Faça um circuito elétrico que multiplique dois bits. A saída deve ter quatro bits.*

3.47. *Uma sala possui uma lâmpada e dois interruptores. Faça um circuito com duas chaves (interruptores) de tal forma que qualquer dos interruptores possa ligar ou desligar a lâmpada. Isto é, se a lâmpada está desligada (ligada), apertando qualquer dos interruptores a liga (desliga).*

Capítulo 4

Lógica de Primeira Ordem

Há inúmeras proposições sobre a Matemática e sobre o mundo que não podem ser expressas no Cálculo Proposicional. Por exemplo, não podemos expressar as seguintes proposições:

1. todos os homens são primatas;
2. todos os primatas são mamíferos;
3. existe um homem que não é inteligente;
4. ao se somar 1 a qualquer número ímpar obtem-se um número par;
5. existe um número primo maior do que 10^{1000};
6. para qualquer número, existe sempre um número maior do que ele.

Estas proposições consideram que: a) todos os membros de um certo conjunto possuem certa propriedade ou b) um membro de um conjunto possui certa propriedade. Para expressar estas proposições podemos utilizar a Lógica de Primeira Ordem (LPO). Esta lógica utiliza os conectivos $\forall$ (lê-se "para todo") e $\exists$ (lê-se "existe"). Certamente esta lógica não é suficiente para expressar as todas verdades do mundo, mas é suficiente para praticamente toda a Matemática.

Como exemplo, expressaremos as proposições acima na linguagem da lógica de primeira ordem:

1. $\forall x\ (H(x) \longrightarrow P(x))$, onde $P(x)$ indica que "x é um primata" e $H(x)$ indica que "x é um homem". Considere que x assume os elementos do conjunto de todos os animais (poderia ser de todas as coisas);
2. $\forall x\ (P(x) \longrightarrow M(x))$, onde $M(x)$ é "x é mamífero". Considere que x assume os elementos do conjunto de todos os animais (poderia ser de todas as coisas);;
3. $\exists x\ \neg I(x)$, onde $I(x)$ é "x é inteligente". Considere que x assume os elementos do conjunto que contém todos os homens. Se considerarmos que x assume os elementos do conjunto de todos os animais ou todas as coisas, esta fórmula deveria ser $\exists x\ H(x) \wedge \neg I(x)$;
4. $\forall x\ (I(x) \longrightarrow P(x+1))$, onde $I(x)$ é "x é ímpar" e $P(x)$ é "x é par". x assume os números Naturais;

5. $\exists x\,(P(x) \wedge (x > 10^{1000}))$, onde $P(x)$ é "x é primo". x assume os números Naturais;

6. $\forall x\,(\exists y\,(x < y))$. Note que a fórmula $\exists y\,(\forall x\,(x < y))$ não faz muito sentido se considerarmos que x e y assumem os elementos do conjunto dos Naturais ou Reais. Esta fórmula diz que existe um y maior do que qualquer x do conjunto;

Os símbolos H, P, M e I são chamados de **predicados**. Eles informam propriedades dos seus argumentos.

Note que podemos fazer deduções com a LPO, como nos exemplos abaixo:

Considerando $\forall x\,(H(x) \longrightarrow P(x))$ e $H(z)$ (z é um homem), concluímos que $P(z)$ (z é um primata).

Considerando $\exists x\,\neg I(x)$, podemos concluir que $\neg\forall x\,I(x)$ (não é verdade que todos os homens são inteligentes).

Um ponto importantíssimo a considerar é que as conclusões obtidas acima **independem** do significado dos predicados H, P M e I. Quaisquer que fossem os significados dos predicados, o resultado seria o mesmo. É exatamente este tipo de dedução que nos interessa, as que independem do mundo real e que podem ser feitas formalmente, observando-se apenas a **forma** das sentenças lógicas.

4.1 A Linguagem da Lógica de Primeira Ordem

A linguagem da Lógica de Primeira Ordem utiliza os seguintes símbolos:

1. variáveis x_1, x_2, x_3, ...;

2. constantes c_1, c_2, c_3, ...;

3. para cada n natural, símbolos de predicados P_1^n, P_2^n, P_3^n, ...;

4. para cada n natural, símbolos de função f_1^n, f_2^n, f_3^n, ...;

5. , (vírgula), (,), $\neg$ e $\longrightarrow$ (tomados do Cálculo Proposicional);

6. $=$ (igual)

7. $\forall$ (quantificador universal)

Utilizamos P_k^n para designar o k-ésimo predicado de aridade n; isto é, este símbolo corresponderá, na semântica, a um predicado de n argumentos. O mesmo se aplica aos símbolos de função. Note que dizemos "símbolos de função" e "símbolos de predicado" e não "função" e "predicado", pois este só existem na semântica (mundo real) e neste instante estamos definindo somente a linguagem.

Definição 4.1. *Um* **termo** *é definido como*

1. *uma variável ou constante é termo;*

2. *se f_k^n é um símbolo de função e t_1, t_2, ... t_k são termos, então $f_k^n(t_1, t_2, ..., t_n)$ é um termo. Observe que um símbolo de função de aridade n deve ser utilizado com n termos;*

3. *nada mais é um termo.*

Definição 4.2. *Uma* **fórmula atômica** *é $t_1 = t_2$ ou possui a forma $P_k^n(t_1, t_2, ..., t_n)$ onde P_k^n é um símbolo de predicado e t_1, t_2, ..., t_n são termos.*

Definição 4.3. *Uma fórmula da linguagem de primeira ordem é definida como*

1. *todo fórmula atômica é fórmula;*

2. *se A e B são fórmulas e x é uma variável qualquer, então $(\neg A)$, $(A \longrightarrow B)$ e $((\forall x)A)$ são fórmulas;*

3. *nada mais é uma fórmula.*

Os conectivos $\wedge$, $\vee$ e $\longleftrightarrow$ são definidos como no CP. O quantificador existencial $(\exists)$ não faz parte da linguagem, sendo definido como

$(\exists x)A$ é uma abreviatura para $\neg((\forall x)(\neg A))$

Ou seja, se existe um x com certas propriedades (A), então não é verdade que para todo x aquela propriedade não vale. O símbolo $\exists$, na semântica de qualquer lógica de primeira ordem, significa "um ou mais". Então, a fórmula $(\exists x)A$, quando interpretada, não diz que A é verdade para apenas um único valor de x, mas para um ou mais valores de x. Da mesma forma, $(\forall x)A$ é uma abreviatura para $\neg((\exists x)(\neg A))$

Note que as variáveis do cálculo proposicional **não** são as mesmas da Lógica de Primeira Ordem. No CP, uma única variável é considerada uma fórmula correta: V_1 é uma fórmula. Na LPO, uma fórmula possui **sempre** um predicado ou o sinal de igualdade. Chamaremos as variáveis do cálculo proposicional de *variáveis proposicionais*.

Retiraremos os parênteses sempre que for possível. Assim, escreveremos $\neg A$, $A \longrightarrow B$ e $\forall x A$ para $(\neg A)$, $(A \longrightarrow B)$ e $((\forall x)A)$. Os quantificadores universal e existencial possuem maior precedência do que $\longrightarrow$ e $\longleftrightarrow$ e menor do que $\wedge$, $\vee$ e $\neg$. Veja a nova tabela de precedência:

$\neg$	maior
$\wedge$	
$\vee$	
$\forall$, $\exists$	
$\longrightarrow$	
$\longleftrightarrow$	menor

Símbolos de mesma precedência se associam da esquerda para a direita: $A \longrightarrow B \longrightarrow C$ é o mesmo que $(A \longrightarrow B) \longrightarrow C$.

Assim,

$\forall x\, P(x) \longrightarrow \exists x\, P(x)$

deve ser lido como

$(\forall x\ P(x)) \longrightarrow (\exists x\ P(x))$

E a fórmula

$\exists x\ A \wedge \forall y\ P(x, y) \vee B$

deve ser lida como

$\exists x\ (A \wedge (\forall y\ (P(x, y) \vee B)))$

Em uma fórmula $\forall x\ A$, x **não** é uma variável da linguagem da LPO. Estas se chamam x_1, x_2, x_3, ... Então, o que é x ? É uma **meta-variável**, um símbolo que representa qualquer variável da linguagem. A fórmula $\forall x\, A$ representa infinitas fórmulas, uma para cada variável da linguagem: $\forall x_1\ A$, $\forall x_2\ A$, ...

Exercícios Triviais

4.1. *O que é uma linguagem de primeira ordem ?*

4.2. *Cite uma proposição que pode ser expressa na lógica de primeira ordem mas não no cálculo proposicional.*

4.3. *Se levarmos a definição de linguagem de primeira ordem estritamente, é $\forall x P(x)$ uma fórmula válida ?*

4.2 Introdução à Semântica da Lógica de Primeira Ordem

Uma teoria de primeira ordem é a parte sintática, o sistema formal, correspondente a uma lógica de primeira ordem. Uma teoria de primeira ordem é uma formalização de uma semântica qualquer. Por exemplo, podemos formalizar a Aritmética, a geometria Euclidiana, as relações de ordem ($<$ e suas propriedades), as relações de equivalência ou uma teoria Matemática qualquer. E podemos formalizar um pedaço qualquer do mundo, desde que ele isto seja possível — pode não ser.

Para fazer esta formalização, tomamos um pedaço do mundo qualquer, descobrimos quais os fatos relevantes a respeito deste pedaço e os colocamos em axiomas e regras. Este "pedaço" do mundo é chamado de **modelo**, que será definido mais precisamente nas seções seguintes. No exemplo abaixo, o modelo utilizado é o de um conjunto de animais de um Zoológico. Vejamos a sua descrição.

Suponha que, em um Zoológico, existam:

1. um elefante chamado Efan;
2. um leão chamado Leo;
3. duas gazelas chamadas Gal e Gel;
4. dois coelhos chamados Eloc e Eloá;
5. uma pantera chamada Pant;

6. um gramado, no cercado dos coelhos e gazelas, contendo duzentos pés de grama chamadas (sim, os pés de grama têm nomes!) g_1, g_2, ..., g_{200}.

Há então um conjunto de $1 + 1 + 2 + 2 + 1 + 200 = 207$ elementos a serem considerados. Este conjunto será chamado de conjunto **universo** do modelo.

Há vários fatos relevantes a respeito deste **modelo**:

- Leo e Pant são carnívoros, Efan, Gal, Gel, Eloc e Eloá são herbívoros e g_i é planta;
- quem é herbívoro devora grama (admitiremos que Efan também devora);
- um carnívoro nunca devora um outro carnívoro;
- qualquer carnívoro pode devorar qualquer herbívoro exceto, naturalmente, Efan;[1]
- um carnívoro nunca devora o que os animais que ele devora devora;[2]
- a pantera (onça) é o único animal americano e todos os outros animais são africanos (inclusive os coelhos !). A grama também é africana;

Assuma que os fatos acima são tudo o que sabemos sobre este **modelo**. Para formalizar este modelo em uma teoria de primeira ordem, precisamos de uma linguagem de primeira ordem como definida na seção anterior. A linguagem que utilizaremos aqui será um pouco diferente, mais informal mas não menos rigorosa. Por exemplo, os predicados não terão os nomes P_1^n, P_2^n, P_3^n, ... e sim os mais significativos Carnívoro, Animal, etc.

Este modelo não possui funções. Os predicados da linguagem são:

1. Carnívoro(x) indica que x é carnívoro;
2. Herbívoro(x) indica que x é herbívoro;
3. Devora(x, y) indica que x pode devorar y;
4. Africano(x) indica que x é africano (animal ou planta), não necessariamente exclusivamente africano;
5. Americano(x) indica que x é americano (das Américas);
6. Animal(x) indica que x é animal;
7. Planta(x) indica que x é planta.

Note que os predicados foram definidos a partir das informações relevantes a respeito do modelo. Agora devemos definir as fórmulas que caracterizam este modelo. Mas espere um pouco. Para que queremos estas fórmulas ? As informações relevantes a respeito do modelo já não estão descritas acima ? Pensando bem, as informações estão todas descritas acima, pois por definição o modelo é o que está descrito. Mas nem todas as informações estão explícitas. Por exemplo, não está

[1] Podem, não necessariamente vão, pois estão em um Zoológico.

[2] Se x come y e y come z, então x não come z.

explícito que Leo não devora Pant. E nem que Efan e Eloá devoram g_i. Há muitas informações implícitas. Da maneira informal como está descrito o modelo, não é fácil fazer deduções nem prová-las que estão corretas. Precisamos de fórmulas que **comprimam** todas as informações dadas informalmente. Esta é a essência de qualquer Ciência: conseguir um conjunto de teorias[3] que comprimam todos os dados experimentais do mundo a respeito de certa área. Por exemplo, a fórmula $s = s_0 + v_0 t + at^2/2$ contém todas as informações do mundo relacionadas à posição de objetos macroscópios em movimento com aceleração uniforme. Todos os dados do universo relacionados a este tipo de movimento estão comprimidos nesta equação.

Abaixo são dados as fórmulas que comprimem as informações do modelo do Zoológico, que chamaremos de Zoo. Não são utilizados os nomes dos animais e plantas — isto torna as fórmulas mais gerais, podem ser utilizados em outras situações, o que se tornará claro em breve.

Definição 4.4. *Seja Γ_{Zoo} o conjunto das fórmulas A1-A9 definidas abaixo. Zoo é um modelo de Γ_{Zoo}.*

A1 $\forall x\,(Herbívoro(x) \longrightarrow Devora(x, g))$, *onde g é uma meta-variável que representa qualquer pé de grama (g_i)*

A2 $\forall x\,\forall y\,(Carnívoro(x) \wedge Carnívoro(y) \longrightarrow \neg Devora(x, y))$

A3 $\forall x\,\forall y\,(Carnívoro(x) \wedge Herbívoro(y) \wedge \neg(y = Efan) \longrightarrow Devora(x, y))$

A4 $\forall x\,\forall y\,\forall z\,(Devora(x, y) \wedge Devora(y, z) \longrightarrow \neg Devora(x, z))$

A5 $\forall x\,\forall y\,(Planta(x) \longrightarrow \neg Devora(x, y))$

A6 $\forall x\,\forall y\,(Americano(x) \wedge Americano(y) \longrightarrow (x = y))$

A7 $\forall x\,(Carnívoro(x) \longrightarrow (\exists y\ Devora(x, y))$

A8 $\forall x\,(Animal(x) \longrightarrow (Carnívoro(x) \vee Herbívoro(x) \vee Planta(x)))$

A9 $\forall x\,(Africano(x) \longrightarrow \neg Americano(x))$

A partir destas fórmulas, podemos utilizar as regras da lógica de primeira ordem para fazer deduções **sem** utilizar qualquer informação do modelo. As deduções podem ser feitas utilizando-se apenas a **forma** das fórmulas de Γ_{Zoo} e o relacionamento que existe entre eles. Ou seja, pode-se fazer deduções **puramente sintáticas** que se revelam verdadeiras no modelo, que se supõe seja uma parte do mundo "real". Esta é a essência da lógica: a partir de um modelo (semântica) constrõem-se axiomas e regras (sintaxe) e então podemos fazer deduções utilizando unicamente as regras da lógica, sem recorrer aos significados "informais" do modelo.

As fórmulas do modelo Zoo (Zoológico) estão dados acima. Dizemos que *Zoo é um modelo para Γ_{Zoo}*. Para fazer deduções sobre Zoo, podemos utilizar as fórmulas do conjunto Γ_{Zoo} e todos os axiomas das teorias de primeira ordem (que serão vistos depois). Não é importante que os axiomas das teorias de primeira ordem ainda não foram vistos. Podemos utilizar a intuição para suprir esta falta. Por exemplo, podemos deduzir que se $\forall x\,P(x)$, então $\exists x\,P(x)$. Da mesma forma, utilizaremos a regra MP do CP e regras intuitivas para obter verdades nas TPO.

[3]Não no sentido em que usamos esta palavra neste texto, mas no sentido usual de teoria científica.

Abaixo são mostradas alguns teoremas feitos com os axiomas acima. O raciocínio empregado para conseguir estes teoremas não será mostrado.

1. $\exists x$ (Herbívoro$(x) \longrightarrow$Devora(x, g)), onde g é uma meta-variável que representa qualquer pé de grama (g_i);
2. $\forall x \, \forall y$ (Devora$(x, y) \longrightarrow (\neg$Carnívoro$(x) \vee \neg$Carnívoro$(y)))$, dado que $B \wedge C \longrightarrow \neg A$ é um teorema sse $A \longrightarrow (\neg B \vee \neg C)$.[4]
3. podemos reescrever A1 como $\forall y$ (Herbívoro$(y) \longrightarrow$Devora(y, g)) e utilizá-lo em A4:
   $\forall x \, \forall y$ (Herbívoro$(y) \wedge$ Devora$(x, y) \longrightarrow \neg$Devora$(x, g)))$
4. $\forall x$ ($\neg$(Americano$(x) \wedge$ Africano$(x))$), tirado de A9;
5. $\neg \exists x$ (Americano$(x) \wedge$ Africano(x)), tirado do item anterior.

Abstraindo os Axiomas

As fórmulas do conjunto Γ_{Zoo} podem ser abstraídas ainda mais. Podemos empregar a linguagem de primeira ordem e substituir os predicados Carnívoro(x), Herbívoro(x), Devora(x, y), Africano(x), Americano(x), Animal(x) e Planta(x) por P_c, P_h, P_d, P_{af}, P_{am}, P_{an} e P_p.[5] O nome Efan é substituído por c, uma constante.

Definição 4.5. *Seja Γ_{abs} o conjunto das fórmulas A1-A9 definidas abaixo. Zoo é um modelo de Γ_{abs}.*

A1 $\forall x \, (P_h(x) \longrightarrow P_d(x, g))$, *onde g é uma meta-variável que representa qualquer pé de grama (g_i)*

A2 $\forall x \, \forall y \, (P_c(x) \wedge P_c(y) \longrightarrow \neg P_d(x, y))$

A3 $\forall x \, \forall y \, (P_c(x) \wedge P_h(y) \wedge \neg(y = c) \longrightarrow P_d(x, y))$

A4 $\forall x \, \forall y \, \forall z \, (P_d(x, y) \wedge P_d(y, z) \longrightarrow \neg P_d(x, z))$

A5 $\forall x \, \forall y \, (P_p(x) \longrightarrow \neg P_d(x, y))$

A6 $\forall x \, \forall y \, (P_{am}(x) \wedge P_{am}(y) \longrightarrow (x = y))$

A7 $\forall x \, (P_c(x) \longrightarrow (\exists y \, P_d(x, y))$

A8 $\forall x \, (P_{an}(x) \longrightarrow (P_c(x) \vee P_h(x) \vee P_p(x)))$

A9 $\forall x \, (P_{af}(x) \longrightarrow \neg P_{am}(x))$

[4] Porque exatamente disto será visto adiante. Mas você pode confirmar este fato fazendo a tabela verdade das duas fórmulas. Como elas são logicamente equivalentes, pelo Teorema 3.3 (Completude), $B \wedge C \longrightarrow \neg A \longleftrightarrow A \longrightarrow (\neg B \vee \neg C)$ é um teorema. Disto poderemos, nas próximas seções, deduzir que $B \wedge C \longrightarrow \neg A$ é teorema sse $A \longrightarrow (\neg B \vee \neg C)$ é.

[5] Note que na linguagem de primeira ordem os predicados se chamam P_i^n e estamos utilizando nomes diferentes aqui. Isto não afeta em nada o raciocínio que se seque.

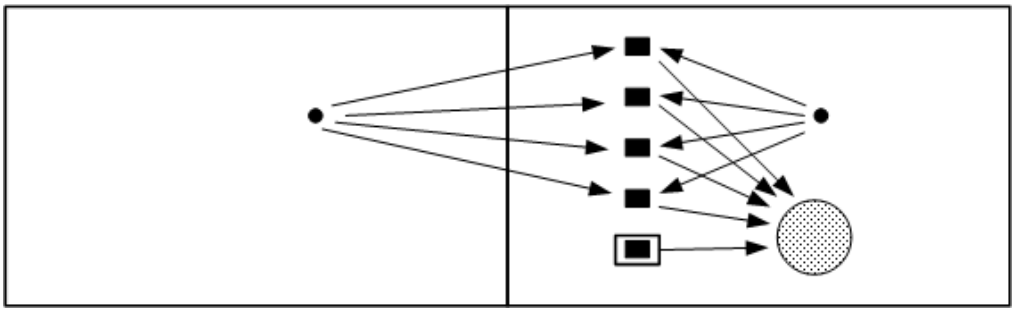

Figura 4.1: Um modelo alternativo para o conjunto de fórmulas Γ_{abs} para o modelo do Zoológico

Mudando os nomes, obtemos conjunto de fórmulas mais abstrato do que o anterior. Uma pergunta surge então naturalmente: será que existe algum outro modelo para Γ_{abs} que não Zoo ? Existe e um outro modelo para Γ_{abs} está na Figura 4.2, que chamaremos de Fig. Considere que

- as bolinhas pretas correspondem aos carnívoros;
- os quadradinhos pretos correspondem aos herbívoros;
- o quadradinho preto embaixo, envolvido por um quadrado, corresponde a Efan;
- os elementos da direita correspondem aos africanos e os da direita aos americanos;
- a bola grande pontilhada representa todos os pés de grama;
- há uma seta de x para y se x devora y;

Os elementos da figura obedecem a todas fórmulas do conjunto Γ_{abs}. De fato, poderíamos ter acrescentado mais bolinhas pretas e quadradinhos e mesmo assim a figura iria satisfazer Γ_{abs}. Este é um dos pontos positivos de sistemas axiomáticos: pode-se obter um conjunto de fórmulas para certo modelo e estas fórmulas podem ser empregadas para modelos completamente diferentes do original. Isto representa um enorme ganho de trabalho, pois nem as fórmulas nem os teoremas que se deduzem delas precisam ser feitos a partir do nada.

Um outro modelo alternativo, chamado Num, pode ser tomado da Aritmética. Considere o conjunto {5, 11, 13, 54, 701} e a correspondência:

- 5 corresponde ao carnívoro americano e 11 a um carnívoro africano;
- todos os números correspondentes aos americanos (só o 5) possuem um único dígito. Todos os números correspondentes aos africanos possuem mais de um dígito;
- o número correspondente à planta, 701, é o único que possui três dígitos;
- os números correspondentes aos herbívoros são 54 e 13, sendo este último correspondente à constante Efan;
- Devora(x, y) corresponde a x divide $y + 1$. Então temos que 5 e 11 dividem $54 + 1$, 54 e 13 dividem $701 + 1$ e 5 e 11 não dividem $701 + 1$.

Exercícios de Treinamento

4.4. *Faça outro modelo para o conjunto Γ_{abs}. Suponha que os professores do ensino fundamental de uma escola façam um curso em uma Universidade onde há pelo menos dois professores, um homem e uma mulher, que não ensinam no ensino fundamental. A relação Devora(x, y) é "x ensina y".*

4.5. *Faça outro modelo com números para as fórmulas Γ_{abs}. Sugestão: acrescente elementos no conjunto universo de Num.*

4.6. *Formalize o modelo Fig.*

4.7. *Crie um conjunto de fórmulas lógicas que caracterizem um conjunto de times de futebol qualquer que está disputando um campeonato. Não faça fórmulas específicas que sirvam, por exemplo, apenas para os times nacionais mais importantes.*

4.8. *Explique a frase: "um conjunto de fórmulas lógicas comprimem informações sobre uma parte do mundo real". Na sua explicação, explique porque as deduções que podem ser feitas utilizando as fórmulas permitem comprimir uma grande quantidade de informação em poucas fórmulas.*

4.9. *Faça um conjunto de fórmulas Γ que caracterizem a copa do mundo de 2006. Isto é, o modelo que queremos espelhar nas fórmulas é a copa do mundo da Alemanha. Na linguagem utilizada, deve existir três constantes c_1, c_2 e c_3 que correspondem ao Brasil, Itália e Alemanha. A linguagem deve ter um símbolo de predicado $G(x)$ tal que G^{Copa}, o predicado do modelo, seja igual a $\{Itália\}$. Este predicado representa o time vencedor da Copa. Obviamente, as suas fórmulas devem ser tais que possa-se obter $\Gamma \vdash G(c_2)$ e $\Gamma \vdash G(x) \longrightarrow (x = c_2)$. Mas $G(c_2)$ não pode ser uma fórmula de Γ.*

4.3 Sintaxe da Lógica de Primeira Ordem

Esta seção apresenta a lógica de primeira ordem como um sistema formal, sem referências à interpretação dos símbolos. Isto significa que os símbolos em si não significam nada. Assim, uma fórmula $\forall x\,(P(x) \longrightarrow Q(x))$ deve ser lida "para qualquer x, $P(x)$ implica $Q(x)$". Mas isto é apenas a forma de se ler a fórmula. Ela não significa que, se $P(x)$ for verdade, $Q(x)$ também o será. Isto é semântica, que será vista na próxima seção. Aqui tudo o que interessa é que temos um conjunto de símbolos e regras para manipular estes símbolos. O que estes símbolos representarão, na semântica, é irrelevante. Poderíamos, por exemplo, trocar $\forall$ por $\Box$ e $\longrightarrow$ por $\circ$ e deixar esta seção exatamente como ela está. Neste caso, uma fórmula $\Box x(P(x) \circ Q(x))$ poderia ser lida, por exemplo, como "Quadrado x, $P(x)$ círculo $Q(x)$". Porém, antes de apresentar os axiomas e regras de dedução, é necessário estudar algumas definições e precauções que devem ser tomadas no estudo deste tipo de sistema formal.

Definição 4.6. *Na fórmula $\forall x\ A$, A é o* **escopo** *do quantificador $\forall x$ que não necessariamente utiliza a variável x. Naturalmente, se não utiliza, o quantificador pode ser retirado e obtemos uma fórmula mais simples e equivalente à original.*

Os símbolos P e f são meta-símbolos que representam um símbolo de predicado qualquer e um símbolo de função qualquer. Assim, $\forall x\, \forall y\, P(x, f(1, y, 3))$ representa todas as fórmulas (infinitas delas) onde P é um símbolo de predicado qualquer de aridade dois e f é um símbolo de função qualquer de aridade três. Note que não adotamos aqui a convenção utilizada no início deste Capítulo onde $H(x)$, $P(x)$, etc eram predicados específicos, representavam exatamente uma propriedade como "homem", "primata", etc.

Podemos também utilizar, nos exemplos, os símbolos de constantes (0, 1, ...), símbolos de função (+, −, /, etc) e predicado convencionais da Aritmética (<, <=).[6] Como exemplo, considere um termo $2 + 3$. Utilizamos a função + na forma normal da Aritmética e não na forma definida pela linguagem. Pela linguagem, as funções de duas variáveis, como o +, têm nomes f_1^2, f_2^2, f_3^2, ... E as constantes têm nomes c_1, c_2, c_3, ... Se fôssemos rigorosos, o termo $2 + 3$ deveria ser escrito como $f_1^2(c_1, c_2)$ com a explicação de que f_1^2 é a soma convencional e que c_1 e c_2 representam o 0 e 1.

Como exemplo, são fórmulas válidas:

1. $\forall x\, \forall y\, (x + y = y + x)$
2. $\forall x\, (x + 0 = x)$
3. $\forall x\, \forall y\, \exists z\, (x < y \longrightarrow (x + z = y \wedge 0 < z))$, se $x < y$, pode-se somar algo a x para alcançar y.

Variáveis livres e ligadas

Uma variável pode ocorrer diversar vezes em uma mesma fórmula. Por exemplo, x em
$x < 0 \wedge \forall x\, P(x)$
aparece três vezes. Mas na primeira ocorrência, a da esquerda, ela está fora do escopo do quantificador existencial e, portanto, representa algo bem diferente das duas outras ocorrências.

Definição 4.7. *Uma ocorrência de uma variável x em uma fórmula A é* **ligada** *se ela está no quantificador ($\forall x$) ou dentro do escopo de um quantificador $\forall x$. Se uma ocorrência de x não é ligada, então ela é* **livre**.

Como exemplo, a primeira ocorrência de x em $x < 0 \wedge \forall x\, P(x)$ é livre. A segunda e a terceria são ligadas.

Utilizaremos $A(x_{i_1}, x_{i_2}, ...x_{i_k})$ para uma fórmula A que possivelmente, mas não necessariamente, possui x_{i_1}, x_{i_2}, ... e x_{i_k} como variáveis **livres**. Se A é
$x < 1 \wedge \forall y\, P(x, y)$
podemos escrever $A(x)$, o que faz sentido, e podemos escrever $A(x, z)$, que não faz sentido mas é correto. E também o mais confuso $A(x, y)$. Escrever $A(x, z)$ ou $A(x, y)$ é equivalente a escrever $A(x)$. Normalmente, queremos que a fórmula $A(x_{i_1}, x_{i_2}, ...x_{i_k})$ tenha x_{i_1}, x_{i_2}, ... e x_{i_k} como variáveis

[6] Um predicado da aritmética deve ser entendido como uma relação. Por exemplo, o conjunto dos pares (x, y) tais que $x < y$ é uma relação. Este conjunto é um subconjunto de $\mathbb{R}^2$.

livres.

Termo livre para uma variável

Freqüentemente será necessário substituir uma variável **livre** por um termo. Por exemplo, dada a fórmula

$\forall x\, \exists z\, (x + z = y)$

pode ser necessário substituir y pelo termo $2 + 3$. Então teríamos

$\forall x\, \exists z\, (x + z = 2 + 3)$

Não houve problema algum, o significado inicial da fórmula continua intacto. O "significado" só será estudado na semântica da LPO mas considere por um momento que esta fórmula se refere aos elementos de $\mathbb{Z}$ (inteiros positivos e negativos) e que possui o significado usual. Neste caso, para todo x, dado um y, existe um z tal que $x + z = y$. Substituindo o y por $2 + 3$ o significado continua válido: para todo x, existe um z tal que $x + z = 2 + 3$. Mas e se o termo tiver variáveis ? Substitua y por $2 + w$:

$\forall x\, \exists z\, (x + z = 2 + w)$

Não houve nenhum problema: para todo x, dado $2 + w$, existe um z tal que $x + z = 2 + w$. Mas e se y for substituído pelo termo $2 + z$? Teremos

$\forall x\, \exists z\, (x + z = 2 + z)$

Agora a interpretação é: para todo x, dado $2 + z$, existe um z tal que $x + z = 2 + z$. O significado mudou. Utilizando as regras usuais da Aritmética, esta fórmula esta dizendo simplesmente que, para todo x, $x = 2$. O que aconteceu ? Colocamos um termo, $2 + z$, que continha uma variável, z, que já estava ligada (pelo $\exists$). Isto sempre mudará o "significado" da sentença. Então este tipo de substituição será proibido. Só se poderá substituir, em uma fórmula A, uma variável y por um termo t se t for livre para y em A.

Se t for $f(w) + 2$, podemos substituir y por t nas seguintes fórmulas

$\forall x\, (y < x)$
$\exists x_1\, \forall x_2\, (f_2(x_1, x_2, y) = 0)$
$\forall y\, (f_1(y) < 0) \longrightarrow (y + 1 = 2)$
$P(y) \wedge \forall x\, (x < y) \longrightarrow \exists w\, P(w)$

Note que a terceira fórmula é $(\forall y (f_1(y) < 0)) \longrightarrow (y+1 = 2)$ e que a terceira e última ocorrência de y é livre nesta fórmula.

A quarta fórmula é $(P(y) \wedge \forall x\, (x < y)) \longrightarrow \exists w\, P(w)$. Substituindo y por $f(w) + 2$ não causa problemas, pois o w deste termo não fica ligado depois da substituição:

$(P(f(w) + 2) \wedge \forall x\, (x < f(w) + 2)) \longrightarrow \exists w\, P(w)$

Um problema acontece sempre que uma variável que é livre no termo se torna ligada na fórmula depois da substituição. Note que só podemos substituir uma **variável livre** por um **termo** em uma fórmula. Nunca substituimos uma variável ligada (a menos que retiremos o quantificador, mas isto é outra estória).

Definição 4.8. *Se A é uma fórmula e t um termo, dizemos que t* **é livre para** *y* **em** *A se nenhuma ocorrência de y em A ocorre dentro do escopo de um quantificador $(\forall w, \exists w)$ onde w é uma variável em t.*

Note que uma variável em um termo é sempre livre, já que termos não possuem quantificadores. Uma variável livre de uma fórmula só deve ser substituída por um termo se o termo for livre para a variável na fórmula. Do contrário obtêm-se resultados diferentes dos esperados.

Exercícios de Treinamento

4.10. *Quando dizemos que A é uma fórmula, isto significa que A não possui variáveis livres ? Quando dizemos que $A(x)$ é uma fórmula com variável livre x, isto significa que x é a única variável livre ?*

4.11. *Represente as seguintes sentenças na linguagem da LPO. Sempre que possível, simplifique as sentenças e depois represente-as novamente em Português. Utilize predicados como $R(x)$ para "x é responsável".*

(a) *Todos os deputados querem que a CPI termine em Pizza.*

(b) *Existe um político que não quer que a CPI termine em Pizza.*

(c) *Não é verdade que se um animal nada ele é um peixe.*

(d) *Não é verdade que se um animal nada ele não é um mamífero.*

(e) *João amava Teresa que amava Raimundo que amava Maria que amava Joaquim que amava Lili, que não amava ninguém.*

(f) *X não gosta de ninguém que goste dele.*

(g) *É tão fácil trocar uma lâmpada que qualquer um pode fazê-lo.*

(h) *Se ele pode fazer a lição, então qualquer um pode.*

(i) *Pelo menos uma pessoa é inteligente.*[7]

(j) *Se João consegue fazer o exercício, Pedro não consegue. E vice-versa. Mas quando Pedro consegue, pelo menos uma outra pessoa da turma consegue fazer o exercício.*

(k) *O barbeiro de uma aldeia faz a barba de todo mundo que não faz a barba de si mesmo.*

(l) *Se dois conjuntos têm os mesmos elementos, eles são iguais.*

(m) *Se L sabia de tudo, L é responsável. Se algum subordinado de L sabia de tudo, então L sabia de tudo. Um subordinado de L sabia de tudo.*

[7] A notação $\exists x$ "significa", na linguagem comum, que existe um x. Mas nada impede que exista mais do que um. Assim, $\exists x$ de fato significa "um ou mais".

(n) *Se L sabia de tudo, L é responsável. Se todos os subordinado de L sabiam de tudo, então L sabia de tudo. Todos sabiam de tudo, inclusive os subordinados de L.*

4.12. *Elimine os parênteses das seguintes fórmulas*

- $(\forall x\,(P(x) \vee Q(x))) \wedge (\exists z\,(z < 0) \longrightarrow R(z))$
- $(x < 1) \wedge (\exists y\,(y < x \wedge P(y)))$
- $\exists x\,(\forall y\,P(x, y)) \longrightarrow (\exists z\,Q(x, z))$
- $\exists x\,(\forall w\,(\forall y\,P(x, y, w))) \longrightarrow (\exists z\,Q(x, z))$

4.13. *Se um termo t é $f(z)$, diga em quais itens abaixo têm-se que t é livre para x na fórmula dada.*

- $\forall y\,(x < 5 \wedge P(y))$
- $\forall y\,(y < 5 \wedge \exists z\,P(x, z))$
- $\forall z\,(z < 1) \longrightarrow \exists y\,P(x, y))$
- $(x < 1) \wedge (\exists y\,(y < x \wedge P(y)))$
- $(x < 1) \wedge (\exists z\,(z < x \wedge P(z)))$

4.14. *Dê exemplo de um termo e de uma fórmula tal que o termo não é livre para uma variável livre x na fórmula.*

4.15. *Explique que problemas podem ocorrer se substituímos, sem cuidados adequados, uma variável livre por um termo qualquer em uma fórmula. Dê um exemplo.*

A Lógica de Primeira Ordem como um Sistema Formal

Uma teoria de primeira ordem é um sistema formal que utiliza pelo menos os esquemas de axiomas e as regras MP e Gen, todos especificados abaixo. Considere que A, B e C são fórmulas quaisquer da linguagem da LPO; isto é, elas são meta-fórmulas. x é uma meta-variável; isto é, x pode ser substituído por qualquer variável da linguagem de primeira ordem.

Axiomas:

(A1) $(A \longrightarrow (B \longrightarrow A))$

(A2) $((A \longrightarrow (B \longrightarrow C)) \longrightarrow ((A \longrightarrow B) \longrightarrow (A \longrightarrow C)))$

(A3) $((\neg B \longrightarrow \neg A) \longrightarrow ((\neg B \longrightarrow A) \longrightarrow B))$

(A4) $(\forall x\,A(x)) \longrightarrow A(t)$ se $A(x)$ é uma fórmula e t é um termo livre para x em $A(x)$

(A5) $(\forall x\,(A \longrightarrow B)) \longrightarrow (A \longrightarrow (\forall x\,B))$ se A não contém ocorrências livres de x

(A6) $x = x$

(A7) $x = y \longrightarrow (A(x,x) \longrightarrow A(x,y))$, onde $A(x,y)$ é a fórmula A onde algumas ou todas as ocorrências livres de x foram substituídas por y. Assume-se que y é livre para x em $A(x,x)$.

Regras de dedução ou inferência:

MP Modus ponens: se A e $A \longrightarrow B$ são teoremas, B também é teorema.

Gen Generalização: se A é teorema, $\forall x\,(A)$ é teorema.

Note que os três primeiros axiomas são os mesmos do CP. Naturalmente, cada um destes esquemas de axiomas pode dar origem a infinitos axiomas. Por exemplo, a meta-variável x em A4 pode ser substituído por qualquer variável da linguagem da LPO: x_1, x_2, x_3, ... Observe que A, B e C são meta-fórmulas e podem ser substituídas por quaisquer fórmulas da linguagem da LPO.

O axioma A7 garante que se $x = y$, então pode-se trocar x por y em qualquer fórmula, desde que a introdução do y em A não modifique o significado original da fórmula. Por exemplo, considere $A(x,x) =_{def} \forall z\,\exists w(z + w = x)$. Claramente $x = y \longrightarrow ((\forall z\,\exists w(z + w = x)) \longrightarrow (\forall z\,\exists w(z + w = y)))$. Mas se y já existir na fórmula $A(x,x)$, a substituição de uma ocorrência de x por y pode tornar uma ocorrência livre em $A(x,x)$ em ligada em $A(x,y)$. Por exemplo, considere $A(x,x) =_{def} \forall y\,(f(x) = f(y))$. Utilizando a interpretação matemática usual, a fórmula A caracteriza uma função f constante, pois, pela regra Gen, se $\vdash A$ então $\vdash \forall x\,\forall y\,(f(x) = f(y))$. Mais uma vez recorremos à semântica para justificar a sintaxe. Mas é isto que deve ser feito: a sintaxe deve espelhar a semântica que se espera, não deve nunca discordar dela.

Se substituirmos a ocorrência livre de x em $\forall y\,(f(x) = f(y))$ por y, obtemos

$$\forall y\, f(y) = f(y)$$

o x, antes livre, agora é ligado. E esta fórmula não caracteriza uma função f constante mas sim uma função qualquer de um argumento, utilizando a interpretação matemática usual. Agora a sintaxe discorda da semântica esperada e é por isto que esta substituição não é válida.

Note que:

- uma teoria de primeira ordem pode ter axiomas específicos para certo domínio, além dos axiomas A1-A5. Por exemplo, uma LPO para formalizar a Aritmética teria axiomas como
  $\forall x\,(x + 0 = x)$
  $\forall x\,\forall y\,(x.(y + 1) = x.y + x)$

  Estes axiomas são chamados de **axiomas próprios** ou **axiomas não lógicos**. É esta última terminologia que empregaremos. *Em todos os teorema que empregaremos nas seções seguintes,* **não** *utilizaremos axiomas não-lógicos*;

- nenhuma LPO precisa de ter novas regras além de MP e Gen. Pode ser provado que qualquer outra regra pode ser simulada com estas duas (talvez com o acréscimo de mais axiomas);

- na regra da generalização, não é necessário que x seja variável livre de A, embora geralmente esta regra seja aplicada neste caso. Ou seja, se

$$P(x, y) \longrightarrow P(y, x)$$

é teorema,

$$\forall z \, (P(x, y) \longrightarrow P(y, x))$$

também é teorema, embora o acréscimo do quantificador não modifique a fórmula em nada.

Definição 4.9. *Uma teoria de primeira ordem sem axiomas não lógicos é chamada de* **cálculo de predicados de primeira ordem**.

Existem infinitos cálculo de predicados de primeira ordem pois, apesar de todos eles compartilharem os axiomas e as regras, os predicados e as funções podem variar. Então um cálculo de predicado pode ter como predicados $P(x)$, $Q(x, y, z)$ e como função $f(x, y, z)$ enquanto que outro cálculo de predicados pode utilizar $R(x, y)$ e função $g(x)$. Um cálculo de predicados pode mesmo não ter nenhum predicado e nenhuma função. É **muito importante** notar que, como um cálculo de predicados não possui *axiomas não lógicos*, nenhuma característica pode ser especificada para os predicados e as funções. Por exemplo, não se pode especificar que um predicado $P(x, y)$ é reflexivo em um cálculo de predicados. Em uma teoria de primeira ordem qualquer, isto pode ser feito utilizando-se um axioma:

$$\forall x \, \forall y \, (R(x, y) \longleftrightarrow R(y, x))$$

Da mesma forma, em um cálculo de predicados não se pode especificar as características das funções. Por exemplo, a função + da aritmética possui a propriedade de que x somado a zero é x:

$$\forall x \, (x + 0 = x)$$

Deste ponto em diante, a palavra "teoria" será utilizada para "teoria de primeira ordem". Utilizaremos $\vdash A$ para significar que A é um teorema de uma teoria de primeira ordem. De qual das infinitas teorias de primeira ordem estamos nos referindo deve ser deduzido do contexto: se, ao provar um teorema sobre uma teoria de primeira ordem qualquer T utilizamos $\vdash A$, então queremos dizer $\vdash_T A$.

Comentários sobre os axiomas

Os axiomas utilizam meta-variáveis que podem ser substituídas por quaisquer variáveis ou por outras meta-variáveis. Então, qualquer variável de uma fórmula A pode ser trocada por outra que não aparece na fórmula. Assim, $P(x, y) \longrightarrow (x = y)$ é equivalente a $P(x, w) \longrightarrow (x = w)$.

No axioma A4, o termo t pode ser simplesmente uma variável x_i. Por exemplo, é uma instância deste axioma

$$(\forall x_1\, P(x_1)) \longrightarrow P(x_1)$$

Logo, se $\forall x_1\, P(x_1)$ é um teorema, por MP podemos deduzir $P(x_1)$, um teorema onde a variável é livre.

O esquema de axioma (ou axioma) A4 exige que t seja livre para x em $A(x)$. Como já foi visto anteriormente, se isto não for exigido obtêm-se resultados diferentes dos esperados.

O axioma A5 exige que A não contenha ocorrências livres de x. E se contiver ? Considere a fórmula

$$(\forall x_1\, (x_1 = 1 \longrightarrow x_1 = 1)) \longrightarrow (x_1 = 1 \longrightarrow (\forall x_1\, (x_1 = 1))$$

que é instância do axioma A5. De fato, uma falsa instância, pois este axioma proíbe que A tenha ocorrências livres de x_i. Neste caso, A é $x_1 = 1$ onde x_1 é livre. Esta instância de axioma, se interpretada com o significado usual, diz que algo verdadeiro, $(\forall x_1\, (x_1 = 1 \longrightarrow x_1 = 1))$, implica algo falso, $(x_1 = 1 \longrightarrow (\forall x_1\, (x_1 = 1))$. Não é porque $x_1 = 1$ que todo x_1 é igual a 1. Note que o primeiro x_1 desta fórmula não se relaciona de forma alguma com o segundo, que está no escopo de um quantificador. Veremos mais sobre isto em breve.

No raciocínio empregado no parágrafo anterior, recorremos à semântica usual dos símbolos $\forall$, $\longrightarrow$, etc. Mas isto não deveria ser importante no estudo da sintaxe, onde os símbolos não têm significado algum. Mas a sintaxe de uma teoria, seus axiomas e regras de dedução, só é feita depois de compreendida a semântica, o que deve ser considerado verdadeiro ou falso. Os axiomas e regras devem **espelhar** a semântica, concordar com ela. Neste caso, consideramos uma instância do axioma A5, $(\forall x_1\, (x_1 = 1 \longrightarrow x_1 = 1)) \longrightarrow (x_1 = 1 \longrightarrow (\forall x_1\, (x_1 = 1))$, e a Aritmética usual. As duas não concordam. Poder-se-ia tomar um exemplo de outra área ou mesmo um exemplo mais abstrato [4]: sejam A e B ambos $P(x)$, um predicado. A "instância" de A5 é

$$(\forall x\, (P(x) \longrightarrow P(x))) \longrightarrow (P(x) \longrightarrow (\forall x\, P(x))$$

Claramente, o antecedente $(\forall x(P(x) \longrightarrow P(x)))$ é sempre verdadeiro mas nem sempre o consequente $(P(x) \longrightarrow (\forall x\, P(x))$ o é. Tome $P(x)$ como "x é par" e considere que os elementos que o x pode assumir são os números inteiros. Esta última fórmula diz que se um x é par, todos os x também são.

Alguns Meta-teoremas sobre as Teorias de Primeira Ordem

Considere uma teoria de primeira ordem T qualquer, que chamaremos simplesmente de teoria. Qualquer menção a "teoria" nesta seção se refere então a uma **específica** teoria T sobre a qual vários meta-teoremas serão enunciados ou provados. Um meta-teorema é um teorema sobre T (diz algo sobre a teoria) e não um teorema de T (que utilizaria a linguagem de primeira ordem).

Teorema 4.1. *Qualquer cálculo de predicados de primeira ordem é consistente.*

Este teorema é muito importante. Ele garante que, se os axiomas não lógicos não forem utilizados, então uma teoria de primeira ordem é consistente.

Definição 4.10. *Seja A uma tautologia no CP e A' uma fórmula da linguagem de primeira ordem (LPO) obtida de A pela substituição das variáveis proposicionais por fórmulas da LPO. Então A' é chamada de* **instância de tautologia**.

Teorema 4.2. *Toda fórmula da LPO que é uma instância de tautologia é um teorema de T.*

Prova. Seja A uma tautologia no CP e A' uma fórmula da linguagem de primeira ordem (LPO) obtida de A pela substituição das variáveis proposicionais por fórmulas da LPO. Pelo Teorema da Completude (página 3.3), A é um teorema do CP. Tome a prova de A no CP e substitua os aparecimentos das variáveis proposicionais pelas fórmulas correspondentes da LPO. Se esta prova utiliza variáveis proposicionais que não aparecem em A, substitua estas variáveis por fórmulas quaisquer da LPO. A prova resultante desta transformação é uma prova de A' em T, linguagem de primeira ordem. Além disso, foram utilizados apenas os axiomas A1-A3 e a regra MP do CP. □

Observação: uma fórmula $\forall x\, P(x) \longrightarrow \forall x\, P(x)$ é um teorema de T pois $B \longrightarrow B$ é uma tautologia no CP. Mesmo envolvendo o quantificador universal, que não está presente no CP, esta fórmula será considerada tautologia em T. Da mesma forma, $(\forall x\ P(x)) \wedge (\forall x\ P(x)) \longrightarrow (\forall x\ P(x))$ é uma tautologia e um teorema de T.

A fórmula $\forall x\ (P(x) \longrightarrow P(x))$ não é uma instância de uma tautologia pois não pode ser obtida pela Definição 4.10. Mas claramente, esta fórmula é um teorema.

Corolário 4.1. *Se considerarmos cada tautologia um axioma, os axiomas* **A1-A3** *podem ser eliminados das teorias de primeira ordem.*

Observação: os axiomas do Cálculo Proposicional são muito difíceis de usar. Qualquer pequena prova de um teorema se transforma em um pesadelo de manipulação de símbolos. Contudo, utilizando o Teorema 4.2 podemos eliminá-los completamente da lista de axiomas das teorias de primeira ordem (TPO). Este teorema diz que, se uma fórmula A é tautologia, então ela é teorema das TPO. Para verificar se certa fórmula é tautologia, podemos utilizar um método semântico como fazer a tabela verdade ou construir o tablô para a fórmula. Ambos são muito mais fáceis do que tentar descobrir uma prova para a fórmula. Assim, de agora em diante os axiomas A1-A3 não serão utilizados, pois consideraremos que, se A é tautologia, A é axioma.

Definição 4.11. *Uma fórmula é* **fechada** *se não possui variáveis livres.*

O teorema da Dedução do CP não pode ser aplicado diretamente em teorias de primeira ordem. A sua aplicação direta resulta em erros. Há várias restrições à sua aplicação e, por simplicidade, enunciaremos uma forma restrita deste teorema.

Teorema 4.3. *(Teorema da Dedução - forma restrita) Se A é uma fórmula fechada e*

$$\Gamma, A \vdash B$$

então

$$\Gamma \vdash A \longrightarrow B$$

Em particular, se $A \vdash B$, então $\vdash A \longrightarrow B$.

Fatos Importantes sobre Teoremas de Teorias de Primeira Ordem

Há inúmeros fatos importantes sobre teorias de primeira ordem que auxiliam em provas de teoremas. Abaixo são detalhados alguns deles. Note que muitos destes fatos se aplicam também ao Cálculo Proposicional.

Lema 4.1. *Sempre que tivermos um teorema ou esquema de teorema A, podemos utilizar A ou qualquer de suas instâncias dentro de uma prova, como é feito no meta-teorema a seguir. Isto é permitido porque, na prova, onde aparece A seria possível expandir a prova repetindo-se todos os passos da prova do próprio A:*

Prova de A:	*Prova de B que utiliza A*
1. A_1	*1.* B_1
2. A_2	*2.* B_2
...	...
...	*k.* A --→ *A prova de A poderia ser inserida aqui*
...	...
n. A	...
	m. B

Lema 4.2. *Se* $\vdash A \longleftrightarrow B$*, então* $\vdash A$ *sse* $\vdash B$

Prova. ($\Longrightarrow$) Assumindo $\vdash A \longleftrightarrow B$, suponha que $\vdash A$ e provaremos $\vdash B$. Toda tautologia é um teorema e vice-versa pelo teorema 3.3 (Completude) e 3.2 (Correção). Então $A \longleftrightarrow B$ e A são tautologias. Logo B é tautologia e portanto teorema.

A prova na direção $\Longleftarrow$ é similar. □

Este lema é muito importante. Ele pode ser aplicado a todas as fórmulas logicamente equivalentes descritas na seção 3.1, página 23. Como exemplo da aplicação deste lema, temos que

$\vdash A \longrightarrow B$ sse $\vdash \neg B \longrightarrow \neg A$

A proposição apresentada abaixo é uma extensão para as teorias de primeira ordem da Proposição 3.3.

Proposição 4.1. *Considere A uma fórmula dentro da qual há uma ou mais ocorrências de uma fórmula B que não contém variáveis livres. Seja* A' *a fórmula obtida a partir de A pela troca de uma ou mais ocorrências de B por* B'*, onde* B' *também não contém variáveis livres. Então, se* $\vdash B \longleftrightarrow B'$*, então* $\vdash A \longleftrightarrow A'$

A proposição seguinte garante que se modificarmos os nomes das variáveis em um teorema ele continua teorema.

Definição 4.12. *Uma fórmula* A' *é uma* **variante** *de A se* A' *pode ser obtida a partir de A pela aplicação sucessiva de zero ou mais das seguintes regras:*

- *troque uma variável livre x em A por y em* A'*, sendo que x e y são meta-variáveis e y não aparece em A nem em* A'*;*
- *troque uma variável quantificada,* $\forall x$ *ou* $\exists x$*, e todas as variáveis x dentro do escopo deste quantificador por y, desde que y não tenha sido utilizada no escopo de* $\forall x$ *em A e seja uma variável nova em* A'*.*

Em resumo, é sempre possível mudar os nomes das variáveis desde que elas não conflitem com nomes de outras variáveis que apareçam na mesma fórmula. Como exemplo, uma fórmula

$\forall x\, P(x) \longrightarrow \exists x\, Q(x,\, c)$ tem como variantes $\forall y\, P(y) \longrightarrow \exists x\, Q(x,\, c)$ e $\forall y\, P(y) \longrightarrow \exists z\, Q(z,\, c)$ mas não $\forall y\, P(x) \longrightarrow \exists w\, Q(y,\, c)$.

As seguintes fórmulas são variantes de $A =_{def} \forall x_1 P(x_1) \longrightarrow \exists y Q(x_1,\, y,\, x_2,\, z)$.

- $\forall x_4 P(x_4) \longrightarrow \exists x Q(x_1,\, x,\, x_2,\, w)$
- $\forall x P(x) \longrightarrow \exists y Q(x,\, y,\, x_3,\, x_2)$
- $\forall x_1 P(x_1) \longrightarrow \exists z Q(w,\, z,\, x_2,\, x_1)$
- $\forall x_1 P(x_1) \longrightarrow \exists y Q(x_1,\, y,\, x_2,\, z)$, a própria fórmula A.

Proposição 4.2. *Seja A' uma variante da fórmula A. Então $\vdash A$ sse $\vdash A'$.*

Esta proposição é importante, pois freqüentemente não podemos utilizar os axiomas A4 e A5 e algumas regras derivadas (dadas a seguir) unicamente por causa de nomes de variáveis.

Considerando os Corolários, os lemas e a proposição acima, estamos em condição de expandir a definição de prova dada na página 3.10.

Corolário 4.2. *Uma seqüência de fórmulas B_1, B_2, ..., B_n é uma prova de B_n se cada B_i é:*

(a) *uma tautologia;*

(b) *uma instância de um dos axiomas* **A4-A7***;*

(c) *resultado da aplicação de MP com B_j e B_k, onde B_k é $B_j \longrightarrow B_i$ e $j, k < i$;*

(d) *resultado da aplicação de Gen com B_j, $j < i$, $B_i =_{def} \forall x\, B_j$;*

(e) *um teorema já provado;*

(f) *logicamente equivalente a B_j, $j < i$; isto é, $B_i \longleftrightarrow B_j$ é uma tautologia;*

(g) *uma variante de uma fórmula B_j, $j < i$.*

Prova. Considerando o Corolário 4.1 e os lemas 4.1 e 4.2 e a Proposição 4.2, a conclusão é imediata. □

Definição 4.13. *Considere um conjunto Γ de fórmulas. Uma seqüência de fórmulas B_1, B_2, ..., B_n é uma prova de B_n considerando Γ com hipóteses se cada B_i é uma fórmula de Γ ou obtido de acordo com o Corolário 4.2. Escrevemos $\Gamma \vdash A$ para "A é teorema tomando-se Γ como hipóteses ou premissas".*

Como exemplo, considere a prova do teorema $C \longrightarrow (A \longrightarrow \forall x\, B), C \vdash \exists x\, \neg B \longrightarrow \neg A$

1. $C \longrightarrow (A \longrightarrow \forall x\, B)$, hipótese
2. C, hipótese
3. $A \longrightarrow \forall x\, B$, MP 1, 2

4. $\neg\forall x\ B \longrightarrow \neg A$, logicamente equivalente a 3

5. $\neg\forall x\ \neg\neg B \longrightarrow \neg A$, logicamente equivalente a 4 pela Proposição 3.3 (que também se aplica a teorias de primeira ordem), pois $B \longleftrightarrow \neg\neg B$.

Agora, pela definição de $\exists$, temos que $\exists x\ C$ é equivalente a $\neg\forall x\ \neg C$. O teorema obtido pode ser escrito como $\exists x\ \neg B \longrightarrow \neg A$ utilizando a definição de $\exists$.

Exercícios Triviais

4.16. *Explique o que é: a) uma meta-variável; b) uma meta-fórmula; c) um meta-teorema e d) um esquema de prova.*

4.17. *O que é uma teoria de primeira ordem.*

4.18. *Quais as regras utilizadas em teorias de primeira ordem ? Uma destas regras, quando aplicada utilizando fórmulas B e C, cria uma nova fórmula A. Esta nova fórmula é maior ou igual a B e C (em número de símbolos) ? A outra regra utiliza apenas uma fórmula B e cria uma fórmula C a partir dela. C é maior do que B ? Com estas informações, e só elas, pode-se deduzir que qualquer teoria de primeira ordem é decidível ?*

4.19. *Explique o axioma A7 em palavras.*

4.20. *O que é um cálculo de predicados de primeira ordem ?*

4.21. *O que é um axioma não lógico ?*

4.22. *Discuta: os axiomas A1-A7 das teorias de primeira ordem não permitem descrever quaisquer características específicas de nenhuma parte da Matemática como Aritmética, Geometria Analítica, Álgebra, etc.*

4.23. *Considere uma teoria de primeira ordem que utiliza uma linguagem com um símbolo de predicado $<$ binário, uma função f unária e constantes c_1, c_2, ... Esta teoria não emprega, como em todo o texto da apostila, axiomas não lógicos. Pode-se afirmar que esta teoria é consistente ? Que teorema garante isto ?*

4.24. *É $\forall x\ (P(x) \longrightarrow P(x))$ uma instância de uma tautologia ? E $\forall x\ P(x) \longrightarrow \forall x\ P(x)$?*

4.25. *Porque pode-se fazer provas em teorias de primeira ordem considerando-se todas as instâncias de tautologias como teoremas ?*

Regras Extras para Teorias de Primeira Ordem

As teorias de primeira ordem vista neste Capítulo utilizam MP e Gen como regras. Uma regra é uma forma de produzir um novo teorema a partir de um ou mais teoremas. Por Modus Ponens, se $\vdash A$ e $\vdash A \longrightarrow B$, então $\vdash B$. Por Gen, se $\vdash A$, então $\vdash \forall x\ A$.

A partir destas regras e dos axiomas **A1-A7**, pode-se deduzir outras regras de inferência, apresentadas a seguir. Convidamos o leitor a prová-las utilizando os teoremas 3.3 (Completude) e 3.2 (Correção).

Lema 4.3. *(Redução ao absurdo) Se* $\neg A \vdash B \wedge \neg B$ *e* A *não possui variáveis livres,* $\vdash A$

Lema 4.4. *Se* $\vdash A$*, então* $\vdash A \vee B$

Lema 4.5. *Se* $\vdash A$*, então* $\vdash B \vee A$

Lema 4.6. $\vdash A \vee B$ *sse* $\vdash B \vee A$

Lema 4.7. $\vdash A \wedge B$ *sse* $\vdash A$ *e* $\vdash B$

Lema 4.8. $\vdash A \wedge B$ *sse* $\vdash B \wedge A$

Lema 4.9. *Se* $\vdash A \vee B$*,* $\vdash A \longrightarrow C$ *e* $\vdash B \longrightarrow C$*, então* $\vdash C$*.*

Lema 4.10. *Se* $\vdash A \vee B$ *e* $\vdash \neg A$*, então* $\vdash B$*.*

Lema 4.11. *Se* $\vdash A \longrightarrow B$ *e* $\vdash B \longrightarrow C$*, então* $\vdash A \longrightarrow C$*.*

Lema 4.12. *Se* $\vdash A \longleftrightarrow B$ *e* $\vdash B \longleftrightarrow C$*, então* $\vdash A \longleftrightarrow C$

Lema 4.13. $\vdash A \longleftrightarrow B$ *sse* $\vdash A \longrightarrow B$ *e* $\vdash B \longrightarrow A$

Lema 4.14. *(Regra da introdução do* $\forall$*) Se* $\vdash A \longrightarrow B$ *e* x *não é livre em* A*, então* $\vdash A \longrightarrow \forall x\, B$*.*

Prova.

1. $A \longrightarrow B$, pois esta fórmula é teorema
2. $\forall x\, (A \longrightarrow B)$, por Gen
3. $(\forall x\, (A \longrightarrow B)) \longrightarrow (A \longrightarrow (\forall x\, B))$, instância de A5
4. $(A \longrightarrow (\forall x\, B))$, MP 2, 3

□

Lema 4.15. *(Regra da substituição) Se* $\vdash A$ *e* A' *é* $A(t_1, t_2, ...t_n)$*, onde* t_i *substitui a variável livre* x_i *de* A*, então* $\vdash A'$*.*

Por esta regra, pode-se substituir as variáveis livres de uma fórmula por outras e a fórmula resultante será teorema se a fórmula original também for. Este resultado também pode ser obtido pela Proposição 4.2.

Lema 4.16. *(Regra da distribuição) Se* $\vdash A \longrightarrow B$*, então* $\vdash \exists x\, A \longrightarrow \exists x\, B$ *e* $\vdash \forall x\, A \longrightarrow \forall x\, B$*.*

Definição 4.14. *O* **fechamento** *de uma fórmula* A *que possui as variáveis livres* x_{i_1}*,* x_{i_2}*, ...,* x_{i_n} *é a fórmula* $\forall x_{i_1}\, \forall x_{i_2}\, \ldots \forall x_{i_n} A$*.*

Lema 4.17. *(Lema da Substituição) Se uma fórmula* A *possui como variáveis livres* x_1*,* x_2*, ...,* x_n*, então*

a) $\vdash A(t_1, t_2, ...t_n) \longrightarrow \exists x_1\, \exists x_2\, \ldots \exists x_n\, A$

b) $\vdash \forall x_1\, \forall x_2\, \ldots \forall x_n\, A \longrightarrow A(t_1, t_2, ...t_n)$

onde t_i substitui x_i em $A(t_1, t_2, ...t_n)$.

Lema 4.18. *(Lema do Fechamento) Se* $\vdash A$ *é o fechamento de A, então* $\vdash A$ *sse* $\vdash A'$.

Há outras formalizações para teorias de primeira ordem, que utilizam outros conjuntos de axiomas, que não **A1-A7**, e outras regras que não MP e Gen. Por exemplo, Shoenfield [6] utiliza os seguintes axiomas

(A1) $A \vee \neg A$

(A2) $A(t) \longrightarrow \exists x\, A$, onde $A = A(x)$

(A3) $x = x$

(A4) $(x_1 = y_1 \wedge x_2 = y_2 \wedge \ldots x_n = y_n) \longrightarrow f(x_1, x_2, \ldots x_n) = f(y_1, y_2, \ldots y_n)$ para todo n natural;

(A5) $(x_1 = y_1 \wedge x_2 = y_2 \wedge \ldots x_n = y_n) \longrightarrow P(x_1, x_2, \ldots x_n) \longleftrightarrow P(y_1, y_2, \ldots y_n)$ para todo n natural;

E as seguintes regras

R1 Se $\vdash A$, então $\vdash B \vee A$ (Regra da expansão)

R2 Se $\vdash A \vee A$, então $\vdash A$ (Regra da contração)

R3 Se $\vdash A \vee (B \vee C)$, então $\vdash (A \vee B) \vee C$ (Regra associativa)

R4 Se $\vdash A \vee B$ e $\vdash \neg A \vee C$, então $\vdash B \vee C$ (Regra do corte)

R5 Se $\vdash A \longrightarrow B$ e x não é livre em B, $\vdash \exists x\, A \longrightarrow B$ (Regra da introdução do $\exists$)

Exercícios de Treinamento

4.26. *Seja A uma fórmula que ocorre dentro de uma fórmula B. A fórmula* B' *é construída a partir de B trocando-se as ocorrências de A por* A' *sendo que* $\vdash A \longleftrightarrow A'$. *Se* $\vdash B$, *então pode-se afirmar que* $\vdash B'$ *? Dê um exemplo de fórmulas A,* A' *e B que obedecem a estas especificações.*

4.27. *Por quê podemos utilizar outras regras na prova de teoremas além de MP e Gen ?*

4.28. *Por quê podemos utilizar um teorema já provado na prova de um outro teorema ?*

4.29. *Se obtivermos* $\Gamma \vdash A$ *e* $\Gamma \vdash \neg A$, *onde* Γ *é um conjunto de fórmulas, isto significa que houve um erro em nossas deduções; isto é, houve um erro nosso ?*

4.30. *Se obtivermos* $\Gamma \vdash A$ *e* $\Gamma \vdash \neg A$, *onde* Γ *é um conjunto de fórmulas, isto significa que* $\Gamma \vdash B$ *onde B é uma fórmula qualquer ?*

4.31. *Cite cinco regras de inferência (ou dedução) para teorias de primeira ordem que não MP e Gen.*

Teoremas de uma LPO

Considere uma teoria de primeira ordem T qualquer. Esta teoria se distingue das demais apenas por empregar diferentes símbolos de predicado e de função e constantes, já que estamos admitindo que T não possui axiomas não lógicos. T emprega pelo menos os axiomas A1-A7 e utiliza as regras MP e Gen. Baseado apenas nestas informações, podemos deduzir vários teoremas. É o que é feito abaixo. Utilizamos a definição de prova dada no Corolário 4.2 que é equivalente à definição original mas mais fácil de trabalhar.

Lema 4.19. $\vdash \forall x\, A \longrightarrow A$ *onde* A *só possui* x *como variável livre.*

Prova. $\forall x\, A(x) \longrightarrow A(x)$ é uma instância de A4 tomando o termo t como x. A fórmula A sempre pode ser escrita como $A(x)$, onde x funciona como um "parâmetro" para a fórmula. Da mesma forma, A pode ser escrita $A(x)$, mesmo que A não tenha x como variável livre. Logo, $\forall x\, A \longrightarrow A$ é também uma instância de A4. □

Lema 4.20. *Se* $\vdash \forall x\, A$*, então* $\vdash A$*.*

Prova.

1. $\forall x\, A$, pois esta fórmula é teorema
2. $\forall x\, A \longrightarrow A$, utilizando o lema 4.19
3. A, MP 1, 2

□

Lema 4.21. $\vdash \forall x\, \forall y\, A \longrightarrow \forall y\, \forall x\, A$ *se* A *não possui variáveis livres.*

Prova. Provaremos $\forall x\, \forall y\, A \vdash \forall y\, \forall x\, A$. Pelo Teorema da Dedução, provamos o teorema.

1. $\forall x\, \forall y\, A$, hipótese
2. $\forall x\, \forall y\, A \longrightarrow \forall y\, A$, instância de A4
3. $\forall y\, A$, MP 1, 2
4. $\forall y\, A \longrightarrow A$, instância de A4
5. A, MP 3, 4
6. $\forall x\, A$, Gen aplicada a 5
7. $\forall y\, \forall x\, A$, Gen aplicada a 6

□

Pode-se fazer uma prova menor utilizando-se o 4.17:

Prova.

1. $\forall x\, \forall y\, A$, hipótese
2. $\forall x\, \forall y\, A \longrightarrow A$, aplicando o Lema 4.17 b) onde t_1 é x e t_2 é y. Lembre-se de que $A(x, y) = A$
3. A, MP 1, 2
4. $\forall x\, A$, Gen aplicada a 3
5. $\forall y\, \forall x\, A$, Gen aplicada a 4

□

Lema 4.22. $\vdash A(t) \longrightarrow \exists x\, A(x)$

Prova.

1. $\forall x\, \neg A(x) \longrightarrow \neg A(t)$, instância de A4
2. $A(t) \longrightarrow \neg \forall x \neg A(x)$, fórmula logicamente equivalente a 1, pelo Teorema 4.2, pois $(C \longrightarrow D) \longleftrightarrow (\neg D \longrightarrow$

Esta última fórmula é uma abreviatura de $A(t) \longrightarrow \exists x\, A(x)$ □

Lema 4.23. $\vdash \forall x\, A \longrightarrow \exists x\, A$ *onde* A *só possui* x *como variável livre.*

Prova. Como $\forall x\, A$ não possui variáveis livres, podemos aplicar o Teorema da Dedução. Então provaremos $\forall x\, A \vdash \exists x\, A$ e teremos a prova do teorema.

1. $\forall x\, A$, hipótese
2. A, lema 4.20
3. $A(x) \longrightarrow \exists x\, A(x)$, pelo lema 4.22 com $t = x$
4. $\exists x\, A(x)$, MP 2, 3

Como $A = A(x)$, temos $\forall x\, A \vdash \exists x\, A$ e $\vdash \forall x\, A \longrightarrow \exists x\, A$. □

Exercícios de Treinamento

4.32. *Prove sintaticamente que as seguintes fórmulas são teoremas.*

(a) $\forall x\, A(x) \longrightarrow A(c)$*, onde a linguagem possui uma constante* c*;*

(b) $(x = y) \longrightarrow (f(x) = f(y))$*, onde a linguagem possui uma função unária* f*;*

(c) $\neg\exists x\, A(x) \longleftrightarrow \forall x\, \neg A(x)$

(d) $A \longleftrightarrow \exists x\, A$, *onde* x *não é livre em* A. *Use o Teorema da Dedução;*

(e) $x = y \longrightarrow y = x$

(f) $t = t$, *onde* t *é um termo qualquer;*

(g) $\exists x\, (x = c)$, *onde* c *é uma constante da linguagem.*

4.33. *Prove sintaticamente as seguintes fórmulas são teoremas dadas as hipótese.*

(a) $\forall x\, A(x) \vdash A(c)$, *onde a linguagem possui uma constante* c;

(b) $A(w) \vdash \exists x\, A(x)$, *onde* A *possui uma variável livre* x *e* $A(w)$ *é a fórmula* A *com* x *substituído por* w;

(c) $A \longrightarrow B$, $\forall x\, \neg B \vdash \neg\exists x\, A$. *Use o Teorema da Dedução;*

(d) A, $A \vee B \longrightarrow C \vdash C$

(e) $\forall x\, A(x), P(c) \vdash \exists x\, (A \wedge P(x))$

4.4 Semântica das Teorias de Primeira Ordem

Esta seção apresenta a semântica associada às teorias de primeira ordem. Considere T uma teoria de primeira ordem sem axiomas não lógicos. Então T possui os axiomas A1-A7, sendo que os teoremas de T são obtidos pela definição de prova dada no Corolário 4.2.

Antes de introduzir formalmente a semântica de uma teoria de primeira ordem, estudaremos novamente o "modelo" Zoológico da Seção 4.2 (página 66), chamado aqui de Zoo. Este "modelo" obedece as fórmulas do conjunto Γ_{abs} — veja Definição 4.5. Mas não é o único modelo de Γ_{abs}. Os "modelos" Fig (página 4.2) e Num (página 4.2) também satisfazem as fórmulas de Γ_{abs}. Utilizamos as fórmulas de Γ_{abs} e não de Γ_{Zoo} porque Γ_{abs} não contém predicados específicos de Zoo (como Devora e Africano).

Estes "modelos" podem ser descritos de maneira formal utilizando-se a notação matemática da teoria dos conjuntos. O objetivo é relacionar o modelo com as fórmulas de Γ_{abs} de tal forma que possamos **interpretar** as fórmulas no modelo. Isto é, dada uma fórmula, podemos mapeá-la no modelo e verificar o que realmente ela significa. O mesmo mapeamento pode ser feito com teoremas. Então é necessário mapear todos os elementos utilizados nas fórmulas de Γ_{abs}, que são os predicados e a constante c, que é utilizada em A3. Além disso, quando uma fórmula utiliza $\forall x$, deve ser especificado exatamente quais os valores que x pode assumir. E estes valores serão sempre os do conjunto universo do modelo, o conjunto de todos os elementos do modelo. No "modelo" Zoo, $\forall x$ significa que x deve assumir todos os animais e plantas do Zoológico.

O "modelo" Zoo é caracterizado por

(a) um conjunto de elementos, Z = {Efan, Leo, Gal, Gel, Eloc, Eloá, Pant, g_1, g_2, ..., g_{200}}, o universo do modelo;

(b) os predicados Carnívoro, Herbívoro, Devora, Africano, Americano, Animal e Planta. Estes predicados se comportam como funções cujo contradomínio é V ou F, verdadeiro ou falso. Na definição formal de modelo, predicados são relações matemáticas:[8]

1. Carnívoro = $\{\text{Leo}, \text{Pant}\}$, associado a P_c;
2. Herbívoro = $\{\text{Efan}, \text{Gal}, \text{Gel}, \text{Eloc}, \text{Eloá}\}$, associado a P_h;
3. Devora = $\{< \text{Leo}, \text{Gal} >, < \text{Leo}, \text{Gel} >, < \text{Leo}, \text{Eloc} >, < \text{Leo}, \text{Eloá} >, < \text{Pant}, \text{Gal} >, < \text{Pant}, \text{Gel} >, < \text{Pant}, \text{Eloc} >, < \text{Pant}, \text{Eloá} >\}$, associado a P_d;
4. Africano = $\{\text{Efan}, \text{Leo}, \text{Gal}, \text{Gel}, \text{Eloc}, \text{Eloá}, \text{g}_1, \text{g}_2, ..., \text{g}_{200}\}$, associado a P_{af};
5. Americano = $\{\text{Pant}\}$, associado a P_{am};
6. Animal = $\{\text{Efan}, \text{Leo}, \text{Gal}, \text{Gel}, \text{Eloc}, \text{Eloá}, \text{Pant}\}$, associado a P_{an};
7. Planta = $\{\text{g}_1, \text{g}_2, \ldots \text{g}_{200}\}$, associado a P_p.

Note que uma relação n-ária é um subconjunto de Z^n. Assim, Carnívoro $\subseteq Z$ e Devora $\subseteq Z^2$.

(c) a associação entre c e Efan; isto é, a constante c utilizada no axioma A3 dos axiomas abstratos A1-A9 da página 69 representa Efan.

Da mesma forma, o "modelo" Num é caracterizado por

(a) um conjunto de elementos $N = \{5, 11, 13, 54, 701\}$

(b) as relações

1. $R_1 = \{5, 11\}$, associado a P_c;
2. $R_2 = \{13, 54\}$, associado a P_h;
3. $R_3 = \{< 5, 54 >, < 5, 701 >, < 11, 54 >, < 11, 701 >\}$, associado a P_d;
4. $R_4 = \{11, 13, 54, 701\}$, associado a P_{af};
5. $R_5 = \{5\}$, associado a P_{am};
6. $R_6 = \{5, 11, 13, 54\}$, associado a P_{an};
7. $R_7 = \{701\}$, associado a P_p.

(c) a associação entre c e 13.

Considere o conjunto Γ_{abs} (Definição 4.5, página 69) que é satisfeito pelos "modelos" Zoo e Num como definidos acima. Isto é, cada fórmula de Γ_{abs} é *verdadeira* em cada um destes modelos. É este o motivo pelo qual chamamos Zoo e Num como *modelos* — as fórmulas realmente espelham algumas informações do Zoológico e do conjunto de números de Num. As fórmulas não necessariamente espelham **todas** as informações. Por exemplo, eles não dizem que o elefante é mais pesado do que um coelho. Ou que uma das gazelas é mais rápida do que o elefante ou que os pés de grama não se movem e são verdes. Mas nas informações que nos interessam, o "mundo real" correspondente ao Zoológico foi precisamente definido pelo conjunto universo do modelo Zoo, as relações e as associações entre as relações/símbolos de predicado e Efan/constante c. É isto que estudaremos

[8] R é uma relação n-ária sobre um conjunto A se $R \subseteq A^n$

de agora em diante. Dado um conjunto de fórmulas, como Γ_{abs}, estudaremos o que podem ser modelos para estas fórmulas e quais as interações entre os modelos e as fórmulas. Esta parte da Lógica é chamada de Teoria dos Modelos.

Definição Formal de Modelo

A linguagem de primeira ordem definida na seção 4.1 utiliza infinitas constantes, infinitos símbolos de função e predicado. Contudo, ao trabalhar com um conjunto específico de fórmulas, como Γ_{abs} da página 69, somente são utilizados um conjunto pequeno de símbolos. Neste caso, são utilizados apenas c, P_c, P_h, P_d, P_{af}, P_{am}, P_{an} e P_p. Estes símbolos, com a definição de linguagem de primeira ordem, definem uma linguagem $\mathcal{L}$. Não faz sentido, nos modelos Zoo e Num, utilizar fórmulas que utilizam outros símbolos que não estes. Por exemplo, não faz sentido verificar se a fórmula $\forall x\,(\text{Mamífero}(x) \longrightarrow \text{Animal}(x))$, que utiliza o símbolo Mamífero, é verdadeira no modelo Zoo.

Mais especificamente, dado um conjunto de fórmulas Γ, existe uma linguagem $\mathcal{L}$ cujas fórmulas são definidas como na Seção 4.1 mas que utiliza apenas as constantes e símbolos de predicado e função que aparecem nas fórmulas de Γ. Em todo o texto abaixo, quando se trabalha com um conjunto de fórmulas qualquer Γ, estará subentendido que utilizamos uma linguagem $\mathcal{L}$ capaz de expressar todas as fórmulas de Γ.

Definição 4.15. *Seja $\mathcal{L}$ a linguagem associada a um conjunto de fórmulas Γ. Uma* **estrutura** $\mathfrak{A}$ *para $\mathcal{L}$ consiste de:*

1. *um conjunto $|\mathfrak{A}| \neq \emptyset$ chamado de universo da estrutura. É necessário que os elementos deste conjunto possam ser comparados com um predicado especial, o de igualdade, cujo símbolo é o mesmo utilizado na definição de linguagem de primeira ordem, $=$. Este predicado possui a seguinte propriedade: b e c representam elementos iguais no conjunto $|\mathfrak{A}|$ se e somente se $b = c$;*

2. *uma função I de interpretação tal que*

   - *para cada símbolo de função f de $\mathcal{L}$ de aridade n, $I(f)$ corresponde a uma função de $|\mathfrak{A}|^n$ em $|\mathfrak{A}|$. $f^{\mathfrak{A}}$ será utilizado para denotar $I(f)$;*
   - *para cada símbolo de predicado P de $\mathcal{L}$ de aridade n, $I(P)$ corresponde a uma relação em $|\mathfrak{A}|^n$. $P^{\mathfrak{A}}$ será utilizado para denotar $I(P)$. Então $P^{\mathfrak{A}} \subseteq |\mathfrak{A}|^n$;*
   - *para cada constante c de $\mathcal{L}$, $I(c)$ corresponde a um elemento fixo de $|\mathfrak{A}|$, que será denotado por $c^{\mathfrak{A}}$.*

Escreveremos $\mathfrak{A} =< |\mathfrak{A}|, I >$ para uma estrutura $\mathfrak{A}$ com conjunto universo $|\mathfrak{A}|$ e função de interpretação I. Para as interpretações usuais utilizaremos uma representação mais suscinta. Por exemplo, para representar uma estrutura associada aos números naturais, utilizaremos $<$ $\mathbb{N}$, $+$, $\centerdot$, 0, 1 $>$. Fica subentendido que esta estrutura utiliza uma linguagem $\mathcal{L}$ com símbolos de função, de aridade 2, $+$ e $\centerdot$ e com constantes 0 e 1. Naturalmente, o símbolo de função $+$ da linguagem corresponde à função $+$ na estrutura e 0 da linguagem corresponde ao elemento 0 do conjunto $\mathbb{N}$. O mesmo se aplica a $\centerdot$ e 1. Apesar de utilizarmos os mesmos símbolos para a linguagem e para a estrutura, o leitor deve ter em mente que se tratam de conceitos diferentes.

Alguns textos utilizam efetivamente símbolos diferentes, como + para a linguagem e $+^{\mathbb{N}}$ para a soma de números naturais. Note que a linguagem utiliza apenas uma constante mas $\mathbb{R}$ contém infinitos elementos.

Pode-se utilizar predicados também, como em $< \mathbb{R}, \leq, +, \centerdot, 0 >$ onde a linguagem possui um único símbolo de predicado $\leq$, símbolos de função + e $\centerdot$ e a constante 0. Sim, $\leq$ é um predicado, uma relação em $\mathbb{R}^2$. Escrevemos $(a,\ b) \in R$ ou $a\ R\ b$ para "a está na relação R com b". Então pode-se escrever $(a,\ b)\ \in \leq$ ou $a \leq b$, que é a notação usual. Uma relação binária R sobre $\mathbb{R}^2$ é tal que $R \subseteq \mathbb{R}$ e

$$R = \{(a,\ b) \in \mathbb{R}^2 : a \text{ e } b \text{ satisfazem certas propriedades } \}$$

Então $\leq\ \subseteq\ \mathbb{R}^2$ e

$$\leq = \{(a,\ b) \in \mathbb{R}^2 : a \text{ é menor do que } b\ \}$$

Como um exemplo, considere a linguagem $\mathcal{L}_{abs}$ utilizada pelas fórmulas do conjunto Γ_{abs} da Definição 4.5. Esta linguagem utiliza uma constante c e os símbolos de predicado P_c, P_h, P_d, P_{af}, P_{am}, P_{an} e P_p. Símbolos de função não são utilizados. Pela definição de estrutura dada acima, Zoo é uma estrutura da linguagem $\mathcal{L}_{abs}$ pois possui um conjunto universo e uma função de interpretação. A função de interpretação faz as seguintes associações:

P_c --→ Carnívoro $=_{def} P_c^{\mathfrak{A}}$
P_h --→ Herbívoro $=_{def} P_h^{\mathfrak{A}}$
P_d --→ Devora $=_{def} P_d^{\mathfrak{A}}$
P_{af} --→ Africano $=_{def} P_{af}^{\mathfrak{A}}$
P_{am} --→ Americano $=_{def} P_{am}^{\mathfrak{A}}$
P_{an} --→ Animal $=_{def} P_{an}^{\mathfrak{A}}$
P_p --→ Planta $=_{def} P_p^{\mathfrak{A}}$
c --→ Efan $=_{def} c^{\mathfrak{A}}$

Então $I(P_c) =$ Carnívoro $=_{def} P_c^{\mathfrak{A}}$, $I(P_h) =$ Herbívoro $=_{def} P_h^{\mathfrak{A}}$, ..., $I(c) =$ Efan $=_{def} c^{\mathfrak{A}}$. O domínio de I é o conjunto das constantes e símbolos de função e predicado de $\mathcal{L}_{abs}$. O contradomínio de I é o conjunto dos elementos de $|\mathfrak{A}|$, as funções $f : |\mathfrak{A}|^n \longrightarrow |\mathfrak{A}|$ e as relações $R \subset |\mathfrak{A}|^n$.

É importante observar que:

1. uma estrutura se refere a uma **linguagem**, não a um conjunto de axiomas;
2. uma função da estrutura é sempre de $|\mathfrak{A}|^n$ para $|\mathfrak{A}|$. Não se pode restringir o domínio ou o contradomínio. Se isto for necessário, utilize relações para simular funções;
3. podem existir várias estruturas para uma mesma linguagem. Por exemplo, o "modelo" Num é também uma estrutura de $\mathcal{L}_{abs}$.

Uma fórmula $\forall x\ (P_h(x) \longrightarrow P_{af}(x))$ sem uma **interpretação** não significa nada. Ela deve ser lida como "para qualquer x, se $P_h(x)$, então $P_{af}(x)$". Contudo, no estudo da sintaxe, o "para qualquer" o "se" e "então" são apenas uma forma de se ler a fórmula, não implicam em nenhuma forma de considerar a fórmula verdadeira ou falsa.

Uma fórmula só é verdadeira ou falsa em uma certa **estrutura**. Então, tomando a estrutura $\mathfrak{A}$ = Zoo definido acima, da linguagem $\mathcal{L}_{abs}$, a fórmula $\forall x\,(P_h(x) \longrightarrow P_{af}(x))$ significa

"para todo b do conjunto universo de Zoo, se $P_h^{\mathfrak{A}}(b)$, então $P_{af}^{\mathfrak{A}}(b)$"

$P_h^{\mathfrak{A}}$ e $P_{af}^{\mathfrak{A}}$ são relações e a notação usual para verificar se um elemento b pertence a uma relação R é $b \in R$. Reescrevendo a frase acima nesta notação temos:

"para todo b do conjunto universo de Zoo, se $b \in P_h^{\mathfrak{A}}$, então $b \in P_{af}^{\mathfrak{A}}$"

$\forall x$ sempre significará "para todo x do conjunto universo da **estrutura**". E agora o "para todo", "se" e "então" assumem o significado usual da nossa língua, como veremos adiante. Esta frase quer dizer exatamente o que está escrito. O modelo Zoo associa $P_h^{\mathfrak{A}}$ a Herbívoro e $P_{af}^{\mathfrak{A}}$ a Africano. Então, esta frase, escrita na linguagem empregada no modelo Zoo, é

"para todo b do conjunto universo de Zoo, se $b \in$ Herbívoro, então $b \in$ Africano"

Ou melhor,

"para todo b do conjunto universo de Zoo, se b é Herbívoro, então b é Africano".

Para cada **estrutura** da linguagem $\mathcal{L}_{abs}$ há uma interpretação da fórmula $\forall x(P_h(x) \longrightarrow P_{af}(x))$. Por exemplo, para Num temos

"para todo b do conjunto universo de Num, se $b \in R_2$, então $b \in R_4$".

Ou melhor,

"para todo b do conjunto $\{5, 11, 13, 54, 701\}$, se $b \in \{13, 54\}$, então $b \in \{11, 13, 54, 701\}$".

Com estas observações, podemos definir formalmente o que é "verdade" e "falso" em uma estrutura.

Satisfabilidade, Verdade e Modelo

Intuitivamente, pode-se definir verdade de uma fórmula A em uma estrutura $\mathfrak{A}$ pela seguinte definição indutiva:

1. se A é $P(t_1, t_2, ...t_n)$ e os valores do universo $|\mathfrak{A}|$ do modelo correspondentes a t_1, t_2, ... t_n forem $t_1^{\mathfrak{A}}$, $t_2^{\mathfrak{A}}$, ... $t_n^{\mathfrak{A}}$, então A é verdade em $\mathfrak{A}$ se e somente se $< t_1^{\mathfrak{A}}, t_2^{\mathfrak{A}}, ...t_n^{\mathfrak{A}} > \in P^{\mathfrak{A}}$. No exemplo do Zoo, $P_h(c)$ é verdade em Zoo sse Efan $\in P_h^{\mathfrak{A}}$ ou melhor, Efan $\in$ Herbívoro. Esta regra *intuitiva* diz que um predicado, interpretado no modelo, é verdade se a relação associada a ele é verdadeira;[9]

[9]Escrevemos $R(x, y)$ ao invés de $(x,\ y) \in R$ assim como escrevemos $x < y$ ao invés de $(x,\ y) \in <$.

2. se A é $\neg B$, A é verdade em $\mathfrak{A}$ se e somente se B é falsa;

3. se A é $C \longrightarrow D$, A é verdade em $\mathfrak{A}$ se e somente se C é falsa ou D é verdadeira;

4. se A é $\forall x B$, A é verdade em $\mathfrak{A}$ se e somente se para cada x do universo de $\mathfrak{A}$, $B(x)$ é verdade.

A definição formal de verdade emprega as noções de satisfabilidade, definida abaixo. Para definir esta noção, é antes necessário definir o que é o valor de um termo t em uma estrutura $\mathfrak{A}$ dados os valores que as suas variáveis livres devem assumir. O "valor de um termo t" é um valor em $|\mathfrak{A}|$ resultado da avalição das possíveis funções em t e da substituição das variáveis e constantes em t pelos respectivos valores. Por exemplo, considere um termo $t =_{def} x + c$ avaliado em um modelo dos números naturais em que x é substituído por 1 e c corresponde a 0. Então o valor de t neste modelo com x substituído por 1 é igual a $1 + 0$, que é 1. Note que em $1 + 0$, o $+$ é a função soma do modelo, os números naturais. O $+$ da definição de t, $x + c$, é apenas um símbolo de função sem significado.

Definição 4.16. *Considere $\mathfrak{A}$ uma estrutura de uma linguagem $\mathcal{L}$. Seja t um termo $t(v_1, v_2, ...v_n)$ com v_i, v_2, ... v_n meta-variáveis[10] livres[11] e $\vec{a} = (a_1, a_2, ...a_n)$ uma seqüência de elementos em $|\mathfrak{A}|$ ($a_i \in |\mathfrak{A}|$). O valor de t em $\mathfrak{A}$, utilizando-se $\vec{a}$, é denotado por $t^{\mathfrak{A}}[\vec{a}]$ e definido indutivamente como:*

1. *a_i se t é v_i;*

2. *$c^{\mathfrak{A}}$ se t é c e $c^{\mathfrak{A}}$ é o valor de $|\mathfrak{A}|$ associado à constante c de $\mathcal{L}$;*

3. *$f^{\mathfrak{A}}(t_1^{\mathfrak{A}}[\vec{a}], t_2^{\mathfrak{A}}[\vec{a}], \ldots t_k^{\mathfrak{A}}[\vec{a}])$ se t é $f(t_1, t_2, \ldots t_k)$.*

Então $t^{\mathfrak{A}}[\vec{a}]$ é na verdade a aplicação de uma função específica para $\mathfrak{A}$ que toma $\vec{a}$ e t como parâmetros e produz um elemento de $|\mathfrak{A}|$ como resultado:

$$g_{\mathfrak{A}} : T^* \times S^* \longrightarrow |\mathfrak{A}|$$

T^* é o conjunto de termos da linguagem e S^* é o conjunto de todas as seqüências de $|\mathfrak{A}|$, de todos os tamanhos. Usamos $\mathfrak{A}$ em $g_{\mathfrak{A}}$ para indicar que g é específico para esta estrutura. Outra estrutura exigiria um g diferente.

Intuitivamente, $t^{\mathfrak{A}}[\vec{a}]$ é o elemento de $|\mathfrak{A}|$ resultado da avaliação de t em $\mathfrak{A}$ utilizando $\vec{a}$ como valores para as variáveis livres. Note que $t^{\mathfrak{A}}[\vec{a}]$ é um valor em $|\mathfrak{A}|$ e t é apenas uma seqüência de símbolos sem significado. Note que $f^{\mathfrak{A}}(t_1^{\mathfrak{A}}[\vec{a}], t_2^{\mathfrak{A}}[\vec{a}], \ldots t_k^{\mathfrak{A}}[\vec{a}])$ é um valor em $|\mathfrak{A}|$ — a função realmente é avaliada.

Como exemplo de cálculo do valor de um termo, se t é $(x + 1) \centerdot y$ e $\vec{a} = (1, 3)$, então $t^{\mathfrak{A}}[\vec{a}] = (1 + 1) \centerdot 3 = 6$. Utilizando a função $g_{\mathfrak{A}}$, temos

$$g_{\mathfrak{A}}((x + 1) \centerdot y, \vec{a}) = 6 \text{ com } \vec{a} = (1, 3)$$

$t^{\mathfrak{A}}[\vec{a}]$ é o valor de t em $\mathfrak{A}$ usando os valores de $\vec{a}$ para as variáveis livres. Note que o valor de t depende não só dos valores de $\vec{a}$, mas também da estrutura utilizada. Poderíamos ter valores diferentes em diferentes interpretações. Por exemplo, suponha que $t = x_1 + x_2$ e $\vec{a} = (1, 1)$. Na estrutura $\mathfrak{B} =_{def} < \mathbb{N}, +, \centerdot, 0, 1 >$ dos números naturais, $t^{\mathfrak{B}}[\vec{a}] = 2$. Mas no modelo $\mathbb{Z}_2$, que contém apenas os elementos 0 e 1 e onde $1 + 1 = 0$, temos $t^{\mathbb{Z}_2}[\vec{a}] = 0$.

[10] Qualquer uma delas pode ser substituída por qualquer outra variável da linguagem.

[11] Variáveis que aparecem em termos sempre são livres.

Definição 4.17. *Seja $A(v_1, v_2, ...v_n)$ uma fórmula em uma linguagem $\mathcal{L}$, v_i, v_2, ... v_n meta-variáveis, $\mathfrak{A}$ uma estrutura de $\mathcal{L}$ e $\vec{a}$ uma seqüência em $|\mathfrak{A}|$. Escrevemos "$\vec{a}$ satisfaz A em $\mathfrak{A}$", denotado por $\mathfrak{A} \vDash A[\vec{a}]$, se uma das condições abaixo é satisfeita.*

1. *A é $t_1 = t_2$ e $t_1^{\mathfrak{A}}[\vec{a}] = t_2^{\mathfrak{A}}[\vec{a}]$. Ou seja,*

   $$\mathfrak{A} \vDash (t_1 = t_2)[\vec{a}] \text{ sse } t_1^{\mathfrak{A}}[\vec{a}] = t_2^{\mathfrak{A}}[\vec{a}]$$

   O símbolo $=$ que aparece em $t_1 = t_2$ é o símbolo da linguagem de primeira ordem. O símbolo $=$ em $t_1^{\mathfrak{A}}[\vec{a}] = t_2^{\mathfrak{A}}[\vec{a}]$ é o igual do modelo $\mathfrak{A}$. Admite-se que todos os modelos utilizam o mesmo símbolo de igualdade e que este símbolo é igual ao símbolo das linguagens de primeira ordem. Uma notação mais precisa utilizaria $=^{\mathfrak{A}}$ para denotar igualdade entre os elementos do modelo $\mathfrak{A}$;

2. *A é $P(t_1, t_2, ...t_n)$ e $(t_1^{\mathfrak{A}}[\vec{a}],\ t_2^{\mathfrak{A}}[\vec{a}], \ldots t_n^{\mathfrak{A}}[\vec{a}]) \in P^{\mathfrak{A}}$. Ou seja,*

   $$\mathfrak{A} \vDash P(t_1, t_2, ...t_n)[\vec{a}] \text{ sse } (t_1^{\mathfrak{A}}[\vec{a}],\ t_2^{\mathfrak{A}}[\vec{a}], \ldots t_n^{\mathfrak{A}}[\vec{a}]) \in P^{\mathfrak{A}}$$

3. *A é $\neg B$ e $\mathfrak{A} \nvDash B[\vec{a}]$. Ou seja,*

   $$\mathfrak{A} \vDash \neg B[\vec{a}] \text{ sse } \mathfrak{A} \nvDash B[\vec{a}]$$

   *Escrevemos $\mathfrak{A} \nvDash B[\vec{a}]$ para "$\vec{a}$ **não** satisfaz A em $\mathfrak{A}$". Isto é, utilizando as regras de satisfabilidade apresentadas aqui, não se consegue provar que $\mathfrak{A} \vDash B[\vec{a}]$;*

4. *A é $B \longrightarrow C$ e $\mathfrak{A} \nvDash B[\vec{a}]$ ou $\mathfrak{A} \vDash C[\vec{a}]$. Ou seja,*

   $$\mathfrak{A} \vDash (B \longrightarrow C)[\vec{a}] \text{ sse } \mathfrak{A} \nvDash B[\vec{a}] \text{ ou } \mathfrak{A} \vDash C[\vec{a}]$$

   B e C não necessariamente possuem todas as variáveis livres de $B \longrightarrow C$. Então poder-se-ia pensar que $B[\vec{a}]$ não tem significado, pois o número de valores em $\vec{a}$ pode ser maior do que o número de variáveis livres em B. Então, considere $B =_{def} B(v_1, v_2, ...v_n)$ e $C =_{def} C(v_1, v_2, ...v_n)$;

5. *A é $\forall x\, B(x)$. Seja y uma variável que não pertence ao conjunto de variáveis utilizadas em A. Então*

   $$\mathfrak{A} \vDash (\forall x\, B(x))[\vec{a}] \text{ sse } \mathfrak{A} \vDash B^x_y[b, \vec{a}] \text{ para todo } b \in |\mathfrak{A}|$$

   onde $|\mathfrak{A}|$ é o universo da estrutura $\mathfrak{A}$. O símbolo B^x_y é a fórmula B com x substituído por y. Naturalmente, b substituirá y nas aplicações indutivas para se descobrir a satisfabilidade de A.

Intuitivamente, $\mathfrak{A} \vDash A[\vec{a}]$, $\vec{a}$ satisfaz A em $\mathfrak{A}$, se A é verdade no modelo $\mathfrak{A}$ segundo a interpretação usual de verdade. Isto é, a fórmula é interpretada como se referisse exclusivamente ao modelo $\mathfrak{A}$: a igualdade é a igualdade no modelo, os símbolos de predicados de A são os predicados do modelo, o $\forall x$ de A refere-se a todos os elementos do universo $|\mathfrak{A}|$ e assim por diante.

Os conectivos derivados $\wedge$, $\vee$ e $\longleftrightarrow$ são definidos a partir de $\neg$ e $\longrightarrow$ e o quantificador existencial $\exists$ é definido usando $\forall$. A definição de satisfação para estes conectivos derivados e para $\exists$ é deduzida utilizando a definição de satisfação para fórmulas que usam $\neg$, $\longrightarrow$ e $\forall$. Então temos

1. $\mathfrak{A} \vDash (B \wedge C)[\vec{a}]$ sse $\mathfrak{A} \vDash B[\vec{a}]$ e $\mathfrak{A} \vDash C[\vec{a}]$
2. $\mathfrak{A} \vDash (B \vee C)[\vec{a}]$ sse $\mathfrak{A} \vDash B[\vec{a}]$ ou $\mathfrak{A} \vDash C[\vec{a}]$

3. $\mathfrak{A} \vDash (B \longleftrightarrow C)[\vec{a}]$ sse ($\mathfrak{A} \vDash B[\vec{a}]$ e $\mathfrak{A} \vDash C[\vec{a}]$) ou ($\mathfrak{A} \nvDash B[\vec{a}]$ e $\mathfrak{A} \nvDash C[\vec{a}]$)

4. $\mathfrak{A} \vDash (\exists x\, B(x))[\vec{a}]$ sse $\mathfrak{A} \vDash B^x_y[b, \vec{a}]$ para algum $b \in |\mathfrak{A}|$

   onde y é uma variável que não pertence ao conjunto de variáveis utilizadas em B e $|\mathfrak{A}|$ é o universo da estrutura $\mathfrak{A}$. O símbolo B^x_y é a fórmula B com x substituído por y.

A definição de satisfação dada acima exige que a estrutura obedeça as regras da lógica clássica. Por exemplo, quando se diz que os números naturais são uma estrutura para uma certa linguagem[12] exige-se mais do que a presença das funções $+$ e $\cdot$. Exige-se, por exemplo, que neste pequeno mundo dos números uma fórmula como $\neg B$ seja considerada "verdadeira" se e somente se B seja considerada "falsa". E que $\forall x\, B(x)$ seja considerada "verdadeira" se e somente se B é "verdadeira" quando x é substituído por cada um dos números naturais. *Então de certa forma as interpretações interpretam também os símbolos da lógica como* $\neg$, $\longrightarrow$ *e* $\forall$. Estes símbolos são interpretados como na lógica clássica, o que pode ser observado nas definições de satisfação acima:

- o $\neg$ é interpretado com um não: $\mathfrak{A} \vDash \neg B[\vec{a}]$ sse $\vec{a}$ **não** satisfaz B em $\mathfrak{A}$;
- o $\longrightarrow$ é interpretado como no CP: $\mathfrak{A} \vDash (B \longrightarrow C)[\vec{a}]$ sse $\vec{a}$ **não** satisfaz B em $\mathfrak{A}$ ou $\vec{a}$ satisfaz C em $\mathfrak{A}$;
- o $\wedge$ é interpretado como um "e" na linguagem natural (Português, Inglês): $\mathfrak{A} \vDash (B \wedge C)[\vec{a}]$ sse $\vec{a}$ satisfaz B em $\mathfrak{A}$ **e** $\vec{a}$ satisfaz C em $\mathfrak{A}$;
- o $\vee$ é interpretado como um "ou" na linguagem natural: $\mathfrak{A} \vDash (B \vee C)[\vec{a}]$ sse $\vec{a}$ satisfaz B em $\mathfrak{A}$ **ou** $\vec{a}$ satisfaz C em $\mathfrak{A}$;
- o $\longleftrightarrow$ é interpretado como um "se e somente se" na linguagem natural: $\mathfrak{A} \vDash (B \longleftrightarrow C)[\vec{a}]$ sse ($\vec{a}$ satisfaz B em $\mathfrak{A}$ **se e somente se** $\vec{a}$ satisfaz C em $\mathfrak{A}$).

Uma estrutura então é composta por uma "parte do mundo" mais as regras da lógica clássica. Note que a definição de estrutura engloba, de certa forma, a definição de tabelas verdade do cálculo proposicional: elas dizem o que deve ser verdadeiro e o que deve ser falso (por enquanto, satisfazível e não satisfazível).

Estudaremos um exemplo de satisfabilidade através da linguagem $\mathcal{L}_\Delta$ que utiliza

1. um símbolo de predicado unário R;
2. um símbolo de predicado binário S;
3. uma função f.

Esta linguagem não utiliza constantes.

Considere uma estrutura $\mathfrak{B}$ de $\mathcal{L}_\Delta$ com universo $|\mathfrak{B}| = \{a, b, c, d, e\}$, predicados $R^{\mathfrak{B}}$ e $S^{\mathfrak{B}}$ e função $f^{\mathfrak{B}}$ tais que

- $R^{\mathfrak{B}}$ é predicado unário e $R^{\mathfrak{B}} = \{a, c, e\}$;

[12] Por exemplo, a linguagem que utiliza os símbolos $+$, $\cdot$, $<$ e a constante 0.

- $S^{\mathfrak{B}}$ é predicado binário e $S^{\mathfrak{B}} = \{(a,b),(b,c),(c,d),(d,e),(e,a)\}$;
- $f^{\mathfrak{B}}$ é função que toma um argumento definida pela tabela

x	$f^{\mathfrak{B}}(x)$
a	b
b	c
c	d
d	e
e	a

Por simplicidade, colocamos os nomes na estrutura (predicados e função) com um nome que já é associado diretamente ao símbolo correspondente na linguagem. Assim, é claro que a função $f^{\mathfrak{B}}$ da estrutura $\mathfrak{B}$ é associada ao símbolo de função f da linguagem $\mathcal{L}_{\Delta}$. E $R^{\mathfrak{B}}$ é associado ao símbolo de predicado R da linguagem.

Verificaremos se algumas fórmulas são satisfeitas por algumas seqüências. Adotaremos a convenção de que se a fórmula possui as variáveis livres x, y e z e é avaliada com a seqüência $\vec{a} = (a_1,\ a_2,\ a_3)$, então x assumirá o valor a_1, y o valor a_2 e z, a_3. Ou seja, os valores serão assumidos na ordem lexicográfica.

(a) $\vec{a} = (a,b)$ e $A =_{def} A(x,\ y) =_{def} S(x,\ f(y))$. Temos que

$$\mathfrak{B} \vDash A[\vec{a}] \text{ sse } (t_1^{\mathfrak{B}}[\vec{a}], t_2^{\mathfrak{B}}[\vec{a}]) \in S^{\mathfrak{B}}$$

onde t_1 é x e t_2 é $f(y)$. Mas $t_1^{\mathfrak{B}}[\vec{a}]$ é igual a a e $t_2^{\mathfrak{B}}[\vec{a}]$ é igual a $f^{\mathfrak{B}}(t_3^{\mathfrak{B}}[\vec{a}])$, onde t_3 é y. Mas $t_3^{\mathfrak{B}}[\vec{a}]$ é b e portanto $t_2^{\mathfrak{B}}[\vec{a}]$ é $f^{\mathfrak{B}}(b)$, que é c. Logo,

$$\mathfrak{B} \vDash A[\vec{a}] \text{ sse } (t_1^{\mathfrak{B}}[\vec{a}], t_2^{\mathfrak{B}}[\vec{a}]) \in S^{\mathfrak{B}} \text{ sse } (a,c) \in S^{\mathfrak{B}}$$

Como $(a,c) \notin S^{\mathfrak{B}}$, então $\mathfrak{B} \nvDash A[\vec{a}]$

(b) $\vec{a} = (b)$ e $A =_{def} A(x) =_{def} S(x,\ f(x))$. Temos que

$$\mathfrak{B} \vDash A[\vec{a}] \text{ sse } (t_1^{\mathfrak{B}}[\vec{a}], t_2^{\mathfrak{B}}[\vec{a}]) \in S^{\mathfrak{B}}$$

onde t_1 é x e t_2 é $f(x)$. Mas $t_1^{\mathfrak{B}}[\vec{a}]$ é igual a b e $t_2^{\mathfrak{B}}[\vec{a}]$ é igual a $f^{\mathfrak{B}}(t_3^{\mathfrak{B}}[\vec{a}])$, onde t_3 é x. Mas $t_3^{\mathfrak{B}}[\vec{a}]$ é b e portanto $t_2^{\mathfrak{B}}[\vec{a}]$ é $f^{\mathfrak{B}}(b)$, que é c. Logo,

$$\mathfrak{B} \vDash A[\vec{a}] \text{ sse } (t_1^{\mathfrak{B}}[\vec{a}], t_2^{\mathfrak{B}}[\vec{a}]) \in S^{\mathfrak{B}} \text{ sse } (b,c) \in S^{\mathfrak{B}}$$

Como $(b,c) \in S^{\mathfrak{B}}$, temos que $\mathfrak{B} \vDash A[\vec{a}]$.

(c) $\vec{a} = (c,a,b)$ e $A =_{def} \forall x S(x, f(x))$. Note que a fórmula A não possui variáveis livres e portanto os valores de $\vec{a}$ não interferem na satisfabilidade da fórmula. Logo $\vec{a}$ pode ser de qualquer tamanho e ter quaisquer elementos. Temos que

$$\mathfrak{B} \vDash A[\vec{a}] \text{ sse para todo elemento } g \in \mathtt{B},\ \mathfrak{B} \vDash A_y^x[g,\vec{a}]$$

A_y^x é a fórmula $S(y,\ f(y))$ que possui y com variável livre e a condição de satisfabilidade se torna

$$\mathfrak{B} \vDash A[\vec{a}] \text{ sse para todo elemento } g \in \mathtt{B},\ \mathfrak{B} \vDash S(y,\ f(y))[g,\vec{a}]$$

É facilmente verificável que, tomando-se qualquer g de B, temos

$$\mathfrak{B} \vDash S(y,\ f(y))[g,\vec{a}]$$

Por exemplo, tome $g = b$. Pelo item 2 acima,

$$\mathfrak{B} \vDash A^x_y[b, \vec{a}]$$

Conclue-se que $\mathfrak{B} \vDash A[\vec{a}]$.

Claramente pode-se chegar às conclusões obtidas acima sem tantas formalidades. O mesmo resultado pode ser obtido de maneira mais informal mas não menos rigorosa:

(a) $\vec{a} = (a, b)$ e $A =_{def} A(x,\ y) =_{def} S(x,\ f(y))$. Temos que

$$\mathfrak{B} \vDash A[\vec{a}] \text{ sse } \mathfrak{B} \vDash S(x,\ f(y))[\vec{a}] \text{ sse } (a,\ f^{\mathfrak{B}}(b)) \in S^{\mathfrak{B}} \text{ sse } (a,\ c) \in S^{\mathfrak{B}}$$

Como $(a,\ c) \notin S^{\mathfrak{B}}$, $\mathfrak{B} \nvDash A[\vec{a}]$

(b) $\vec{a} = (b)$ e $A =_{def} A(x) =_{def} S(x,\ f(x))$. Temos que

$$\mathfrak{B} \vDash A[\vec{a}] \text{ sse } \mathfrak{B} \vDash S(x,\ f(x))[\vec{a}] \text{ sse } (b, f^{\mathfrak{B}}(b)) \in S^{\mathfrak{B}} \text{ sse } (b, c) \in S^{\mathfrak{B}}$$

Como $(b, c) \in S^{\mathfrak{B}}$, temos que $\mathfrak{B} \vDash A[\vec{a}]$.

(c) $\vec{a} = (c, a, b)$ e $A =_{def} \forall x\, S(x,\ f(x))$.

$$\mathfrak{B} \vDash A[\vec{a}] \text{ sse para todo elemento } g \in \mathtt{B},\ \mathfrak{B} \vDash A^x_y[g, \vec{a}]$$

$$\text{sse para todo elemento } g \in \mathtt{B},\ \mathfrak{B} \vDash S(y,\ f(y))[g, \vec{a}]$$

$$\text{sse para todo elemento } g \in \mathtt{B},\ (g, f^{\mathfrak{B}}(g)) \in S^{\mathfrak{B}}$$

Tomando-se um elemento $g \in \mathtt{B}$ qualquer, tem-se que $f^{\mathfrak{B}}(g)$ é o elemento seguinte a g na seqüência $\{a,\ b,\ c,\ d,\ e\}$ com a seguindo-se a e. Na definição do predicado $S^{\mathfrak{B}}$, os elementos são todos do tipo (x, elemento seguinte a x). Portanto, para todo $g \in \mathtt{B}$,

$$(g, f(g)) \in S^{\mathfrak{B}}$$

e $\mathfrak{B} \vDash A[\vec{a}]$.

No item c) acima ocorreu algo interessante: não utilizamos somente a relação ou predicado $S^{\mathfrak{B}}$ para verificar a satisfabilidade de uma fórmula. Utilizamos também o conhecimento que obtivemos observando os elementos da relação — descobrimos que eles seguiam um certo padrão. Neste exemplo finito e pequeno, isto não seria necessário. Poderíamos ter tomado elemento a elemento de $\mathtt{B}$ e substituído na fórmula. Contudo, não podemos fazer isto em interpretações com um número infinito de elementos. Nestes casos é fundamental utilizar o conhecimento que não está completamente formalizado. Por exemplo, uma estrutura dos números naturais nunca poderá ser colocada explicitamente em predicados e funções. O predicado $<^{\mathbb{N}}$, por exemplo, seria infinito:

$$<^{\mathbb{N}} = \{(0,\ 1), (0,\ 2), \ldots (1,\ 2), (1,\ 3), \ldots\}$$

Exercícios de Treinamento

4.34. *(i4d4) Verifique se as seguintes fórmulas são satisfazíveis na estrutura do Zoológico dada na página 87. Assuma que a seqüência $\vec{a}$ possui valores para as variáveis livres na ordem lexicográfica. Isto é, se as variáveis* **livres** *de uma fórmula são x, y e z e a seqüência for (v_1, v_2, v_3), então assume-me que v_1 é o valor associado a x, v_2 a y e assim por diante.*

(a) $\vec{a} = (Pant)$ *e* $A =_{def} Carnívoro(x) \wedge Americano(x)$

(b) $\vec{a} = (Leo, Eloá)$ *e* $A =_{def} Planta(x) \longrightarrow \neg Devora(x, y)$

(c) $\vec{a} = (Efan)$ *e* $A =_{def} \neg\exists x\, Devora(x, y)$

(d) $\vec{a} = (Efan, Leo, Pant)$ *e* $A =_{def} \exists x\, Herbívoro(x) \wedge \neg\exists y\, Devora(y, x)$

(e) $\vec{a} = (Gal, g_1)$ *e* $A =_{def} Devora(x, y) \wedge \neg Devora(y, x)$

(f) $\vec{a} = (Gal, g_1)$ *e* $A =_{def} Devora(x, y) \wedge \neg\exists z Devora(z, x)$

4.35. *Defina estrutura para uma linguagem. Dê um exemplo de linguagem e de duas estruturas para ela.*

4.36. *As funções de uma estrutura podem ser de um subconjunto do universo para outro subconjunto ?*

4.37. *Escreva, em Português, o significado da fórmula* $\forall x\, (P_d(x,\, y) \longrightarrow \neg\exists z\, P_d(y,\, z))$ *utilizando o modelo Zoo. Verifique se esta fórmula é verdadeira no modelo Zoo.*

4.38. *Defina modelo de um conjunto de fórmulas. Faça um exemplo de modelo de um conjunto de duas fórmulas.*

4.39. *O valor de um termo* t *em uma estrutura* $\mathfrak{A}$ *é um objeto de que tipo ? É símbolo da linguagem ? Constante ? Um número natural ou real ?*

4.40. *Explique em palavras com se calcula o valor de um termo para uma estrutura utilizando* $\vec{a}$ *como seqüência.*

4.41. *Dê um exemplo de modelo e de verificação que* $\mathfrak{A} \vDash (\forall x\, A(x))[\vec{a}]$ *para certa fórmula* A *com variável livre* x *e seqüência* $\vec{a}$*.*

4.42. *Explique: uma estrutura representa parte do mundo em forma de conjuntos e associações com símbolos de uma linguagem mais as regras da lógica clássica.*

Definição 4.18. *Uma fórmula* A *de uma linguagem* $\mathcal{L}$ *é* **verdadeira** *na estrutura* $\mathfrak{A}$ *de* $\mathcal{L}$ *se e somente se toda seqüência* $\vec{a}$ *de* $|\mathfrak{A}|$ *satisfaz* A*. Isto é,* $\mathfrak{A} \vDash A[\vec{a}]$ *para toda sequência* $\vec{a}$*. Escrevemos* $\mathfrak{A} \vDash A$*.*

Definição 4.19. *Uma fórmula* A *de uma linguagem* $\mathcal{L}$ *é* **falsa** *na estrutura* $\mathfrak{A}$ *de* $\mathcal{L}$ *se e somente se nenhuma seqüência* $\vec{a}$ *de* $|\mathfrak{A}|$ *satisfaz* A*. Isto é,* $\mathfrak{A} \nvDash A[\vec{a}]$ *para toda sequência* $\vec{a}$*. Escrevemos* $\mathfrak{A} \nvDash A$*.*

Estas definições são muito, muito importantes. Elas dizem quando uma fórmula é **verdadeira** e quando é **falsa**. Estas definições correspondem, aproximadamente, no Cálculo Proposicional, a uma fórmula ser tautologia (verdadeira) e contradição (falsa). Uma estrutura $\mathfrak{A}$ na Lógica de Primeira Ordem corresponde às tabelas verdades dos conectivos no CP. E uma seqüência $\vec{a}$ em $\mathfrak{A}$ na LPO corresponde a uma atribuição de valores às variáveis de uma fórmula no CP; isto é, uma linha da tabela verdade.

Há algumas observações importantes sobre as definições acima:

- uma certa fórmula pode não ser verdadeira nem falsa. Por exemplo, na estrutura do Zoológico, a fórmula

  $$\text{Carnívoro}(x) \wedge \text{Devora}(x, y)$$

  não é verdadeira nem falsa. Depende da seqüência $\vec{a}$. Se $\vec{a} = (\text{Leo}, \text{Gal})$, a fórmula é satisfeita. Se $\vec{a} = (\text{g}_1, \text{Efan})$, a fórmula não é satisfeita. Contudo, uma fórmula é satisfeita por uma seqüência $\vec{a}$ em um modelo $\mathfrak{A}$ **ou** não é; uma das duas coisas ocorre mas não ambas;

- se uma fórmula é verdadeira então ela não é falsa. Se é verdadeira, pela definição ela não pode ser falsa. Se é falsa, pela definição de fórmula falsa ela não pode ser verdadeira;

- uma fórmula fechada, sem variáveis livres, é sempre verdadeira ou falsa.

Definição 4.20. *Uma estrutura $\mathfrak{A}$ de uma linguagem $\mathcal{L}$ é um* **modelo** *para uma fórmula A em $\mathcal{L}$ se A é* **verdadeira** *em $\mathfrak{A}$. Escrevemos $\mathfrak{A} \vDash A$.*

Definição 4.21. *Uma estrutura $\mathfrak{A}$ de uma linguagem $\mathcal{L}$ é um* **modelo** *para um* **conjunto** *de fórmulas Γ em $\mathcal{L}$ se $\mathfrak{A}$ é modelo para cada uma das fórmulas de Γ. Escrevemos $\mathfrak{A} \vDash \Gamma$.*

Então $\mathfrak{A}$ é modelo de Γ se e somente se

$$\mathfrak{A} \vDash A \text{ para todo } A \in \Gamma$$

Definição 4.22. *Escrevemos $\Gamma \vDash A$ para indicar que A é verdadeira em todos os modelos do conjunto Γ.*

Isto é, para qualquer $\mathfrak{A}$ modelo de Γ, então $\mathfrak{A} \vDash A$.

O leitor pode se perguntar porque a definição de **estrutura** e **modelo**. Estas definições foram feitas sob medida para representar estruturas matemáticas como os números naturais e suas operações, os números reais, as diferentes geometrias, grupos, etc. Uma característica importante de qualquer estrutura é que nela todos os axiomas das teorias de primeira ordem (Seção 4.3, página 75) são verdadeiros. Antes de provar este ponto, veremos alguns lemas importantes.

Lema 4.24. *A é falso em uma estrutura $\mathfrak{A}$ se e somente se $\neg A$ é verdadeiro.*

Prova. A é falso em uma estrutura $\mathfrak{A}$ (definição de fórmula falsa) sse nenhuma seqüência $\vec{a}$ de $|\mathfrak{A}|$ satisfaz A em $\mathfrak{A}$ sse (expandindo "nenhuma seqüência") toda seqüência $\vec{a}$ de $|\mathfrak{A}|$ não satisfaz A em $\mathfrak{A}$ sse (pela definição 4.17 de $\nvDash$) para toda seqüência $\vec{a}$ de $|\mathfrak{A}|$ tem-se $\mathfrak{A} \nvDash A[\vec{a}]$ sse (pela definição 4.17 de satisfação de $\neg B$) para toda seqüência $\vec{a}$ de $|\mathfrak{A}|$ tem-se $\mathfrak{A} \vDash \neg A[\vec{a}]$ sse (definição de fórmula verdadeira em uma estrutura) $\neg A$ é verdadeira. □

Esta prova poderia ser resumida [4] para : uma seqüência $\vec{a}$ satisfaz $\neg A$ sse $\vec{a}$ não satisfaz A. Portanto, todas as seqüências satisfazem $\neg A$ sse nenhuma seqüência satisfaz A. Ou seja, $\neg A$ é verdadeira sse A é falsa.

É importante observar que, se A possui variáveis livres, A poderia não ser verdadeiro nem falso em uma estrutura. Mas se A é verdadeiro em uma estrutrura $\mathfrak{A}$, $\neg A$ é falso. E se A é falso em $\mathfrak{A}$,

$\neg A$ é verdadeiro. Então temos três possíves "valores verdade" para uma fórmula A com variáveis livres: V, F, I. Este último símbolo representa "a fórmula não é nem verdadeira nem falsa".

Pode-se concluir que, se uma fórmula A não é verdadeira, A pode ter tanto o valor verdade F como I. Então se A não é verdadeira não se pode concluir que A possui o valor verdade F, que A é falsa.

Proposição 4.3. *Os axiomas das teorias de primeira ordem são verdadeiros em qualquer estrutura.*

Prova. Não faremos as provas de que todos os axiomas A1-A7 são verdadeiros. De fato, provaremos apenas que as instâncias do axioma A1,

$$(A \longrightarrow (B \longrightarrow A))$$

são verdadeiras em qualquer estrutura. Considere um modelo $\mathfrak{A}$ de uma linguagem $\mathcal{L}$ e seja $\vec{a}$ uma seqüência qualquer em $|\mathfrak{A}|$. Provaremos que

$$\mathfrak{A} \vDash (A \longrightarrow (B \longrightarrow A))[\vec{a}] \text{ para toda seqüência } \vec{a}$$

ou seja, toda seqüência $\vec{a}$ em $|\mathfrak{A}|$ satisfaz $(A \longrightarrow (B \longrightarrow A))$ em $\mathfrak{A}$, que resumiremos para

$$\text{todo } \vec{a} \text{ satisfaz } (A \longrightarrow (B \longrightarrow A))$$

pois o universo $|\mathfrak{A}|$ e o modelo $\mathfrak{A}$ ficam subentendidos. Então temos

$\vec{a}$ satisfaz $(A \longrightarrow (B \longrightarrow A))$ sse
(definição 4.17 de satisfação) $\vec{a}$ não satisfaz A ou $\vec{a}$ satisfaz $B \longrightarrow A$ sse
(definição 4.17 de satisfação) $\vec{a}$ não satisfaz A ou ($\vec{a}$ não satisfaz B ou $\vec{a}$ satisfaz A)

Como a última frase é sempre verdade em nossa lógica informal, qualquer $\vec{a}$ satisfaz $(A \longrightarrow (B \longrightarrow A))$ e esta fórmula é **verdadeira** em $\mathfrak{A}$. Como $\mathfrak{A}$ é uma estrutura qualquer, este axioma é verdadeiro em qualquer estrutura. Note que convertemos os símbolos lógicos para palavras em Português ("ou" neste caso). Pela definição de **estrutura** isto sempre acontecerá.

Se a fórmula A utilizada no axioma $A \longrightarrow (B \longrightarrow A)$ tiver variáveis livres, é possível que ela seja satisfeita para algumas seqüências $\vec{a}$ e não seja satisfeita para outras (em um dado modelo $\mathfrak{A}$). Mas, fixado um $\vec{a}$, a fórmula $A \longrightarrow (B \longrightarrow A)$ é sempre satisfeita no modelo. Isto é, $\mathfrak{A} \vDash (A \longrightarrow (B \longrightarrow A))[\vec{a}]$ se $\mathfrak{A} \vDash A[\vec{a}]$ ou $\mathfrak{A} \nvDash A[\vec{a}]$. Como para toda seqüência $\vec{a}$ temos $\mathfrak{A} \vDash (A \longrightarrow (B \longrightarrow A))[\vec{a}]$, então $\mathfrak{A} \vDash A \longrightarrow (B \longrightarrow A)$ e o axioma é verdadeiro na estrutura $\mathfrak{A}$. Como tomamos uma estrutura $\mathfrak{A}$ qualquer, este axioma é verdadeiro em todas as estrutras da linguagem $\mathcal{L}$ utilizada pelo axioma. Como a linguagem pode ser qualquer uma, o axioma é verdadeiro em todas as linguagens.

Usando as tabelas verdade de $\neg$ e $\longrightarrow$, podemos deduzir que todos os axiomas do cálculo proposicional são tautologias. Nas teorias de primeira ordem, as verdades são deduzidas a partir da noção de satisfabilidade em uma estrutura (Definição 4.17), que possuem as informações presentes nas tabelas verdade. □

Definição 4.23. *Uma fórmula A de uma linguagem $\mathcal{L}$ é* **logicamente válida** *se e somente se ela é* **verdadeira** *em todas as estruturas de $\mathcal{L}$.*

Uma fórmula é verdadeira em uma estrutura se ela é verdadeira naquele pedaço "mundo real" que chamamos de **estrutura** de uma linguagem (veja definição 4.18). Se uma fórmula é verdadeira em todas as estruturas, então é claro que a veracidade dela não depende de algo particular de nenhuma parte do "mundo real", a sua veracidade não depende das estruturas. Isto significa que

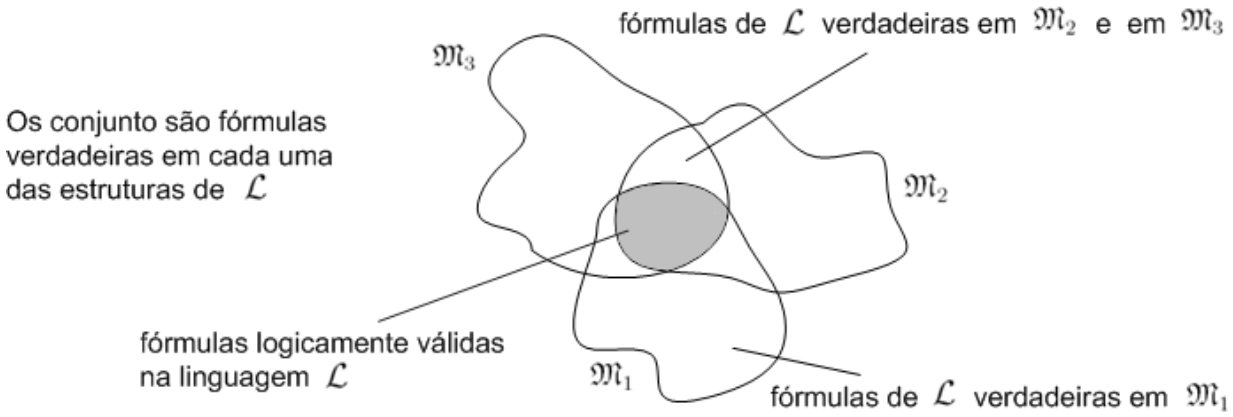

Figura 4.2: Conjuntos representando fórmulas verdadeiras nas estruturas de uma linguagem $\mathcal{L}$. Assuma que as três estruturas representadas representem todas as estruturas desta linguagem.

a fórmula é **logicamente válida** por causa da definição 4.18 de verdade, que por sua vez depende da definição 4.17 de satisfação. A definição de satisfação depende das *regras da lógica usuais*, como $A \longrightarrow B$ é verdade se A é falsa ou B é verdade. Então a definição de **logicamente válida** coincide com a definição usual, que é ser verdade independente de uma estrutura específica. É "verdade" apenas pelas regras da lógica.

A Figura 4.4 mostra conjuntos de fórmulas verdadeiras em três estruturas de uma linguagem $\mathcal{L}$. Assuma que estas três estruturas representem todas as possíveis estruturas desta linguagem. As fórmulas logicamente válidas em $\mathcal{L}$ estão na intersecção de todos os três conjuntos representados na figura, que é a área em cinza.

Pode uma linguagem ter apenas três estruturas ? Claramente não. Tome uma estrutura qualquer de uma linguagem. Sempre é possível acrescentar novos elementos no universo da estrutura, novos elementos nas relações, modificar as funções e os elementos associados às constantes, etc. Por exemplo, considere a linguagem $\mathcal{L}_\Delta$ da página 94. Esta linguagem utiliza um símbolo de predicado unário R, um símbolo de predicado binário S e uma função f. Constantes não são utilizadas. Podemos contruir um modelo $\mathfrak{A}$ com universo $|\mathfrak{A}| = \{a, b, c, d, e, f, g\}$, predicados $R^{\mathfrak{A}} = \{a, c, e, f\}$, $S^{\mathfrak{A}} = \{(a, b), (b, c), (c, d), (d, e), (e, a), (f, g), (g, f)\}$ e função $f^{\mathfrak{A}}$ tal que $f^{\mathfrak{A}}(x) = x$.

Temos alguns fatos muito **importantes** sobre estruturas:

1. qualquer linguagem possui infinitas estruturas;

2. nem todos os elementos de uma estrutura $\mathfrak{A}$ da linguagem $\mathcal{L}$ precisam estar associados a contantes de $\mathcal{L}$. A linguagem pode ter uma única constante c e $|\mathfrak{A}|$ pode ter dois, três ou mesmo infinitos elementos. É mais ou menos assim em nossa linguagem natural. Descrevemos os números naturais, por exemplo, sem nunca falar ou escrever todos os símbolos disponíveis para todos os números naturais (isto nem seria possível, dado que são infinitos). Durante toda a vida uma pessoa sempre utilizará um número finito de símbolos correspondentes aos números naturais;

3. um mesmo conjunto de elementos, como $\mathbb{N}$ pode pertencer a várias estruturas de maneiras diferente. Isto é verdade se as estruturas pertencem à mesma linguagem ou não. Por exemplo, considere uma linguagem com uma função f e constante c. Uma estrutura $\mathfrak{A}$ para uma

linguagem com função f e constante c cujo universo $|\mathfrak{A}|$ é $\mathbb{N}$ e que associa f a $+$ e c a 0. Uma outra estrutura pode ter universo $\{0, 1, 2\}$ e associar f a $\cdot$ e c a 2;

4. uma estrutura ou modelo não necessariamente utiliza todas as relações (predicados), funções e elementos do conceito que a origina. Por exemplo, os números naturais dão origem a diversas estruturas e modelos que não necessariamente utilizam todas as relações e funções que usualmente atribuímos aos números naturais. Por exemplo, uma estrutura baseada em $\mathbb{N}$ pode utilizar apenas a função f e uma constante c (associados a $+$ e 0, por exemplo). Outra estrutura pode utilizar apenas $<$ e nenhuma constante. E ainda outra estrutura pode utilizar $+$, $\cdot$, $<$ e 0, 1, 2, ...

Pela definição de logicamente válido, temos que todos os axiomas são logicamente válidos.

Como um exemplo de fórmula logicamente válida temos

- $\forall x\ A \longrightarrow \exists x\ A$, que quer dizer, em uma estrutura, que se todos os elementos do modelo satisfazem certas propriedades dadas por A, então existe um elemento que satisfaz aquelas propriedades;
- $A \longrightarrow A$, óbvio;
- $\exists x\, B \longrightarrow (\forall x\, A(x) \longrightarrow A(t))$, pois $(\forall x\, A(x) \longrightarrow A(t))$ é uma instância do axioma A4 (página 75) e portanto é logicamente válido. Pelas propriedades do conectivo $\longrightarrow$, a fórmula toda é logicamente válida.[13]

Definição 4.24. *Considere as fórmulas A e B de uma mesma linguagem $\mathcal{L}$. Dizemos que uma fórmula A* **logicamente implica** *uma fórmula B se, para toda estrutura de $\mathcal{L}$, toda seqüência que satisfaz A também satisfaz B.*

Por exemplo,

$$\forall x\ (\text{Herbívoro}(x) \longrightarrow \text{Devora}(x, g))$$

implica logicamente

$$\forall x\ (\neg\text{Devora}(x, g) \longrightarrow \neg\text{Herbívoro}(x))$$

Não é possível ter uma estrutura com uma seqüência que satisfaz a primeira fórmula sem satisfazer a segunda.

Definição 4.25. *Considere as fórmulas A e B de uma mesma linguagem $\mathcal{L}$. Dizemos que uma fórmula A* **é logicamente equivalente** *a uma fórmula B se A* **implica logicamente** *B e vice-versa.*

Na verdade, o exemplo dado acima é de fórmulas logicamente equivalentes:

[13] Se B ou $\neg A$ são logicamente válidos, então $A \longrightarrow B$ é logicamente válido.

$$\forall x\,(\text{Herbívoro}(x) \longrightarrow \text{Devora}(x, g))$$

é logicamente equivalente a

$$\forall x\,(\neg\text{Devora}(x, g) \longrightarrow \neg\text{Herbívoro}(x))$$

Ou seja,

$$\forall x\,(\text{Herbívoro}(x) \longrightarrow \text{Devora}(x, g)) \longleftrightarrow \forall x\,(\neg\text{Devora}(x, g) \longrightarrow \neg\text{Herbívoro}(x))$$

é logicamente válido.

Definição 4.26. *Considere a fórmula A e um conjunto Γ de fórmulas de uma linguagem $\mathcal{L}$. Uma fórmula A é uma* **conseqüência lógica** *de um conjunto de fórmulas Γ se A é* **verdadeira** *em todos os modelos de Γ. Isto é, se $\mathfrak{A} \vDash \Gamma$ para certo modelo $\mathfrak{A}$, então $\mathfrak{A} \vDash A$. Escrevemos $\Gamma \vDash A$ como na Definição 4.22.*

No exemplo do Zoológico da página 3, a fórmula

$$\forall x\,\forall y\,\forall z\,(\text{Herbívoro}(y) \wedge \text{Devora}(x, y) \longrightarrow \neg\text{Devora}(x, g)))$$

é uma conseqüência lógica do conjunto de axiomas para Zoo (página 68). Ela foi deduzida utilizando-se os axiomas e as regras da lógica. E nada mais. Esta fórmula, quando colocada na linguagem utilizada pelos axiomas abstratos para o Zoológico (página 69), é também válida para os outros modelos que compartilham estes axiomas, que são Fig e Num.

Por outro lado, a fórmula

$$\forall x\,(\text{Americano}(x) \longrightarrow \text{Carnívoro}(x))$$

apesar de ser válida nos modelos Zoo, Fig e Num, não é válida em todos os modelos dos axiomas. Poderíamos criar um modelos dos axiomas em que esta fórmula não seja válida. Já a fórmula

$$\exists x\,\exists y\,((\text{Planta}(x) \wedge \text{Planta}(y) \wedge \text{Devora}(\text{Efan}, x) \wedge \text{Devora}(\text{Efan}, y)) \longrightarrow \neg(x = y))$$

é verdadeira no modelo Zoo mas não o é nos modelos Fig e Num. Ela diz que há pelo menos dois pés de grama.

Para provar que $\Gamma \vDash A$, tome um modelo $\mathfrak{A}$ qualquer (genérico, não específico) de Γ e prove que $\mathfrak{A} \vDash A$. Por exemplo, provaremos que

$$\forall x_1\, Q(x_1) \vDash Q(x_2)$$

Seja $\mathfrak{A}$ um modelo de $\forall x_1\, Q(x_1)$. Então para todo $b \in |\mathfrak{A}|$, $\mathfrak{A} \vDash (\forall x_1\, Q(x_1))[b]$ sse para todo $b \in |\mathfrak{A}|$, $\mathfrak{A} \vDash Q(y)[b]$ sse para todo $b \in |\mathfrak{A}|$, $Q^{\mathfrak{A}}(b)$. Note que utilizamos b ao invés de $\vec{a}$, pois

neste caso $\vec{a}$ teria apenas um elemento. Partindo da fórmula $\forall x_1\, Q(x_1)$ conseguimos informações a respeito da relação $Q^{\mathfrak{A}}$ do modelo $\mathfrak{A}$ — em todos os modelos $\mathfrak{A}$ desta fórmula obrigatoriamente "para todo $b \in |\mathfrak{A}|$, $Q^{\mathfrak{A}}(b)$". Utilizando esta informação podemos provar que o modelo $\mathfrak{A}$ é também modelo de $Q(x_2)$.

Queremos provar que $\mathfrak{A} \vDash Q(x_2)$; isto é, para todo $b \in |\mathfrak{A}|$, $\mathfrak{A} \vDash Q(x_2)[b]$ sse para todo $b \in |\mathfrak{A}|$, $Q^{\mathfrak{A}}(b)$. Como esta última afirmação é verdade, $\mathfrak{A} \vDash Q(x_2)$.

Para provar que $\Gamma \nvDash A$, basta encontrar um modelo de Γ em que A é falsa. Por exemplo, suponha que queremos provar que

$$Q(c) \vDash \forall x\, Q(x)$$

Considere o modelo $\mathfrak{A}$ com universo $|\mathfrak{A}| = \{0, 1\}$ que associa à constante c da linguagem o elemento 0. O predicado Q é $\{0\}$. Então

$\mathfrak{A} \vDash Q(c)[b]$ para todo $b \in |\mathfrak{A}|$, pois $0 \in Q^{\mathfrak{A}}$ e b não é utilizado.

mas $1 \notin Q^{\mathfrak{A}}$ e portanto não é verdade que

$\mathfrak{A} \vDash (\forall x\, Q(x))[b]$ para todo $b \in |\mathfrak{A}|$. Isto significaria $0 \in Q^{\mathfrak{A}}$ e $1 \in Q^{\mathfrak{A}}$.

Exercícios de Treinamento

4.43. *Defina fórmula verdadeira e falsa em uma estrutura.*

4.44. *Faça uma estrutura e uma fórmula tal que a fórmula não seja nem verdadeira nem falsa na estrutura.*

4.45. *Considere modelos $\mathfrak{A}_i$, $i \in \mathbb{N}$ e fórmulas B, C e D tais que:*

(a) $\mathfrak{A}_i \vDash B$ *para i par;*

(b) $\mathfrak{A}_i \vDash C$ *para i primo;*

(c) $\mathfrak{A}_i \vDash D$ *para i ímpar.*

Verifique se as afirmações abaixo são verdadeiras.

(a) $\mathfrak{A}_2 \vDash \{B, C\}$

(b) $\mathfrak{A}_i \vDash \{C, D\}$ *para i primo;*

(c) $\mathfrak{A}_i \vDash \{B, C, D\}$ *para $i \in \mathbb{N}$;*

Justifique.

4.46. *Prove que o axioma $A \longrightarrow (B \longrightarrow A)$ é verdadeiro em qualquer estrutura.*

4.47. *Represente graficamente as fórmulas logicamente válidas de uma linguagem (como feita na apostila). Mostre onde estão estas fórmulas no desenho.*

4.48. *Pode uma fórmula que represente uma informação específica de uma estrutura ser logicamente válida ?*

4.49. *Faça uma fórmula A de uma linguagem $\mathcal{L}$ tal que todos os modelos de A tenham apenas um elemento.*

4.50. *Faça uma fórmula A de uma linguagem $\mathcal{L}$ tal que todos os modelos de A tenham exatamente dois elementos.*

4.51. *Uma estrutura de uma linguagem $\mathcal{L}$ pode associar duas constantes da linguagem a um mesmo elemento do universo da estrutura ? Dê um exemplo.*

4.52. *Uma estrutura de uma linguagem $\mathcal{L}$ pode associar dois símbolos de função da linguagem a uma mesma função da estrutura ? Dê um exemplo.*

4.53. *(i4d3) Encontre modelos em que as fórmulas abaixo não são verdadeiras. Então estas fórmulas não são logicamente válidas.*[14]

(a) $\exists x\, P(x) \longrightarrow P(c)$, *onde c é uma constante da linguagem;*

(b) $\exists x\, \exists y\, \neg(x = y) \longrightarrow \neg(f(x) = f(y))$

(c) $\forall x\, (P(x) \vee Q(x)) \longrightarrow (\forall x\, P(x) \vee \forall x\, Q(x))$

(d) $(\exists x\, P(x) \wedge \exists x\, Q(x)) \longrightarrow \exists x\, (P(x) \wedge Q(x))$

(e) $\forall x\, \neg(x = c)$, *onde c é uma constante da linguagem;*

(f) $\forall x\, \exists y\, (f(x) = f(y) \longrightarrow (x = y))$

(g) $\forall x\, P(x) \longrightarrow \exists y\, (Q(y) \longrightarrow \neg P(y))$

4.54. *(i4d2) Considere a linguagem $\mathcal{L}$ com os símbolos de predicado $D(x,\, y)$, $A(x)$ e $P(x,\, y)$ e a constante c. Suponha que um modelo $\mathfrak{A}$ interpreta estes símbolos como*

- *$D(x,\, y)$, x é uma disciplina mais difícil do que y;*
- *$A(x)$, x é uma disciplina que possui uma apostila;*
- *$P(x,\, y)$, as provas de x são mais difíceis do que as provas de y;*
- *c é a disciplina "Introdução à Lógica".*

Considere que $|\mathfrak{A}|$ seja o conjunto de todas as disciplinas.

Faça fórmulas na linguagem $\mathcal{L}$ que representem as frases seguintes quando interpretadas no modelo $\mathfrak{A}$:

(a) *qualquer disciplina é mais difícil do que Introdução à Lógica;*

[14] Se fossem seriam verdadeiras em todos os modelos. Evidentemente, assume-se que cada fórmula está em uma linguagem $\mathcal{L}$ e que o modelo encontrado é estrutura de $\mathcal{L}$.

(b) *há uma disciplina que é mais difícil do que todas as demais;*

(c) *se as provas de uma disciplina D_1 são mais difíceis do as da disciplina D_2, então a disciplina D_1 é mais difícil do que D_2. Os símbolos D_1 e D_2 representam disciplinas quaisquer, não constantes da linguagem ou do universo do modelo;*

(d) *se uma disciplina tem apostila, então ela é a mais fácil de todas;*

(e) *existe uma disciplina que é mais fácil que todas as outras.*

(f) *existe uma única disciplina;*

(g) *existe apenas uma outra disciplina além de Introdução à Lógica;*

(h) *se uma disciplina é mais difícil do que todas as outras, então esta disciplina possui provas mais difíceis do que todas as outras;*

(i) *se uma disciplina é mais difícil do que todas as outras, então existe uma disciplina que possui provas mais fáceis do que esta;*

(j) *Introdução à Lógica possui provas mais fáceis do que todas as outras disciplinas;*

(k) *se uma disciplina é mais difícil do que alguma outra, então esta disciplina não é Introdução à Lógica.*

Cuidado com as relações: $D(x,\, x)$ é sempre falso, assim como $P(x,\, x)$.

4.55. *(i4d2) Usando os dados da questão anterior, escreva o que significam as fórmulas abaixo.*

(a) $\forall x\, (A(x) \longrightarrow \exists y\, D(y,\, x))$

(b) $\forall x\, (A(x) \longrightarrow (\exists y\, P(y, x) \vee \exists y\, D(y,\, x)))$

(c) $\forall x\, (x = c \vee D(x,\, c) \vee P(x,\, c))$

(d) $\forall x\, \neg D(x,\, y) \longrightarrow y = c$

(e) $\exists x\, (\forall y\, (\neg(x = y) \longrightarrow D(x,\, y)) \longrightarrow \neg(x = c))$

4.56. *(i4d3) Considere uma linguagem com o símbolo de função binário f e as constantes a e b. Uma estrutura $\mathfrak{A}$ para esta linguagem associa a constante "a" a 0, a constante "b" a 1 e f à seguinte função*

x	y	$f^{\mathfrak{A}}(x,\, y)$
0	*0*	*0*
0	*1*	*1*
1	*0*	*1*
1	*1*	*0*

Considere $|\mathfrak{A}| = \{0,\, 1\}$.

Verifique quais das fórmulas abaixo são satisfeitas neste modelo

1. $f(a,\, b) = f(b,\, a)$

2. $\forall x\, \exists y\, f(x,\, y) = a$

3. $\exists x\, \forall y\, f(x,\, y) = a$

4. $\forall x\, \forall y\, f(x,\, f(y,\, x)) = x$

5. $\exists x\, \forall y\, f(f(y,\, x),\, f(y,\, y)) = x$

4.57. *Considere uma linguagem com os símbolos usuais da Aritmética:* $+$, $-$, $\cdot$, $<$, 0 *e* 1. *A linguagem possui apenas estes símbolos — não possui* 2 *como símbolo, por exemplo, nem* $/$ *(de divisão). Naturalmente, assuma que uma estrutura para esta linguagem é a Aritmética usual que possui todos os números naturais. Faça fórmulas que correspondam aos seguintes predicados:*

(a) $M(x,\, y)$, *x é maior do que y;*

(b) $D(x,\, y)$, *x divide y;*

(c) $R(x,\, y,\, z)$, *z é o resultado da divisão de x por y;*

(d) $P(x)$, *x é um número primo;*

(e) $E(x,\, y,\, z)$, $x^y = z$;

(f) $Q(x)$, *x é um primo da forma* $2^{2^n} - 1$.

Escrevemos acima, em cada item, o nome do símbolo de predicado, seus possíveis argumentos e a sua interpretação na Aritmética.

4.58. *(i4d2) Utilizando a linguagem e os símbolos de predicado do exercício anterior, faça fórmulas que, quando interpretadas na Aritmética, representem as seguintes frases*

(a) *dado um número primo, existe outro primo maior do que ele (existem infinitos primos);*

(b) *existem* x, y *e* z *tal que* $x^2 + y^2 = z^2$;

(c) *não existem* x, y *e* z *e* $n > 2$ *tal que* $x^n + y^n = z^n$;

(d) *se* $x > y$ *e* $y > z$, *então* $x > z$;

(e) x *somado a qualquer número diferente de* 0 *é maior do que* x;

(f) *para todo* x, x *é menor do que* $x + 1$;

(g) *Se* $x + 1 = y + 1$, *então* $x = y$.

4.59. *(i3d3) Um conjunto de sentenças é satisfazível se existe um modelo para elas e neste modelo há pelo menos uma seqüência que satisfaça a todas elas. Verifique se as sentenças abaixo são satisfazíveis.*

(a) $\{\forall x\, \neg P(x),\, Q(x) \longrightarrow P(x)\}$

(b) $\{\forall x\, \exists y\, P(x,\, y),\, \neg\forall x\, P(x,\, x)\}$

(c) $\{\exists x\, P(x),\, \neg P(c),\, \forall x\, (P(f(x))\longrightarrow Q(x,\, c))\}$

(d) $\{\neg\exists x\, P(x) \vee \exists y\, Q(x),\, \neg\forall x\, (P(x) \wedge Q(x))\}$

(e) $\{\forall x\, (P(x)\longrightarrow Q(x)),\, P(c),\, Q(a),\, \neg\exists x\, (Q(x)\longrightarrow P(x))\}$

4.5 Relação entre Sintaxe e Semântica

Esta Seção apresenta as relações entre a sintaxe e a semântica. Uma teoria de primeira ordem possui os axiomas A1-A7 da Seção 4.3 que são verdadeiros em qualquer modelo. Contudo, estes axiomas não são suficientes para expressar as verdades de domínios específicos. Por exemplo, não se pode provar que $1+0=1$ utilizando-se apenas estes axiomas. Para tanto devem ser feitos axiomas específicos para a Aritmética (ou qualquer outra parte da Matemática ou do "mundo real"). O leitor interessado pode consultar a página 113 em que são apresentados, em uma linguagem de primeira ordem, os axiomas que caracterizam a Aritmética.

Os axiomas específicos para certo domínio como a Aritmética, teoria dos conjuntos, teoria dos grupos, geometria euclidiana, etc **não** serão considerados, neste texto, "axiomas" de uma teoria de primeira ordem. Eles serão colocados em conjuntos de fórmulas Γ, o que para todos os efeitos práticos é equivalente a considerá-los axiomas de uma teoria de primeira ordem. Assim temos um conjunto Γ_P de fórmulas que caracterizam a Aritmética, um conjunto $\mathcal{L}_G$ para a teoria dos grupos e assim por diante.

Proposição 4.4. *Se A e $A\longrightarrow B$ são verdadeiros em um modelo $\mathfrak{A}$, então B é verdadeiro neste modelo. Logo, se A e $A\longrightarrow B$ são logicamente válidos, B é logicamente válido.*

Em resumo, a regra Modus Pones toma duas fórmulas logicamente verdadeiras e produz uma fórmula logicamente verdadeira, como seria esperado. A primeira frase desta proposição, em símbolos, é

$$\text{Se } \mathfrak{A} \vDash A \text{ e } \mathfrak{A} \vDash A\longrightarrow B, \text{ então } \mathfrak{A} \vDash B$$

Proposição 4.5. *A é verdadeiro em um modelo $\mathfrak{A}$ se e somente se $\forall x\, A$ é verdadeiro em $\mathfrak{A}$. Logo, A é verdadeiro em um modelo $\mathfrak{A}$ se e somente se o fechamento de A é verdadeiro em $\mathfrak{A}$.*

A regra Gen produz uma fórmula verdadeira em um modelo a partir de outra fórmula verdadeira neste modelo. As regras fazem parte da sintaxe da linguagem, não são utilizadas na semântica, o estudo dos modelos, que é o objeto de estudo desta seção. Contudo, é absolutamente fundamental que a sintaxe e semântica concordem entre si. As duas proposições acima nos garantem que pelo menos as regras de dedução só produzem verdades a partir de verdades.

Teorema 4.4. *(Correção) Considere uma linguagem $\mathcal{L}$. Os teoremas em $\mathcal{L}$ das teorias de primeira ordem são verdadeiros em qualquer estrutura de $\mathcal{L}$ (são logicamente válidos).*

Prova. Os axiomas são verdadeiros em qualquer estrutura (logicamente válidos) pela proposição 4.3 e as regras de dedução MP e Gen preservam a validade pelas proposições acima. Como todo teorema é deduzido a partir dos axiomas e aplicações de MP e Gen, todo teorema é verdadeiro em qualquer estrutura.

□

Definição 4.27. *Um conjunto de fórmula* Γ *é* **consistente** *se existe uma fórmula* A *tal que* $\Gamma \nvdash A$. *Isto é, não se pode deduzir qualquer fórmula a partir de* Γ.

Em outras palavras, dada uma fórmula qualquer A, se $\vdash A$ então $\nvdash \neg A$. O que significa que, se $\vdash \neg A$, então $\nvdash A$. Mas isto não significa que se $\nvdash A$ então $\vdash \neg A$, pois neste caso se A não é teorema, obrigatoriamente $\neg A$ é teorema. Isto só seria verdade se tivéssemos um "sse": $\vdash A$ sse $\nvdash \neg A$. Conclui-se que um conjunto de fórmulas consistente não necessariamente é completo: pode existir uma fórmula A tal que $\nvdash A$ e $\nvdash \neg A$.

Teorema 4.5. *(Teorema da Completude de Gödel, 1930) Um conjunto de fórmulas fechadas (sentenças)* Γ *em uma linguagem* $\mathcal{L}$ *é consistente se e somente se* Γ *tem modelo em* $\mathcal{L}$.

Prova. ($\Longleftarrow$) Seja $\mathfrak{A}$ um modelo de Γ e suponha que Γ seja inconsistente. Então existe uma fórmula A tal que $\Gamma \vdash A$ e $\Gamma \vdash \neg A$. Pelo teorema da Correção (4.4), A e $\neg A$ são verdadeiros em qualquer estrutura de $\mathcal{L}$. Como $\mathfrak{A}$ é estrutura de $\mathcal{L}$, $\mathfrak{A} \vDash A$ e $\mathfrak{A} \vDash \neg A$. Contradição, pois $\mathfrak{A} \vDash \neg A$ implica em $\mathfrak{A} \nvDash A$.

($\Longrightarrow$) Esta prova não será feita, mas faremos um comentário a respeito da prova. O problema é encontrar um modelo tal que todas as fórmulas de Γ sejam verdadeiras nele. Gödel construiu um modelo partindo da própria sintaxe da linguagem, mais precisamente, dos teoremas que podem ser obtidos tomando-se Γ como hipótese. Os predicados do modelo são relações construídas a partir dos teoremas que podem ser obtidos de Γ; isto é um predicado $R^{\mathfrak{A}}$ do modelo construído possui certos elementos de acordo com quais fórmulas são teoremas. O mesmo se aplica a funções do modelo e elementos associados às constantes. Gödel projeta, literalmente, estes predicados, funções e elementos do universo do modelo de tal forma que todas as fórmulas de Γ sejam verdadeiras no modelo construído. □

O teorema da completude pode também ser colocado em uma outra forma, dada abaixo, que é equivalente à forma acima.

Teorema 4.6. *(Teorema da Completude de Gödel, 1930) Dado um cálculo de predicados T que utiliza uma linguagem* $\mathcal{L}$, *uma fórmula fechada (sentença) de* $\mathcal{L}$ *logicamente válida é um teorema de T.*

Isto significa que um cálculo de predicados T que utiliza uma linguagem $\mathcal{L}$ captura precisamente as fórmulas logicamente válidas de $\mathcal{L}$. Ou seja, tome todas as estruturas de $\mathcal{L}$. Há algumas fórmulas que são válidas em todas as estruturas, o que inclui todos os axiomas das teorias de primeira ordem (página 75). Este teorema diz que todas estas fórmulas são teoremas de T. Em símbolos, temos

Se $\mathfrak{A} \vDash A$ para toda estrutura $\mathfrak{A}$ de $\mathcal{L}$, então $\vdash A$

O teorema da correção dado acima garante o contrário, que todos os teoremas de T são verdadeiros em todas as estruturas. Em símbolos, temos

Se $\vdash A$, então $\mathfrak{A} \vDash A$ para toda estrutura $\mathfrak{A}$ de $\mathcal{L}$.

É sempre importante lembrar que o cálculo de predicados T é dado na linguagem $\mathcal{L}$ e que a definição de fórmula *logicamente válida* é feita sobre as estruturas de $\mathcal{L}$.

Teorema 4.7. *(Teorema da Completude de Gödel, 1930) Dado um conjunto de fórmulas fechadas* Γ *e uma fórmula fechada* A *em uma linguagem* $\mathcal{L}$*, então* $\Gamma \vDash A$ *se e somente se* $\Gamma \vdash A$*.*

Este teorema é deduzido a partir do teorema 4.6 e do teorema 4.4 da Correção.

Há várias maneiras de se provar que uma fórmula é logicamente válida, verdadeira em todos os modelos da sua linguagem:

1. fazendo uma dedução formal através dos axiomas. Pelo teoremas da Completude e Correção, uma fórmula é teorema sse é logicamente verdadeira;

2. fazendo uma prova que utiliza a definição de satisfabilidade (Definição 4.17);

3. utilizando um tablô (a ser visto nas próximas seções).

Para provar que uma fórmula não é logicamente válida, deve-se encontrar um modelo em que ela não é verdadeira.

Como exemplo, provaremos que $P(c) \longrightarrow \forall x\, P(x)$ não é logicamente válida. Considere o modelo $\mathfrak{A}$ com universo $|\mathfrak{A}| = \{0, 1\}$ e onde $P^{\mathfrak{A}} = \{0\}$. A constante c da linguagem é associada a 0. Então $P(c)$ é verdadeiro mas $\forall x\, P(x)$ não é. Ou seja, qualquer que seja $\vec{a}$, seqüência em $|\mathfrak{A}|$, $\mathfrak{A} \vDash P(c)[\vec{a}]$ mas para nenhuma $\vec{a}$, $|\mathfrak{A}| \vDash (\forall x\, P(x))[\vec{a}]$. Isto é, $\mathfrak{A} \vDash P(c)$ mas $\mathfrak{A} \nvDash \forall x\, P(x)$. Em palavras, $P(c)$ é verdadeiro no modelo $\mathfrak{A}$ e $\forall x\, P(x)$ é falso, o que implica que $P(c) \longrightarrow \forall x\, P(x)$ é falso neste modelo. Nesta prova utilizamos a definição 4.17 de satisfabilidade e a definição 4.18 de verdade em uma estrutura.

Uma fórmula que não é logicamente válida também não é um teorema (Teorema da Correção 4.4). Mas provar que uma fórmula não é teorema é mais complexo. Se a teoria utilizada for **completa**; isto é, ou $\vdash A$ ou $\vdash \neg A$ para toda fórmula A sem variáveis livres, então pode-se provar que A não é teorema encontrando-se uma prova formal para $\neg A$.

Uma pergunta interessante sobre modelos é se qualquer conjunto de fórmulas Γ possui modelos de qualquer tamanho. Ou se qualquer conjunto de fórmulas admite um modelo de tamanho infinito. A resposta é não para ambos os casos. Se Γ contém unicamente a fórmula

$$A =_{def} \forall x\, \forall y\, (x = y)$$

então todos os modelos de Γ contém apenas um único elemento. Esta conclusão está baseada no fato de que o símbolo "=" que aparece na fórmula acima é interpretado como "=" em um modelo qualquer de Γ e este igual possui a seguinte propriedade, descrita na Definição 4.15 de estrutura: b e c representam elementos iguais no conjunto universo do modelo (que é uma estrutura) se e somente se $b = c$, onde este símbolo "=" é o igual no modelo. Assim, não podemos ter uma estrutura (ou modelo) composta por todos os triângulos possíveis que satisfaçam a fórmula A dada acima. O universo deste modelo teria todos os possíveis triângulos de todos os tamanhos e todos os ângulos. Este modelo não satisfaz a fórmula A acima porque um triângulo equilátero

seria considerado diferente de um triângulo retângulo pela relação de igualdade "=".

Regras Semânticas

As regras extras para teorias de primeira ordem dadas na página 82 são regras sintáticas. Elas dizem que se tal fórmula é teorema, outra fórmula é teorema. Isto é, se existe uma prova para tal fórmula, existe uma prova formal para a outra fórmula também. Pelo Teorema da Completude 4.7, tomando-se $\Gamma = \emptyset$, temos

$\vDash A$ se e somente se $\vdash A$

Isto significa que, se $\vDash A$, então $\vDash A \vee B$. Vejamos a prova:

$\vDash A$ sse (Teorema da Completude) $\vdash A$ então (Lema 4.4) $\vdash A \vee B$ sse (Teorema da Completude) $\vDash A \vee B$

Logo, se $\vDash A$, então $\vDash A \vee B$. Observe que nesta última relação, temos um "se-então". Não poderia ser "se e somente se"? Não, pois na dedução acima temos muitos "sse" mas no meio deles há um "se-então" que força a direção de dedução da esquerda para a direita somente. Em símbolos, temos algo como

$$C_1 \Longleftrightarrow C_2 \Longleftrightarrow C_3 \Longrightarrow C_4 \Longleftrightarrow C_5 \Longleftrightarrow C_6$$

e portanto temos $C_1 \Longrightarrow C_6$, onde $\Longleftrightarrow$ é um símbolo para "sse" e $\Longrightarrow$ representa "se-então".

Nenhum modelo é contraditório. Considerando B fechada, sem variáveis livres, se $\vDash \neg B$, obrigatoriamente $\nvDash B$ e, se $\nvDash \neg B$, então $\vDash B$. Isto é, não pode existir um modelo tal que $\vDash B \wedge \neg B$. Este fato pode ser utilizado para fazer uma prova por absurdo. Para provar que A é uma fórmula logicamente válida, suponha que exista um modelo $\mathfrak{A}$ para $\neg A$. Se conseguimos obter que $\neg B \wedge B$ é verdadeira em $\mathfrak{A}$, então A é logicamente válida pois este modelo $\mathfrak{A}$ não pode existir. Ou seja, em nenhum modelo temos $\mathfrak{A} \vDash \neg A$ e portanto $\mathfrak{B} \vDash A$ em qualquer modelo $\mathfrak{B}$. De onde pode-se concluir $\vDash A$, A é logicamente válida. Em símbolos, se $\mathfrak{A} \vDash \neg B \wedge B$ (ou $\mathfrak{A} \vDash \neg B$ e $\mathfrak{A} \vDash B$), então A é logicamente válida.

As regras semânticas são apresentadas abaixo, em sua maioria, sem provas.

Se $\vDash A$, então $\vDash B \vee A$

$\vDash A \vee B$ sse $\vDash B \vee A$

$\vDash A \wedge B$ sse $\vDash A$ e $\vDash B$

$\vDash A \wedge B$ sse $\vDash B \wedge A$

Se $\vDash A \vee B$, $\vDash A \longrightarrow C$ e $\vDash B \longrightarrow C$, então $\vDash C$

Se $\vDash A \vee B$ e $\vDash \neg A$, então $\vDash B$

Se $\vDash A \longrightarrow B$ e $\vDash B \longrightarrow C$, então $\vDash A \longrightarrow C$. A prova é:

$\vDash A \longrightarrow B$ e $\vDash B \longrightarrow C$ sse (Teorema da Completude) $\vdash A \longrightarrow B$ e $\vdash B \longrightarrow C$, então (Lema 4.11) $\vdash A \longrightarrow C$ sse (Teorema da Completude) $\vDash A \longrightarrow C$. Logo, Se $\vDash A \longrightarrow B$ e $\vDash B \longrightarrow C$, então $\vDash A \longrightarrow C$.

Se $\vDash A \longleftrightarrow B$ e $\vDash B \longleftrightarrow C$, então $\vDash A \longleftrightarrow C$

$\vDash A \longleftrightarrow B$ sse $\vDash A \longrightarrow B$ e $\vDash B \longrightarrow A$

Se $\vDash A \longrightarrow B$ e x não é livre em A, então $\vDash A \longrightarrow \forall x\, B$

Se $\vDash A$ e A' é $A(t_1, t_2, ...t_n)$, onde t_i substitui a variável livre x_i de A, então $\vDash A'$

Se $\vDash A \longrightarrow B$, então $\vDash \exists x\, A \longrightarrow \exists x\, B$ e $\vDash \forall x\, A \longrightarrow \forall x\, B$

Se uma fórmula A possui como variáveis livres x_1, x_2, ..., x_n, então

a) $\vDash A(t_1, t_2, ...t_n) \longrightarrow \exists x_1\, \exists x_2\, \ldots \exists x_n\, A$

b) $\vDash \forall x_1\, \forall x_2\, \ldots \forall x_n\, A \longrightarrow A(t_1, t_2, ...t_n)$

onde t_i substitui x_i em $A(t_1, t_2, ...t_n)$.

Se $\vDash A$ é o fechamento de A, então $\vDash A$ sse $\vDash A'$

Note que uma fórmula A é uma instância de uma tautologia se e somente se

$$\vDash A$$

Então, por exemplo,

$$\vDash A \longrightarrow (B \longrightarrow A)$$

Exercícios de Treinamento

4.60. *Explique o que é sintaxe e o que é semântica na lógica de primeira ordem.*

4.61. *Enuncie o teorema da Correção para lógicas de primeira ordem.*

4.62. *Faça um conjunto de fórmulas Γ inconsistente sem que este conjunto contenha fórmulas da forma A e $\neg A$.*

4.6 Alguns Exemplos de Modelos

Grupos

Um grupo é um conjunto G e uma operação $\circ$ tal que

1. para todo x, $y \in G$, $x \circ y \in G$. A operação $\circ$ é uma função
$$\circ : G^2 \longrightarrow G$$
2. para todo x, y, $z \in G$,
$$(x \circ y) \circ z = x \circ (y \circ z)$$
3. existe um elemento $e \in G$ tal que para todo $x \in G$, $x \circ e = x$. O elemento e é chamado de **elemento neutro** ou **identidade** do grupo;

4. para cada $x \in G$, existe um elemento $y \in G$ tal que $x \circ y = e$. O elemento y é chamado de **inverso** de x e é escrito como x^{-1}.

A linguagem $\mathcal{L}_G$ para grupos utiliza o símbolo de função $\circ$ de aridade dois e uma constante e. Os axiomas que descrevem grupos são:

A1 $(x \circ y) \circ z = x \circ (y \circ z)$

A2 $x \circ e = x$

A3 $\exists y\, (x \circ y = e)$

Neste texto, não se utiliza axiomas não lógicos em teorias de primeira ordem. Mas isto não tem importância. Definimos que as fórmulas acima, os axiomas de grupos, compõem um conjunto Γ_G de fórmulas que caracterizam grupos. Então uma fórmula $e \circ x = x$ é teorema se $\Gamma_G \vdash e \circ x = x$.

Algumas observações a respeito destes axiomas são necessárias:

1. não é necessário colocar nenhum axioma para a especificação "para todo x, $y \in G$, $x \circ y \in G$". Na definição de uma estrutura (e de modelo), toda função toma elementos do universo e produz um elemento do mesmo universo;

2. por causa da regra Gen, não é necessário colocar quantificadores universais ($\forall$) no início dos axiomas;

3. pode-se escrever A1 na notação usual de função: $\circ(\circ(x, y), z) = \circ(x, \circ(y, z))$.

Um grupo G de n elementos será chamado de grupo de ordem n. Não necessariamente um grupo é comutativo; isto é, $x \circ y = y \circ x$ para todo x, $y \in G$.

Há inúmeros exemplos de grupos na Matemática. Mostraremos alguns deles:

1. o conjunto $\mathbb{Z}$ com a operação $+$ e elemento neutro 0;

2. os números racionais sem o zero, $\mathbb{Q} - \{0\}$, com a operação $\cdot$ de multiplicação e 1 como elemento neutro;

3. o conjunto $\mathbb{R}$ dos números reais com a operação $+$ e 0 como elemento neutro;

4. o conjunto $\{0,\, 1,\, 2,\, \ldots n-1\}$ com a operação $\tilde{+}$ definida como

   $a\tilde{+}0 = a$

   $a\tilde{+}(b\tilde{+}1) = (a+b)+1$ se $a+b < n$

   $a\tilde{+}(b\tilde{+}1) = (a+b)-n$ se $a+b >= n$

   onde $+$ é a soma usual em $\mathbb{N}$ e $\tilde{+}$ é a operação deste grupo, chamado de $\mathbb{Z}_n$. Normalmente utilizaremos o símbolo de soma, $+$, para $\mathbb{Z}_n$.

5. o conjunto das matrizes quadradas de ordem 3 não singulares (determinante diferente de zero, que possuem inversa) com a operação de multiplicação de matrizes. O elemento neutro é a matriz identidade de ordem 3.

Os Números Naturais

A Aritmética pode ser caracterizada pelos axiomas de Peano. A descrição informal é

A1 0 é um número natural;

A2 cada número natural x tem um sucessor denominado x';

A3 $0 \neq x'$ para qualquer natural x;

A4 se $x' = y'$, então $x = y$;

A5 seja $P(x)$ uma propriedade sobre o número natural x. Então se $P(0)$ e se $P(x)$ implicar $P(x')$, para x qualquer, então para qualquer número natural y, $P(y)$. Este é o princípio da indução.

A versão em uma linguagem de primeira ordem dos axiomas de Peano é dada abaixo. Utiliza-se uma linguagem $\mathcal{L}_P$ com uma única constante 0, os símbolos de função $+$, $\cdot$ e $'$. Este último símbolo será utilizado como x' e, *quando interpretado* nos números naturais, significa o sucessor de x, que é $x + 1$.

F1 $x = y \longrightarrow (x = z \longrightarrow y = z)$

F2 $x = y \longrightarrow (x' = y')$

F3 $0 \neq x'$, que é o mesmo que $\neg(0 = x')$

F4 $x' = y' \longrightarrow x = y$

F5 $x + 0 = x$

F6 $x + y' = (x + y)'$

F7 $x \cdot 0 = 0$

F8 $x \cdot (y') = (x \cdot y) + x$

F9 $(A(0) \wedge (\forall x\, (A(x) \longrightarrow A(x')))) \longrightarrow \forall x\, A(x)$ para qualquer fórmula A.

Considere que Γ_P seja o conjunto de fórmulas **F1-F9**. Supõe-se que estas fórmulas sejam suficientes para caracterizar a Aritmética, que é um modelo $\mathfrak{B}$ no qual valem as "verdades" usuais sobre números, como $1 + 1 = 2$, $\forall x\, \exists y\, (x + 0' = y)$ e $(x + 0'') \cdot y = (x \cdot y) + 0'' \cdot y$. Este modelo têm as seguintes características:

1. $|\mathfrak{B}| = \{0,\ 1,\ 2,\ 3,\ \ldots\}$, o universo do modelo;

2. o símbolo de função unária $'$ da linguagem deve ser interpretado pela função $s(x) = x + 1$ do modelo $\mathfrak{B}$, onde $+$ é o símbolo utilizado na Aritmética usual, não o símbolo de função da linguagem $\mathcal{L}_P$. Assim, se representarmos uma função $s(x)$ pelo seu gráfico; isto é, pelo conjunto dos pares $(x,\ s(x))$ para todo x do domínio, temos que

$$Grafico(s) = \{(0,\ 1),\ (1,\ 2),\ (2,\ 3),\ \dots\}$$

3. o símbolo de função binária $+$ da linguagem deve ser interpretado pela função $+$ do modelo; isto é, pelo $+$ da Aritmética. O gráfico de $+$ possui triplas da forma $(a,\ b,\ (a + b))$. Assim,

$$Grafico(+) = \{(0,\ 0,\ 0),\ (0,\ 1,\ 1),\ \dots (1,\ 0,\ 1),\ (1,\ 1,\ 2),\ \dots,\ \dots\}$$

4. o símbolo de função binária $\centerdot$ da linguagem deve ser interpretado pela função $\centerdot$ do modelo; isto é, pelo $\centerdot$ da Aritmética (multiplicação). O gráfico de $\centerdot$ possui triplas da forma $(a, b, (a \centerdot b))$. Assim,

$$Grafico(\centerdot) = \{(0,\ 0,\ 0),\ (0,\ 1,\ 0),\ \dots (1,\ 0,\ 0),\ (1,\ 1,\ 1),\ \dots,\ \dots\}$$

5. a constante 0 da linguagem corresponde ao elemento 0 de $|\mathfrak{B}|$.

Note que o que chamamos de Aritmética usual, o modelo $\mathfrak{B}$, é composto pelos conjuntos acima mais as regras da lógica clássica. Então assumimos, não surpreendentemente, as regras da lógica clássica ao nos referirmos às verdades da Aritmética.

Os conjuntos acima, tanto $|\mathfrak{B}|$ como os gráficos das funções contêm "..." significando que eles não foram completados. Isto era esperado, já que os conjuntos são mesmo infinitos. Mas como então confirmar que as fórmulas do conjunto Γ_P são verdadeiras no modelo $\mathfrak{B}$? Só existe uma possibilidade: utilizar informações que conhecemos intuitivamente e que não estão escritas completamente no papel. Não podemos escrever completamente o gráfico da função $+$, que é infinito, mas através de palavras, intuitivamente, conhecemos todas as caracteristicas que esta função deve ter. Então pode-se provar um meta-teorema[15] que afirma que as fórmulas de Γ_P são verdadeiras na Aritmética, o modelo $\mathfrak{B}$.

A discussão acima nos remete à questão "O que é uma prova na Matemática" ? Certamente não é uma prova sintática, obtida pelo usos de axiomas e aplicação das regras MP e Gen. Mas o que é então ? Para responder a esta pergunta, devemos assumir que, em uma prova Matemática, utiliza-se um conjunto de axiomas e outras fórmulas já provadas anteriormente. Por exemplo, em uma prova sobre números inteiros, utiliza-se os axiomas acima, mesmo que este fato não esteja explícito na prova. O desenvolvimento da prova se faz principalmente por argumentos semânticos — relembre a definição de verdade em uma estrutura (página 97) e satisfação (Definição 4.17 na página 93). A definição de verdade utiliza a definição de satisfação. Contudo, qualquer argumento pode ser utilizado em uma prova matemática desde que os matemáticos concordem com a sua utilização — desde que ela seja razoável. O que é ou não razoável é uma discussão filosófica que está fora do escopo deste livro.

[15] É meta-teorema porque é um teorema sobre a lógica de primeira ordem, não um teorema da teoria de primeira ordem que utiliza a linguagem $\mathcal{L}_P$ e as fórmulas de Γ_P.

De qualquer forma, espera-se que qualquer prova matemática possa ser convertida em uma prova sintática formal (Definição 4.2). Em geral, esta prova pode ser feita em uma lógica de primeira ordem. Contudo, nem sempre isto é possível — pode ser absolutamente necessário utilizar uma lógica de segunda ordem. Neste tipo de lógica existem algumas variáveis que, quando interpretadas em uma estrutura, referem-se a conjuntos de elementos da estrutura. Por exemplo, considere a fórmula

$$\forall X\, \exists x\, (x \in X \wedge \forall y\, ((y \in X \wedge \neg(x = y)) \longrightarrow (y < x)))$$

interpretada em um modelo $\mathfrak{A}$ da Aritmética que utiliza o símbolo $<$ no qual $b < c$ significa "b é menor do que c". Se esta fórmula é verdadeira neste modelo, então qualquer conjunto de elementos de $|\mathfrak{A}|$ possui um elemento que é menor do que todos os outros.

Apesar da lógica de segunda ordem ser necessária em alguns casos, praticamente toda a Matemática pode ser expressa na lógica de primeira ordem.

A Teoria dos Conjuntos

A teoria de conjuntos Zermelo-Fraenkel , (ZF) utiliza seis axiomas [1]. Estes axiomas mais o axioma da escolha (A7 abaixo) constituem-se na teoria ZFC que constitui base para formalizar toda a Matemática. A partir desta teoria pode-se deduzir todos os axiomas da Aritmética dados no item anterior, desde que os símbolos $+$, $\cdot$, $'$ e 0 sejam representados de uma certa forma (não mostrada) utilizando-se somente o símbolo $\in$. Aliás, este é o único predicado utilizado pelos axiomas. Funções e constantes não são necessários. Contudo, para facilitar o entendimento dos axiomas, utilizaremos o conjunto vazio, $\emptyset$, como constante. Nos comentários a respeito das fórmulas, interpretamos as fórmulas dadas em um modelo da teoria dos conjuntos.

A1 $\forall z\, (z \in x \longleftrightarrow z \in y) \longrightarrow (x = y)$, axioma da extensionalidade. Este axioma significa o seguinte: se x e y têm os mesmos elementos, eles são iguais;

A2 $F_\varphi \longrightarrow (\exists b\, \forall y\, (y \in b \longleftrightarrow \exists x\, (x \in a \wedge \varphi(x, y))))$, axioma da substituição. F_φ é a fórmula $\forall x\, \forall y\, \forall z\, (\varphi(x,\, y) \wedge \varphi(x,\, z) \longrightarrow (y = z))$. Este axioma garante que podemos construir um conjunto b que seja a imagem de uma fórmula φ que é usada como uma função tendo a como domínio;

A3 $\exists y\, \forall x\, (x \in y \longleftrightarrow \forall z\, (z \in x \longrightarrow z \in a))$, axioma das partes. Este axioma garante que existe o conjunto das partes de um conjunto a se a existe;

A4 $\exists y\, \forall y\, (x \in y \longleftrightarrow \exists z\, (x \in z \wedge z \in a))$, axioma da reunião. Este axioma garante a existência de um conjunto que é a união de todos elementos de a. Naturalmente, na interpretação deste axioma na teoria dos conjuntos usuais, a é composto de conjuntos. Em outras palavras, este axioma garante a existência de $\{x : \exists z\, (x \in z \wedge z \in a)\}$;

A5 $\exists y\, (y \in x) \longrightarrow \exists y\, (y \in x \wedge \forall z \neg(z \in x \wedge z \in y))$, axioma da regularidade. Este axioma garante que um conjunto não está contido em si mesmo direta ou indiretamente. Este é o mais complexo de todos os axiomas de ZF;

A6 $\exists w\, ((\emptyset \in w) \wedge \forall x\, (x \in w \longrightarrow x \cup \{x\} \in w))$, axioma da infinidade. Este axioma garante a existência de um conjunto infinito. A saber, $\{\emptyset,\, \{\emptyset\},\, \{\emptyset,\, \{\emptyset\}\},\, \{\emptyset,\, \{\emptyset,\, \{\emptyset\}\}\},\, \ldots\}$. O símbolo $\cup$ é um meta-predicado. $x \cup y$ é uma abreviação de $\exists z\, (w \in z \longleftrightarrow (w \in x \vee w \in y))$;

A7 $(\forall x\,(x \in z \longrightarrow \neg(x = \emptyset)) \wedge (\forall x\,\forall y\,(x \in z \wedge y \in z \wedge \neg(x = y)) \longrightarrow (x \cap y = \emptyset))) \longrightarrow \exists u\,\forall x\,\exists v\,(x \in z \longrightarrow u \cap x = \{v\})$, axioma da escolha. O símbolo $\cap$ é um meta-predicado. $x \cap y$ significa $\exists z\,(w \in z \longleftrightarrow (w \in x \wedge w \in y))$.

Este axioma garante que, dado um conjunto z que possui como elementos outros conjuntos, existe um conjunto u tal que u possui exatamente um elemento em comum com cada elemento de z (que é um conjunto). Exige-se que os elementos de z não sejam vazios e que dois a dois não tenham elementos em comum. Estudando a fórmula,

1. $\forall x\,(x \in z \longrightarrow \neg(x = \emptyset))$ garante que todos os elementos de z sejam diferentes de $\emptyset$;
2. $(\forall x\,\forall y\,(x \in z \wedge y \in z \wedge \neg(x = y)) \longrightarrow (x \cap y = \emptyset))$ garante que quaisquer dois elementos de z, que são conjuntos, tem intersecção vazia;
3. $\exists u\,\forall x\,\exists v\,(x \in z \longrightarrow u \cap x = \{v\})$ garante que existe u que tem exatamente um elemento v em comum com cada elemento x de z dado que os itens 1 e 2 acima, quando interpretados, são verdadeiros. Como $\exists v$ aparece depois de $\exists x$, para cada x de z podemos ter um elemento v diferente — de fato, todos eles são diferentes, já os elementos de z são dois a dois disjuntos.

Exercícios de Treinamento

4.63. *Prove sintaticamente que as seguintes fórmulas são teoremas. Utilizamos* 1 *como uma abreviatura para* $0'$ *e* n *como abreviatura para* $0''...'$ *onde o símbolo* $'$ *aparece* n *vezes.*

(a) $1 = 1$

(b) $x + y \cdot 0 = x$

(c) $1 + 0 = 1$

(d) $2 = 3 \longrightarrow 1 = 2$

4.7 Incompletude

A cada fórmula de uma linguagem de primeira ordem $\mathcal{L}$ se pode associar um número inteiro univocamente. Isto é, existe uma função

$$f : F^* \longrightarrow \mathbb{N}$$

injetora que toma uma fórmula da linguagem de primeira ordem e retorna um número inteiro. Assuma que F^* é o conjunto de todas as fórmulas da linguagem $\mathcal{L}$. É importante notar que duas fórmulas sempre terão números diferentes. Mas pode existir um número inteiro que não está associado a nenhuma fórmula. Uma numeração deste tipo, injetora, é chamada de **numeração de Gödel**. Há inúmeras formas de se construir tal função. Mostraremos a mais fácil delas.

Pode-se associar qualquer fórmula a um número. Mas nesta Seção estaremos interessados em associar apenas as fórmulas da linguagem $\mathcal{L}_A$ da Aritmética (página 113) a números. Esta linguagem é a utilizada no conjunto Γ_P e possui símbolos de função $+$, $'$, $\cdot$ e 0, além dos símbolos usuais de todas as linguagens de primeira ordem (veja a definição na Seção 4.1, página 64). A cada símbolo da linguagem associaremos um número:

0	9
x_1	1
x_2	11
x_3	111
x_n	n números 1 em seqüência
c_1	2
c_2	22
c_3	222
c_n	n números 2 em seqüência
$+$	30
$\cdot$	33
$'$	34
,	35
(	36
)	37
$\neg$	5
$\longrightarrow$	6
$=$	7
$\forall$	8

O leitor deve se convencer de que duas fórmulas diferentes sempre correspondem a números diferentes. Se uma fórmula utilizar conectivos derivados ($\wedge$, $\vee$, $\longleftrightarrow$ e $\exists$), pode-se utilizar a definição destes conectivos em termos dos conectivos primitivos ($\longrightarrow$, $\neg$ e $\forall$) e obter uma fórmula apenas com estes conectivos.

Vejamos alguns exemplos de numeração de fórmulas:

(a)

$\neg$	(	0	$=$	x_1	$'$	)
5	36	9	7	1	34	37

Ou seja, o número correspondente a "$\neg(0 = x_1')$" é 5369713437.

(b)

$\forall$	x_3	(	x_3	$\cdot$	0	$=$	0	)
8	111	36	111	33	9	7	9	37

O número correspondente a "$\forall x_3\,(x_3 \cdot 0 = 0)$" é 8111361113397937.

Utilizando a numeração de Gödel dada acima pode-se obter uma correspondência 1-1 entre fórmulas e números naturais. Apenas é necessário eliminar os números que não correspondem a fórmulas:

Número	fórmula	É fórmula ?
0		não
1	x_1	não (termo)
2	c_1	não (termo)
3		não
4		não
5	$\neg$	não
6	$\longrightarrow$	não
7	$=$	não
8	$\forall$	não
9	0	não
...		
171	$x_1 = x_1$	sim
172	$x_1 = c_1$	sim
173		não
...		
179	$x_1 = 0$	sim
...		

Assim, 0 da nova numeração corresponde à fórmula "$x_1 = x_1$", 1 à "$x_1 = c_1$", 2 a "$x_1 = 0$" e assim por diante.

Neste ponto estamos prontos para responder à seguinte questão: pode-se fazer uma fórmula $A(x)$, com uma variável livre x, que corresponde a qualquer propriedade dos números naturais ? Isto é, dada uma propriedade qualquer dos naturais, como "x é par", "x é um número primo" ou "x é a soma de dois primos ao quadrado", pode-se criar uma fórmula que a represente ? Para verificar este fato, utilizaremos o símbolo n na linguagem $\mathcal{L}_P$ para representar 0 seguido de n símbolos $'$. Isto é, n na linguagem representará o número n no modelo $\mathfrak{B}$ da Aritmética.

Sabemos que podemos fazer fórmulas $A(x)$ tal que

1. $\Gamma_P \vdash A(n)$ sse n é par. Esta fórmula A é $\exists y\,(0'' \cdot y = x)$;
2. $\Gamma_P \vdash A(n)$ sse n é primo. $A =_{def} \forall y\ (D(y,\ x) \longrightarrow (y = 1 \vee y = x) \wedge \neg(y = 0))$ onde $D(y,\ x) =_{def} \exists z(z \cdot y = x \wedge \neg(y = 0))$.
3. $\Gamma_P \vdash A(n)$ sse n é a soma de dois primos ao quadrado (Exercício).

Mas poderemos fazer fórmulas para quaisquer propriedades dos números naturais ? A resposta é não. Uma propriedade dos naturais pode ser caracterizada pelo conjunto dos naturais que a satisfazem. Por exemplo, as propriedades dadas acima correspondem aos conjuntos

1. $\{0,\ 2,\ 4,\ 6,\ \ldots\}$
2. $\{2,\ 3,\ 5,\ 7,\ 11,\ 13,\ 17,\ \ldots\}$
3. $\{11,\ 29,\ 34,\ \ldots\}$

Mas quantos conjuntos dos números naturais existem ? Este conjunto é $\mathcal{P}(\mathbb{N})$, o conjunto das partes dos números naturais. Pode ser provado que não existe nenhuma função bijetora entre $\mathbb{N}$ e

$\mathcal{P}(\mathbb{N})$ embora exista uma função bijetora entre $\mathbb{N}$ e um subconjunto de $\mathcal{P}(\mathbb{N})$ (a saber, o conjunto $\{\{0\}, \{1\}, \{2\}, \ldots\}$). Dizemos que a cardinalidade de $\mathbb{N}$ é menor do que a cardinalidade de $\mathcal{P}(\mathbb{N})$. A propósito, a cardinalidade de $\mathcal{P}(\mathbb{N})$ é igual à do conjunto $\mathbb{R}$.

Concluímos que há propriedades dos números naturais que não podem ser expressas na lógica de primeira ordem. Ou seja, esta lógica tem limitações, ela não pode ser utilizada para representar todas as propriedades dos números naturais.

Proposição 4.6. *Pode-se enumerar todas as fórmulas que são teoremas tomando-se Γ_P como hipóteses. Isto é, pode-se enumerar as fórmulas A tal que $\Gamma_P \vdash A$.*

Prova. Uma seqüência de fórmulas B_1, B_2, ... B_k pode ser representado por um número $n_1 4 n_2 4 \ldots 4 n_k$ onde n_i, $1 \leq i \leq k$, é o número correspondente à fórmula B_i. Pode se provar que existe um algoritmo que, dado um número $n = n_1 4 n_2 4 \ldots 4 n_k$, verifica se n corresponde a uma prova de B_k usando Γ_P como hipóteses. Isto é, o algoritmo verifica se cada n_i corresponde a uma instância de um dos axiomas lógicos A1-A7 ou a uma fórmula de Γ_P ou a uma aplicação de MP e Gen utilizando fórmulas cujos números aparecem em n com índice menor do que i. Suponha que este algoritmo seja uma função `prova(n)` que retorna `true` se n codifica uma prova sintática que utiliza Γ_P como hipóteses e `false` caso contrário.

Então pode-se enumerar (neste caso, imprimir) os teoremas A tal que $\Gamma_P \vdash A$ pelo seguinte algoritmo:

```
n = 1
while true do
   begin
   if prova(n)
   then
        Se n codifica a prova B_1, B_2, ... B_k, imprima B_k
   n = n + 1
   end
```

□

Em toda a discussão abaixo, assumiremos a razoável hipótese de que o conjunto Γ_P é consistente. Isto é, não se pode deduzir uma fórmula A e $\neg A$ a partir deste conjunto.

Qualquer programa de computador é equivalente a uma função $f : \mathbb{N}^k \longrightarrow \mathbb{N}$. As entradas de um programa podem ser convertidos em números inteiros e a saída também. Afinal, tudo o que um computador digital conhece são zeros e uns e estes, em seqüência, formam números inteiros.

Considere uma fórmula $A(x_1, x_2, \ldots x_k, y)$ com x_i, $1 \leq i \leq k$ e y livres e uma função $f : \mathbb{N}^k \longrightarrow \mathbb{N}$. Dizemos que a função f está **representada** por A em Γ_P se $f(n_1, n_2, \ldots n_k) = m$ implicar:

(a) $\Gamma_P \vdash A(n_1, n_2, \ldots n_k, m)$

(b) $\Gamma_P \vdash \neg A(n_1, n_2, \ldots n_k, p)$ para todo $p \neq m$.

Note que o n_i e m em $f(n_1, n_2, \ldots n_k) = m$ são os números naturais e n_i e m em $\Gamma_P \vdash A(n_1, n_2, \ldots n_k, m)$ são abreviaturas de 0 seguido de n_i e m símbolos $'$.

Proposição 4.7. *Qualquer programa de computador é representável por uma fórmula na linguagem $\mathcal{L}_P$ em Γ_P.*

Esta proposição será apresentada sem provas. Ela diz que a qualquer programa corresponde uma fórmula da lógica de primeira ordem que é teorema tomando-se Γ_P como hipóteses. Isto só é possível porque utilizamos o conjunto Γ_P que, esperamos, caracterize os números naturais.

Proposição 4.8. *O conjunto de fórmulas Γ_P é indecidível; isto é, dada uma fórmula A, não existe um algoritmo que diz se $\Gamma_P \vdash A$.*

As fórmulas que caracterizam a Aritmética, Γ_P, possuem uma complexidade tal que nenhum algoritmo jamais será capaz de tomar uma fórmula como parâmetro e dizer se aquela fórmula pode ou não ser deduzida a partir de Γ_P.

Este resultado foi obtido por Gödel em 1931, juntamente com o seu famoso teorema, chamado teorema de Gödel. Este teorema afirma que:

(a) existe uma fórmula A tal que nem A nem $\neg A$ podem ser deduzidas a partir de Γ_P mas nós, observadores fora do sistema formal, sabemos que A é verdadeira. Então Gödel obteve uma fórmula verdadeira da Aritmética que não pode ser demonstrada usando Γ_P. A adição de novas fórmulas a este conjunto não resolve o problema. Por mais que fórmulas sejam adicionadas, sempre haverá uma fórmula que sabemos ser verdadeira mas que não pode ser deduzida. Intuitivamente, a fórmula que Gödel obteve é "eu sou indemonstrável usando Γ_P". Gödel realmente construiu a fórmula A, não apenas provou que alguma fórmula deste tipo existe;

(b) não se pode provar usando o conjunto Γ_P que Γ_P é consistente; isto é, que se $\Gamma_P \vdash A$, então $\Gamma_P \nvdash \neg A$. Gödel criou uma fórmula B que, intuitivamente, é "Γ_P é consistente". Então ele provou que B não pode ser deduzido usando Γ_P; isto é, $\Gamma_P \nvdash B$.

O teorema de Gödel então afirma que o conjunto Γ_P é incompleto, assim como qualquer extensão dele. Isto é, há verdades na Aritmética que não podem ser demonstradas usando Γ_P.

Exercícios de Treinamento

4.64. *Dado um conjunto Γ qualquer, se $\Gamma \nvdash A$, então necessariamente $\Gamma \vdash \neg A$?*

4.65. *Explique, sem provar, o teorema da Completude de Gödel: um conjunto de fórmulas fechadas (sentenças) Γ em uma linguagem $\mathcal{L}$ é consistente se e somente se Γ tem modelo em $\mathcal{L}$.*

4.66. *Explique, sem provar, o teorema da Completude de Gödel: dado um conjunto de fórmulas fechadas Γ e uma fórmula fechada A em uma linguagem $\mathcal{L}$, então $\Gamma \vDash A$ se e somente se $\Gamma \vdash A$.*

Para as questões abaixo, assuma que $\mathcal{L}_P$ é a linguagem utilizada pelos axiomas de Peano expressos em uma linguagem de primeira ordem e Γ_P é o conjunto de fórmulas que representam estes axiomas em primeira ordem.

4.67. *Invente uma nova maneira de associar números a fórmulas das linguagens de primeira ordem.*

4.68. *Associe números a fórmulas de primeira ordem utilizando números primos: uma fórmula $\neg A$ poderia ser codificada como 2^n onde n é o número de A, $A \longrightarrow B$ poderia ser codificada como $3^n 5^m$ onde n é o número de A e m é o número de B e assim por diante.*

4.69. *Como uma propriedade sobre os números naturais pode ser representada por uma fórmula da linguagem $\mathcal{L}_P$ com uma variável livre ?*

4.70. *Qualquer propriedade que é ou não é satisfeita por qualquer número natural pode ser escrita por uma fórmula da linguagem $\mathcal{L}_P$ com uma variável livre ?*

4.71. *Pode-se fazer um algoritmo que imprime, um por vez, os elementos do conjunto $R = \{A : \Gamma_P \vdash A\}$? Justifique.*

4.72. *Pode-se fazer um algoritmo que imprime, um por vez, os elementos do conjunto $R = \{A : \Gamma_P \nvdash A\}$? Justifique.*

4.73. *Represente por fórmulas na linguagem $\mathcal{L}_P$ as seguintes funções da Aritmética:*

(a) $+$

(b) $f : \mathbb{N} \longrightarrow \mathbb{N}$ *tal que* $f(n)$ *é* 1 *se* n *é par e* 0 *se* n *é ímpar;*

(c) $g : \mathbb{N} \longrightarrow \mathbb{N}$ *tal que* $g(n) = 0$ *para qualquer* n*;*

(d) $h : \mathbb{N}^2 \longrightarrow \mathbb{N}$ *tal que* $g(n, m) = (n + m) \,.\, 2$.

4.74. *Pode-se encontrar um algoritmo que não pode ser representado por uma fórmula de $\mathcal{L}_P$?*

4.75. *O que é Aritmética ?*

4.76. *Existe um algoritmo que diz se uma fórmula A de $\mathcal{L}_P$ é teorema tomando-se Γ_P como hipóteses ?*

4.77. *Com relação ao Teorema de Gödel (Incompletude), comente as seguintes frases, corrigindo-as se for necessário:*

(a) *Gödel encontrou uma fórmula A tal que $\Gamma_P \nvdash A$ e $\Gamma_P \nvdash \neg A$ e nós não sabemos se A é verdadeira na Aritmética ou não;*

(b) *Gödel encontrou uma fórmula A tal que nem A nem $\neg A$ são verdadeiras no modelo da Aritmética, mas nós sabemos que A é verdadeira;*

(c) *Gödel encontrou uma fórmula A tal que $\Gamma_P \nvdash A$ e $\Gamma_P \nvdash \neg A$ e nós sabemos que A é verdadeira na Aritmética;*

(d) *Gödel provou que existe uma fórmula tal que nem ela nem a sua negação são teoremas tomando-se Γ_P como hipóteses;*

(e) *Gödel provou que existe uma fórmula tal que nem ela nem a sua negação são teoremas tomando-se Γ_P como hipóteses. Mas ele não mostrou esta fórmula, apenas provou que existe;*

(f) *considere uma fórmula A que significa, intuitivamente, "eu sou indemonstrável usando Γ_P". Então se $\Gamma_P \vdash A$ temos uma contradição;*

(g) *considere uma fórmula A que significa, intuitivamente, "eu sou indemonstrável usando Γ_P", ou seja "A é indemonstrável usando Γ_P". Então se $\Gamma_P \vdash \neg A$ temos uma contradição pois $\neg A$ significa "A é demonstrável usando Γ_P" e então teríamos deduzido que A é demonstrável quando na verdade o que é demonstrável é $\neg A$. Assuma que Γ_P é um conjunto consistente de fórmulas.*

Apêndice A

Respostas dos Exercícios Selecionados

Respostas do Capítulo 2:

2.3 Meta-teorema: Mxy é um teorema onde x contém $2^n - 3 \cdot m$ I´s e y contém m U´s para todo $n \geq 2$ e $m \geq 2^n/3$, n e m inteiros.

2.7 Alguns dos teoremas são $E + T$, $T * N$, $0 + 1 * 2$, $T * 1$. Os teoremas que não contém E, T ou N são precisamente expressões aritméticas com números como $0 + 1 * 2$ ou $0 + 1 + 2 + 3$.

Respostas dos exercícios do Capítulo 3:

3.24 $(A \wedge B \longrightarrow (\neg B \longrightarrow (B \longrightarrow A))) \wedge C$

$\neg A \longrightarrow B \vee C \longleftrightarrow (A \wedge B \vee C \longleftrightarrow A)$

$A \wedge B \wedge C$

$A \vee B \longleftrightarrow A \longrightarrow \neg B$

Referências Bibliográficas

[1] Coniglio, Marcelo E. Teoria Axiomática de Conjuntos: uma Introdução.

[2] Gardner, Martin. The fantastic combinations of John Conway's new solitaire game "life". Scientific American, Vol. 223, October 1970, páginas 120-123. Disponível em http://ddi.cs.uni-potsdam.de/HyFISCH/Produzieren/lis_projekt/proj_gamelife/ConwayScie

[3] Hofstadter, Douglas R. . Gödel, Escher, Bach: an Eternal Golden Braid. Vintage Books, 1979.

[4] Mendelson, Elliott . Introduction to Mathematical Logic. Wadsworth Publishing Co., 1997.

[5] Mortari, César. Introdução à Lógica. Fundação Editora da UNESP, 2001.

[6] Shoenfield, Joseph R. Mathematical Logic. Addison-Wesley, 1967.

[7] Cellular automaton. Disponível em http://en.wikipedia.org/wiki/Cellular_automaton.

[8] Boolean Algebra. Disponível em http://en.wikipedia.org/wiki/Boolean_algebra.

Índice Remissivo