Lógica - CONTEÚDO

MÓDULO 1

LANGUAGE, PROOF AND LOGIC

JON BARWISE & JOHN ETCHEMENDY

Tradução Resumida da 1ª Parte

1. Texto da Introdução e Capítulos 1 a 8

2. Resumo Geral dos Capítulos

3. Todos os Mundos (do programa Mundo de Tarski)

4. Tabela com os Predicados da Linguagem dos Blocos e *Tabela 1.2*, de Predicados Coloquiais

5. Tabela com Descrição das Regras do Sistema F

Texto da Parte I do Livro

Introdução

O Papel Especial da Lógica na Investigação Racional

O que os campos da astronomia, economia, finanças, direito, matemática, medicina, física e sociologia têm em comum? Com certeza, não muito com relação ao assunto. E definitivamente nada com relação à metodologia. O que eles de fato têm em comum, uns com os outros e com muitos outros campos, é sua dependência de um certo padrão de racionalidade. Em cada um destes campos, assume-se que os participantes podem diferenciar entre argumentação racional baseada em princípios e evidências assumidas de especulação livre ou *non sequitur*, afirmações que de modo algum seguem-se das evidências assumidas. Em outras palavras, todos estes campos pressupõem uma aceitação subjacente de princípios de lógica.

lógica e investigação racional

Por esta razão, toda investigação racional depende de lógica, da habilidade das pessoas em raciocinar corretamente a maior parte do tempo, e, quando falharem em raciocinar corretamente, da habilidade de outros apontar as lacunas de seus raciocínios.

Ainda que as pessoas possam não concordar em tudo, elas parecem capazes de concordar sobre o que pode ser legitimamente concluído de informações dadas. A aceitação desses princípios geralmente seguros da racionalidade é o que diferencia a investigação racional de outras formas de atividade humana.

Quais exatamente são os princípios de racionalidade pressupostos por estas disciplinas? E quais são as técnicas com as quais podemos distinguir um raciocínio correto ou "válido" de um incorreto ou "inválido"? Mais basicamente, o que é que faz com que uma afirmação "se siga, logicamente", de um dado conjunto de informações, enquanto uma outra afirmação não se segue do mesmo conjunto?

lógica e convenção

Muitas respostas a estas perguntas têm sido exploradas. Algumas pessoas têm afirmado que as leis da lógica são meramente resultado de convenções. Se for assim, é presumível que possamos decidir mudar as convenções e adotar princípios lógicos diferentes, da mesma forma que podemos decidir qual é o lado correto da estrada em que se deve dirigir. No entanto, há uma forte intuição de que as leis da lógica são de algum modo mais fundamentais, menos sujeitas a revogação, do que as leis do direito, ou até mesmo que as leis da física. Nós conseguimos imaginar um país no qual a luz vermelha do semáforo signifique *siga*, e um mundo no qual a água flua morro acima. Mas nós não conseguimos nem imaginar um mundo no qual haja e não haja nove planetas.

leis da lógica

A importância da lógica tem sido reconhecida desde a antiguidade. Afinal de contas, nenhuma ciência pode ser mais certa que sua vinculação mais fraca. Se há algo arbitrário sobre a lógica, então o mesmo deve valer para toda a investigação racional. Deste modo torna-se crucial entender exatamente quais são as leis da lógica e, até mais importante, *por que* elas são leis da lógica. Estas são as questões que se leva em consideração quando se estuda lógica. Estudar lógica é utilizar os métodos da investigação racional sobre a própria racionalidade.

Durante o século passado, o estudo da lógica experimentou avanços rápidos e importantes. Impulsionada por problemas lógicos na mais dedutiva das disciplinas, a matemática, a lógica desenvolveu-se em uma disciplina independente, com seus próprios conceitos, métodos, técnicas e linguagem. A *Enciclopédia Britânica* lista a lógica como um dos sete ramos principais do conhecimento. Mais recentemente, teve papel fundamental no desenvolvimento dos computadores modernos e das linguagens de programação. A lógica continua desempenhando um papel importante na ciência da computação; de fato, tem-se dito que a ciência da computação é apenas lógica implementada pela engenharia elétrica.

objetivos do livro

O objetivo deste livro é introduzi-lo a alguns dos mais importantes conceitos e ferramentas da lógica. Nossa meta é fornecer respostas sistemáticas e detalhadas às questões levantadas acima. Queremos que você entenda exatamente como as leis da lógica se seguem inevitavelmente dos significados das expressões que usamos para fazer afirmações. Convenção é crucial para atribuir significado à linguagem, mas uma vez que o significado é estabelecido, as leis da lógica se seguem inevitavelmente.

Mais particularmente, temos dois objetivos. O primeiro é ajudá-lo a aprender uma linguagem nova, a linguagem da lógica de primeira ordem. O segundo é ajudá-lo a aprender sobre a noção de conseqüência lógica, e sobre como proceder para estabelecer se alguma afirmação é ou não uma conseqüência lógica de outras afirmações aceitas. Mesmo que haja muito mais em lógica do que podemos sequer mencionar neste livro, ou do que qualquer pessoa poderia aprender no tempo de uma vida, nós podemos, pelo menos, cobrir estes assuntos mais básicos.

Por que aprender uma linguagem artificial?

Esta linguagem da lógica de primeira ordem é muito importante. Como Latim, a linguagem não é falada, mas diferentemente de Latim, ela é usada todos os dias por matemáticos, filósofos, cientistas da computação, lingüistas, e praticantes da inteligência artificial. De fato, de certo modo, ela é uma linguagem universal, a *lingua franca* das ciências simbólicas. Apesar de não ser tão freqüentemente utilizada em outras formas da investigação racional, como medicina e finanças, ela é também uma ferramenta valiosa para o entendimento dos princípios da racionalidade subjacentes também a estas disciplinas.

FOL

Esta linguagem atende por vários nomes: o mais baixo cálculo de predicados, o cálculo funcional, a linguagem da lógica de primeira ordem e FOL[1]. Este último é o nome que usaremos.

Certos elementos da linguagem FOL remetem-se à Aristóteles, mas a linguagem como é conhecida hoje emergiu durante os últimos cem anos. Os nomes mais fortemente associados com o seu desenvolvimento são os de Gottlob Frege, Giuseppe Peano e Charles Sanders Peirce. No final do século XIX, estes três lógicos desenvolveram independentemente os elementos mais importantes da linguagem, conhecidos como *quantificadores*. Desde então tem ocorrido um processo de padronização e simplificação, resultando na linguagem em sua forma atual. Mesmo assim, ainda existem alguns dialetos de FOL, que diferem principalmente na escolha dos símbolos específicos usados para expressar as noções básicas da linguagem. Nós utilizaremos o dialeto mais comum em matemática, no entanto, durante o percurso, informaremos sobre vários outros dialetos. FOL é usada de diferentes formas em diferentes campos. Em matemática, é bastante usada de uma maneira informal. Os diversos conectivos e quantificadores são bastante utilizados no discurso matemático, tanto formalmente quanto informalmente, como em um apontamento de aula. Neste contexto você encontrará elementos de FOL intercalados com português ou com a linguagem matemática nativa. Se você alguma vez fez um curso de cálculo, provavelmente já viu fórmulas como:

lógica e matemática

$$\forall \varepsilon > 0 \; \exists \delta > 0 \ldots$$

Aqui, as letras tortas incomuns foram tiradas diretamente de FOL.

lógica e filosofia

Em filosofia, FOL e algumas de suas expansões são usadas de duas maneiras diferentes. Como em matemática, a notação de FOL é utilizada quando claridade absoluta, rigor e ausência de ambigüidade são essenciais. Mas ela também é usada como um estudo de

[1] FOL: abreviação da expressão inglesa "first-order language" (linguagem de primeira ordem). Em português é muito comum utilizar a abreviação da expressão traduzida: LPO. **N. do T.**

caso para a transformação de noções informais (como gramaticalidade, significado, verdade e prova) em definições precisas e rigorosas. Suas aplicações em lingüística se originam deste uso, uma vez que a lingüística se ocupa, em grande medida, com o entendimento destas noções informais.

lógica e inteligência artificial

Em inteligência artificial, FOL é também utilizada de duas formas. Alguns pesquisadores aproveitam-se da estrutura simples das sentenças de FOL para usá-la como uma forma de codificar o conhecimento que será gravado e utilizado por um computador. O pensamento é modelado através de manipulações envolvendo sentenças de FOL. O outro uso é como uma linguagem de especificação precisa para declaração de axiomas e demonstração de resultados sobre agentes artificiais.

lógica e ciência da computação

Em ciência da computação, FOL teve uma influência ainda mais profunda. A própria idéia de uma linguagem artificial que é precisa e ainda rica o suficiente para programar computadores foi inspirada nesta linguagem. Além disso, todas as linguagens de programação que existem tomaram emprestadas algumas noções de um ou outro dialeto de FOL. Finalmente, há as assim chamadas linguagens de programação lógica, como *Prolog*, cujos programas são seqüências de sentenças em um certo dialeto de FOL. Nós discutiremos um pouco sobre as bases lógicas do Prolog na Parte III deste livro.

linguagens artificiais

FOL serve como o exemplo prototípico do que é conhecido como uma linguagem artificial. Estas são linguagens que foram designadas para propósitos especiais, e são contrastadas com as assim chamadas linguagens naturais, linguagens como Português e Grego, que as pessoas realmente falam. O projeto de linguagens artificiais dentro das ciências simbólicas é uma atividade importante, que é baseada no sucesso de FOL e de seus descendentes.

lógica e linguagem comum

Mesmo que você não vá prosseguir aprendendo lógica ou qualquer ciência simbólica, o estudo de FOL pode ter benefícios reais. É por isso que ela é tão amplamente ensinada. Pelo menos uma coisa é certa, aprender FOL é uma maneira fácil de desmistificar muito do trabalho formal. Também ajudara você a entender mais sobre a sua própria língua, e sobre as leis da lógica que a suportam. Em primeiro lugar, FOL, ainda que seja bastante simples, incorpora de uma maneira limpa algumas das mais importantes características das linguagens humanas. Isto ajuda a tornar estas características muito mais transparentes. A principal destas características é a relação entre a linguagem e o mundo. Em segundo lugar, conforme você aprende a traduzir sentenças do português em sentenças de FOL, você também ganhará maior capacidade para reconhecer e apreciar a enorme sutileza que reside no Português, sutileza que não pode ser capturada por FOL ou linguagens similares, pelo menos ainda não. Finalmente, você se tornará consciente da enorme ambigüidade presente em quase todas as sentenças do Português, ambigüidade que de algum modo não nos impede de nos entendermos uns aos outros na maioria das situações.

Conseqüência e prova

conseqüência lógica

Anteriormente, perguntamos o que faz com que uma afirmação se siga de outras: convenção ou alguma outra coisa? Dar uma resposta a esta questão para FOL ocupará uma parte significativa deste livro. Mas uma resposta curta pode ser dada aqui. A lógica moderna nos ensina que uma afirmação (**P**) é uma conseqüência lógica de outra (**Q**) se não há modo da última (**Q**) ser verdadeira sem que a primeira (**P**) também o seja.

Esta é a noção de conseqüência lógica implícita em toda a investigação racional. Todas as disciplinas racionais pressupõem que esta noção faz sentido, e que podemos utilizá-la para extrair conseqüências daquilo que já conhecemos, ou de suposições que possamos ter. Esta noção também é usada para invalidar uma teoria. Pois se uma afirmação particular é conseqüência lógica de uma teoria, e descobrimos que a afirmação é falsa, então nós sabemos que a própria teoria deve ser incorreta de uma forma ou outra. Se nossa teoria

física tem como uma de suas conseqüências que as órbitas planetárias são circulares quando de fato elas são elípticas, então há alguma coisa errada com nossa física. Se nossa teoria econômica diz que a inflação é uma conseqüência necessária da baixa taxa de desemprego, mas o baixo desemprego dos dias de hoje não tem causado inflação, então nossa teoria econômica precisa de reavaliação.

Investigação racional, no sentido que utilizamos, não está limitada às disciplinas acadêmicas e, portanto, também não estão os princípios da lógica. Se nossas crenças sobre um amigo íntimo implicam logicamente que ele nunca espalharia boatos sobre nós, mas você descobre que ele fez isso, então suas crenças precisam de revisão. Conseqüência lógica é central, não apenas para as ciências, mas para virtualmente todos os aspectos da vida diária.

Uma de nossas maiores preocupações neste livro foi examinar esta noção de conseqüência lógica conforme ela se aplica especificamente à linguagem FOL. Mas ao fazer isso, nós aprenderemos também muito sobre a relação de conseqüência lógica nas linguagens naturais. Nosso interesse principal será aprender como reconhecer quando uma afirmação específica se segue logicamente de outras, e, reciprocamente, quando ela não se segue. Esta é uma habilidade extremamente valiosa, mesmo se você nunca tenha oportunidade de utilizar FOL outra vez depois deste curso. Muito tempo de nossas vidas é gasto tentando convencer outras pessoas de certas coisas, ou sendo convencido de certas coisas por outras pessoas, independentemente se o assunto é inflação, desemprego, que tipo de carro comprar, ou como aproveitar uma noite. A habilidade de distinguir raciocínio bom de ruim irá ajudá-lo a reconhecer quando seus próprios argumentos poderiam ser fortalecidos, ou quando os argumentos de outros deveriam ser rejeitados, a despeito de sua plausibilidade superficial.

prova e contraexemplo

Não é sempre óbvio quando uma afirmação é conseqüência lógica de outras, mas métodos poderosos têm sido desenvolvidos para tratar deste problema, pelo menos para a linguagem FOL. Neste livro, exploraremos o método das provas – como podemos *provar* que uma afirmação é conseqüência lógica de outra – e também o método para mostrar que uma afirmação não é conseqüência lógica de outras. Juntamente com a própria linguagem FOL, estes dois métodos, o método das provas e o método dos contraexemplos, constituem o assunto principal deste livro.

Capítulo 1 - Sentenças Atômicas

FOL não é, na verdade, uma única linguagem, mas uma família de linguagens, todas com a mesma gramática e compartilhando certos itens de vocabulário, conhecidos como conectivos e quantificadores.

As diferenças que podem existir nas linguagens da família de FOL estão em certos itens de vocabulário com os quais construímos suas sentenças mais básicas, as sentenças atômicas.

Sentenças Atômicas

Em português, são as sentenças mais simples, consistindo de alguns nomes conectados por um predicado.

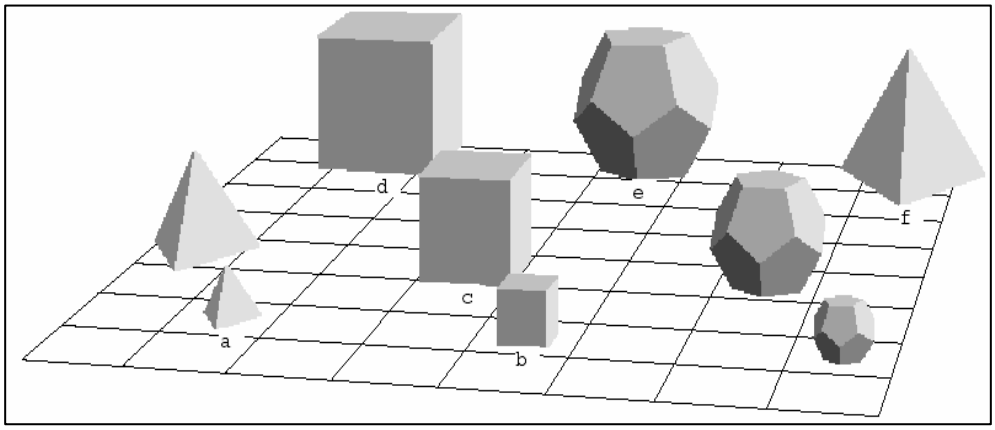
Exemplos: *Max correu*
Max viu Claire
Claire deu Scruffy para Max *(Scruffy é o nome de um animal de estimação)*

Em FOL, as sentenças atômicas são formadas pela combinação de nomes (ou constantes individuais, como são comumente chamados) e predicados.

Nomes e Predicados

Diferentes versões de FOL (ou seja, diferentes linguagens da família FOL) têm diferentes nomes e predicados disponíveis.

Usaremos bastante uma linguagem de primeira ordem projetada para descrever blocos dispostos em um tabuleiro de xadrez, em arranjos que você poderá criar com o programa Mundo de Tarski (Tarski's World).



A Linguagem dos Blocos

Esta linguagem terá nomes tais como **b**, **e**, e n_2, e predicados como **Cube**, **Larger**, e **Between**.

Alguns exemplos de sentenças atômicas nesta linguagem são: **Cube(b)**, **Larger(c, f)**, e **Between(b, c, d)**.

Estas sentenças dizem, respectivamente, que **b** é um cubo, que **c** é maior do que **f** e que **b** está entre **c** e **d**.

1.1. Constantes Individuais

Constantes individuais são simplesmente símbolos que usamos para nos referirmos a algum objeto individual fixado. Elas são o análogo em FOL dos nomes. Em geral não são escritos com letras maiúsculas.

Podemos, em FOL, utilizar **max** como uma constante individual para denotar uma pessoa específica cujo nome é Max.

As constantes individuais funcionam em FOL basicamente da mesma forma que funcionam os nomes em português.

A nossa linguagem de blocos utilizará as letras **a** até **f** acrescidas de n_1, n_2,... como seus nomes.

Nomes em FOL

A principal diferença entre os nomes em português e as constantes individuais de FOL é que é exigido para as constantes individuais que elas se refiram a exatamente um objeto (um e apenas um).

É claro que quando usamos o nome Max em português podemos estar nos referindo a muitas pessoas diferentes. Em FOL isso é proibido. Se utilizamos o nome **max**, então **max** se refere a exatamente UM objeto específico.

Há também nomes em português que não se referem a nenhum objeto que de fato exista. Papai Noel, por exemplo. Este tipo de nome também não é permitido em FOL.

O que é possível em FOL é que um determinado objeto tenha mais do que um nome. Assim, duas constantes individuais, tais como **matthew** e **max** podem ambas se referir ao mesmo indivíduo.

Também será permitido lidar com objetos que não tenham nenhum nome.

LEMBRE-SE

Em FOL

- Cada constante individual deve nomear um objeto (existente).
- Nenhuma constante individual pode nomear mais de um objeto.
- Um objeto pode ter mais de um nome ou mesmo nenhum nome.

1.2. Predicados

Símbolos de predicado são símbolos usados para expressar alguma propriedade de objetos ou alguma relação entre objetos. São por isso também chamados de símbolos de relação.

Como em português, predicados são expressões que, quando combinadas com nomes, formam sentenças atômicas.

Em português a sentença *Max ama Claire* tem um sujeito (*Max*) seguido por um predicado (*ama Claire*).

Sujeitos Lógicos

Em FOL, ao contrário, encaramos esta afirmação como envolvendo dois "sujeitos lógicos", os nomes *Max* e *Claire*, e um predicado, *ama*, que expressa uma relação entre os referentes dos nomes.

Os sujeitos lógicos são chamados de "argumentos" do predicado.

Argumentos de um Predicado

Em português, alguns predicados têm argumentos opcionais. Podemos, por exemplo, dizer *Claire emprestou*, *Claire emprestou Scruffy*, ou *Claire emprestou Scruffy a Max*. Aqui o predicado *emprestou* está sendo usado com um, dois e três argumentos, respectivamente.

Aridade de um Predicado

Mas em FOL, cada predicado tem um número fixo de argumentos. Uma aridade fixa. Este número diz quantas constantes individuais o símbolo de predicado precisa para formar uma sentença.

Se a aridade de um símbolo de predicado **Pred** é 1, então **Pred** será usado para expressar alguma propriedade de objetos.

Nós podemos utilizar o símbolo de predicado unário **Home** para expressar a propriedade de "estar em casa". Podemos então combiná-lo com o nome **max** para obter a expressão **Home(max)**, que expressa a afirmação de que Max está em casa.

Se a aridade de **Pred** é 2, então **Pred** será usado para representar uma relação entre dois objetos. Assim, poderíamos utilizar a expressão **Taller(claire, max)** para expressar a afirmação de que Claire é mais alta do que Max.

Em FOL podemos ter símbolos de predicado de qualquer aridade. No entanto, na linguagem de blocos usada no Mundo de Tarski restringiremos nossos predicados às aridades 1, 2 e 3.

A linguagem de blocos possui 18 predicados (e mais a identidade "=") que o Mundo de Tarski atribui interpretações fixas consistentes com suas correspondentes frases verbais. Assim, **Cube** corresponde a *é um cubo*, **BackOf** corresponde a *está atrás de*, e assim por diante.

A tabela a seguir lista sentenças atômicas que utilizam todos os 19 predicados e apresenta suas interpretações e aridade.

Predicados da Linguagem de Blocos		
Sentença atômica	**Interpretação**	**Aridade**
Tet(a)	**a** é um tetraedro	1 propriedades
Cube(a)	**a** é um cubo	
Dodec(a)	**a** é um dodecaedro	
Small(a)	**a** é pequeno	
Medium(a)	**a** é médio	
Large(a)	**a** é grande	
SameSize(a, b)	**a** tem o mesmo tamanho que **b**	2 relações binárias
SameShape(a, b)	**a** tem o mesmo formato que **b**	
Larger(a, b)	**a** é maior que **b**	
Smaller(a, b)	**a** é menor que **b**	
SameCol(a, b)	**a** está na mesma coluna que **b**	
SameRow(a, b)	**a** está na mesma linha que **b**	
Adjoins(a, b)	**a** e **b** estão em quadros adjacentes (não diagonais) da grade	
LeftOf(a, b)	**a** está mais próximo da borda esquerda que **b**	
RightOf(a, b)	**a** está mais próximo da borda direita que **b**	
FrontOf(a, b)	**a** está mais próximo da borda da frente que **b**	
BackOf(a, b)	**a** está mais próximo da borda de traz que **b**	
a = b	**a** e **b** nomeiam o mesmo objeto	
Between(a, b, c)	**a**, **b** e **c** estão na mesma linha, coluna, ou diagonal, e **a** está entre **b** e **c**	3 - relação ternária

Incerteza ("vaguidade")

Em português, os predicados são muitas vezes vagos. Isso ocorre quando não é muito claro se um indivíduo particular tem a propriedade em questão ou não.

Por exemplo, Claire, que tem dezesseis anos, é jovem. Alguém que tenha 96 anos certamente não é jovem, mas não há uma idade determinada que defina quando alguém deixa de ser jovem. Isto é um processo gradual.

Propriedade Determinável

FOL, no entanto, assume que todo predicado é interpretado por uma propriedade ou relação determinável.

E, por uma propriedade *determinável* entendemos uma propriedade para a qual, dado qualquer objeto, existe um fato definido que indica se o objeto tem ou não a propriedade.

É por isso que dizemos que os predicados da linguagem de blocos são um tanto consistentes com seus predicados correspondentes em português (ou inglês), mas não são necessariamente idênticos a eles.

LEMBRE-SE

Em FOL

- Cada símbolo de predicado tem uma única (e fixa) "aridade", um número que diz a você quantos nomes ele necessita para formar uma sentença atômica.
- Cada predicado é interpretado por uma propriedade ou relação determinável da mesma aridade que o predicado.

1.3. Sentenças Atômicas

Uma sentença formada por um predicado de aridade *n* seguida por *n* nomes é chamda *sentença atômica*.

Taller(claire, max) e **Cube(a)** são exemplos de sentenças atômicas.

Notaçao Infixa x Notação Prefixa

Para o caso do símbolo de identidade, colocaremos os dois nomes requeridos um de cada lado do predicado, como em **a = b**. Isto é chamado de notação "infixa".

Para os outros predicados usaremos a notação "prefixa": o símbolo de predicado precede os argumentos.

A Ordem dos nomes é importante

A ordem dos nomes em uma sentença atômica é bastante importante. Da mesma forma que *Claire é mais alta do que Max* significa algo diferente de *Max é mais alto do que Claire*, também **Taller(claire, max)** significa algo completamente diferente de **Taller(max, claire)**.

Afirmações

Predicados e nomes designam propriedades e objetos, respectivamente, que são partes de sentenças.

Mas o que torna as sentenças especiais é que elas fazem afirmações (ou expressam proposições).

Uma afirmação é algo que é verdadeiro ou falso.

Valor de Verdade

Verdadeiro (**TRUE**) e falso (**FALSE**) são chamados de valores de verdade.

Assim, por exemplo, **Taller(daniel, dapaz)** expressa uma afirmação cujo valor de verdade é **TRUE** e **Taller(dapaz, daniel)** expressa uma afirmação cujo valor de verdade é falso.

LEMBRE-SE

Em FOL,

- Sentenças atômicas são formadas colocando-se um predicado de aridade *n* na frente de *n* nomes (cercados por parênteses e separados por vírgulas).
- Sentenças atômicas também são construídas a partir do predicado de identidade, =, usando notação infixa: os argumentos são colocados em cada um dos lados do predicado.
- A ordem dos nomes é crucial na formação de sentenças atômicas.

1.4. Linguagens de Primeira-Ordem Gerais

As linguagens de primeira-ordem diferem nos nomes e predicados que elas contém e, portanto, nas sentenças atômicas que podem ser construídas.

O que elas compartilham são os conectivos e quantificadores que nos habilitam a construir sentenças mais complexas a partir destas partes mais simples.

Tradução

Quando se traduz sentenças do português para FOL, temos algumas vezes uma linguagem de primeira-ordem "predefinida" que precisamos usar. Como a linguagem de blocos do Mundo de Tarski. Quando isso ocorre, temos que produzir uma tradução que capture ao máximo o significado das sentença original, dados os nomes e predicados disponíveis nesta linguagem predefinida.

Outras vezes, contudo, você não terá uma linguagem predefinida na qual deve fazer a tradução. Quando isto ocorre, a primeira coisa a fazer é decidir que nomes e predicados serão necessários para fazer a tradução. Você terá que projetar, livremente, uma linguagem de primeira-ordem nova capaz de expressar as sentenças do português que você quer traduzir.

Projetando Linguagens

Há sempre muitas alternativas ao se projetar uma linguagem. Por exemplo, a sentença *Claire deu Scruffy a Max* pode ser traduzida criando-se dois nomes, **max** e **claire** e um predicado de dois lugares (aridade 2 ou binário) **GaveScruffy(x, y)**, que significa *x deu Scruffy a y*. Então a sentença original pode ser traduzida como **GaveScruffy(claire, max)**.

Alternativamente, você pode introduzir o predicado de três lugares (aridade 3 ou ternário) **Gave(x, y, z)**, que significa *x deu y a z*, e traduzir a sentença como: **Gave(claire, scruffy, max)**.

Não há nada de errado com nenhum destes dois predicados, nem com as sentenças traduzidas, contanto que você especifique claramente o que o predicado significa.

Escolhendo Predicados

É claro que o predicado binário é mais restritivo que o ternário. A sentença *Claire deu Max para Scruffy* não pode ser traduzida em uma linguagem que tenha apenas o predicado binário **GaveScruffy(x, y)**, mas com o predicado ternário **Gave(x, y, z)** ela pode ser assim traduzida: **Gave(claire, max, scruffy)**.

Se quiséssemos traduzir esta sentença com um predicado binário, teríamos que criar outro, tal como **GaveMax(x, y)** e então a tradução seria: **GaveMax(claire, scruffy)**.

Em geral, quando projetamos uma linguagem de primeira-ordem, tentamos economizar nos predicados, e para isso utilizamos os mais flexíveis possíveis. Em nosso exemplo, **Gave(x, y, z)** é preferível ao par **GaveScruffy(x, y)** e **GaveCarl(x, y)**.

Objetos

Nomes podem ser introduzidos em uma linguagem de primeira ordem para se referir a qualquer coisa que possa ser considerada um objeto.

Nossa noção de objeto é extremamente flexível. Qualquer coisa sobre a qual nós podemos fazer afirmações pode ser considerado um objeto.

Suponha, por exemplo, que queiramos traduzir as sentenças:

Claire deu Scruffy para Max no sábado
Domingo, Max deu Scruffy para Evan

Podemos introduzir um predicado de quatro lugares **Gave(w, x, y, z)** que signifique *w deu x para y no dia z*. Se introduzirmos nomes para os dias particulares, tais como sábado e domingo, podemos, então, traduzir as duas sentenças acima como:

Gave(claire, scruffy, max, saturday)
Gave(max, scruffy, evan, sunday)

Ou seja, tanto seres humanos, quanto animais, quanto números, quanto sólidos geométricos, quanto dias da semana, quanto qualquer outra coisa sobre a qual possamos falar, pode ser um objeto e, portanto, ter algum nome que a represente em uma linguagem de primeira ordem.

Projetar uma linguagem de primeira-ordem com apenas os nomes e predicados necessários requer uma certa habilidade. Usualmente, o objetivo principal é produzir uma linguagem que possa dizer tudo o que você quer, mas que use o menor "vocabulário" (nomes e predicados) possível.

1.5. Símbolos de Função

Algumas linguagens de primeira-ordem têm, além dos nomes e predicados, outras expressões que podem aparecer nas sentenças atômicas. São os ***símbolos de função***.

Os símbolos de função nos permitem construir termos do tipo dos nomes a partir de nomes.

Eles nos permitem expressar, usando sentenças atômicas, afirmações complexas que não poderiam ser tão bem expressas utilizando apenas nomes e predicados.

Os principais exemplos do português são as "frases nominais". Além de nomes como *Max* e *Claire*, as frases nominais incluem expressões como: *O pai de Max*, *A mãe de Claire*, *Toda garota que conhece Max*, *Nenhum garoto que conheça Claire*, *Alguém*, e assim por diante...

Cada um destes exemplos quando combinados com uma "frase verbal" singular, tal como *gosta de pipoca sem manteiga* forma uma sentença diferente:

A mãe de Claire gosta de pipoca sem manteiga
Nenhum garoto que conheça Claire gosta de pipoca sem manteiga

Estas são duas sentenças distintas, com propriedades lógicas distintas em que duas frases nominais diferentes, ambas envolvendo Claire, são combinadas com a mesma frase verbal.

Elas têm propriedades lógicas distintas pois a primeira sentença afirma que um indivíduo específico gosta de pipoca sem manteiga. Já da segunda sentença não podemos tirar esta conclusão.

Termos

Devido a suas propriedades lógicas diferentes, as "frases nominais" são tratadas diferentemente em FOL. Aquelas que, como "*A mãe de Claire*", intuitivamente se referem a um indivíduo, são chamadas de "termos", e se comportam como as constantes individuais.

De fato, as constantes individuais representam os termos mais simples. Termos mais complexos são construídos a partir destes através do uso dos símbolos de função.

Já frases nominais como *Nenhum garoto que conheça Claire* são manipuladas através de dispositivos bem diferentes conhecidos como quantificadores.

Termos Complexos

A forma análoga em FOL para a frase nominal *O pai de Max* é o termo **father(max)**. Ele é formado colocando-se o símbolo de função **father** na frente da constante individual **max**. O resultado é um termo complexo que usaremos para nos referir ao pai da pessoa referida pelo nome **max**.

É possível repetir construções deste tipo quantas vezes quisermos, formando termos mais complexos ainda, tais como:

father(father(max))
mother(father(claire))
mother(mother(mother(claire)))

O primeiro termo se refere ao avô paterno de Max, o segundo à avó paterna de Claire, e assim por diante.

Estes símbolos de função são chamados unários, porque, como os predicados unários, têm apenas um argumento.

Os termos resultantes funcionam exatamente como os nomes e podem ser usados para formar sentenças atômicas. Por exemplo, a sentença:

Taller(father(max), max)

afirma que o pai de Max é mais alto do que Max. Assim, em uma linguagem que contém símbolos de função, a definição de sentenças atômicas precisa ser modificada para permitir que termos complexos também sejam utilizados como argumentos, da mesma forma que os nomes.

Sïmbolos de Função x Predicados

Os estudantes freqüentemente confundem símbolos de função com predicados, porque ambos têm termos como argumentos. Mas há uma grande diferença entre eles.

Quando você combina uma função unária com um termo, você não obtém uma sentença, mas outro termo: alguma coisa que se refere (ou deveria se referir) a um objeto. É por isso que os símbolos de função podem ser reaplicados várias vezes.

Como vimos, a seguinte construção é perfeitamente compreensível e faz sentido:

father(father(max))

Já esta, por outro lado, é completamente sem sentido:

Dodec(Dodec(a))

Para ajudar a evitar esta confusão, as letras iniciais de todos os predicados de FOL serão MAIÚSCULAS, enquanto que as funções serão escritas com todas as suas letras minúsculas.

Aridade dos Símbolos de Função

FOL permite símbolos de função de qualquer aridade. Por exemplo, podemos ter um símbolo de função **sum** que combinado com dois termos t_1 e t_2 nos dá um novo termo **sum(t_1, t_2)** que se refere à soma dos números referidos pelos termos t_1 e t_2. Assim, o termo complexo **sum(3, 5)** seria uma outra forma de nos referirmos ao número 8.

Da mesma forma que em FOL exigimos que cada nome se refira a um objeto existente, também assumimos que cada termo complexo se refira a exatamente um objeto, ainda que esta seja um exigência um tanto artificial, pois talvez não haja nenhum objeto ao qual o termo **mother(adão)** se refira, e certamente não há nenhum objeto ao qual **mother(3)** se refere.

Ao projetar uma linguagem de primeira-ordem com funções, você deve se assegurar de que os seus termos complexos sempre tenham referência a um único e existente indivíduo.

Sïmbolos de Função para a Linguagem de Blocos

A linguagem de blocos, conforme implementada no Mundo de Tarski, não tem símbolos de função, mas nós poderíamos facilmente estendê-la para incluir alguns.

Suponha, por exemplo, que introduzimos os símbolos de função **fm**, **bm**, **lm** e **rm**, com os quais podemos formar termos complexos tais como:

fm(a)
lm(bm(c))
rm(rm(fm(d)))

Poderíamos interpretar estes símbolos de função como: **fm(a)** se refere ao bloco mais à frente na mesma coluna que o bloco com nome **a** (*frontmost*). Assim, se há vários blocos na coluna de **a**, **fm(a)** se refere ao que está mais à frente, mesmo que seja o próprio **a**.

Analogamente, poderíamos interpretar **bm**, **lm** e **rm** respectivamente como *backmost* (o bloco mais atrás na coluna), *leftmost* (o bloco mais à esquerda na linha) e *rightmost* (o bloco mais à direita na linha).

Com estas interpretações, o termo **lm(bm(c))** se referiria ao bloco mais à esquerda da linha em que está o bloco mais atrás na coluna do bloco **c**.

A sentença atômica **Larger(lm(bm(c)), c)** seria verdadeira se e somente se este bloco fosse maior que **c**.

LEMBRE-SE

Em uma linguagem com símbolos de função,

- Termos complexos são tipicamente formados colocando-se um símbolo de função de aridade *n* na frente de *n* termos (simples ou complexos).
- Termos complexos são usados exatamente como os nomes (termos simples) na construção de sentenças atômicas.
- Em FOL, assumimos que os termos complexos também se referem a um e apenas um objeto.

Capítulo 2 - A Lógica das Sentenças Atômicas

O principal assunto em lógica é o conceito de *conseqüência lógica*: Quando uma sentença, declaração ou afirmação se segue logicamente de outras?

De fato, uma das principais motivações para criação de FOL foi tornar o conceito de conseqüência lógica tão claro quanto possível.

Neste capítulo, explicaremos, o que significa dizer que uma sentença é "*conseqüência lógica*" de outras ou, equivalentemente, o que significa dizer que um argumento é "*logicamente válido*".

Apesar de ser um conceito fácil de entender, ele pode ser diabolicamente difícil de aplicar em situações específicas. Os matemáticos sabem bem disso.

Descreveremos também as principais técnicas utilizadas para mostrar que uma determinada afirmação é ou não é uma conseqüência de outras afirmações.

Vamos, por fim, iniciar a apresentação do que é conhecido como um *sistema formal de dedução*, um sistema que nos permite mostrar que uma sentença de FOL é uma conseqüência de outras.

Este sistema continuará sendo desenvolvido nos próximos capítulo do livro, conforme formos aprendermos mais sobre FOL.

2.1. Argumentos Válidos e Corretos

Argumentos, Premissas e Conclusões

Um *argumento* é qualquer série de declarações (afirmações) na qual uma (chamada de *conclusão*) supostamente se segue, ou é justificada pelas outras (chamada de *premissas*).

Argumentos aparecem em todas as formas de discurso científico e racional, além de editoriais de jornais, livros escolares, nossas conversas, romances,...

Expressões como "***por isso***", "***conseqüentemente***", "***assim***", "***deste modo***", "***portanto***", "***então***", são usadas para indicar que o que segue é a conclusão de um argumento.

Expressões como "***porque***", "***desde que***", "***já que***", "***uma vez que***", "***afinal de contas***", são usadas para indicar as premissas.

Exemplos:

(1) Todos os homens são mortais. Sócrates é homem. Então, Sócrates é mortal.

(2) Lucrecius é um homem. Afinal de contas, Lucrecius é mortal e todos os homens são mortais.

Conseqüência Lógica

Diremos que o primeiro argumento acima é ***logicamente válido***, ou que sua conslusão é uma ***conseqüência lógica*** de suas premissas.

O motivo disso é que, no primeiro argumento, é impossível para a conclusão ser falsa se as premissas são verdadeiras.

Em contraste, a conclusão do segundo argumento acima poderia ser falsa (suponha que Lucrecius é o nome de meu peixe), ainda que as premissas sejam verdadeiras (peixes de aquário são sabidamente mortais).

Assim, no segundo argumento, a conclusão não é conseqüência lógica das premissas.

Argumentos Logicamente Válidos

Grosseiramente falando, um argumento é logicamente válido se e somente se, sob a hipótese de que suas premissas são verdadeiras, sua conclusão é obrigatoriamente verdadeira.

Note que isto não significa que as premissas de um argumento têm que ser verdadeiras para que ele seja válido.

Nosso primeiro argumento continuaria sendo um argumento válido, mesmo se descobríssemos que Sócrates, ao invés de ser humano, não passasse de um personagem criado por Platão.

Mesmo nesta hipótese, continuaria sendo impossível às premissas serem verdadeiras e a conclusão falsa.

Neste caso, ainda que o argumento seja logicamente válido, não poderíamos garantir que sua conclusão é verdadeira, uma vez que ele tem uma premissa falsa.

Um outro argumento logicamente válido, que envolve a linguagem dos blocos é o seguinte:

> Suponha que lhe digam que **Cube(c)** e que **c = b**. Então, disso se segue que **Cube(b)**.

Note que não há possibilidade de a conclusão ser falsa na hipótese das premissas serem verdadeiras. Podemos reconhecer este fato mesmo sem sabermos se as premissas são de fato verdadeiras.

O que garante a validade do argumento é nossa capacidade de reconhecer, sem sombra de dúvidas que, se as premissas forem verdadeiras, a conclusão também será.

Argumentos Corretos

É claro que quando utilizamos argumentos queremos não apenas que eles sejam válidos, mas que tenham premissas verdadeiras. Pois apenas deste modo garantimos a verdade da conclusão.

Dizemos que um argumento válido que tenha premissas verdadeiras é ***correto***.

Portanto, apenas um argumento correto assegura a verdade de sua conclusão.

O argumento sobre Sócrates acima, além de válido é também correto. Há muitos indícios de que Sócrates de fato viveu.

Veja um exemplo de um argumento válido mas não correto:

> Todos os atores ricos são bons atores. Brad Pitt é um ator rico. Logo, ele tem que ser um bom ator.

Como a primeira premissa é falsa, este argumento não garante a verdade de sua conclusão. Ainda que Brad Pitt possa ser um bom ator, isto certamente não se deve ao fato dele ser rico e de que todos os atores ricos são bons.

A lógica estuda a validade dos argumentos não sua correção.

O que os lógicos podem lhe dizer é como raciocinar corretamente a partir das informações que você sabe ou acredita que são verdadeiras.

A lógica não é a ciência de tudo. Ela apenas nos ajuda a reconhecer e produzir argumentos válidos. Garantir a correção dos argumentos está fora dos limites da lógica!!

O Formato de Fitch para Argumentos

Usaremos, neste livro, um formato especial para apresentar argumentos. O "*formato de Fitch*", que indica quais sentenças são premissas e qual é a conclusão.

O argumento acima fica assim disposto no formato de Fitch:

Todos os atores ricos são bons atores
Brad Pitt é um ator rico

Brad Pitt é um bom ator

As sentenças acima da linha horizontal, chamada de ***barra Fitch***, são as premissas, e a sentença abaixo é a conclusão.

LEMBRE-SE

1. Um ***argumento*** é uma série de afirmações na qual uma, chamada de ***conclusão***, é tomada como ***conseqüência*** das outras, chamadas de ***premissas***.
2. Um argumento é ***válido*** se a conclusão for verdadeira em qualquer circunstância na qual as premissas são verdadeiras. Diremos que a *conclusão* de um argumento logicamente válido é uma ***conseqüência lógica*** de suas premissas.
3. Um argumento é ***correto*** se for válido e suas premissas forem todas verdadeiras.

2.2. Métodos de Prova

Apesar de sua clareza, nossa descrição acima da relação de conseqüência lógica não é operacional. Ou seja, ela não nos indica como mostrar que uma dada conclusão **S** se segue ou não se segue de algumas premissas **P**, **Q**, **R**,...

Os exemplos que vimos eram bastante óbvios, mas nem sempre isto ocorre.

Neste curso você aprenderá os métodos fundamentais para comprovar quando afirmações se seguem de outras afirmações e quando elas não se seguem. Ou seja, quando a conclusão de um argumento é conseqüência lógica de suas premissas e quando ela não é.

A técnica fundamental para mostrar que uma determinada conclusão *não se segue* de algumas premissas é encontrar uma circunstância possível na qual as premissas são verdadeiras, mas a conclusão é falsa. Mais adiante analisaremos com mais detalhe esta técnica.

Prova

A noção chave envolvida na tarefa de mostrar que uma determinada afirmação é conseqüência lógica de algumas premissas é a noção de **prova**.

Uma prova é uma demonstração passo-a-passo de que uma conclusão (digamos **S**) se segue de algumas premissas (digamos **P**, **Q**, **R**).

O modo como uma prova funciona é através do estabelecimento de uma série de conclusões intermediárias, cada uma delas correspondendo a uma conseqüência óbvia das premissas originais e das conclusões intermediárias previamente estabelecidas.

A prova termina quando **S** (a conclusão desejada) for finalmente estabelecida como uma conseqüência óbvia das premissas originais e das conclusões intermediárias.

Por exemplo, de **P**, **Q**, **R** pode ser óbvio que S_1 se segue. Destas 4 (**P**, **Q**, **R**, S_1) pode ser óbvio que S_2 se segue. Finalmente, de todas estas 5 afirmações, nós poderíamos ser capazes de obter a conclusão desejada **S**.

Esta prova mostrará que **S** é, de fato, conseqüência de **P**, **Q**, **R**. Isso porque na hipótese de as premissas serem todas verdadeiras, se cada um dos passos individuais estiver correto, então cada uma das conclusões intermediárias (S_1 e S_2) deve também ser verdadeira e, neste caso, nossa conclusão final (**S**) deve também ser verdadeira.

Um exemplo. Suponha que queiramos provar que o seguinte argumento é logicamente válido:

P_1	Sócrates é um homem.
P_2	Todos os homens são mortais.
P_3	Nenhum mortal vive para sempre.
P_4	Todos que cedo ou tarde morrerão se preocupam com isso de vez em quando.
S	Sócrates se preocupa com a morte de vez em quando.

Uma prova de que a conclusão é conseqüência lógica das premissas poderia ser obtida através dos passos intermediários S_1 e S_2 abaixo:

P_1	Sócrates é um homem.	
P_2	Todos os homens são mortais.	
P_3	Nenhum mortal vive para sempre.	
P_4	Todos que cedo ou tarde morrerão se preocupam com isso de vez em quando.	
S_1	Sócrates é mortal.	(de P_1 e P_2)
S_2	Sócrates cedo ou tarde morrerá.	(de S_1 e P_3)
S	Sócrates se preocupa com a morte de vez em quando.	(de S_2 e P_4)

Isto prova que **S** é conseqüência lógica apenas de P_1,...,P_4, porque se estas premissas forem todas verdadeiras, S_1 será verdadeira. Isso garante a verdade de S_2 que, por sua vez, garante a verdade de **S**.

Assim, a verdade das premissas basta para garantir a verdade de cada um dos passos intermediários e, portanto, da conclusão desejada.

Demanda por Rigor

Uma prova de que **S** se segue das premissas $P_1,\ldots,P_n$ pode ser bastante longa e complicada. No entanto, cada passo da prova deve prover evidência absolutamente incontestável de que a conclusão intermediária se segue do que foi anteriormente estabelecido.

Neste ponto a lógica é extremamente rigorosa.

Não basta mostrar que cada passo de uma suposta prova se segue, quase que certamente, dos passos anteriores. É preciso que haja 100% de certeza em cada passo intermediário.

Em nossos argumentos do dia-a-dia, talvez possamos abdicar dos 100% de certeza, mas se quisermos demonstrar que **S** *tem* que ser verdadeiro quando $P_1,\ldots,P_n$ o são, precisamos ser mais rigorosos.

Há uma razão prática para este rigor. Os raciocínios passo-a-passo que eventualmente fazemos no dia-a-dia têm geralmente um pequeno número de passos.

Mesmo que pequenas incertezas sejam acrescentadas nos passos intermediários, seu acúmulo não compromete muito o resultado final.

Mas em muitos tipos de argumentação não é este o caso. Os geômetras, por exemplo, baseiam sua disciplina em um pequeno número de axiomas (os 5 axiomas de Euclides). Vão provando conclusões, a partir destes axiomas, chamadas de teoremas. Os teoremas já provados são utilizados na prova de novos e mais interessantes teoremas.

A prova de um teorema inclui, portanto, as provas de todos os teoremas nela utilizados. Assim, se fôssemos escrevê-la completamente, uma prova aparentemente simples poderia ter centenas, talvez milhares de passos.

Se nós permitíssemos mesmo a mais sutil incerteza nos passos individuais, estas incertezas poderiam se multiplicar tanto, até que uma alegada "prova" tornasse a verdade de sua conclusão não mais confiável que sua falsidade.

Métodos de Prova

Sempre que introduzirmos novos tipos de expressões em nossa linguagem, discutiremos os novos métodos de prova que tais expressões suportam.

Isso será feito através de discussões informais dos métodos de prova mais usados em matemática, ciência e no dia-a-dia, seguidas pela formalização destes métodos, incorporando-os dentro do que chamamos de ***sistema formal*** de dedução.

Sistemas Formais

Um sistema formal de dedução utiliza um conjunto fixo de regras que especificam o que vale como passo aceitável de uma prova.

Provas Informais

A diferença entre uma prova informal e uma prova formal *não está no rigor, mas no estilo*. Uma prova informal, do tipo que os matemáticos fazem, é tão rigorosa quanto uma prova formal.

No entanto, não está escrita em um sistema formal, mas em português e, além disso, ela pode deixar de fora alguns dos passos mais óbvios.

Provas Formais

Uma prova formal, por contraste, emprega um estoque fixo de regras e um método de apresentação altamente estilizado.

Por exemplo, o argumento simples que mostra que de **Cube(c)** e **c = b** segue-se **Cube(b)** terá, em nosso sistema formal o seguinte aspecto:

Aprenderemos, no decorrer desta disciplina como fazer provas tanto informais quanto formais.

Não queremos dar a impressão de que as provas formais são de algum modo melhores que as informais. Ao contrário, em muitas situações os métodos informais são preferíveis.

As provas formais, no entanto, têm sua razão de ser. Primeiro elas podem ser mecanicamente verificadas. É mais fácil encontrar erros em provas formais do que nas informais. Segundo porque elas nos permitem provar coisas sobre a nossa própria capacidade de provar coisas, tais como os Teoremas de Completude e Incompletude de Gödel que serão discutidos na seção final deste livro.

LEMBRE-SE

1. Uma prova para uma sentença **S** a partir das premissas P_1,..., P_n é uma demonstração passo a passo que mostra que **S** deve ser verdadeira em quaisquer circunstâncias nas quais as premissas P_1,..., P_n sejam todas verdadeiras.
2. Provas formais e informais diferem apenas no estilo, não no rigor.

2.2.1 Provas Envolvendo o Símbolo de Identidade (=)

Indiscernibilidade de Idênticos

Já vimos um exemplo de um importante método de prova.

Se pudermos provar, não importa as premissas, que **b=c**, então sabemos que qualquer coisa que for verdadeira para **b** também será verdadeira para **c**. Afinal de contas, **b** e **c** são apenas nomes diferentes do mesmo objeto.

Em filosofia esta simples observação tem o pomposo nome de "***Princípio de Indiscernibilidade de Idênticos***". Algumas vezes também chamado de ***Princípio da Substituição***.

Eliminação da Identidade

Chamaremos a regra formal correspondente ao princípio da indiscernibilidade de idênticos de **Eliminação da Identidade**, que abreviaremos para **(=Elim)**.

A razão deste nome é que uma aplicação desta regra "elimina" um uso de um símbolo de identidade quando nos movemos das premissas de um argumento para a sua conclusão.

Reflexividade da Identidade – Introdução da Identidade

Outro princípio, tão simples que é freqüentemente esquecido é a chamada **Reflexividade da Identidade**, que diz que algo é sempre idêntico a si mesmo.

A regra formal que corresponde a este princípio é chamada de **Introdução da Identidade**, ou **(=Intro)**, uma vez que permite-nos introduzir sentenças com o símbolo de identidade nas provas.

Ele nos diz que qualquer sentença com a forma **a=a** pode ser validamente inferida a partir de quaisquer premissas, ou mesmo de nenhuma premissa.

Este princípio só é válido porque, em FOL, os nomes se referem a um e apenas um objeto. Não há como um nome se referir a objetos diferentes.

Simetria da Identidade

Mais um princípio, um pouco mais útil, é o da simetria da identidade. Ele nos permite concluir **b=a** a partir de **a=b**.

Na verdade, se quisermos, é possível obter este princípio como uma conseqüência dos primeiros dois princípios através da seguinte prova informal:

Prova: suponha que **a=b**. Nós sabemos, pela reflexividade da identidade que **a=a**. Agora, se substituirmos o primeiro **a** de **a=a** por **b**, usando para isso o princípio da indiscernibilidade de idênticos (**=Elim**), então obtemos **b=a.**

O parágrafo acima é outro exemplo de uma **prova informal**. Uma prova informal freqüentemente inicia-se pelo estabelecimento das premissas e suposições da prova, e então segue-se uma explicação no estilo passo-a-passo sobre como podemos obter, a partir destas suposições, a conclusão desejada.

Não há regras estritas sobre quão detalhada deve ser uma prova informal. Isso depende de para quem você está fazendo a prova.

A única regra é que cada passo deve estar escrito de forma não ambígua, e sua validade deve ser aparente.

Na seção 3 aprenderemos como formalizar a prova acima.

Transitividade da Identidade

Um terceiro princípio sobre a identidade que beira a trivialidade é a chamada **transitividade**.

Ele estabelece que se **a=b** e **b=c**, então **a=c**. Demonstraremos tal princípio em um exercício mais adiante, utilizando, para isso a indiscernibilidade de idênticos.

Estes princípios de identidade valem até para os termos complexos de uma linguagem que utiliza símbolos de função.

Se sabemos, por exemplo, que **Happy(john)** e que **john = father(max)**, então você pode utilizar a eliminação da identidade para concluir que **Happy(father(max))**, ainda que **father(max)** seja um termo complexo.

LEMBRE-SE

Há quatro princípios importantes válidos para a relação de identidade:

1. **(=Elim)** : Se **b=c**, então tudo o que é válido para **b** é valido para **c**. Este princípio também é conhecido como indiscernibilidade de idênticos.
2. **(=Intro)** : Sentenças da forma **b=b** são sempre verdadeiras em FOL. Isto também é conhecido como reflexividade da identidade.
3. **Simetria da Identidade** : se **b=c**, então **c=b**.
4. **Transitividade da Identidade** : Se **a=b** e **b=c**, então **a=c**.

Os últimos dois princípios se seguem dos dois primeiros.

2.2.2 Provas Envolvendo Outros Predicados e Relações

Algumas vezes ocorrem certas **dependências lógicas** entre os predicados de uma linguagem de primeira ordem. Dependências similares às que acabamos de discutir envolvendo o símbolo de identidade.

Este é o caso, por exemplo, para os predicados de nossa linguagem dos blocos. Quando isso ocorre, as provas podem ter que explorar estas dependências e relações.

Relações Transitivas

Por exemplo, a sentença **Larger(a,c)** é uma conseqüência de **Larger(a,b)** e **Larger(b,c)**. Isto ocorre porque a relação "maior que" (expressa por **Larger**), da mesma forma que a identidade, é transitiva.

É por isso que em qualquer mundo no qual que as duas últimas sentenças são verdadeiras, a primeira também será.

Não há como catalogar todas as inferências legítimas envolvendo símbolos de predicado e relações em todas as linguagens que podemos criar. Mas o exemplo da identidade nos dá algumas referências dos tipos de propriedades que temos que procurar.

Muitas relações serão **reflexivas**: por exemplo, é claro que a sentença **SameSize(a,a)** é sempre verdadeira.

Muitas relações serão **simétricas**: Se **SameShape(a,b)** é verdadeira, então também é **SameShape(b,a)**.

Relações Inversas

Você poderia ser informado de que **b** é maior que **c** e solicitado a inferir que **c** é menor do que **b**. Isto seria válido, porque "maior que" e "menor que" são **relações inversas**.

Tabela de Dependência de Predicados

A tabela abaixo apresenta um exemplo de dependências mais comuns que as relações expressas por predicados podem possuir. Ela também indica quais os predicados da linguagem dos blocos que possuem cada uma destas dependências.

NOME	DEPENÊNCIA	EXEMPLO NA LINGUAGEM DOS BLOCOS
Reflexividade	Vale **P(a, a)**	SameSize, SameShape, SameCol, SameRow
Simetria	Se vale **P(a, b)**, então vale **P(b, a)**	SameSize, SameShape, SameCol, SameRow, Adjoins
Transitividade	Se valem **P(a, b)** e **P(b, c)**, então vale **P(a, c)**	SameSize, SameShape, Larger, Smaller, SameCol, SameRow, LeftOf, RIghtOf, FrontOf, BackOf,
Não-reflexividade	Não vale **P(a, a)**	Larger, Smaller, Adjoins, LeftOf, RightOf, FrontOf, BackOf, Between
Antissimetria	Se vale **P(a, b)**, então não vale **P(b, a)**	Larger, Smaller, LeftOf, RightOf, FrontOf, BackOf
Não-transitividade	Não é verdade que: "Se valem **P(a, b)** e **P(b, c)**, então vale **P(a, c)**"	Adjoins
Relações Inversas	**P(a, b)** vale, se e somente se **Q(b, a)** também vale (**P** – **Q** são inversos)	Larger – Smaller, LeftOf – RightOf, FrontOf – BackOf

Mais um exemplo de prova informal. Suponha que você seja solicitado a provar o seguinte argumento:

$$\begin{array}{l} \mathrm{RightOf}(b, c) \\ \mathrm{LeftOf}(d, e) \\ b = d \\ \hline \mathrm{LeftOf}(c, e) \end{array}$$

Uma prova informal poderia ser como a seguinte:

> **Prova**: Uma das premissas é que **b** *está à direita* de **c**. Então, **c** deve estar *à esquerda* de **b**, uma vez que *estar a esquerda* e *estar a direita* são relações inversas. E como **b=d** (premissa), então, por indiscernibilidade de idênticos, **c** está à esquerda de **d**. Mas também sabemos que **d** está à esquerda de **e** (outra premissa). Conseqüentemente, **c** está à esquerda de **e**, pois *estar à esquerda* é uma relação transitiva. Esta é a conclusão desejada.

2.3 Provas Formais

Sistemas Dedutivos

Nesta seção iniciaremos a construção de nosso sistema formal para apresentação de provas. Nosso "**sistema dedutivo**".

O Sistema $\mathcal{F}$

Existem muitos estilos diferentes de sistemas dedutivos. O sistema que será apresentado nas duas primeiras partes deste livro será chamado de sistema $\mathcal{F}$, um sistema de dedução natural ao estilo dos sistemas criados pelo lógico Frederic Fitch.

No sistema $\mathcal{F}$, uma prova de uma conclusão **S** a partir das premissas **P**, **Q** e **R** se parece muito com um argumento apresentado no formato Fitch.

A diferença principal é que a prova apresenta, além da conclusão **S**, todas as conclusões intermediárias $S_1, \ldots, S_n$ que derivamos das premissas para chegar a **S**.

É importante notarmos as duas linhas: a vertical e a horizontal.

A **linha vertical** direciona nossa atenção para o fato de que temos uma única prova que consiste de uma seqüência de vários passos.

A **linha horizontal** (chamada de barra Fitch) indica a divisão entre as afirmações que estamos assumindo (nossas premissas) e aquelas que se seguem delas (as conclusões intermediárias e final).

Assim, como **P**, **Q** e **R** estão acima da barra Fitch, isso indica que elas são as premissas de nossa prova.

E como **S_1, ..., S_n** e **S** estão abaixo da barra Fitch, isso indica que estas sentenças devem ser conseqüências lógicas das premissas.

Justificativa

Note que à direita de cada passo abaixo da barra Fitch, apresentamos uma justificativa para o passo.

Uma justificativa indica qual a regra nos permite dar este passo, e a quais passos anteriores da prova a regra é aplicada.

Também numeraremos os passos das provas para que possamos fazer referências a eles em justificativas de passos posteriores.

Aqui está um exemplo de uma formalização de nossa prova da simetria da identidade.

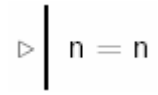
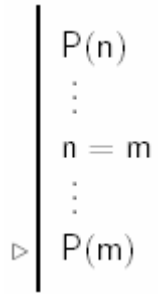
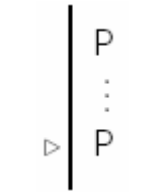
Repare que as justificativas que aparecem do lado direito das sentenças abaixo da barra Fitch representam as regras sobre a igualdade que introduzimos na seção 2.2.1.

Os números à direita da indicação **=Elim**, na justificativa do passo 3, indicam que este passo se segue dos passos 2 e 1 através da utilização da regra citada (eliminação da igualdade).

Regras (=Intro), (=Elim) e (Reit)

Na tabela abaixo, são apresentadas as primeiras regras do sistema $\mathcal{F}$, com a notação que utilizaremos para indicá-las, e uma descrição de sua utilização.

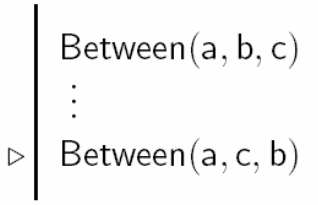
O símbolo ► indicará exatamente qual é o passo que está sendo autorizado pela regra.

Primeiras Regras do Sistema Formal $\mathcal{F}$		
Nome	**Notação**	**Descrição**
(=Intro)		• Regra que permite a introdução da asserção **n = n**, para qualquer nome ou termo complexo **n** em uso em uma prova. • À direita de um passo autorizado por esta regra escrevemos: "**=Intro**". • Note que não há necessidade de citar nenhum passo anterior na justificativa.
(=Elim)		• Se já provamos uma sentença contendo o nome **n** (o que será indicado escrevendo-se **P(n)**) e uma sentença da forma **n=m**, então temos justificativa para afirmar qualquer sentença que resulte de **P(n)** pela substituição de algumas ou todas ocorrências de **n** por **m**. Esta nova sentença será indicada por **P(m)**. • Resumo: Permite substituir termos idênticos nas sentenças de uma prova. • Não importa quem vem primeiro na prova **P(n)** ou **n**, desde que ambos ocorram antes de **P(m)**. • À direita de um passo autorizado por esta regra escrevemos: "**=Elim: *j*, *k***", onde *j* é o número do passo da sentença **P(n)** e *k* o número do passo da sentença **n = m**.
(Reit)		• Permite repetir sentenças nas provas. • À direita de um passo autorizado por esta regra nós escrevemos: "**Reit: *x***", onde *x* é o número da ocorrência anterior da sentença que estamos repetindo. • Esta regra não é tecnicamente necessária, mas sua utilização fará algumas provas parecerem mais naturais.

Dependências Lógicas de Predicados não Constituem Regras Formais

Poderíamos incluir em nossos sistemas formais regras que expressassem dependências lógicas dos outros predicados, além da identidade, que podemos criar em FOL.

Por exemplo, poderíamos criar uma regra que expressasse a bidirecionalidade do predicado **Between** da linguagem dos blocos. Tal regra seria como:



Não faremos isso, simplesmente porque o número de predicados possíveis com as mais variadas dependências lógicas que podemos criar em **FOL** é infinito.

Precisaríamos de um número infinito de regras para expressar todas as dependências lógicas possíveis para todas as linguagens de **FOL**.

Estas dependências lógicas serão tratadas de uma outra maneira, que veremos mais adiante.

Exemplo 1

Suponha que queiramos provar **SameRow(b, a)** a partir das premissas **SameRow(a, a)** e **a = b**.

Nossa prova poderia ser a seguinte:

1. **SameRow(a, a).**
2. **a = b**

---

3. **SameRow(b, a)** (=Elim: 1, 2)

Exemplo 2

Suponha agora que queiramos provar a mesma sentença **SameRow(b, a)** só que desta vez a partir das premissas **SameRow(a, a)** e **b = a**.

A situação é quase idêntica ao exemplo anterior. A diferença é que lá a segunda premissa é **a = b** e aqui é **b = a**.

A princípio pode parecer que a prova deste segundo exemplo deve ser similar à anterior, com apenas uma aplicação da regra **(=Elim)**.

Note, no entanto, que a formulação que fizemos da regra **(=Elim)** indica que podemos derivar sentenças com ocorrências do termo do lado direito da identidade (**a** neste exemplo, pois a premissa é **b = a**), a partir sentenças com ocorrências do termo do lado esquerdo da identidade (**b**).

No entanto, o exemplo nos dá **SameRow(a, a)** e **b = a** e pede que derivemos **SameRow(b, a)**.

Para podermos aplicar a regra **(=Elim)** teríamos que ter uma igualdade **a = b**, mas temos **b = a**.

Como resolver este problema?

É simples. Em primeiro lugar, já vimos que a identidade é simétrica, ou seja, se **a = b**, é claro que **b = a**.

Em segundo lugar, vimos acima uma prova formal deste princípio. Logo, para construir a prova que queremos temos, primeiramente, que "repetir" a prova da simetria da identidade, obtendo **a = b** a partir de **b = a** e em seguida aplicar a regra de **(=Elim)**.

A prova ficaria como:

1.	**SameRow(a, a).**	
2.	**b = a**	
3.	**b = b**	(=Intro)
4.	**a = b**	(=Elim: 3, 2)
5.	**SameRow(b, a)**	(=Elim: 1, 4)

Este exemplo mostra muito bem como os sistemas formais são rígidos e muitas vezes chatos, e mostra também porque, na maioria das vezes, é mais fácil (e mais natural) fazermos provas informais do que provas formais.

2.4 Construindo provas com o Programa Fitch

A nossa sorte é que utilizaremos o Programa Fitch para construir provas formais, que ameniza um pouco esta rigidez dos sistemas formais, tornando a tarefa de construir e verificar provas um pouco mais agradável.

O Programa Fitch

Além de tornar menos dolorosa a tarefa de construir provas formais, o programa Fitch também verifica as suas provas, indicando-lhe se elas estão corretas ou não, e caso não estejam, indicando exatamente quais os passos em que você cometeu erros.

Programa Fitch x Sistema $\mathcal{F}$

O programa Fitch é mais flexível que o sistema $\mathcal{F}$. Ele permite que você tome certos atalhos que são logicamente corretos mas que, estritamente falando, não seguem exatamente as regras do sistema $\mathcal{F}$.

Um destes atalhos, por exemplo, é que você pode utilizar a regra **(=Elim)** nos dois sentidos da identidade, tornando a prova do Exemplo 2 acima tão simples quanto a do Exemplo 1.

Para se familiarizar com o programa Fitch, faça os exercícios **Tente Isto** da Lista de Exercícios 2.

Conseqüência Analítica

Como já mencionamos, o sistema de provas $\mathcal{F}$ não tem regras para dependências lógicas de predicados (além das regras da igualdade).

O programa Fitch, no entanto, apesar de também não ter estas regras, possui um mecanismo que, entre outras coisas, permite-nos verificar, para alguns predicados da linguagem de blocos[1], se sentenças atômicas são conseqüência de outras sentenças atômicas.

Por exemplo, este mecanismo nos permitirá inferir a sentença **Larger(a, c)** a partir das sentenças **Larger(a, b)** e **Larger(b, c)**, uma vez que a relação " ser maior que", expressada pelo predicado **Larger**, é transitiva e, portanto, qualquer mundo que torne as duas últimas verdadeiras tornará também a primeira.

Regra (AnaCon)

Chamaremos a este mecanismo de regra **AnaCon**. Ela nos permitirá citar algumas sentenças como suporte de uma afirmação se qualquer mundo que torne as sentenças citadas verdadeiras, torne também a conclusão verdadeira, em virtude do significado do predicado usado no Mundo de Tarski.

Exemplo. Uma prova de **Larger(a, c)** a partir das premissas **Larger(a, b)** e **Larger(b, c)** utilizando o mecanismo **AnaCon** seria como:

1. **Larger(a, b)**
2. **Larger(b, c)**
3. **Larger(a, c)** (AnaCon: 1, 2)

Restrições no Uso dos Mecanismos Con

AnaCon não é de fato uma regra do sistema $\mathcal{F}$, mas um mecanismo do programa Fitch que nos ajuda, entre outras coisas que veremos mais adiante, a obter conseqüências de sentenças atômicas.

Por causa disso você jamais deve utilizar **AnaCon** em seus exercícios a menos que o enunciado explicitamente indique que você deva.

Se tiver dúvidas sobre isto, quando estiver fazendo um exercício que envolva o programa Fitch, você pode clicar no botão ***Goal Constraints*** para verificar o tipo de regras e mecanismos que podem ser usados para resolver o exercício em questão.

LEMBRE-SE

- O sistema dedutivo que você está aprendendo é um sistema dedutivo do estilo Fitch, chamado $\mathcal{F}$.
- A aplicação de computador que auxilia você na construção de provas em $\mathcal{F}$ é, por isso, chamada de programa Fitch.
- Quando você escreve suas provas no papel, você está utilizando o sistema $\mathcal{F}$, quando o faz no computador, você está utilizando o programa Fitch.

2.5 Demonstrando Não-conseqüência

As provas podem ter diferentes formas.

Provas de Conseqüência

Quando um promotor prova que o réu é culpado, ele está provando que uma particular afirmação ***se segue*** de certas informações aceitas (as provas).

Provas de Não-conseqüência

Quando um advogado de defesa mostra que o crime poderia ter sido cometido por uma outra pessoa além de seu cliente, o advogado está tentando provar que a culpa do réu ***não se segue*** das evidências que se tem no caso.

Uma prova de não-conseqüência para um dado argumento procura estabelecer que seria possível a conclusão ser falsa mesmo que todas as premissas fossem verdadeiras.

[1] Este mecanismo não funciona para os predicados **Adjoins** e **Between**, devido à complexidade dos modos em que os significados destes predicados interagem com os outros. Apesar de perfeitamente possível, seria computacionalmente muito complexo implementar tal mecanismo para estes predicados.

Os sistemas formais e suas regras representam métodos para provar conseqüência, ou seja, métodos para provar que um argumento é logicamente válido, que sua conclusão é conseqüência lógica de suas premissas.

Veremos nesta seção o método mais importante para demonstrar não-conseqüência, ou seja, para demonstrar que uma determinada sentença não é conseqüência lógica de um conjunto de premissas.

De acordo com a definição de validade que apresentamos, um argumento é válido se qualquer circunstância que torne suas premissas verdadeiras também torna sua conclusão verdadeira.

Em outras palavras, um argumento é ***inválido*** se existe alguma circunstância que torna as premissas verdadeiras e a conclusão falsa.

Contra-exemplos

Assim, para mostrar que uma sentença **Q** não é conseqüência das sentenças $\mathbf{P_1}$, ..., $\mathbf{P_n}$, devemos mostrar que o argumento cujas premissas são $\mathbf{P_1}$, ..., $\mathbf{P_n}$ e a conseqüência é **Q** é inválido.

Isso requer que demonstremos que é possível que as sentenças $\mathbf{P_1}$, ..., $\mathbf{P_n}$ sejam todas verdadeiras e a conclusão **Q** seja falsa.

Temos então que mostrar que existe uma situação possível (ou circunstância) em que isso ocorra: as premissas são todas verdadeiras e a conclusão é falsa.

Tal circunstância é chamada de um ***contra-exemplo*** para o argumento.

Provas Informais de Não-conseqüência

Devemos, simplesmente, descrever claramente o que seria uma situação possível na qual as premissas são verdadeiras e a conclusão falsa.

Esta é a tática usada por advogados de defesa, que tenta criar uma dúvida razoável de que seu cliente seja culpado (a conseqüência do promotor), a despeito da evidência que se tenha para o caso (as premissas do promotor)

Considere, como exemplo, o seguinte argumento:

Maluf é um político
Raramente os políticos são homestos
∴ Maluf é desonesto

Se as premissas deste argumento forem verdadeiras, a conclusão é provavelmente verdadeira.

Mas mesmo assim o argumento não é logicamente válido: a conclusão não é uma conseqüência lógica de suas premissas.

Como podemos ver isso?

Imagine a seguinte situação: uma convenção em que estejam 10.000 políticos e que Maluf seja o único político homesto entre todos nesta convenção.

Nesta situação as duas premissas são verdadeiras mas a conclusão é falsa.

Esta situação é um contra-exemplo para o argumento. Ela demonstra que o argumento é inválido.

O que apresentamos foi nada mais do que uma prova informal de não-conseqüência.

Provas Formais de Não-Conseqüência

Será que há provas formais de não-conseqüência similares às provas formais de validade que podemos construir no sistema $\mathcal{F}$? **Em geral não**.

No entanto, definiremos a noção de provas formais de não-conseqüência para a linguagem dos blocos.

Para a linguagem dos blocos, diremos que uma prova formal de que **Q** não é uma conseqüência de $\mathbf{P_1}$, ..., $\mathbf{P_n}$, consistirá de um arquivo de sentenças com $\mathbf{P_1}$, ..., $\mathbf{P_n}$ rotuladas como premissas, e a sentença **Q** rotulada como conclusão, e um arquivo de mundo que faça com que $\mathbf{P_1}$, ..., $\mathbf{P_n}$ sejam todas verdadeiras e **Q** seja falsa.

O mundo descrito no arquivo de mundo será chamado de contra-exemplo para o argumento do arquivo de sentenças.

LEMBRE-SE

Para demonstrar a invalidade de um argumento com premissas P_1, ..., P_n e conclusão **Q**, encontre um contra-exemplo: uma circunstância possível em que P_1, ..., P_n sejam todas verdadeiras, mas **Q** seja falsa. Tal contra-exemplo mostra que **Q** não é conseqüência de P_1, ..., P_n.

Capítulo 3 - Os Conectivos Booleanos

Até agora estudamos apenas as sentenças atômicas. Para construir afirmações complexas, FOL dispõe de conectivos e quantificadores.

Conectivos Booleanos

Neste capítulo abordaremos os três conectivos mais simples: ***conjunção***, ***disjunção*** e ***negação***, que correspondem aos usos simples das expressões portuguesas: "*e*", "*ou*" e "*não é o caso que*".

George Boole foi o primeiro lógico a estudá-los sistematicamente no início do século XIX. Daí o nome *booleanos*.

Conectivos Verofuncionais

Os conectivos booleanos são chamados de verofuncionais porque o valor de verdade de uma sentença complexa construída pelo uso destes conectivos depende apenas do valor de verdade das sentenças mais simples utilizadas na construção da sentença complexa.

Matematicamente, eles podem ser descritos como funções de verdade (verofuncionais).

Tabelas de Verdade

Por serem funções de verdade, os conectivos booleanos podem ser descritos através de uma *tabela de verdade* que mostra como o valor de verdade de uma sentença construída com o conectivo depende dos valores de verdade de suas partes imediatas.

O Jogo de Henkin-Hintikka

Uma forma interessante de entender o significado dos conectivos booleanos é através de um jogo, conhecido como o de Henkin-Hintikka.

O jogo é simples. Imagine que duas pessoas, por exemplo, Maria da Paz e Daniel, descordem sobre o valor de verdade de uma sentença complexa. Da Paz acha que a sentença é verdadeira e Daniel acha que ela é falsa.

Os dois se desafiam repetidamente a justificar suas afirmações em termos de afirmações menores, componentes da afirmação inicial, até que a discórdia se reduza ao valor de verdade de uma sentença atômica.

Neste ponto eles podem examinar o mundo e verificar se sentença atômica é verdadeira ou falsa, resolvendo a discórdia.

Mais adiante descreveremos detalhadamente como o jogo funciona. Há uma implementação dele no software Mundo de Tarski, onde você pode jogar com o computador.

Símbolo de Negação: ¬

O símbolo ¬ será usado para expressar negação em nossa linguagem (FOL).

Em lógica de primeira ordem, sempre aplicaremos este símbolo na frente da sentença que queremos negar.

Traduzimos a sentença "*John não está em casa*", em FOL, por uma do tipo: **¬Home(john)**.

Esta sentença será verdadeira se e somente se **Home(john)** não for verdadeira. Ou seja, exatamente no caso em que John não está em casa.

Você pode por um símbolo de negação na frente de qualquer sentença. Inclusive de uma sentença que já contenha uma negação. Por exemplo, a sentença **¬¬Home(john)** nega a sentença **¬Home(john)**. Portanto, (a primeira) será verdadeira se e somente se John estiver em casa.

Literais

Diremos que uma sentença é um *literal* se ela for uma sentença atômica ou a negação de uma sentença atômica.

Assim, **Home(john)** e **¬Home(john)** são literais, mas **¬¬Home(john)** não é.

Símbolo de não-identidade: (≠)

Considere uma sentença atômica que afirma uma identidade, tal como **a = b**. A negação desta sentença, **¬(a = b)**, poderá ser abreviada através do símbolo de não-identidade ≠ por: **a ≠ b**.

A Semântica da Negação

Dada qualquer sentença **P** de FOL (atômica ou complexa) existe uma outra sentença **¬P** cujo valor de verdade será sempre o oposto do valor de **P**.

A seguinte tabela mostra isso:

Tabela de Verdade para a Negação

P	¬P
TRUE	FALSE
FALSE	TRUE

Regra para a Negação no Jogo Henkin-Hintikka

A regra é muito simples, pois não há muito a fazer neste caso. Mais adiante veremos casos bem mais interessantes.

Uma vez que você se compromete com a verdade de **¬P**, isto é o mesmo que se comprometer com a falsidade de **P**. Similarmente se você se compromete com a falsidade de **¬P**, é o mesmo que se comprometer com a verdade de **P**.

Faça o primeiro "Tente Isto" da lista de exercícios 3 para testar por si mesmo o jogo com sentenças negadas.

LEMBRE-SE

1. Se **P** é uma sentença de FOL, então **¬P** também é.
2. A sentença **¬P** é verdadeira se e somente se **P** não for.
3. Uma sentença que é ou atômica ou a negação de uma sentença atômica é chamada de ***literal***.

Símbolo de Conjunção: ∧

O símbolo ∧ será usado para expressar conjunção em nossa linguagem (FOL).

Conjunção é a noção que normalmente expressamos em português através da palavra "***e***".

Além de "*e*", expressões como: "*mas*", "*além disso",* entre outras, também expressam a noção de conjunção.

Na lógica, este conectivo é sempre colocado entre duas sentenças. Enquanto que em português sua utilização é mais livre.

Por exemplo:

João e Maria estão em casa
João está em casa e Maria está em casa

Estas duas sentenças têm a mesma tradução em FOL:

Home(joao) ∧ Home(maria)

Esta sentença será verdadeira se e somente se João está em casa e Maria está em casa. Outro exemplo. A sentença

João dormiu e caiu

Traduz-se em FOL por:

Dormiu(joao) ∧ Caiu(joao)

Esta sentença será verdadeira apenas se as sentenças atômicas **Dormiu(joao)** e **Caiu(joao)** forem ambas verdadeiras.

Muitas vezes uma sentença de FOL conterá o símbolo ∧ mesmo quando não há nenhum sinal visível de conjunção na sentença em português correspondente.

Como você acha que a sentença do português "*d é um cubo grande*" seria na Linguagem dos Blocos? Se você sugeriu

$$\text{Large(d)} \wedge \text{Cube(d)}$$

você acertou. Esta sentença será verdadeira se e somente se **d** for grande e **d** for um cubo. Ou seja, se **d** for um cubo grande.

Alguns usos da palavra "*e*" em português **não** são espelháveis com precisão pelo símbolo de conjunção de FOL.

Por exemplo, suponha que estejamos falando de uma noite em que Max e Claire estavam juntos (tendo um encontro!).

Se disséssemos *Max foi para casa e Claire foi dormir*, nossa asserção poderia estar carregando uma implicação temporal. A de que Max foi para casa *antes* de Claire ir dormir.

Similarmente, se utilizássemos a ordem reversa da sentença: *Claire foi dormir e Max foi para casa*, isto poderia sugerir uma situação bastante diferente.

No entanto, nenhuma dessas implicações (implícitas ou explícitas) está presente quando usamos o símbolo $\wedge$ na sentença:

$$\text{FoiCasa(max)} \wedge \text{FoiDormir(claire)}$$

Esta sentença é verdadeira exatamente nas mesmas circunstâncias que

$$\text{FoiDormir(claire)} \wedge \text{FoiCasa(max)}$$

A Semântica da Conjunção

Considere **P** e **Q** duas sentenças quaisquer de FOL. Note que **P** e **Q** podem ser simples (atômicas) ou complexas.

Uma sentença $\mathbf{P} \wedge \mathbf{Q}$ é verdadeira se e somente se **P** e **Q** forem ambas verdadeiras. Deste modo, $\mathbf{P} \wedge \mathbf{Q}$ é falsa se e somente se ou alguma das duas ou ambas forem falsas.

Isto pode ser resumido pela seguinte tabela:

Tabela de Verdade para a Conjunção

P	Q	P ∧ Q
T	T	T
T	F	F
F	T	F
F	F	F

Regra para a Conjunção no Jogo Henkin-Hintikka

Aqui o jogo do Mundo de Tarski é um pouco mais interessante.

Se você se comprometer com a verdade de $\mathbf{P} \wedge \mathbf{Q}$ então você, implicitamente se comprometeu com a verdade de cada uma das sentenças **P** e **Q**. O Mundo de Tarski, então, escolherá uma dessas duas sentenças mais simples (**P** ou **Q**) e lhe pedirá que confirme seu compromisso com sua verdade.

Qual sentença o programa escolherá? Bem, se houver alguma falsa, ele escolherá esta, pois tenta ganhar o jogo de você. Se as duas forem verdadeiras, ele escolhe uma aleatoriamente e continua jogando, na esperança de que você cometa um erro mais adiante.

Se você se comprometer com a falsidade de $\mathbf{P} \wedge \mathbf{Q}$, então você considera que pelo menos uma delas **P** ou **Q** é falsa. Neste caso, o Mundo de Tarski pedirá que você escolha qual delas considera falsa. Se a sentença que escolheu não for falsa, você poderá perder o jogo.

Faça o "Tente Isto 2" da lista de exercícios 3 para testar por si mesmo como o jogo funciona com a conjunção.

LEMBRE-SE

1. Se **P** e **Q** são sentenças de FOL, então $\mathbf{P} \wedge \mathbf{Q}$ também é.
2. A sentença $\mathbf{P} \wedge \mathbf{Q}$ é verdadeira se e somente se **P** e **Q** forem ambas verdadeiras.

Símbolo de Disjunção: $\vee$

O símbolo $\vee$ será usado para expressar disjunção em nossa linguagem (FOL).

Disjunção é a noção que normalmente expressamos em português pela palavra "***ou***".

Assim como a conjunção, em FOL este conectivo é sempre colocado entre duas sentenças. Enquanto que em português sua utilização é mais livre.

Por exemplo:

João ou Maria estão em casa
João está em casa ou Maria está em casa

Estas duas sentenças têm a mesma tradução em FOL:

$$\text{Home(joao)} \vee \text{Home(maria)}$$

Disjunção Exclusiva x Disjunção Inclusiva

O ***ou*** do português algumas vezes é usado com um sentido "exclusivo", para dizer que *exatamente* uma (isto é uma mas não mais do que uma) das partes de uma disjunção é verdadeira.

Em lógica, no entanto, $\vee$ sempre tem uma interpretação "inclusiva". Isto significa que pelo menos uma e, possivelmente ambas as sentenças de uma disjunção são verdadeiras.

Assim, no exemplo acima a sentença é verdadeira se João está em casa e Maria não, se Maria está em casa e João não, e se ambos, Maria e João estão em casa.

Se quiséssemos expressar o sentido exclusivo do *ou* no exemplo acima, poderíamos escrever a seguinte sentença em FOL:

$$(\text{Home(joao)} \vee \text{Home(maria)}) \wedge \neg(\text{Home(joao)} \wedge \text{Home(maria)})$$

A Semântica da Disjunção

Considere **P** e **Q** duas sentenças quaisquer de FOL (atômicas ou não). Podemos combiná-las utilizando $\vee$ para formar uma nova sentença $\mathbf{P} \vee \mathbf{Q}$.

$\mathbf{P} \vee \mathbf{Q}$ será verdadeira se pelo menos uma das sentenças **P** ou **Q** for verdadeira. Caso contrário, será falsa.

Isto pode ser resumido pela seguinte tabela:

Tabela de Verdade para a Disjunção

P	Q	P ∨ Q
T	T	T
T	F	T
F	T	T
F	F	F

Regra para a Disjunção no Jogo Henkin-Hintikka

As regras do jogo do Mundo de Tarski para $\vee$ são exatamente o reverso das regras para $\wedge$.

Se você se comprometer com a verdade de $\mathbf{P} \vee \mathbf{Q}$, você implicitamente está se comprometendo com a verdade de pelo menos uma das sentenças **P** ou **Q**. O mundo de Tarski, então, solicita que você escolha qual das duas considera verdadeira.

Se você se comprometer com a falsidade de $\mathbf{P} \vee \mathbf{Q}$, então você implicitamente está se comprometendo com a falsidade de ambas as sentenças. O mundo de Tarski escolherá uma delas e lhe pedirá que confirme o seu compromisso com a falsidade desta sentença. (Claro que o programa, tentando ganhar o jogo, escolherá uma sentença que seja verdadeira, caso haja alguma)

Faça o "Tente Isto 3" da lista de exercícios 3 para testar por si mesmo como o jogo funciona com a disjunção.

LEMBRE-SE

5. Se **P** e **Q** são sentenças de FOL, então **P** ∨ **Q** também é.
6. A sentença **P** ∨ **Q** é verdadeira se e somente se **P** é verdadeira ou **Q** é verdadeira (ou ambas são verdadeiras).

A tabela abaixo resume as regras do jogo de Henkin-Hintikka para os conectivos ¬, ∧ e ∨.

forma da sentença	seu compromisso	Quem joga	a jogada
P ∨ **Q**	TRUE	Você	Escolher uma das sentenças **P** ou **Q** que seja verdadeira.
	FALSE	o programa	
P ∧ **Q**	TRUE	o programa	Escolher uma das sentenças **P** ou **Q** que seja falsa.
	FALSE	Você	
¬**P**	qualquer que seja	-	Substituir ¬**P** por **P** e trocar o compromisso.

Ambigüidade e Parênteses

A sentença:

Max está em casa ou Claire está em casa e Carl está feliz.

pode ser interpretada de duas formas bem diferentes. Em uma delas a sentença afirma que:

Ou Claire está em casa e Carl está feliz, ou max está em casa.

Outra interpretação para a sentença seria:

Max ou Claire estão em casa e Carl está feliz.

Considere a situação (circunstância) em que Max está em casa e Carl está infeliz.

Nesta circunstância, a primeira interpretação da sentença é verdadeira e a segunda é falsa. Você percebe isso?

Este tipo de ambigüidade é evitado em FOL através da utilização dos parênteses. Assim, em FOL não há uma sentença com dupla interpretação, mas duas sentenças diferentes:

Home(max) ∨ (Home(claire) ∧ Happy(carl))
(Home(max) ∨ Home(claire)) ∧ Happy(carl)

Os parênteses na primeira indicam que a sentença é uma disjunção cujo segundo disjunto é uma conjunção. Na segunda, os parênteses indicam que a sentença é uma conjunção cujo primeiro "conjunto" é uma disjunção.

Esta diferença é a mesma que encontramos em expressões aritméticas como:

2 + (3 x 5)
(2 + 3) x 5

Estas são claramente duas expressões diferentes. O resultado da primeira é 17 e o da segunda é 25.

Em lógica os parênteses evitam ambigüidade da mesma forma que nas expressões algébricas da matemática. Em FOL, além dos parênteses, podemos usar colchetes "[", "]" e chaves "{", "}".

Escopo da Negação

Os parênteses também são utilizados para indicar o "escopo" de um símbolo de negação quando ele aparece em uma sentença complexa.

As duas sentenças abaixo, por exemplo, significam coisas bem diferentes.

¬Home(claire) ∧ Home(max)
¬[Home(claire) ∧ Home(max)]

Economizando Parênteses

A primeira sentença acima é uma conjunção de literais, o primeiro deles declarando que Claire não está em casa e o segundo que Max está em casa. Já a segunda sentença é a negação de uma sentença que é, ela própria, uma conjunção. Ela declara que Claire e Max não estão ambos em casa.

Muitos livros de lógica exigem que você coloque parênteses em volta de qualquer par de sentenças unidas por um conectivo binário (tal como $\wedge$ ou $\vee$). Nestes livros, uma sentença da forma

$$P \wedge Q \wedge R$$

não está bem escrita. Seria necessário reescrevê-la com uma das formas abaixo:

$$((P \wedge Q) \wedge R)$$
$$(P \wedge (Q \wedge R))$$

A versão de FOL que estamos usando neste livro não é tão exigente.

Em primeiro lugar, ela permite que você faça conjunções de qualquer número de sentenças sem utilizar parênteses, desde que o resultado não seja ambíguo, e o mesmo vale para disjunções.

Em segundo lugar, ela permite que você deixe de fora os parênteses mais externos de uma sentença, uma vez que eles não servem a nenhum propósito.

Você, no entanto, pode adicionar parênteses extras o quanto quiser, para facilitar a leitura das sentenças.

> **LEMBRE-SE**
>
> Os parênteses devem ser utilizados sempre que possa haver ambigüidade caso sejam omitidos. Na prática, isto significa que conjunções e disjunções precisam ser envolvidas por parênteses sempre que estiverem combinadas com outros conectivos.

Formas Equivalentes de Dizer as Coisas

Em qualquer linguagem há muitas formas de dizer as mesmas coisas. FOL não é uma exceção a esta regra.

Por exemplo, a linguagem dos blocos não perderia seu poder expressivo se retirássemos dela o predicado **RightOf**, pois tudo o que podemos dizer através do predicado **RightOf** pode ser dito de outra forma com o predicado **LeftOf**.

As sentenças **RightOf(a, b)** e **LeftOf(b, a)**, por exemplo, representam duas formas diferentes de dizer a mesma coisa na linguagem de blocos.

As sentenças complexas $P \wedge Q$ e $Q \wedge P$ também expressam a mesma coisa em qualquer linguagem de primeira ordem.

Leis de DeMorgan

O exercício 3.16 (da Lista 3) ilustra exemplos mais interessantes de maneiras diferentes de dizer as mesmas coisas. São equivalências conhecidas como Leis de DeMorgan:

$\neg(P \wedge Q)$ expressa exatamente a mesma coisa que (é logicamente equivalente a) $\neg P \vee \neg Q$
$\neg(P \vee Q)$ é logicamente equivalente a $\neg P \wedge \neg Q$

Estas leis são simples conseqüências dos significados dos conectivos booleanos ($\neg$, $\wedge$ e $\vee$).

Se escrevermos $S_1 \Leftrightarrow S_2$ para indicar que S_1 e S_2 são logicamente equivalentes, podemos expressar as leis de DeMorgan da seguinte maneira:

$$\neg(P \wedge Q) \Leftrightarrow (\neg P \vee \neg Q)$$
$$\neg(P \vee Q) \Leftrightarrow (\neg P \wedge \neg Q)$$

Dupla Negação

Há muitas outras equivalências que decorrem dos significados dos conectivos booleanos.

Talvez a mais simples seja a que é conhecida como princípio da *dupla negação*, que diz que uma sentença da forma $\neg\neg P$ é equivalente à sentença P.

No próximo capítulo estas e outras equivalências serão sistematicamente discutidas. Por enquanto, queremos apenas que você as perceba, antes de continuarmos.

LEMBRE-SE

(Dupla Negação e Leis de DeMorgan) Para quaisquer sentenças **P** e **Q**:

1. Dupla negação: $\neg\neg P \Leftrightarrow P$
2. DeMorgan: $\neg(P \wedge Q) \Leftrightarrow (\neg P \vee \neg Q)$
3. DeMorgan: $\neg(P \vee Q) \Leftrightarrow (\neg P \wedge \neg Q)$

Tradução

Estamos construindo uma linguagem artificial para estudar lógica (FOL). Por isso, uma habilidade essencial para a utilização da lógica é a de traduzir sentenças do português para FOL e vice versa.

Tradução Correta

Intuitivamente, uma tradução correta é uma sentença com o mesmo significado da que está sendo traduzida. Mas o que é significado?

FOL lida com esta questão reduzindo a noção de significado à noção de "***condições de verdade***".

O que é necessário para uma tradução correta em FOL é que a sentença traduzida seja verdadeira exatamente nas mesmas circunstâncias que a sentença original.

Se duas sentenças são verdadeiras exatamente nas mesmas circunstâncias, dizemos que elas têm as mesmas ***condições de verdade***.

Para as sentenças do Mundo de Tarski, isto significa que as sentenças serão verdadeiras exatamente nos mesmos mundos. (qualquer mundo que torne uma delas verdadeira, torna verdadeira também a outra)

LEMBRE-SE

Para que uma sentença de FOL seja uma boa tradução de uma sentença do português, é suficiente que as duas sentenças tenham os mesmos valores de verdade em todas as circunstâncias possíveis, ou seja, que elas tenham as mesmas ***condições de verdade***.

Diante disso, dada uma sentença do português $\mathcal{S}$ e uma boa tradução dela em FOL, digamos **S**, qualquer outra sentença **S'** que for equivalente a **S** também será uma tradução aceitável de $\mathcal{S}$, uma vez que **S** e **S'** têm as mesmas condições de verdade.

Mas há uma questão de estilo. Algumas traduções aceitáveis são melhores do que outras. Queremos sentenças que mantenham os conectivos de FOL tão próximos quanto possível da sentença em português.

Por exemplo, uma boa tradução da sentença:

Não é verdade que Claire e Max estão ambos em casa.

poderia ser:

$$\neg(\text{Home(claire)} \wedge \text{Home(max)})$$

Esta sentença (de acordo com a primeira lei de DeMorgan) é equivalente à sentença abaixo, que portanto também é uma tradução aceitável da sentença em inglês:

$$\neg\text{Home(claire)} \vee \neg\text{Home(max)}$$

No entanto, é claro que o estilo da primeira tradução é melhor, pois se aproxima mais da sentença original. Afinal de contas a sentença original está escrita como uma negação de uma conjunção e não como uma disjunção de negações.

Não há regras exatas que determinam a melhor tradução de uma sentença. Além disso, há muitas sutilezas estilísticas do português que nada têm a ver com as condições de verdade de uma sentença e, portanto, não podem ser capturadas por uma tradução em FOL.

Veja o seguinte exemplo:

Carl está com fome, mas está feliz.

Esta sentença nos diz duas coisas, que Carl está com fome e que ele está feliz. Então, poderia ser traduzida em FOL por:

$$\text{Hungry(carl)} \wedge \text{Happy(carl)}$$

Mas, porém, contudo, todavia, no entanto

Quando consideramos apenas as condições de verdade, "***mas***" expressa a mesma função de verdade que "***e***". No entanto, é claro que "***mas***" carrega uma sugestão adicional que "***e***" não carrega.

É a sugestão de que o ouvinte poderia ficar um pouco surpreso com a segunda parte da sentença, dadas as expectativas criadas pela primeira parte.

No exemplo acima, mesmo estando com fome, Carl está feliz. A palavra "***mas***" expressa de alguma forma esta sutil contrariedade. A palavra "***e***" não expressa. No entanto, é claro que as duas sentenças abaixo

Carl está com fome mas está feliz.
Carl está com fome e feliz.

possuem as mesmas condições de verdade, ou seja, são verdadeiras nas mesmas circunstâncias e, portanto, em FOL seriam adequadamente traduzidas por:

$$\text{Hungry(carl)} \wedge \text{Happy(carl)}$$

Ou...ou , Ambos

As expressões do português *ou...ou* e *ambos* nos ajudam, muitas vezes, a evitar ambigüidades em sentenças complexas. Elas funcionam de modo parecido aos parênteses combinados com $\vee$ e $\wedge$ em FOL.

Por exemplo, a sentença:

Ou Max está em casa e Claire está em casa, ou Carl está feliz.

Não é ambígua. Uma tradução para FOL poderia ser:

$$[\text{Home(max)} \wedge \text{Home(claire)}] \vee \text{Happy(carl)}$$

Se não houvesse o *ou* inicial, a sentença seria ambígua da mesma forma que no caso discutido anteriormente.

A sentença abaixo, por exemplo, também não é ambígua, sendo traduzida em FOL da mesma forma que a anterior.

Max e Claire estão ambos em casa, ou Carl está feliz.

Neste caso, é a expressão *ambos* que suprime a ambiguidade.

LEMBRE-SE

1. A expressão "***e***" do português algumas vezes sugere ordem temporal. A expressão $\wedge$ de FOL nunca dá este tipo de sugestão.
2. As expressões do português "***mas***", "***porém***", "***todavia***", "***contudo***", "***entretanto***", "***no entanto***" são todas variantes estilizadas da expressão "***e***".
3. As expressões do português "***ou ... ou***" e "***ambos***" são frequentemente usadas como parênteses para evitar possíveis ambigüidades das sentenças.

Capítulo 4 - A Lógica dos Conectivos Booleanos

Os conectivos $\wedge$, $\vee$ e $\neg$ são funções de verdade (*conectivos verofuncionais*). Isso, como já dissemos, significa que o valor de verdade de uma sentença complexa construída com estes símbolos pode ser completamente determinado pelos valores de verdade das sentenças mais simples que a constituem.

Assim, para saber se $P \vee Q$ é verdadeira, precisamos apenas saber os valores de verdade de **P** e de **Q**.

Operadores Verofuncionais x Operadores Não-Verofuncionais

Outros conectivos que poderíamos estudar, não são tão simples assim. Considere a sentença:

É necessariamente o caso que **S**.

Uma vez que algumas asserções são necessariamente verdadeiras, ou seja, jamais poderiam ser falsas (**a = a** por exemplo), enquanto outras asserções não são necessariamente verdadeiras (**Cube(a)** por exemplo), nós não poderíamos descobrir o valor de verdade da sentença completa (*É necessariamente o caso que* **S**) sabendo apenas o valor de verdade de **S**.

Portanto, o operador *"é necessariamente o caso"*, diferentemente de *"não é o caso"* ($\neg$), não é uma função de verdade.

Por serem funções de verdade, os conectivos booleanos podem ser estudados por uma técnica bastante simples, a técnica das tabelas de verdade, que são extensões das tabelas que expressam os significados dos conectivos booleanos.

Neste capítulo estudaremos o que as tabelas de verdade podem nos dizer a respeito de quatro importantes noções lógicas:

Conseqüência Lógica ***Necessidade Lógica***
Equivalência Lógica ***Possibilidade Lógica***

Necessidade Lógica

Já dissemos que uma sentença **S** é uma conseqüência lógica de um conjunto de premissas $P_1, \ldots, P_n$ se for impossível a conclusão **S** ser falsa quando todas as premissas são verdadeiras.

Ou seja, a conclusão ***tem que*** ser verdadeira se as premissas forem verdadeiras.

Há, no entanto, algumas sentenças que são conseqüência lógica de ***qualquer*** conjunto de premissas, até do conjunto vazio.

Isso ocorre com qualquer sentença que for uma necessidade lógica, ou seja, que for verdadeira em qualquer circunstância.

Por exemplo, como em FOL uma *constante individual* (nome) não pode se referir a mais de um objeto, a sentença **a = a** é necessariamente verdadeira. Não há nenhuma circunstância logicamente possível que a falsifique.

Imagine um conjunto de sentenças qualquer, e as tome como premissas de um argumento em que **a = a** é conclusão. Não importa quais sejam essas premissas, é impossível que ocorra a situação de todas elas serem verdadeiras e a conclusão **a = a** ser falsa, simplesmente por que é impossível que **a = a** seja falsa.

Verdade Lógica $\Leftrightarrow$ Necessidade Lógica

Sentenças logicamente necessárias também são chamadas de ***verdades lógicas***. Portanto os termos **verdade lógica** e **necessidade lógica** são sinônimos.

É importante não confundir o conceito de "verdade lógica" com o de "ser verdadeira", ou mais precismante, "ter valor de verdade T". Verdade lógica é uma noção absoluta. Independe do mundo, pois são verdades lógicas as sentenças que são verdadeiras em todas as circunstâncias logicamente possíveis. Em todos os mundos possíveis.

Já ter o valor de verdade T (ou ser verdadeira) é uma noção relativa. Uma sentença da linguagem dos blocos, por exemplo, será verdadeira ou falsa apenas quando confrontada com algum mundo.

Justamente para evitar este tipo de confusão o termo necessidade lógica é preferível ao verdade lógica. Mas como os dois são utilizados por aí, é importante conhecê-los.

Possibilidade Lógica e Necessidade Lógica

Intuitivamente, uma sentença é logicamente possível se ela pode ser (ou poderia ter sido) verdadeira. Pelo menos do ponto de vista lógico.

Pode haver outras razões (razões físicas, por exemplo) para que uma sentença seja falsa, mas se não há nenhuma razão lógica, a sentença é logicamente possível.

Por exemplo, não é fisicamente possível mover-se mais rápido que a velocidade da luz, no entanto tal movimento é, sim, logicamente possível. Imaginá-lo não viola nenhum princípio lógico. (O teletransporte da Jornada nas Estrelas !!!)

Por outro lado a sentença **a ≠ a** não é nem logicamente possível, uma vez que **a** é o nome de um e apenas um objeto. Qualquer situação imaginável em que esta sentença fosse verdadeira violaria os princípios lógicos da identidade.

Temos, portanto, as seguintes definições:

> Possibilidade Lógica: uma afirmação é *logicamente possível* se há alguma circunstância logicamente possível (ou situação, ou mundo) na qual a afirmação é verdadeira.
>
> Necessidade Lógica: similarmente, uma sentença é *logicamente necessária* se ela for verdadeira em todas as circunstâncias logicamente possíveis.

Tornando Noções Vagas mais Precisas

Apesar de bastante importantes para a lógica, estas duas noções são irritantemente vagas para os padrões que buscamos para as definições lógicas.

Introduziremos vários conceitos precisos que nos ajudarão a clarificar estas noções. Um deles será a noção de *tautologia*.

Mas como pode um conceito preciso clarificar uma noção intuitiva imprecisa?

Vamos pensar, por um momento, na noção de possibilidade lógica com respeito a sentenças da linguagem dos blocos.

Presumivelmente uma sentença da linguagem dos blocos será logicamente possível se pode haver um mundo (de blocos) no qual ela é verdadeira. Mas alto lá. A coisa não é tão simples assim.

É claro que se nós conseguirmos construir um mundo no **Mundo de Tarski** que torna a sentença verdadeira, então isso demonstra que a sentença é, de fato, logicamente possível.

Por outro lado, há sentenças da linguagem dos blocos que são logicamente possíveis mas que não podem ser tornadas verdadeiras pelos mundos que podemos construir com o programa **Mundo de Tarski**. Por exemplo, a sentença

(1) $\neg$(Tet(b) $\vee$ Cube(b) $\vee$ Dodec(b))

é com certeza *logicamente* possível. Basta que imaginemos que **b** é uma esfera, ou um icosaedro.

No entanto, não conseguimos construir com o **Mundo de Tarski** um mundo que a torne verdadeira, mas isso não é culpa da lógica, da mesma forma que não é culpa da lógica que não se pode viajar mais rápido que a luz.

O **Mundo de Tarski** tem as suas leis e restrições **não-lógicas**, da mesma forma que o mundo físico também tem as suas.

TW-Possibilidade

Uma sentença é ***TW-possível*** se ela for verdadeira em algum mundo que pode ser construído usando o programa **Mundo de Tarski**.[1]

Assim, o que o exemplo acima nos mostra é que qualquer sentença TW-possível é logicamente possível, mas o contrário não é verdade em geral. Algumas sentenças logicamente possíveis não são TW-possíveis. A sentença (1), acima, é um exemplo deste caso.

O programa **Mundo de Tarski** nos dá um método preciso para mostrar que algumas sentenças da linguagem dos blocos são logicamente possíveis, pois o que quer que seja possível no **Mundo de Tarski** é logicamente possível.

Veremos um outro método preciso que poderá nos mostrar se uma sentença construída com conectivos verofuncionais é logicamente necessária.

Tabelas de Verdade e Necessidade Lógica

O método das tabelas de verdades será útil para, entre outras coisas, mostrar que certas sentenças não podem ser falsas devido simplesmente aos significados dos conectivos verofuncionais que elas contém.

O método das tabelas de verdade, para este caso, também funciona apenas em uma direção:

[1] **TW** – vem de "*Tarski's World*", Mundo de Tarski em inglês.

Quando ele diz que uma sentença é logicamente necessária, então ela definitivamente é. Por outro lado, algumas sentenças são logicamente necessárias por razões que o método das tabelas de verdade não consegue detectar.

O Método das Tabelas de Verdade

Suponha que tenhamos uma sentença complexa **S** com *n* sentenças atômicas constituintes A_1, ..., A_n.

Para construir uma tabela de verdade para **S**, escrevemos as sentenças atômicas A_1, ..., A_n na linha do topo de uma página, com a sentença **S** à direita. É costume traçar uma coluna grossa separando as sentenças atômicas de **S**.

A tabela de verdade terá uma linha para cada forma possível de atribuir **TRUE** ou **FALSE** para as sentenças atômicas.

Número de Linhas em uma Tabela de Verdade

Uma vez que há dois valores possíveis para cada sentença atômica (**TRUE** ou **FALSE**), então com *n* sentenças atômicas teremos 2^n linhas, Se n=1, haverá duas linhas, se n=2 haverá 4 linhas, se n=3 haverá 8 linhas e assim por diante.

Preenchendo as Colunas de Referência

As colunas à esquerda, abaixo das sentenças atômicas de uma tabela de verdade, são chamadas de *colunas de referência*.

É costume fazer com que a coluna de referência mais à esquerda tenha o valor **TRUE** para as linhas da metade superior da tabela, e **FALSE** para as linhas da metade inferior. A coluna seguinte divide cada uma destas metades, tendo as linhas do primeiro e terceiro quarto marcadas com **TRUE** e as linhas do segundo e quarto quartos marcadas com **FALSE**.

Este processo de divisão pela metade se repete para as linhas mais à direita, e termina com a última coluna de referência tendo alternadamente os valores **TRUE** e **FALSE**.

Vejamos um exemplo bastante simples. Considere a seguinte sentença: **Cube(a)** ∨ ¬**Cube(a)**.

Uma vez que ela é construída a partir de uma única sentença atômica **Cube(a)** (que se repete duas vezes), a tabela terá apenas duas linhas, uma para cada possível valor de verdade para **Cube(a)**.

Cube(a)	Cube(a) ∨ ¬ Cube(a)
T	
F	

Preenchendo as Colunas dos Conectivos

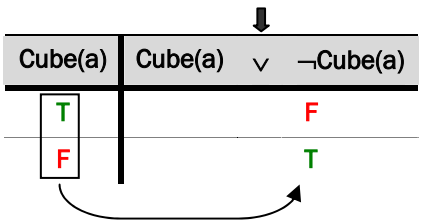
Uma vez que as colunas de referência tenham sido preenchidas, da maneira que descrevemos acima, estamos prontos para preencher o resto da tabela.

Para fazer isso construímos colunas com **T**s e **F**s abaixo de cada conectivo na sentença alvo **S**. Estas colunas são preenchidas uma por uma, utilizando como base as tabelas de verdade para os vários conectivos.

Iniciamos trabalhando com os conectivos que são aplicados apenas às sentenças atômicas e seguimos trabalhando com os conectivos que se aplicam a sentenças cujos conectivos principais já tenham suas colunas preenchidas. Continuamos este processo até que o conectivo principal de **S** tenha sua coluna preenchida.

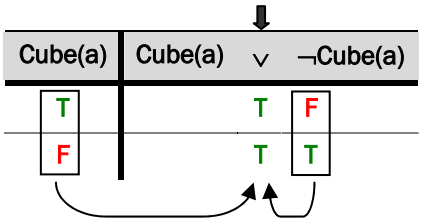
É esta coluna que mostra como o valor de verdade de **S** depende do valor de verdade de suas partes atômicas.

Então, para o nosso exemplo, o primeiro passo será calcular os valores de verdade da coluna do conectivo mais interno, que é, neste caso, a negação (¬). Faremos isso utilizando como referência os valores da coluna **Cube(a)**, trocando-os, de acordo com o significado de ¬.



Com esta coluna preenchida, podemos determinar os valores de verdade que devem ficar abaixo de ∨. Basta consultar os valores sob **Cube(a)** e aqueles abaixo de ¬. Isso porque são eles que correspondem aos valores dos dois disjuntos aos quais ∨ é aplicado. (Você entende isso?)

Como há pelo menos um **T** em cada linha, a coluna final da tabela de verdade (marcada com a seta grossa) fica assim:



Nossa tabela nos mostra que a sentença **Cube(a) ∨ ¬Cube(a)** não pode ser falsa. Ela é o que nós chamaremos de uma ***tautologia***, que é um caso simples de verdade lógica, que mais adiante definiremos com precisão.

Lei do Terceiro Excluído

Nossa sentença é, de fato, uma instância (um caso) de um princípio (**P ∨ ¬P**) que é conhecido como a *lei do terceiro excluído*.

Qualquer instância deste princípio é uma tautologia. Dito em outras palavras, a substituição de **P** por qualquer sentença tem como resultado uma tautologia.

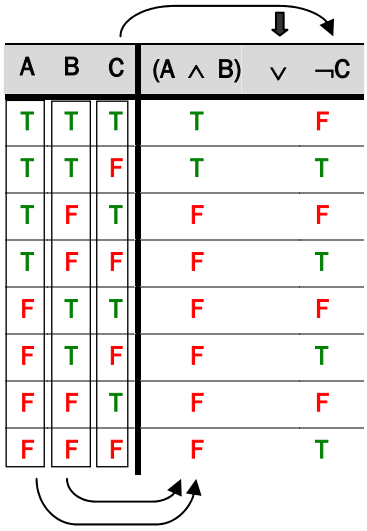
Vejamos uma tabela de verdade mais complexa. Considere a seguinte sentença: **(Cube(a) ∧ Cube(b)) ∨ ¬Cube(c)**

Para simplificar a construção da tabela, abreviaremos as três sentenças atômicas presentes na sentença acima (**Cube(a)**, **Cube(b)** e **Cube (c)**) por **A**, **B** e **C**, respectivamente.

Como são 3 sentenças atômicas, nossa tabela terá 2^3 (ou seja 8) linhas. Examine cuidadosamente a tabela abaixo e perceba que todos os arranjos possíveis para os valores de verdade das sentenças atômicas **A**, **B** e **C** estão representados nas linhas da tabela. Certifique-se de entender, também, por que a disjunção é o conectivo principal.

A	B	C	(A ∧ B) ∨ ¬C
T	T	T	
T	T	F	
T	F	T	
T	F	F	
F	T	T	
F	T	F	
F	F	T	
F	F	F	

Há dois conectivos na sentença alvo que se aplicam a sentenças atômicas. A conjunção ∧ e a negação ¬. Podemos preencher suas colunas consultando os valores de **A**, **B** e **C** das colunas de referência e as tabelas de verdade de ∧ e ¬. Então a tabela fica:



A	B	C	(A ∧ B)	∨	¬C
T	T	T	T		F
T	T	F	T		T
T	F	T	F		F
T	F	F	F		T
F	T	T	F		F
F	T	F	F		T
F	F	T	F		F
F	F	F	F		T

Resta apenas preencher a coluna do conectivo principal da sentença. Fazemos isso usando os valores de verdade das colunas que acabamos de preencher e consultando a tabela de verdade da disjunção ∨.

A	B	C	(A ∧ B)	∨	¬C
T	T	T	T	T	F
T	T	F	T	T	T
T	F	T	F	F	F
T	F	F	F	T	T
F	T	T	F	F	F
F	T	F	F	T	T
F	F	T	F	F	F
F	F	F	F	T	T

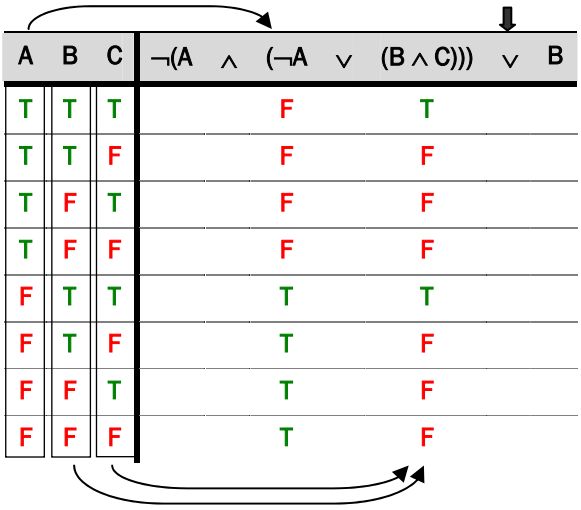
Analisando a coluna final desta tabela, a que está abaixo do conectivo ∨, percebemos que a sentença será falsa em qualquer circunstância onde **Cube(c)** é verdadeira e uma das sentenças **Cube(a)** ou **Cube(b)** é falsa.

Esta tabela mostra que nossa sentença não é uma tautologia. Ademais, como há claramente mundos de blocos nos quais **c** é um cubo e **a** ou **b** não são, a asserção feita pela sentença original não é *logicamente necessária*.

Vamos olhar para mais um exemplo, desta vez para uma sentença da forma

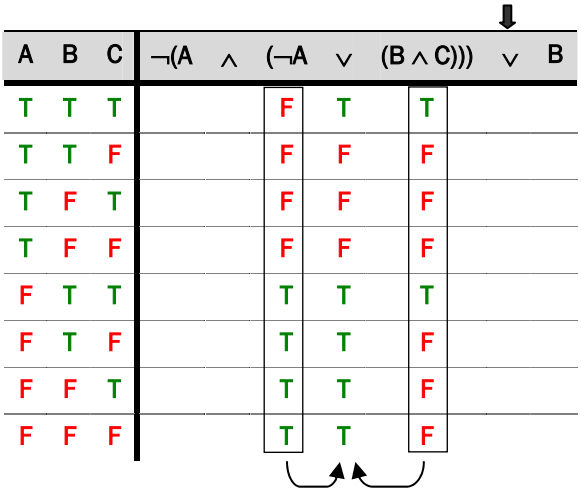
$$\neg(A \wedge (\neg A \vee (B \wedge C))) \vee B$$

Iniciamos a tabela de verdade preenchendo as colunas de referência (para as sentenças atômicas) e a dos conectivos que se aplicam diretamente a sentenças atômicas. Veja:



A	B	C	¬(A	∧	(¬A	∨	(B ∧ C)))	∨	B
T	T	T		F			T		
T	T	F		F			F		
T	F	T		F			F		
T	F	F		F			F		
F	T	T		T			T		
F	T	F		T			F		
F	F	T		T			F		
F	F	F		T			F		

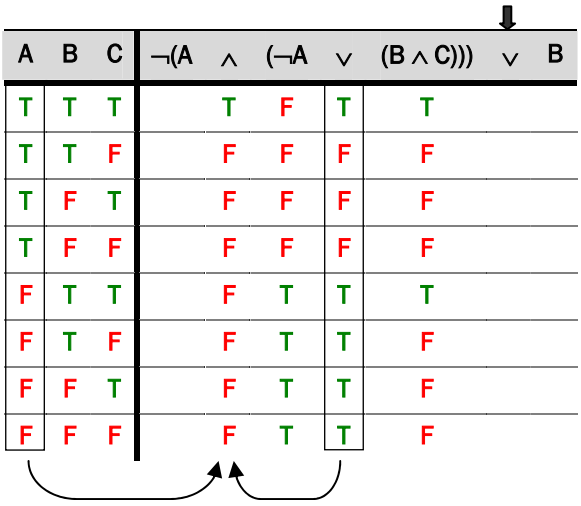
Podemos agora preencher a coluna sob a disjunção $\vee$ que conecta $\neg A$ e $B \wedge C$ nos referindo aos valores das colunas que acabamos de preencher. Como se trata de uma disjunção, só haverá F nas linhas em que ambos os seus constituintes forem falsos.



A	B	C	¬(A	∧	(¬A	∨	(B ∧ C)))	∨	B
T	T	T			F	T	T		
T	T	F			F	F	F		
T	F	T			F	F	F		
T	F	F			F	F	F		
F	T	T			T	T	T		
F	T	F			T	T	F		
F	F	T			T	T	F		
F	F	F			T	T	F		

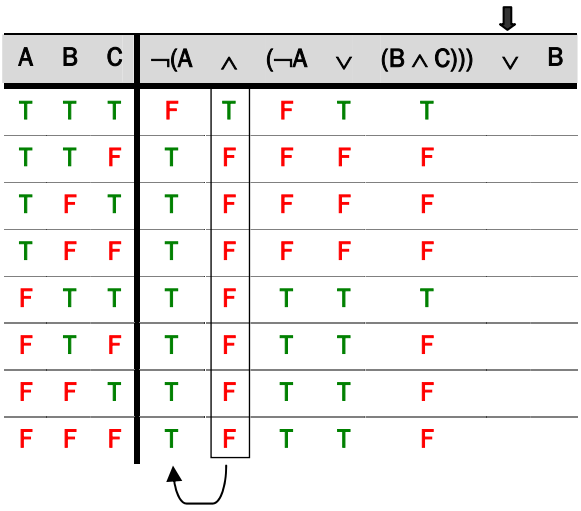
Agora preenchemos a coluna abaixo do símbolo $\wedge$ que ainda resta. Para fazer isso, precisamos nos referir aos valores da coluna de A e da coluna que acabamos de preencher. Como se trata de uma conjunção, só haverá T nas linhas em que os dois valores destas colunas sejam T.

A melhor maneira de fazer isso é colocando um dedo em cada uma destas colunas (A e a coluna de $\vee$ recém preenchida) e, linha por linha, verificar se ambas marcam T. Em caso positivo, preencha a coluna da conjunção com T. Em caso negativo, preencha com F.



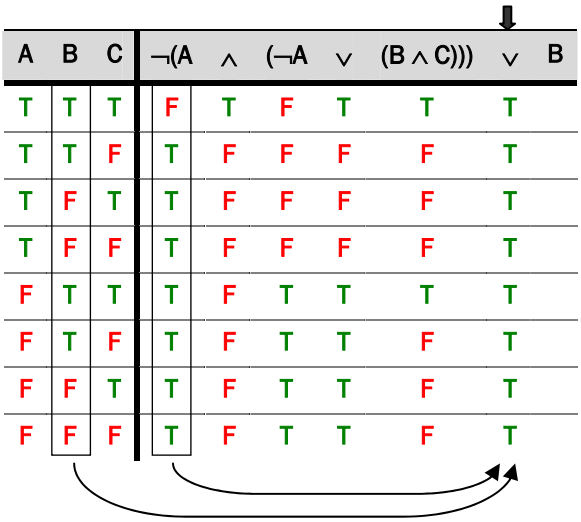
A	B	C	¬(A	∧	(¬A	∨	(B ∧ C)))	∨	B
T	T	T		T	F	T	T		
T	T	F		F	F	F	F		
T	F	T		F	F	F	F		
T	F	F		F	F	F	F		
F	T	T		F	T	T	T		
F	T	F		F	T	T	F		
F	F	T		F	T	T	F		
F	F	F		F	T	T	F		

Agora já é possível preencher a coluna da negação ¬ que resta, referindo-se à coluna que acabamos de completar. Isso é feito apenas trocando os valores T e F uns pelos outros.



A	B	C	¬(A	∧	(¬A	∨	(B ∧ C)))	∨	B
T	T	T	F	T	F	T	T		
T	T	F	T	F	F	F	F		
T	F	T	T	F	F	F	F		
T	F	F	T	F	F	F	F		
F	T	T	T	F	T	T	T		
F	T	F	T	F	T	T	F		
F	F	T	T	F	T	T	F		
F	F	F	T	F	T	T	F		

Finalmente podemos preencher a coluna abaixo do conectivo principal de nossa sentença. Fazemos isso através do método dos dois dedos: deslizamos os dedos pela coluna de referência **B** e pela coluna da negação recém completada. Como se trata de uma disjunção, preenchemos a linha com **F** sempre que ambos os valores nestas colunas forem **F**, caso contrário, preenchemos com **T**.



A	B	C	¬(A	∧	(¬A	∨	(B ∧ C)))	∨	B
T	T	T	F	T	F	T	T	T	
T	T	F	T	F	F	F	F	T	
T	F	T	T	F	F	F	F	T	
T	F	F	T	F	F	F	F	T	
F	T	T	T	F	T	T	T	T	
F	T	F	T	F	T	T	F	T	
F	F	T	T	F	T	T	F	T	
F	F	F	T	F	T	T	F	T	

Tautologia

Diremos que uma ***tautologia*** é qualquer sentença cuja tabela de verdade apresenta apenas **T**s na coluna abaixo de seu conectivo principal.

A tabela de verdade acima mostra, portanto, que qualquer sentença da forma

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

é uma tautologia.

Faça o **Tente Isto (1)** da Lista de Exercícios 4 para aprender como criar uma tabela de verdade utilizando o programa **Boole**.

Há um pequeno problema com nossa definição de tautologia, uma vez que ela assume que toda sentença tem um conectivo principal. Isto ocorre quase sempre, mas não em sentenças como:

$$P \wedge Q \wedge R$$

Apenas para os propósitos de construir tabelas de verdade, assumiremos que o conectivo principal em conjunções ou disjunções com mais de dois elementos é sempre o conectivo mais à direita.

Assim, uma sentença da forma:

$$P_1 \wedge P_2 \wedge P_3 \wedge \ldots \wedge P_4$$

deve ser interpretada como:

$$(((P_1 \wedge P_2) \wedge P_3) \wedge \ldots) \wedge P_4$$

Tautologias e Necessidade Lógica

Toda tautologia é logicamente necessária. Afinal de contas, sua verdade é garantida simplesmente por sua estrutura e pelos significados de seus conectivos verofuncionais.

Tautologias são necessidades lógicas em um sentido bastante forte. Sua verdade é independente tanto do modo como é o mundo, quanto dos significados das sentenças atômicas com as quais elas são compostas.

Deve ficar claro, no entanto, que nem todas as asserções logicamente necessárias são tautologias. O exemplo mais simples de uma afirmação logicamente necessária que não é uma tautologia é a sentença de FOL **a = a**.

Uma vez que ela é uma sentença atômica sua tabela de verdade se limita a:

a = a
T
F

O método das tabelas de verdade é muito grosseiro para reconhecer que a linha contendo F não representa uma possibilidade genuína.

Você deve ser capaz de perceber que um grande número de sentenças não são tautologias, e apesar disso parecem logicamente necessárias. Por exemplo, a sentença

$$\neg(\text{Larger(a, b)} \wedge \text{Larger(b, a)})$$

não pode ser falsificada. No entanto, uma tabela de verdade não mostrará isso. A sentença será falsa na linha em que se atribui valor T para **Larger(a, b)** e para **Larger(b, a)**. Você percebe isso?

TW-Necessidade

Se limitarmos a definição de necessidade lógica aos mundos que podemos construir com o programa **Mundo de Tarski** chegamos à definição de TW-necessidade:

Uma sentença é uma ***TW-necessidade*** se for verdadeira em todos os mundos nos quais ela tem algum valor de verdade (isto é, nos mundos em que todos os nomes que ocorrem na sentença referenciam objetos)

TT-Necessidade

As tautologias também são chamadas de TT-necessidades.[2] Ou seja, uma sentença que tenha valor de verdade T em todas as linhas de sua tabela de verdade é chamada tanto de *tautologia* quanto de ***TT-necessária***.

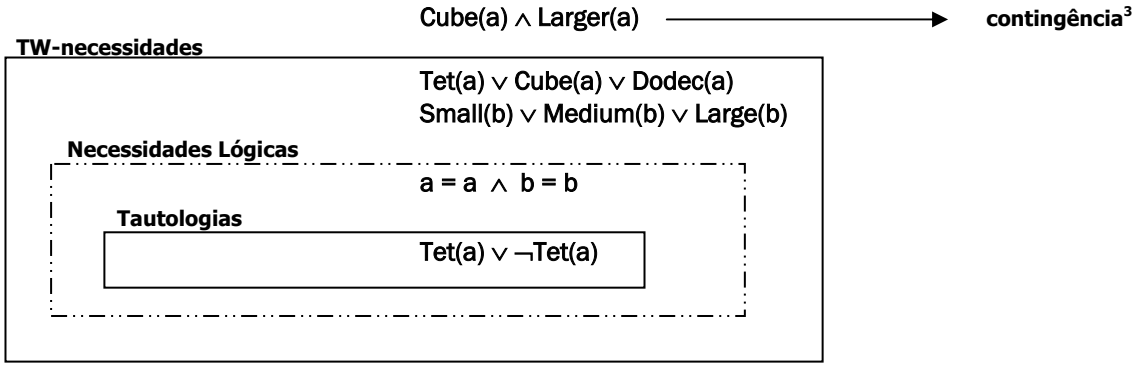
TT-Possibilidade

Analogamente, quando uma sentença é verdadeira em **pelo menos uma** linha de sua tabela de verdade, ela é chamada de ***TT-possível***.

Nenhum destes conceitos corresponde exatamente às noções vagas de necessidade lógica e possibilidade lógica. Mas há importantes relações entre estas noções.

Do lado da necessidade nós sabemos que todas as tautologias são logicamente necessárias e que todas as necessidades lógicas são TW-necessidades. Repare o diagrama abaixo.

Diagrama da Necessidade - Tautologias ⊂ Necessidades Lógicas ⊂ TW-necesidades



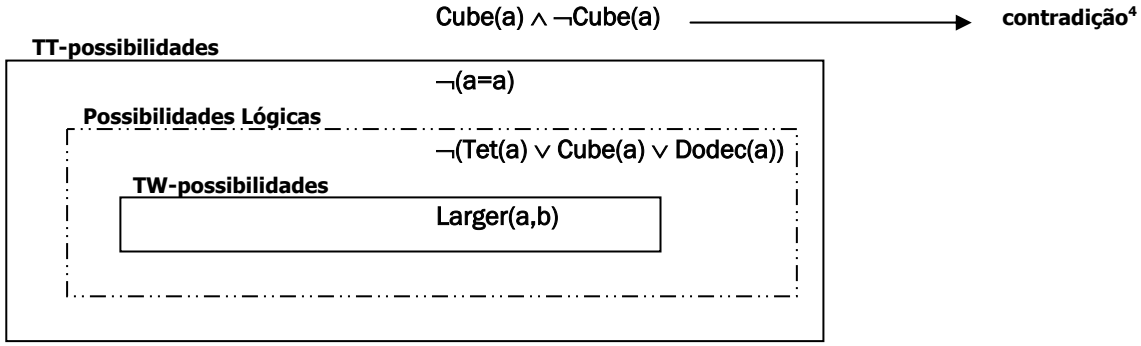
Tanto o conceito de tautologia, quanto o de TW-necessidade são tentativas de tornar precisa a noção vaga de necessidade lógica (também chamada de verdade lógica).

O diagrama acima mostra como a definição de tautologia é muito restritiva. Ela garante que tudo o que for tautologia será uma verdade lógica, no entanto ela é incompleta, pois não consegue captar todas as verdades lógicas. Existem claramente sentenças logicamente necessárias que não são tautologias (ex: **a = a ∧ b = b**).

[2] **TT** vem de "*truth table*" a expressão inglesa para tabela de verdade.
[3] Intuitivamente, uma sentença é uma **contingência** quando em determinadas circunstâncias ela é verdadeira e em outras ela é falsa.

O diagrama também mostra que a definição de TW-necessidade é muito permissiva. Ela garante que todas as verdades lógicas da linguagem dos blocos são TW-necessidade, no entanto, admite como TW-necessidade sentenças que não reconhecemos como necessidades lógicas (ex: **Tet(a) ∨ Cube(a) ∨ Dodec(a)**).

Diagrama da Possibilidade - TW-Possibilidades ⊂ Possibilidades Lógicas ⊂ TT-Possibilidades



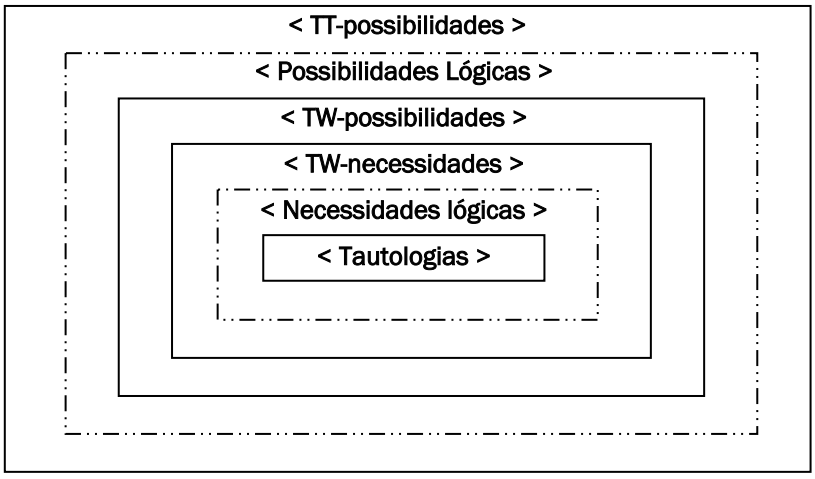
Analogamente, tanto o conceito de TT-possibilidade quanto o de TW-possibilidade são tentativas de tornar precisa a noção vaga de possibilidade lógica.

O diagrama acima mostra como a definição de TW-possibilidade é muito restritiva. Ela garante que tudo o que for TW-possível será logicamente possível, no entanto ela é incompleta, pois não consegue captar todas as possibilidades lógicas para a linguagem dos blocos. Existem claramente sentenças logicamente possíveis que não são TW-possíveis (ex: **¬(Tet(a) ∨ Cube(a) ∨ Dodec(a))**).

O diagrama também mostra que a definição de TT-possibilidade é muito permissiva. Ela garante que todas as possibilidades lógicas são TT-possíveis, no entanto, admite como TT-possibilidade sentenças que não reconhecemos como logicamente possíveis (ex: **¬(a=a)**).

Note que, como todas as TW-necessidades são TW-possibilidades, se fôssemos encadear os dois diagramas acima, o diagrama de necessidade ficaria dentro do nível mais interno do diagrama de possibilidade.

O diagrama geral seria como:



Prova e Necessidade Lógica

Há, de fato, outro método, utilizando a técnica de provas, para mostrar que uma sentença é uma verdade lógica.

[4] Intuitivamente, uma sentença é uma **contradição** quando não existem circunstâncias que a tornem verdadeira. Neste caso específico estamos nos referindo a sentenças TT-impossíveis.

Se você conseguir construir uma prova para uma sentença sem usar nenhuma premissa, então a sentença é logicamente necessária.

Nos capítulos seguintes, apresentaremos a vocês alguns métodos de prova que o habilitarão a provar que sentenças são logicamente necessárias, sem construir suas tabelas de verdade.

LEMBRE-SE

Seja **S** uma sentença de FOL construída a partir de sentenças atômicas utilizando-se apenas conectivos verofuncionais. Uma tabela de verdade para **S** mostra como o valor de verdade de **S** depende dos valores de verdade de suas partes atômicas.

1. **S** é uma tautologia se e somente se toda linha da tabela de verdade para **S** leva o valor **TRUE**.
2. Se **S** é uma tautologia, então **S** é uma verdade lógica (ou seja, é logicamente necessária)
3. Algumas verdades lógicas não são tautologias
4. **S** é TT-possível se e somente se pelo menos uma linha de sua tabela de verdade tem o valor **TRUE**.

Equivalência Lógica

Vimos no capítulo anterior que duas ou mais sentenças são logicamente equivalentes quando têm os mesmos valores de verdade para cada circunstância possível. Neste caso dizemos que as sentenças têm as mesmas ***condições de verdade***.

Mas esta noção de equivalência lógica, da mesma forma que a noção de necessidade lógica, é um tanto vaga.

Podemos, no entanto, como fizemos anteriormente, introduzir conceitos precisos que se aproximam desta noção intuitiva e nos ajudam a entendê-la melhor. Um destes conceitos é o de *equivalência tautológica*.

Equivalência Tautológica

Duas sentenças são ***tautologicamente equivalentes*** quando elas são equivalentes em virtude apenas dos significados de seus conectivos verofuncionais.

Como seria de se esperar, podemos verificar equivalência tautológica utilizando o método das tabelas de verdade.

Tabelas de Verdade Conjuntas

Suponha que queiramos verificar se duas sentenças **S** e **S'** são tautologicamente equivalentes.

O que fazemos é construir uma única tabela de verdade em que na coluna de referência estão todas as sentenças atômicas que ocorrem nas duas sentenças originais (**S** e **S'**).

À direita das colunas de referência colocamos estas duas sentenças, separadas por uma linha vertical, e preenchemos os valores de verdade abaixo de seus conectivos da forma usual.

Chamamos a isto de ***tabela de verdade conjunta*** para as sentenças **S** e **S'**.

Com a tabela preenchida, comparamos a coluna sob o conectivo principal de **S** com a coluna sob o conectivo principal de **S'**. Se estas colunas forem idênticas, então sabemos que as condições de verdade destas duas sentenças são idênticas e dizemos que elas são ***tautologicamente equivalentes***.

Façamos com exemplo um teste para verificar se as sentenças da primeira lei de DeMorgan são tautologicamente equivalentes, usando **A** e **B** como sentenças atômicas arbitrárias.

A	B	¬ (A	∧	B)	¬A	∨	¬B
T	T	F	T		F	F	F
T	F	T	F		F	T	T
F	T	T	F		T	T	F
F	F	T	F		T	T	T

Como as colunas para o conectivo principal de cada uma das sentenças são idênticas, sabemos que, quaisquer que sejam os valores de verdade das sentenças atômicas que as constituem, as sentenças finais sempre terão os mesmos valores de verdade. Portanto, as sentenças são tautologicamente equivalentes.

Vejamos um segundo exemplo. Será que a sentença $\neg((A \vee B) \wedge \neg C)$ é tautologicamente equivalente a $(\neg A \wedge \neg B) \vee C$?

A	B	C	¬((A	∨	B)	∧	¬C)	(¬A	∧	¬B)	∨	C
T	T	T	T	T		F	F	F	F	F	T	
T	T	F	F	T		T	T	F	F	F	F	
T	F	T	T	T		F	F	F	F	T	T	
T	F	F	F	T		T	T	F	F	T	F	
F	T	T	T	T		F	F	T	F	F	T	
F	T	F	F	T		T	T	T	F	F	F	
F	F	T	T	F		F	F	T	T	T	T	
F	F	F	T	F		F	T	T	T	T	T	

Mais uma vez, se percorrermos as colunas sob o conectivo principal de cada uma das sentenças percebemos que elas são idênticas, o que revela que as sentenças são tautologicamente equivalentes e, portanto, são logicamente equivalentes.

Equivalência Tautológica $\subset$ Equivalência Lógica

Todas as sentenças tautologicamente equivalentes são logicamente equivalentes, mas o contrário não é sempre verdade.

Equivalência tautológica é uma forma restrita de equivalência lógica. Existem sentenças que são logicamente equivalentes mas não são tautologicamente equivalentes.

Considere as seguintes sentenças:

(1) $a = b \wedge Cube(a)$

(2) $a = b \wedge Cube(b)$

A seguinte prova informal demonstra que elas são logicamente equivalentes.

Leia atentamente a prova abaixo, procurando entender por que ela demonstra que as duas sentenças são logicamente equivalentes.

> **Prova**: Suponha que a sentença **(1)** seja verdadeira. Então, **a = b** e **Cube(a)** são ambas verdadeiras. Logo, usando o princípio de indiscernibilidade de idênticos **(=Elim)**, concluímos que **Cube(b)** também é verdadeiro, e portanto a sentença **(2)** é verdadeira. Assim, provamos até agora que a verdade de **(1)** implica na verdade de **(2)**.
>
> A relação reversa também é válida. Suponha que **(2)** seja verdadeira. Então, **a = b** e **Cube(b)** são ambas verdadeiras. Novamente por indiscernibilidade de idênticos, concluímos que **Cube(a)** também é verdadeiro, e portanto, a sentença **(1)** é verdadeira. Assim, provamos também que a verdade de **(2)** implica a verdade de **(1)**.
>
> Logo, provamos que **(1)** é verdadeira se e somente se **(2)** é verdadeira. Ou seja, **(1)** e **(2)** têm os mesmos valores de verdade em qualquer circunstância possível e são, por isso, logicamente equivalentes.

Mas veja o que acontece quando construímos a tabela de verdade conjunta para as sentenças **(1)** e **(2)**.

a = b	Cube(a)	Cube(b)	a = b ∧ Cube(a)	a = b ∧ Cube(b)
T	T	T	T	T
T	T	F	T	F
T	F	T	F	T
T	F	F	F	F
F	T	T	F	F
F	T	F	F	F
F	F	T	F	F
F	F	F	F	F

Em primeiro lugar, repare que se fizéssemos duas tabelas de verdade separadas, uma para cada sentença, cada tabela teria apenas 4 linhas, uma vez que em cada sentença ocorrem apenas duas sentenças atômicas. Mas a tabela de verdade conjunta tem 8 linhas, pois o número total de sentenças atômicas que ocorrem nas duas sentenças é 3. Você percebe isso?

Em segundo lugar, e mais importante, repare que as duas sentenças não são tautologicamente equivalentes, uma vez que seus valores de verdade diferem na segunda e terceira linhas da tabela.

Note que nestas duas linhas o valor de verdade de **a = b** é **T** enquanto que **Cube(a)** e **Cube(b)** recebem valores de verdade diferentes.

Devido ao significado do símbolo de identidade, sabemos que nenhuma destas linhas corresponde a uma circunstância logicamente possível, pois se **a** e **b** são idênticos, então eles nomeiam o mesmo objeto e portanto não faz sentido que **Cube(a)** e **Cube(b)** tenham valores de verdade diferentes.

Acontece que o método das tabelas de verdade não detecta esta sutileza, pois investiga apenas o significado dos conectivos verofuncionais. As tabelas de verdade não olham para "dentro" das sentenças atômicas e, portanto, não são sensíveis ao significado dos predicados. Em particular, não interpretam a identidade.

Quando acrescentarmos quantificadores em nossa linguagem, descobriremos muitas outras equivalências lógicas que não são equivalências tautológicas.

Apesar disso, para muitos casos envolvendo conectivos verofuncionais o método das tabelas de verdade é útil.

LEMBRE-SE

Sejam **S** e **S'** sentenças de FOL construídas a partir de sentenças atômicas utilizando-se apenas conectivos verofuncionais. Para verificar se são equivalências tautológicas, construímos tabelas de verdade conjuntas para as duas sentenças.

1. **S** e **S'** são tautologicamente equivalentes se e somente se cada linha da tabela de verdade conjunta atribui os mesmos valores para **S** e **S'**.
2. Se **S** e **S'** são tautologicamente equivalentes, então eles são logicamente equivalentes.
3. Algumas sentenças logicamente equivalentes não são tautologicamente equivalentes.

Conseqüência Lógica é o Conceito Fundamental da Lógica

Nosso interesse principal neste livro é com a relação de **conseqüência lógica**. Necessidade lógica, equivalência lógica e possibilidade lógica podem ser entendidas como casos especiais de conseqüência lógica. Veja como:

Necessidade Lógica: uma sentença é *logicamente necessária* (ou uma necessidade lógica) quando for *conseqüência lógica* de qualquer conjunto de premissas.

Equivalência Lógica: duas sentenças são *logicamente equivalentes* se cada uma for *conseqüência lógica* da outra.

Possibilidade Lógica: uma sentença **S** é *logicamente possível* (ou uma possibilidade lógica) quando sua negação **¬S** não for conseqüência lógica de qualquer conjunto de premissas. Ou seja, quando **¬S** não for uma necessidade lógica.

Conseqüência Lógica e Conseqüência Tautológica

Como você já deve ter adivinhado, as tabelas de verdade nos permitem definir com precisão a noção de ***conseqüência tautológica***, que é uma forma restrita de conseqüência lógica, da mesma forma que nos permitiu definir precisamente tautologia e equivalência tautológica, que são versões restritas das noções de verdade lógica e equivalência lógica.

Considere duas sentenças **P** e **Q**, construídas a partir de sentenças atômicas utilizando apenas conectivos verofuncionais. Suponha que você queira saber se **Q** é uma conseqüência de **P**.

O que você tem a fazer é criar uma tabela de verdade conjunta para **P** e **Q**, da mesma forma que fizemos para verificar equivalência tautológica.

Conseqüência Tautológica

Com a tabela conjunta para **P** e **Q** já preenchida, percorra as colunas sob o conectivo principal de cada sentença.

Em particular, olhe para cada linha de sua tabela na qual **P** é verdadeira. Se em cada uma destas linhas **Q** também for verdadeira, então diremos que **Q** é uma *conseqüência tautológica* de **P**.

A tabela de verdade estará mostrando, neste caso, que se **P** é verdadeira, então **Q** deve também ser verdadeira, e isso devido apenas ao significado dos conectivos verofuncionais das duas sentenças.

Da mesma forma que as tautologias são logicamente necessárias, qualquer conseqüência tautológica **Q** de uma sentença **P** deve também ser uma conseqüência lógica de **P**.

Podemos verificar isso fazendo uma prova informal que mostra que se **Q** não for uma conseqüência lógica de **P**, então o teste da tabela de verdade falha e, portanto, **Q** não será uma conseqüência tautológica de **P**.

> Prova: Suponha que **Q** não é uma conseqüência lógica de **P**. Então, com base em nossa definição de conseqüência lógica, deve haver alguma circunstância possível em que **P** é verdadeira e **Q** é falsa. Esta circunstância determinará valores de verdade para as sentenças atômicas de **P** e **Q**, e estes valores corresponderão a uma linha em uma tabela de verdade conjunta para **P** e **Q**, uma vez todas as atribuições de valores possíveis para as sentenças atômicas de **P** e **Q** estão representadas nesta tabela. Além disso, como **P** e **Q** são construídas a partir destas sentenças atômicas pelo uso de conectivos verofuncionais e, como já dissemos, na circunstância que gerou esta linha **P** é verdadeira e **Q** é falsa, então **P** receberá o valor de verdade T e **Q** receberá o valor de verdade F nesta linha. Logo, **Q** não é conseqüência tautológica de **P**.

A prova acima mostra que:

(1) Sempre que **Q** não é conseqüência lógica de **P**, também não é conseqüência tautológica.

Logo, é claro que se **Q** for conseqüência tautológica de **P**, tem necessariamente que ser conseqüência lógica, caso contrário teríamos um caso em que **P** não é conseqüência lógica de **Q**, mas é conseqüência tautológica, o que claramente desobedece (1). Você entende este argumento? Difícil? Pense com calma! Mais adiante neste livro estudaremos com detalhes este tipo de argumento.

Não se preocupe muito com a dificuldade aqui. O importante é você perceber que se **Q** é conseqüência tautológica de **P**, então é também conseqüência lógica e, portanto, a definição precisa de conseqüência tautológica é uma forma restrita da noção mais vaga de conseqüência lógica.

Vamos fazer um exemplo bastante simples para verificar se $A \vee B$ é uma conseqüência lógica de $A \wedge B$:

A	B	$A \wedge B$	$A \vee B$
T	T	T	T
T	F	F	T
F	T	F	T
F	F	F	F

Está claro que as sentenças não são tautologicamente equivalentes, uma vez que suas colunas são distintas. Mas o que queremos saber é se $A \vee B$ é conseqüência lógica de $A \wedge B$. Para isso, como vimos acima, basta que examinemos apenas as linhas da tabela em que $A \wedge B$ é verdadeira. Isso ocorre somente na primeira linha e ali, $A \vee B$ também é verdadeira.

Portanto, de acordo com a definição que demos acima, esta tabela mostra que **A ∨ B** é sim conseqüência tautológica (e portanto conseqüência lógica) de **A ∧ B**.

Dizer isso não significa nada mais do que dizer que em todas as circunstâncias nas quais **A ∧ B** é verdadeira, **A ∨ B** também será verdadeira. O que é bastante razoável, dado os significados de ∧ e ∨.

Repare que esta mesma tabela de verdade mostra que **A ∧ B** não é uma conseqüência tautológica de **A ∨ B**, uma vez que há linhas em que a última é verdadeira e a primeira falsa.

Será que isso mostra que **A ∧ B** não é uma conseqüência lógica de **A ∨ B** ? É preciso ter cuidado aqui, pois veremos que da mesma forma que para a equivalência lógica, há casos de conseqüência lógica que esta definição de conseqüência tautológica não consegue perceber.

Conseqüência Tautológica ⊂ Conseqüência Lógica

Nem toda conseqüência lógica de uma sentença é uma conseqüência tautológica.

Por exemplo, a sentença **a = c** é uma conseqüência lógica da sentença (**a = b ∧ b = c**), mas não é uma conseqüência tautológica dela.

Pense sobre a linha que atribui **T** para as sentenças **a = b** e **b = c**, mas atribui **F** para **a = c**. Esta linha, que impede que **a = c** seja uma conseqüência tautológica de (**a = b ∧ b = c**) não respeita o significado das sentenças atômicas envolvidas.

Ela não corresponde a uma circunstância genuinamente possível, no entanto o método das tabelas de verdade não detecta isso.

Conseqüência Tautológica com Mais de Uma Premissa

O método das tabelas de verdade para verificação de conseqüência tautológica não é restrito a uma premissa apenas. Ele pode ser aplicado a argumentos com qualquer número de premissas P_1, ..., P_n e uma conclusão **Q**.

Para fazer isso é preciso construir uma tabela de verdade conjunta para todas as sentenças P_1, ..., P_n e **Q**.

Feito isso, você deve verificar se **Q** é verdadeira em cada linha na qual todas as premissas sejam P_1, ..., P_n conjuntamente verdadeiras.

Se isso ocorrer, a conclusão é conseqüência tautológica das premissas.

Vejamos alguns exemplos simples. Suponha que queiramos verificar se **B** é conseqüência das premissas **A ∨ B** e **¬A**.

A tabela de verdade conjunta é a seguinte:

A	B	A ∨ B	¬A	B
T	T	T	F	T
T	F	T	F	F
F	T	T	T	T
F	F	F	T	F

Percorrendo as colunas abaixo das duas premissas, percebemos que há apenas uma linha em que ambas são verdadeiras: a terceira linha. Nesta linha, a conclusão **B** também é verdadeira.

Então, **B** é de fato uma conseqüência tautológica (e portanto conseqüência lógica) das premissas **A ∨ B** e **¬A**.

Suponha agora que queiramos aplicar o método para saber se **A ∨ C** é uma conseqüência de **A ∨ ¬B** e **B ∨ C**. A tabela verdade conjunta para estas três sentenças é:

A	B	C	A ∨ ¬B	B ∨ C	A ∨ C
T	T	T	T	T	T
T	T	F	T	T	T
T	F	T	T	T	T
T	F	F	T	F	T
F	T	T	F	T	T
F	T	F	F	T	F
F	F	T	T	T	T
F	F	F	T	F	F

Há 4 linhas em que as premissas são verdadeiras, as linhas 1, 2, 3 e 7. E em cada uma destas linhas a conclusão **A ∨ C** também é verdadeira. Portanto, esta inferência, de **A ∨ C** a partir de **A ∨ ¬B** e de **B ∨ C** é logicamente válida.

Fica como exercício, para você entregar junto com a Lista de Exercícios 4, mostrar que a sentença **A ∨ ¬B** não é conseqüência tautológica das premissas **B ∨ C** e **A ∨ C**. Construa a tabela de verdade conjunta e indique especificamente as linhas que demonstram este fato.

LEMBRE-SE

Sejam P_1, ..., P_n e **Q** sentenças de FOL construídas a partir de sentenças atômicas utilizando-se apenas conectivos verofuncionais. Em uma tabela de verdade conjunta para todas estas sentenças, podemos verificar os seguintes fatos:

1. **Q** é conseqüência tautológica de $P_1,...,P_n$ se e somente se toda linha que atribui **T** para cada $P_1,...,P_n$, também atribui **T** para **Q**.
2. Se **Q** é uma conseqüência tautológica de $P_1, ..., P_n$, então **Q** também é uma conseqüência lógica de $P_1, ..., P_n$.
3. Algumas conseqüências lógicas não são conseqüências tautológicas. (exemplo **a=c** é conseqüência lógica, mas não conseqüência tautológica de **a=b ∧ b=c**).

Conseqüência Tautológica no Programa FITCH

Verificar se uma sentença **Q** é conseqüência tautológica de de $P_1, ...,P_n$ é um procedimento mecânico. Se as sentenças são grandes pode requerer um monte de trabalho tedioso, mas não requer absolutamente nenhuma originalidade.

É exatamente este tipo de tarefa em que os computadores são bons. Por causa disso, construímos um mecanismo no programa **Fitch** chamado **TautCon**.

O Mecanismo TautCon

TautCon é similar ao mecanismo **AnaCon**, com a diferença que ele verifica se uma determinada sentença é conseqüência tautológica das sentenças citadas como suporte.

TautCon, assim como **AnaCon** não é de fato uma regra de inferência, mas é útil para rapidamente testar se uma sentença se segue tautologicamente de outras.

Faça o **Tente Isto (2)** da Lista de Exercícios 4 para experimentar a utilização do mecanismo **TautCon** no programa **Fitch**.

Os Mecanismos TautCon, FOCon e AnaCon

TautCon, **FOCon** e **AnaCon** são três mecanismos que o programa **Fitch** proporciona para verificarmos a relação de *conseqüência lógica*.

TautCon é o mais fraco deles. Dada uma sentença, em uma prova, que utiliza **TautCon** como justificativa, ele verifica se esta sentença se segue das sentenças citadas, testando apenas ao significado dos conectivos verofuncionais. Ele ignora os significados de qualquer predicado das sentenças. **TautCon** também ignorará os quantificadores, que são conectivos especiais (não-verofuncionais) que introduziremos mais adiante.

FOCon, que deve seu nome a *"first-order conseqüênce"*[5], leva em consideração os conectivos verofuncionais, os quantificadores e o predicado da identidade (**=**) para verificar se uma determinada sentença que utiliza **FOCon** como justificativa é conseqüência lógica das sentenças citadas por este passo. **FOCon** pode, por exemplo, identificar que **a = c** é conseqüência lógica de **a = b ∧ b = c**. Ele é mais forte do que **TautCon** pois toda conseqüência lógica que **TautCon** reconheça, **FOCon** também reconhecerá.

AnaCon é o mais forte destes três mecanismos, pois ele tenta identificar conseqüências lógicas devidas aos conectivos verofuncionais, aos quantificadores, à identidade e à maioria dos predicados da linguagem dos blocos. (**AnaCon** ignora os predicados **Between** e **Adjoins** simplesmente por razões computacionais práticas). Qualquer inferência de conseqüência lógica que possa ser feita com **TautCon** ou **FOCon** pode, em princípio, ser feita utilizando o mecanismo **AnaCon**.[6]

Como já dissemos anteriormente, você só pode usar qualquer um destes mecanismos **Con** quando o enunciado do exercício indica, explicitamente, que sua utilização é permitida.

Faça o **Tente Isto (3)** da Lista de Exercícios 4 para experimentar por você mesmo o funcionamento destes mecanismos no programa **Fitch**.

[5] Em português: conseqüência de primeira-ordem.

[6] Na prática o mecanismo **AnaCon** pode falhar por restrições computacionais (falta de memória, tempo de processamento demasiado longo,...) em situações onde **TautCon** e **FOCon** não falhariam. Mas, em princípio, se tivéssemos um computador com MUITA memória e que fosse MUITO rápido, o mecanismo **AnaCon** não falharia nestes casos.

Capítulo 5 - Métodos de Prova para a Lógica Booleana

O método das tabelas de verdade é bom para mostrar a validade de argumentos simples que dependem apenas do significado dos conectivos verofuncionais, mas o método tem duas limitações bastante significativas:

Limitações do Método das Tabelas de Verdade

(1) As tabelas de verdade tornam-se extremamente longas conforme o número de sentenças atômicas aumenta. Um argumento com 7 sentenças atômicas (o que ainda é um número pequeno) exige uma tabela com 128 linhas para ser analisado. Se dobrarmos o número de sentenças atômicas para 14, precisaríamos de uma tabela com mais de **16 mil** linhas para analisá-lo. Este crescimento exponencial das tabelas de verdade praticamente elimina o valor prático deste método.

(2) Não é possível estender facilmente o método das tabelas de verdade para argumentos cuja validade depende de outros elementos além dos conectivos verofuncionais. Esta limitação impede a aplicação do método para a grande maioria dos argumentos que fazemos em nosso dia a dia, que além dos conectivos booleanos, se baseiam também em outros tipos de expressão.

Precisamos, pois, de um método para identificar conseqüências lógicas que vá além das conseqüências tautológicas, e que possa ser aplicado tanto a conectivos booleanos quanto a outros princípios racionais. Tal método será o **método das provas**.

Discutiremos, neste capítulo, os **padrões de inferência** legítimos que surgem quando introduzimos os conectivos booleanos em nossa linguagem, e mostraremos como aplicar estes padrões a **provas informais**. No capítulo 6 estenderemos nosso sistema **F** para incluir as regras formais derivadas destes padrões.

A principal vantagem do método das provas sobre ao método das tabelas de verdade é que seremos capazes de utilizá-lo mesmo quando a validade do argumento que queremos provar depende de mais elementos do que apenas os conectivos booleanos.

Passos de Inferência Válidos (EM PROVAS INFORMAIS)

REGRA INICIAL: Ao fazer uma prova informal de **Q** a partir de algumas premissas, se é sabido que **Q** é uma conseqüência lógica de $P_1, \ldots, P_n$, e cada uma das sentenças $P_1, \ldots, P_n$ já foram provadas a partir das mesmas premissas, então é possível afirmar **Q** na prova.

Você já aprendeu neste curso várias equivalências lógicas conhecidas. Portanto, sinta-se livre em usá-las quando lhe for pedido para fazer uma prova informal. Por exemplo, você pode usar livremente o princípio da dupla negação ($P \Leftrightarrow \neg\neg P$) ou DeMorgan, em qualquer prova informal.

Um caso especial desta regra inicial é o seguinte:

REGRA INICIAL (caso especial): Se você já sabe que uma determinada sentença **Q** é uma verdade lógica (necessariamente verdadeira), então você pode afirmar **Q** em qualquer ponto de sua prova.

Já vimos este principio em ação no Capítulo 2 quando utilizamos a regra **(=Intro)** que nos permite afirmar qualquer sentença da forma **a = a** em qualquer ponto de uma prova. Da mesma forma, podemos, de acordo com este princípio, afirmar verdades lógicas simples, como o princípio do terceiro excluído ($P \vee \neg P$), em qualquer ponto de uma prova.

Os três passos de inferência válidos que veremos a seguir são tão óbvios que, em geral, são utilizados sem sequer serem citados nas provas informais.

Eliminação da Conjunção

De uma conjunção com qualquer número de sentenças ($P_1 \wedge \ldots \wedge P_n$) é correto inferir qualquer um de seus "conjuntos" (P_i - $1 \leq i \leq n$)

Isto porque é claro que qualquer P_i é conseqüência lógica da conjunção, uma vez que para ela ser verdadeira todos os conjuntos têm que ser individualmente verdadeiros (não é possível a conjunção ser verdadeira e algum P_i individual ser falso).

Introdução da Conjunção

Para provar uma conjunção de várias sentenças ($P_1 \wedge \ldots \wedge P_n$), basta provarmos cada um dos "conjuntos" (P_i - $1 \leq i \leq n$) separadamente.

Isto porque é claro que $(P_1 \wedge \ldots \wedge P_n)$ é conseqüência lógica da totalidade das premissas $P_1, \ldots, P_n$ (já que não é possível que $(P_1 \wedge \ldots \wedge P_n)$ seja falsa quando $P_1, \ldots, P_n$ são todas verdadeiras).

Introdução da Disjunção

Se você provou uma sentença **Q**, então você pode inferir qualquer disjunção que tenha **Q** como um de seus disjuntos.

Isto porque, suponha que **Q** seja algum P_i (com $1 \leq i \leq n$). Quaisquer que sejam as outras sentenças $P_1, \ldots, P_n$, a disjunção $(P_1 \vee \ldots \vee P_i \vee \ldots \vee P_n)$ é conseqüência lógica de P_i (já que não é possível que $(P_1 \vee \ldots \vee P_i \vee \ldots \vee P_n)$ seja falsa quando P_i é verdadeira).

Alguns estudantes iniciantes se confundem com este passo de inferência, pois eles se perguntam, com razão, porque alguém iria querer concluir, por exemplo, a sentença **Cube(a) ∨ Tet(a)** quando já sabe que **Cube(a)** é verdadeira? Ao fazer isso não estamos partindo de uma informação precisa e tornando-a mais imprecisa?

Isso é verdade, mas veremos mais adiante que, quando combinado com os outros passos de inferência, este passo se tornará bastante útil.

Questão de Estilo

Provas informais servem a dois propósitos:

(1) São um método de descoberta: elas nos permitem extrair informação nova de informações que já possuíamos.

(2) São um método de comunicação: elas nos permitem comunicar nossas descobertas aos outros.

Sendo um método de comunicação, as provas informais podem ser feitas com mais ou menos qualidade. Elas também sempre envolvem aspectos estilísticos, como qualquer de nossas comunicações.

Independentemente do estilo, devemos buscar duas qualidades em cada passo de nossas provas informais: ele deve ser de ***fácil entendimento*** e deve ser ***significativo***. Em outras palavras, cada passo de nossas provas não deve dar muito trabalho para ser compreendido por um leitor (fácil entendimento) e deve, também, ser informativo, e não uma perda de tempo para o leitor (significativo).

Quão fácil é o entendimento de um passo e quão significativo ele é, depende da audiência! Se você está fazendo uma prova informal para ser apresentada em um congresso de lógica, onde sua audiência será de lógicos profissionais, você certamente considerará que muitas passagens dificílimas a alunos iniciantes, que exigiriam muita explicação são, para estes lógicos, passos de fácil entendimento que podem ser rapidamente descritos. Já se a mesma prova fosse apresentada em uma turma de alunos iniciantes, estas passagens complicadas deveriam ser subdivididas em vários passos compreensíveis aos estudantes, mas que, para a audiência especializada, seriam pouco significativos.

LEMBRE-SE

1. Em uma prova informal a partir de algumas premissas, se sabemos que **Q** é uma conseqüência lógica das sentenças $P_1, \ldots, P_n$ e se cada $P_1, \ldots, P_n$ já foi provado a partir dessas mesmas premissas, então podemos afirmar **Q** em nossa prova.
2. Cada passo de uma prova informal deve ser ***significativo*** mas ***de fácil entendimento***.
3. Se um passo é significativo ou de entendimento fácil, isso depende da audiência para quem a prova é endereçada.
4. Os seguintes padrões de inferência válidos são geralmente utilizados em prova informais sem serem mencionados:
    - De $P \wedge Q$ infere-se **P**.
    - De **P** e **Q** infere-se $P \wedge Q$.
    - De **P** infere-se $P \vee Q$.

Além dos passos de inferência triviais descritos acima, os conectivos booleanos propiciam dois métodos de prova inteiramente novos que são explicitamente aplicados em todos os tipos de raciocínio rigoroso. O primeiro deles é o método de *prova por casos*, que em nosso sistema formal **F** será chamado de *eliminação da disjunção*.

Prova por Casos

Vamos iniciar ilustrando uma prova por casos utilizada em um tipo de argumento muito comum em matemática. O argumento prova que:

(1) Existem números irracionais *b* e *c* tais que b^c é um número racional.

Apenas relembrando, um número é *racional* se ele pode ser expresso como uma fração *n/m*, onde *n* e *m* são números inteiros. Se ele não puder ser assim expresso, dizemos que é um número *irracional*. Então, 2 é racional (pois 2 = 2/1), mas $\sqrt{2}$ não é (mas adiante provaremos isso também). Segue abaixo nossa prova da proposição (1).

> **PROVA**: Considere o número $\sqrt{2}^{\sqrt{2}}$.
>
> Sabemos que $\sqrt{2}$ é irracional. Mas e quanto a $\sqrt{2}^{\sqrt{2}}$?
>
> Ou ele é um número racional ou é irracional.
>
> Se $\sqrt{2}^{\sqrt{2}}$ é racional, já encontramos **b** e **c** irracionais tais que $\mathbf{b^c}$ é racional. São eles **b=c=** $\sqrt{2}$.
>
> Se $\sqrt{2}^{\sqrt{2}}$ é irracional, considere então o seguinte número $\left(\sqrt{2}^{\sqrt{2}}\right)^{\sqrt{2}}$.
>
> Note que $\left(\sqrt{2}^{\sqrt{2}}\right)^{\sqrt{2}} = \sqrt{2}^{\sqrt{2}\cdot\sqrt{2}} = \sqrt{2}^{2} = \sqrt{2}\cdot\sqrt{2} = 2$ (que é um número racional)
>
> Mas então, se fizermos **b=** $\sqrt{2}^{\sqrt{2}}$ e **c=** $\sqrt{2}$, como estamos considerando $\sqrt{2}^{\sqrt{2}}$ irracional, temos **b** e **c** dois números irracionais tais que $\mathbf{b^c}$ = 2 é racional.
>
> Ou seja, para cada uma das duas alternativas que consideramos ($\sqrt{2}^{\sqrt{2}}$ como racional e $\sqrt{2}^{\sqrt{2}}$ como irracional), encontramos **b** e **c** irracionais tais que $\mathbf{b^c}$ é um número racional. Isso prova o teorema.

O que nos interessa nesta prova não é tanto o resultado, mas o método da prova por casos. De uma maneira geral, o padrão de argumentação conhecido como prova por casos é o seguinte:

PROVA POR CASOS: desejamos provar uma afirmação, digamos **S**, e conhecemos (temos como premissa) uma disjunção, digamos $\mathbf{P \vee Q}$. Temos então que mostrar duas coisas: (1) que **S** se segue quando assumimos **P** como premissa, e (2) que **S** também se segue quando assumimos **Q** como premissa. Uma vez que sabemos que uma das duas (**P** ou **Q**) deve ser verdadeira, já que nossa premissa inicial é $\mathbf{P \vee Q}$, então nós concluímos que **S** deve ser verdadeira.

Na verdade, não estamos limitados a apenas dois casos. Se em qualquer estágio de uma prova, temos uma disjunção de *n* disjuntos, digamos $\mathbf{P_1 \vee \ldots \vee P_n}$, podemos, então, quebrar nossa prova em *n* casos. No primeiro assumimos $\mathbf{P_1}$ como hipótese, no segundo $\mathbf{P_2}$, e assim por diante para cada disjunto. Se provarmos a sentença desejada, **S**, em cada um destes casos, então podemos afirmar **S** (temos justificativa para inferir **S**).

Vejamos um exemplo ainda mais simples. Suponha que queiramos provar que **Small(c)** é uma conseqüência lógica de

$$\mathbf{(Cube(c) \wedge Small(c)) \vee (Tet(c) \wedge Small(c))}$$

É um argumento óbvio cuja prova, no entanto, envolve prova por casos. Veja como seria uma prova informal deste argumento:

> **PROVA**: Nossa premissa é $\mathbf{(Cube(c) \wedge Small(c)) \vee (Tet(c) \wedge Small(c))}$. Dividiremos a prova em dois casos correspondentes aos dois disjuntos desta premissa.
>
> Primeiro assuma que $\mathbf{Cube(c) \wedge Small(c)}$ é verdadeira. Neste caso, por eliminação da conjunção, que nem precisaríamos mencionar, sabemos que **Small(c)** é verdadeira.
>
> Da mesma forma, assuma agora que $\mathbf{Tet(c) \wedge Small(c)}$ é verdadeira. Então, **Small(c)** é verdadeira.
>
> Assim, em ambos os casos concluímos **Small(c)**. Portanto, pelo método da prova por casos, **Small(c)** é conseqüência da disjunção inicial.

O exemplo seguinte mostra de que forma aquele curioso passo da introdução da disjunção (de **P** infere-se **P** $\vee$ **Q**) pode ser útil em uma prova por casos.

Suponha que saibamos que ou Max está em casa e Cleo está feliz, ou Claire está em casa e Miau está feliz. Isto é:

$$(\mathbf{Home(max)} \wedge \mathbf{Happy(cleo)}) \vee (\mathbf{Home(claire)} \wedge \mathbf{Happy(miau)})$$

Queremos provar que Cleo ou Miau estão felizes, ou seja:

$$\mathbf{Happy(cleo)} \vee \mathbf{Happy(miau)}$$

> PROVA: Assumindo a disjunção da premissa sabemos que: ou **(Home(max)** $\wedge$ **Happy(cleo)** ou **Home(claire)** $\wedge$ **Happy(miau)**.
>
> Se a primeira alternativa é verdadeira, então é claro que **Happy(cleo)** e, portanto, por introdução da disjunção, **Happy(cleo)**$\vee$ **Happy(miau)** também é verdadeira.
>
> Se a segunda alternativa é verdadeira, então é claro que **Happy(miau)** e, portanto **Happy(cleo)** $\vee$ **Happy(miau)** também é verdadeira.
>
> Logo, em qualquer um dos casos obtemos a conclusão desejada. Portanto, nossa conclusão é conseqüência da disjunção inicial.

Raciocinar por casos é extremamente útil em argumentos do dia a dia. Por exemplo, um dos autores deste livro (vamos chamá-lo **J**) e sua esposa um dia desses perceberam que a cartela de estacionamento (área azul) que tinham deixado no carro estava expirada fazia várias horas. **J** argumentou da seguinte forma para defender a posição de que eles não precisavam se apressar para voltar ao carro (os lógicos costumam argumentar assim, não case com um [ou uma]):

> PROVA: No presente momento, ou já fomos multados ou não fomos. Se já recebemos uma multa, não receberemos outra durante o tempo que levará para chegarmos no carro, então, neste caso não há razão para nos apressarmos. Por outro lado, se mesmo com a cartela expirada há várias horas ainda não recebemos uma multa, é extremamente improvável que receberemos uma nos próximos minutos. Então, neste caso também não há razão para nos apressarmos. Em qualquer um dos casos, não há razão para pressa em voltar ao carro.

A esposa de **J** respondeu com o seguinte contra-argumento (mostrando que muitos anos de casamento com um lógico tem lá seu impacto):

> PROVA: Ou nós vamos receber uma multa nos próximos minutos ou não vamos. Se vamos, então podemos evitá-la apressando-nos em voltar ao carro, o que seria uma boa coisa. Se não vamos receber uma multa nos próximos minutos, apressarmo-nos em voltar ao carro seria um bom exercício e também demonstraria nosso respeito à lei; e ambas estas coisas são boas. Então, em qualquer evento, apressarmo-nos em voltar ao carro é uma boa coisa a ser feita.

A esposa de **J** ganhou a discussão!!

> LEMBRE-SE
>
> Prova por casos: Para provar **S** a partir de $\mathbf{P_1} \vee \ldots \vee \mathbf{P_n}$ usando este método, prove **S** a partir de cada um dos $\mathbf{P_1}, \ldots, \mathbf{P_n}$.

Prova Inidireta: prova por absurdo

Um dos mais importantes métodos de prova é conhecido como *prova por absurdo*. Ele também é chamado de *prova indireta*, *redução ao absurdo*, algumas vezes em latim, *reductio ad absurdum*. Sua contrapartida formal em nosso sistema **F** será chamada de *introdução da negação*. A idéia básica é a seguinte:

PROVA POR ABSURDO: suponha que você queira provar uma sentença negativa, digamos **¬S**, a partir de algumas premissas, digamos **P_1 ,..., P_n**. Uma forma de fazer isso é assumir temporariamente **S** como uma premissa e mostrar que isso leva a uma contradição (das premissas **P_1 ,..., P_n** juntamente com **S** segue-se uma contradição).

POR QUE ?: Por que a prova de uma contradição a partir de **S, P_1 ,..., P_n** demonstra que **¬S** é conseqüência lógica de **P_1 ,..., P_n**? Pense com calma. Ao chegar a uma contradição você provou que as sentenças **S, P_1 ,..., P_n** não podem ser todas elas simultaneamente verdadeiras (pois caso sejam a contradição também será verdadeira, e isso não pode ocorrer, pois violaria algum princípio lógico). Portanto, em qualquer circunstância na qual **P_1 ,..., P_n** sejam verdadeiras, **S** deve ser falsa. E isso é o mesmo que dizer que sempre que **P_1 ,..., P_n** forem todas verdadeiras, **¬S** deve também ser verdadeira. Ou seja, **¬S** é conseqüência lógica de **P_1 ,..., P_n**.

Vejamos um exemplo simples: assuma, como premissas **Cube(c) ∨ Dodec(c)** e **Tet(b)** e vamos provar **¬(b=c)**.

PROVA: Para provar ¬(**b=c**) vamos mostrar que admitir **b=c** como premissa leva a uma contradição.

Da primeira premissa sabemos que ou **Cube(c)** ou **Dodec(c)** é verdadeira.

Logo, assumindo **b=c**, sabemos que ou **Cube(b)** ou **Dodec(b)** é verdadeira. (por **=Elim**)

Então, caso valha **Cube(b)**, temos uma contradição com a segunda premissa, que afirma **Tet(b)**.

Analogamente, caso valha **Dodec(b)**, também temos uma contradição com a segunda premissa (**Tet(b)**).

Assim, a adoção da hipótese **b=c** leva inevitavelmente a uma contradição.

Portanto, temos uma prova por contradição (ou indireta) da negação desta hipótese. Ou seja, provamos, a partir das premissas dadas, que **¬(b=c)**.

Vejamos agora um exemplo mais interessante e famoso deste método de prova. Os gregos antigos ficaram chocados ao descobrir que a raiz quadrada de 2 não podia ser expressa como uma fração. Dito nos termos que os matemáticos costumam utilizar: a raiz quadrada de 2 é um número irracional!

Antes de fazermos a prova, vamos nos lembrar de alguns fatos numéricos simples que já eram conhecidos dos gregos:

(1) qualquer número racional pode ser expresso como uma fração p/q na qual pelo menos um dos dois números (p ou q) é impar (se não for, divida os dois números por 2 e continue dividindo até que um deles seja impar)

(2) O quadrado de um número ímpar é sempre um número impar. Ou seja, se n^2 é par, então n também é par.

(3) Se n^2 é par, então ele é divisível por 4 (o que é conseqüência do fato (2)).

Vamos agora à prova de que $\sqrt{2}$ é irracional.

PROVA: objetivando chegar a uma contradição, vamos assumir que $\sqrt{2}$ é racional.

(*i*) Sob esta suposição, $\sqrt{2}$ pode ser expressa da forma p/q onde pelo menos um destes dois números é ímpar.

Como $\sqrt{2} = p/q$, se elevamos ao quadrado os dois lados desta identidade obtemos:

(*) $2 = p^2/q^2$

Multiplicando os dois lados por q^2 obtemos: $2q^2 = p^2$.

Esta igualdade mostra que p^2 é um número par.

(*ii*) Então, por (2), sabemos que p é um número par

E por (3) sabemos que p^2 é divisível por 4.

Olhando mais uma vez para a equação $2q^2 = p^2$, vemos que se p^2 é divisível por 4, então $2q^2$ também é divisível por 4 e, portanto, q^2 deve ser divisível por 2.

(*iii*) Neste caso, por (2) q também deve ser par.

Assim, em (*ii*) e (*iii*), mostramos que p e q são números pares, o que contradiz o fato (expresso em (*i*)) de que um dos dois deve ser um número impar.

Logo, nossa suposição de que $\sqrt{2}$ é racional nos levou a uma contradição. Portanto, concluímos (por absurdo) que $\sqrt{2}$ é irracional.

O Método da Prova Indireta Também Demonstra Sentenças Afirmativas

Nos dois exemplos anteriores, usamos o método da prova indireta para provar sentenças negadas (**¬S**) (que iniciam por uma negação). Mas é possível usar este método para provar uma sentença afirmativa **S**. Para fazer isso, você deve iniciar assumindo como hipótese **¬S**, obter uma contradição e então concluir a negação de **¬S**, ou seja, **¬¬S**, que, como já sabemos, é equivalente a **S**.

Contradição

Para compreender e aplicar bem o método da prova por absurdo, é importante que você entenda o que é uma contradição, pois o método exige como, justificativa para a prova de **¬S**, que se obtenha uma contradição a partir da suposição temporária **S**.

Intuitivamente, uma contradição é qualquer asserção que não é possível que seja verdadeira, ou qualquer conjunto de asserções que não podem ser conjuntamente verdadeiras.

Alguns exemplos de contradições:

O conjunto de sentenças:	**Q** , **¬Q**
O conjunto de asserções:	**Cube(c)** , **Tet(c)**
A sentença:	**a ≠ a**
O conjunto de asserções:	***x*<*y*** , ***y*<*x***
A sentença:	**Q ∧ ¬Q**

Conjunto Inconsistente de Sentenças

Podemos, portanto, falar sobre a noção de um **conjunto contraditório ou inconsistente de sentenças** como sendo qualquer conjunto de sentenças para as quais não há nenhuma situação (ou circunstância) que tornem todas elas verdadeiras.

Símbolo do Absurdo: (⊥)

O símbolo ⊥ é freqüentemente utilizado como uma forma abreviada de dizer que uma contradição foi obtida.

Costumamos ler o símbolo ⊥ como "absurdo". Mas alguns lógicos o chamam de "o falso", outros de "contradição", alguns ainda de "*bottom*".

Qualquer que seja o nome, o importante é que o símbolo ⊥ indica que uma conclusão logicamente impossível foi obtida, ou que várias conclusões foram obtidas que, tomadas conjuntamente, são impossíveis.

Repare que uma sentença **S** é uma impossibilidade lógica (logicamente impossível) se e somente se sua negação **¬S** for logicamente necessária. Portanto, os métodos que temos para demonstrar que uma sentença é logicamente necessária também demonstram que sua negação é logicamente impossível, ou seja, que é uma contradição.

Por exemplo, se uma tabela de verdade demonstra que **¬S** é uma tautologia, então sabemos que **S** é uma contradição (lembre-se que **S** é logicamente equivalente a **¬¬S**).

TT-Contradição

O método das tabelas de verdade, além de poder mostrar que uma sentença é uma contradição, pode também mostrar que determinadas sentenças são mutuamente contraditórias.

Construa uma tabela de verdade conjunta para as sentenças (digamos P_1 ,..., P_n). Estas sentenças são **TT-contraditórias** se em cada linha da tabela há pelo menos uma sentença assinalada com **F**.

Se as sentenças são **TT-contraditórias**, nós sabemos, então, que elas não podem ser conjuntamente verdadeiras na mesma circunstância simplesmente em virtude dos significados de seus conectivos verofuncionais.

Usos da Prova Indireta em Argumentações do Cotidiano

O método de prova por absurdo, da mesma forma que o de prova por casos, também é utilizado com muita freqüência em nossos argumentos do dia-a-dia, embora a contradição derivada fique implícita algumas vezes.

Com freqüência as pessoas assumem uma hipótese e então mostram que desta suposição se segue uma falsidade. Elas então concluem a negação da hipótese originalmente assumida. Este tipo de raciocínio é, de fato, uma prova indireta.

Vejamos um exemplo: imagine um advogado de defesa apresentando o seguinte argumento aos jurados:

> O promotor afirma que meu cliente matou o dono do KitKat Club. Assumam que ele esteja correto. Vocês ouviram os especialistas da própria promotoria testemunharem que o assassinato ocorreu às 17:15 h. Nós também sabemos que o réu ainda estava trabalhando, na prefeitura, às 16:45 h, de acordo com os testemunhos de cinco colegas de trabalho. Logo meu cliente teria que ter ido da prefeitura ao KitKat Club em 30 minutos ou menos. Mas este percurso, na melhor das circunstâncias de trânsito possível, demora 35 minutos. Além disso, os arquivos policiais indicam que o trânsito estava bastante engarrafado no dia do assassinato. Eu, portanto, declaro que meu cliente é inocente.

Usamos argumentos como este todo o tempo: assumimos alguma coisa e então descartamos esta suposição com base em suas conseqüências falsas. Algumas vezes estas conseqüências nem são contradições, nem mesmo coisas que sabemos serem falsas, mas são apenas conseqüências futuras que consideramos inaceitáveis.

Você poderia, por exemplo, assumir que irá ao Rio de Janeiro nas férias de julho. Aí você calcula o impacto desta viagem em suas finanças e avalia sua capacidade para terminar todos os trabalhos das disciplinas que está cursando a tempo de viajar. Então você conclui, relutantemente, que não pode fazer a viagem. Quando você raciocina assim, está utilizando nada mais nada menos do que o método da prova indireta.

LEMBRE-SE

Prova por Absurdo (ou Indireta): Para provar $\neg S$ usando este método, assuma S e prove uma contradição $\bot$.

Argumentos com Premissas Inconsistentes

O que se segue de um conjunto inconsistente de premissas?

Se você pensar um pouco na definição de conseqüência lógica, concordará que qualquer sentença é conseqüência lógica de um conjunto inconsistente de premissas.

POR QUE ? Imagine um argumento com um conjunto inconsistente de premissas e uma conclusão. Se as premissas são contraditórias, então não há circunstâncias nas quais elas sejam todas verdadeiras. Portanto, qualquer que seja a sentença da conclusão, não há circunstâncias nas quais as premissas sejam verdadeiras e a conclusão falsa. Logo, qualquer sentença é conseqüência lógica de um conjunto inconsistente de premissas!

Esta conclusão nos leva a mais um passo de inferência válido.

Do Absurdo Tudo se Segue

Se, partindo de um conjunto de premissas, conseguirmos estabelecer uma contradição $\bot$, então estamos justificados a estabelecer qualquer sentença deste conjunto de premissas.

Muitos estudantes consideram este método de argumentação bastante esquisito. E por uma boa razão: um argumento deste tipo, apesar de *válido*, jamais será *correto*.

Vamos lembrar. Um argumento é *correto* quando for *válido* e tiver premissas verdadeiras. Portanto, mesmo sendo válidos, os argumentos com premissas inconsistentes não são jamais corretos, pois não há circunstância em que suas premissas sejam todas verdadeiras.

Por esta razão, um argumento com um conjunto inconsistente de premissas não tem muito valor por si próprio.

Argumentos Com Premissas Inconsistentes Não Garantem a Verdade de Sua Conclusão

Como um argumento com premissas inconsistentes nunca é correto, ele jamais garantirá a verdade de sua conclusão. Ou seja, ele é tão ruim para justificar a verdade de sua conclusão quanto um argumento inválido.

Em geral, os métodos de prova não nos permitem mostrar que um argumento é incorreto. Afinal de contas a verdade ou falsidade das premissas de um argumento não é um assunto da lógica, mas depende de como é o mundo. Mas no caso

de argumentos com premissas inconsistentes, nossos métodos de prova nos permitem mostrar que pelo menos uma das premissas é falsa (embora não saibamos qual delas), e então que o argumento é incorreto. Para fazer isso basta provar que das premissas se deduz uma contradição.

Suponha, por exemplo, que você prove o seguinte argumento:

$$\begin{array}{l} \text{Home(max)} \vee \text{Home(claire)} \\ \neg\text{Home(max)} \\ \neg\text{Home(claire)} \\ \hline \text{Home(max)} \wedge \text{Happy(carl)} \end{array}$$

Mesmo sendo verdade que a conclusão deste argumento é uma conseqüência lógica de suas premissas, sua conclusão não inspira muita confiança.

Usando prova por casos podemos mostrar que suas premissas são inconsistentes, e portanto, que o argumento é incorreto. De fato, não há razão para nos convencermos da verdade da conclusão de um argumento incorreto.

Mas não se engane, apesar de não terem valor sozinhos, os argumentos com premissas inconsistentes representam sim padrões de inferência que se mostrarão úteis quando combinados com outros padrões de inferência em uma prova. Esta utilidade se tornará evidente quando estivermos fazendo provas formalizadas no capítulo seguinte e nos próximos.

LEMBRE-SE

A prova de uma contradição $\perp$ a partir das premissas $P_1, \ldots, P_n$ (sem suposições adicionais) mostra que estas premissas são inconsistentes. Um argumento com premissas inconsistentes é sempre ***válido***, mas, mais importante ainda, é sempre ***incorreto***.

Capítulo 6 - Provas Formais e Lógica Booleana

Dedução Natural

O sistema dedutivo **F** que estamos utilizando é conhecido como um sistema de *Dedução Natural*. Pretende-se que tais sistemas sejam modelos para os princípios válidos de inferência usados nas provas informais. Neste capítulo apresentaremos as regras de inferência de **F** que correspondem aos princípios de inferência booleanos que discutimos no capítulo anterior. Você reconhecerá com facilidade as regras aqui apresentadas como sendo as contrapartes formais daqueles princípios de inferência.

Apesar de tentarem modelar nossas inferências informais, sistemas de dedução natural, como o sistema **F**, reproduzem estas inferências em versões relativamente longas e um pouco exageradas. Por exemplo, dissemos que, ao fazer uma prova informal, você sempre pode pressupor passos que você e sua audiência saibam que são logicamente válidos. Assim, se uma determinada equivalência lógica não é o objeto em questão de uma prova, você pode simplesmente aplicá-la em um passo único de uma prova informal. Contudo, o sistema **F** lhe dará uma coleção bastante elegante mas restrita de regras de inferência que você pode aplicar na construção de uma prova formal. Muitos dos passos de inferência válidos que já vimos (como as lies de DeMorgan) não são permitidos como passos únicos; eles precisam ser justificados em termos de passos mais básicos. A vantagem desta abordagem "magra mas significativa" é que ela torna mais fácil provar resultados *sobre* o sistema dedutivo, uma vez que quanto menos regras tiver, mais simples é o sistema.

Regras de Introdução e Eliminação

Sistemas de dedução natural como **F** utilizam duas regras para cada conectivo, uma que nos permite provar sentenças que contenham o símbolo e outra que nos permite provar coisas a partir de sentenças que contenham o símbolo. As primeiras são chamadas de *regras de introdução*, uma vez que elas nos permitem introduzir estes símbolos nas provas. Por contraste, as últimas são chamadas de *regras de eliminação*. Este tratamento é similar ao utilizado para o predicado da identidade visto no Capítulo 2.

Todas as regras formais de **F** foram implementadas no programa **Fitch**, permitindo que você construa suas provas formais muito mais facilmente do que tivesse que escrevê-las à mão. Na verdade, a interpretação de **Fitch** para as regras de introdução e eliminação são um pouco mais generosas do que em **F**. Elas não permitem que você faça nada que **F** não pudesse também permitir, mas há casos onde **Fitch** permitirá que você faça em apenas um passo coisas que levariam vários passos em **F**.

Aplicações *Default* de Regras no Programa FITCH

Além disso, muitas das regras de **Fitch** possuem "*aplicações default*" que podem economizar bastante tempo. Para utilizar a aplicação *default* de alguma regra, basta que você especifique qual é a regra e cite os passos de suporte. Então você clica no botão ***Check Step*** e o programa **Fitch** preencherá automaticamente a conclusão apropriada.

Ao final de cada seção abaixo explicaremos a aplicação *default* das regras introduzidas na seção.

[1] Regras para a Conjunção $\wedge$

As regras de eliminação da conjunção (**∧Elim**) e introdução da conjunção (**∧Intro**) são os princípios mais simples de serem formalizados.

Eliminação da Conjunção (∧Elim)

A regra de eliminação da conjunção nos permite afirmar qualquer 'conjunto' P_i a partir de uma sentença $P_1 \wedge \ldots \wedge P_i \wedge \ldots \wedge P_n$ que já tenha sido derivada em uma prova. (a propósito, P_i pode ser qualquer conjunto, incluindo o primeiro e o último – $1 \leq i \leq n$). O novo passo é justificado citando-se o passo que contém a conjunção.

O seguinte esquema simboliza esta regra:

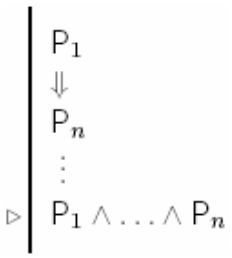
$$\begin{array}{|l} P_1 \wedge \ldots \wedge P_i \wedge \ldots \wedge P_n \\ \vdots \\ \triangleright\ P_i \end{array}$$

Faça o Tente Isto (1) da Lista de Exercícios 5-6 para exercitar o uso desta regra no programa **Fitch**.

Introdução da Conjunção (∧Intro)

A regra de introdução da conjunção permite a você afirmar a conjunção $P_1 \wedge \ldots \wedge P_n$ contanto que você já tenha estabelecido cada um de seus conjuntos constituintes P_1 até P_n em uma prova.

Simbolizaremos esta regra da seguinte forma:



A notação $P_1 \Downarrow P_n$

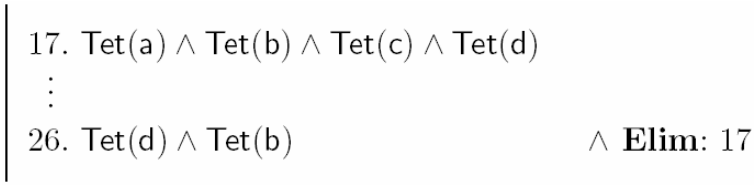
Apenas indica que cada um dos P_1 até P_n deve aparecer na prova antes de que você possa afirmar a conjunção deles. Eles não precisam aparecer em ordem nem em uma seqüência compacta (um logo depois do outro). Eles apenas precisam ocorrer em alguma parte anterior da prova.

Veja um exemplo simples de uso conjunto das duas regras da conjunção. É uma prova de $C \wedge B$ a partir de $A \wedge B \wedge C$.

Pratique a utilização destas regras no programa **Fitch** fazendo o Tente Isto (2) da Lista de Exercícios 5-6.

Usos Generosos das regras ∧ em FITCH

Como dissemos, o programa **Fitch** é generoso em sua interpretação das regras de inferência de **F**. Por exemplo, **Fitch** considera o seguinte exemplo um uso aceitável da regra **(∧Elim)**.



O que fizemos aqui foi pegar 2 conjuntos da conjunção do passo 17 e asserir a conjunção deles no passo 26. Tecnicamente, **F** deveria requerer que nós derivássemos os dois conjuntos separadamente, fazendo duas aplicações de **(∧Elim)**, e então, com uma aplicação de **(∧Intro)**, juntássemos os dois na conjunção da linha 26. **Fitch** permite que façamos isso em um único passo.

Uma vez que o programa **Fitch** permite que você pegue qualquer coleção dos conjuntos de uma sentença citada e afirme a conjunção deles em qualquer ordem, então a interpretação de **Fitch** para a regra **(∧Elim)** permite provar, em um único passo, que a conjunção é "*comutativa*". Em outras palavras, você pode usar **(∧Elim)** para reordenar os conjuntos de uma conjunção da forma que quiser:

Pratique estas possibilidades de aplicação fazendo o Tente Isto (3) da Lista 5-6.

A regra (∧Intro) implementada em **Fitch** também é menos restritiva do que a nossa discussão da regra formal poderia sugerir. Em primeiro lugar, **Fitch** não se importa com a ordem na qual você cita as sentenças de suporte. Segundo, se você cita uma sentença, ela pode aparecer mais do que uma vez como um conjunto da sentença resultante.

Aplicações *Default* das Regras ∧ em Fitch

Se em um passo novo de uma prova você especifica a regra como (∧Elim) e cita uma conjunção como suporte, então se verificar o passo (clicar no botão ***Check Step***) **Fitch** preencherá o passo em branco com o conjunto mais à esquerda da sentença citada.

Se você especificar a regra (∧Intro) e citar várias sentenças em suporte, **Fitch** preencherá o passo em branco com uma conjunção formada pelas sentenças dos passos citados, na ordem em que eles foram citados.

Pratique estas aplicações generosas e *default* fazendo o Tente Isso (4) da Lista 5-6.

Parênteses e as Regras da Conjunção

Ao aplicar a regra de introdução da conjunção, você precisa ser cuidadoso com os parênteses. Se um dos conjuntos é, ele mesmo, uma conjunção, então é claro que não há necessidade de colocá-lo entre parênteses antes de formar a conjunção maior, a menos que você queira. Por exemplo, as duas seguintes aplicações da regra (∧Intro) são corretas. (A primeira delas é a que o mecanismo *default* de **Fitch** lhe daria.)

Correto:

1. A ∧ B
2. C

---

3. (A ∧ B) ∧ C ∧ Intro: 1, 2

Correto:

1. A ∧ B
2. C

---

3. A ∧ B ∧ C ∧ Intro: 1, 2

No entanto, se um dos conjuntos é ele próprio uma disjunção (ou alguma sentença complexa), para evitar ambigüidade você deve colocá-lo entre parênteses na sentença resultante. Assim, o primeiro exemplo abaixo é correto, mas o segundo está errado, uma vez que a sentença final é ambígua.

Correto:

1. A ∨ B
2. C

---

3. (A ∨ B) ∧ C ∧ Intro: 1, 2

Errado:

1. A ∨ B
2. C

---

3. A ∨ B ∧ C ∧ Intro: 1, 2

[2] Regras para a Disjunção ∨

Está bem, sabemos que as regras da conjunção dão tédio de tão simples. Mas as da disjunção não. Especialmente a regra de eliminação da disjunção.

Introdução da Disjunção (∨Intro)

A regra de introdução da disjunção permite a você inferir, a partir de uma sentença P_i, qualquer disjunção que tenha P_i entre os seus disjuntos constituintes, digamos $P_1 \vee \ldots \vee P_i \vee \ldots \vee P_n$. Sua forma esquemática é:

Reforçamos que P_i pode ser inclusive o primeiro ou último disjunto da conclusão. Além disso, os mesmos alertas que fizemos sobre os parênteses na regra (∧Intro) são pertinentes também para esta regra para evitar ambigüidade.

Antes de apresentarmos um exemplo interessante de aplicação desta regra, precisamos ter disponível a outra regra da disjunção.

Eliminação da Disjunção (∨Elim)

Esta é a primeira regra que corresponderá ao que chamamos no capítulo anterior de um *método de prova*. As regras anteriores apenas formalizam o que chamamos de passos válidos de inferência, que são bastante triviais.

A regra da eliminação da disjunção é a contraparte formal do *método de prova por casos*. Lembremos que as provas por casos nos permitem concluir uma sentença **S** a partir de uma disjunção $P_1 \vee \ldots \vee P_n$, se conseguirmos provar **S** a partir de cada um dos P_1 até P_n individualmente.

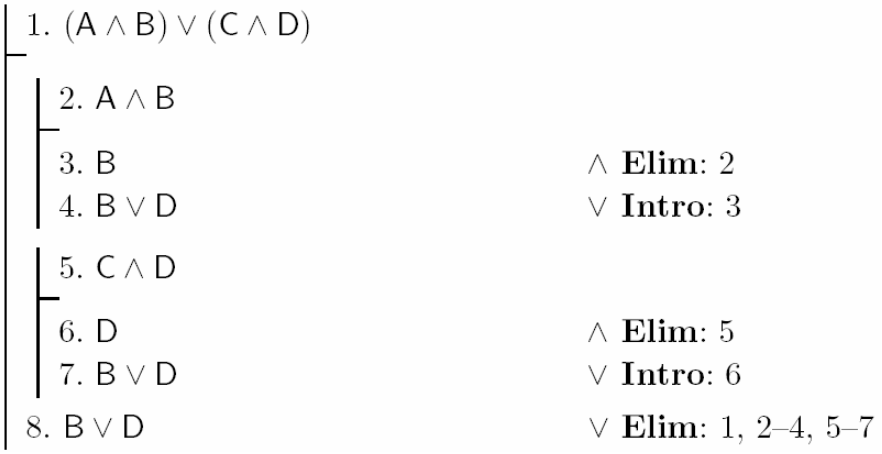
A forma desta regra exige a discussão de uma nova e importante característica estrutural dos sistemas de dedução do estilo Fitch. É a noção de ***subprova***.

Subprovas

Uma subprova, como o nome sugere, é uma prova que ocorre dentro do contexto de uma prova maior.

Da mesma forma que qualquer prova, uma subprova geralmente começa com uma suposição, separada do resto da subprova pela barra Fitch. Mas a suposição de uma subprova, diferentemente de uma premissa da prova principal, é assumida apenas temporariamente. Ao longo da subprova, a suposição funciona exatamente como uma premissa adicional. Mas após a subprova, a suposição não está mais em vigor.

Antes de apresentarmos a forma esquemática da eliminação da disjunção, vamos olhar para uma prova particular que usa a regra. Ela servirá como um exemplo concreto de como as subprovas aparecem em **F**.



Se substituirmos **A**, **B**, **C** e **D** por **Home(max)**, **Happy(cleo)**, **Home(claire)** e **Happy(miau)**, esta prova se torna a formalização da prova que apresentamos na página 4 do Capítulo 5. Ela contém duas subprovas. Uma delas vai da linha 2 à 4, e mostra que $B \vee D$ é obtido quando assumimos (temporariamente) $A \wedge B$. A outra vai da linha 5 à 7, e mostra que a mesma conclusão ($B \vee D$) é obtida quando assumimos $C \wedge D$.

Estas duas subprovas, juntas com a premissa $(A \wedge B) \vee (C \wedge D)$, são exatamente o que precisamos para aplicar o método da prova por casos ou, conforme nós iremos chamá-lo, a regra da eliminação da disjunção.

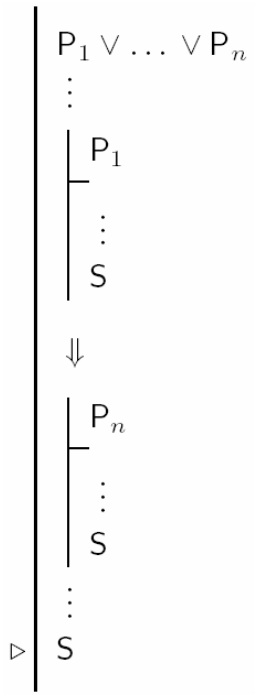
Examine cuidadosamente esta prova e compare-a com a prova informal apresentada na página 4 do Capítulo 5 para ver se você entende o que está acontecendo.

Repare que os passos das suposições das duas subprovas não precisam ser justificados por nenhuma regra, da mesma forma que a premissa da prova maior (principal) também não requer justificação. Isto porque nós não estamos

reivindicando que estas suposições se sigam do que ocorre antes delas na prova, mas estamos apenas assumindo-as e mostrando que certas sentenças se seguem destas suposições.

Note também que utilizamos a regra **(∨Intro)** duas vezes nesta prova, uma vez que esta é a única forma de derivar a sentença desejada em cada subprova. Apesar de parecer que estamos jogando informação fora quando inferimos **B ∨ D** a partir da afirmação mais forte **B**, quando você considera a prova como um todo, torna-se claro que **B ∨ D** é a afirmação mais forte que se segue da premissa original.

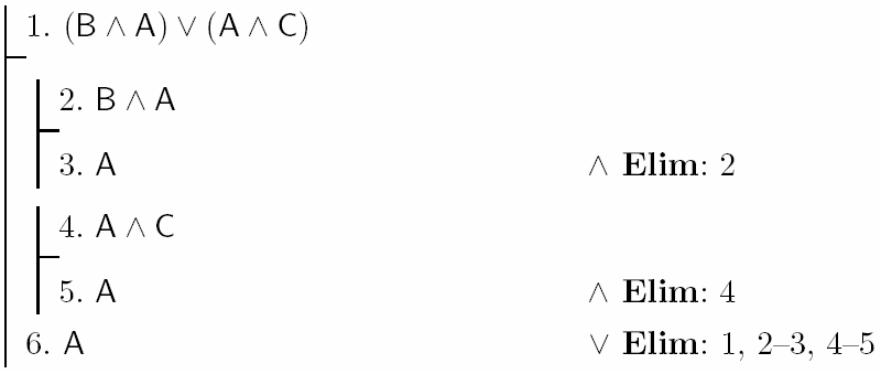
Podemos agora apresentar a versão esquemática da eliminação da disjunção. **(∨Elim)**:



O que o esquema acima denota é que se você obteve a disjunção **P1 ∨ ... ∨ Pn**, e mostrou que **S** se segue de cada um dos disjuntos, de **P1** até **Pn**, entÃo você pode concluir **S**.

Novamente, não importa a ordem em que as subprovas aparecem, ou mesmo se elas estão depois da disjunção. Quando aplicar a regra, você deve citar o passo contendo a disjunção mais cada uma das subprovas requeridas.

Vejamos outro exemplo desta regra, para enfatizar como as justificações (citações) envolvendo subprovas deve ser feitas. Aqui abaixo temos uma prova que mostra que **A** é conseqüência da sentença **(B ∧ A) ∨ (A ∧ C)**.



A justificativa para o passo 6 mostra a forma que devemos citar subprovas. A citação "*n*-*m*" é o modo que usamos para nos referirmos a uma subprova que começa na linha *n* e termina na linha *m*.

Algumas vezes, quando usar eliminação da disjunção, você achará natural usar a regra de reiteração introduzida no Capítulo 3. Por exemplo, suponha que modifiquemos a prova acima para mostrar que **A** se segue de **(B ∧ A) ∨ A**.

Aqui, a suposição da segunda subprova é **A**, que é exatamente a sentença que queremos provar. Então, tudo o que temos a fazer é repetir a sentença para que tenhamos uma subprova de **A** a partir de **A**. (Poderíamos também deixar a subprova com apenas um passo, a suposição, mas, para estes casos, é mais natural usar a reiteração.)

Faça o Tente Isto (5) da Lista 5-6 para experimentar o uso das regras da disjunção.

Usos Generosos das Regras ∨ em FITCH

Há algumas formas nas quais **Fitch** é menos rigoroso na verificação de **(∨Elim)** do que é sugerido pela forma estrita da regra.

Primeiro, a sentença **S** não precisa ser a última sentença da subprova, embora usualmente ela seja. **S** simplesmente precisa aparecer no "*nível principal*" de cada subprova.

Segundo, se você inicia com uma disjunção contendo mais de dois disjuntos, digamos **P ∨ Q ∨ R**, **Fitch** não exige três subprovas. Se você tem uma subprova começando com **P** e uma começando com **Q ∨ R**, ou uma iniciando com **Q** e outra com **P ∨ R**, então **Fitch** ficará satisfeito e permitirá a utilização de **(∨Elim)**, com tanto que você tenha provado **S** em cada uma delas.

Aplicações *Default* das Regras ∨ em FITCH

Quando você cita o suporte apropriado para a regra **(∨Elim)** (isto é, a disjunção e as subprovas para cada disjuto) e verifica o passo (sem digitar a sentença resultante), **Fitch** olhará para as subprovas citadas e, se todas elas terminam com a mesma sentença, preencherá esta sentença no passo final.

Se você cita uma sentença e aplica **(∨Intro)** sem preencher a sentença resultante, **Fitch** insere a sentença citada, seguida por ∨, deixando o cursor de inserção logo após a ∨ para que você digite o restante da disjunção que tinha em mente.

Faça o Tente Isto (6) da Lista 5-6 para praticar estas aplicações *Default*.

[3] Regras para a Negação ¬

Para terminar, as regras da negação. Veremos que a introdução da negação será a mais interessante e mais complexa regra dos conectivos booleanos.

Eliminação da Negação (¬Elim)

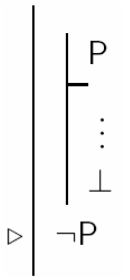
A regra de eliminação da negação corresponde ao passo de inferência válido bastante trivial em que, a partir de **¬¬P** infere-se **P**. Esquematicamente:

A eliminação da negação nos dá uma das direções do princípio da dupla negação (de $\neg\neg P$ obtém-se P). Seria razoável esperar que a regra de introdução da negação nos desse a outra direção (de P obtém-se $\neg\neg P$). Mas a história não é bem essa.

Introdução da Negação (¬Intro)

A regra de introdução da negação corresponde ao *método de prova indireta* (ou prova por absurdo) que estudamos informalmente no capítulo anterior. Da mesma forma que (∨Elim) ela envolve o uso de subprovas, e o mesmo se dará com todos os métodos de prova não triviais que estudaremos.

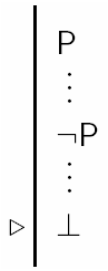
A regra diz que se você puder provar uma contradição $\bot$ com base em uma suposição adicional P, então você pode inferir $\neg P$ a partir das premissas originais. Esquematicamente:



Há diferentes formas de entender esta regra, dependendo de como interpretamos o símbolo do absurdo $\bot$. Alguns autores interpretam-no como uma abreviação para qualquer contradição da forma $Q \wedge \neg Q$. Se tivéssemos construído nossa interpretação desta forma, não precisaríamos falar mais nada sobre a regra. Mas trataremos $\bot$ como um símbolo independente que deve ser lido como "absurdo" (ou contradição). Isto tem várias vantagens que se tornarão aparentes quando você começar a usar a regra. A única desvantagem é que, como inserimos um símbolo a mais em nossa linguagem (FOL), precisamos de regras de inferência para este símbolo também. Apresentamos estas regras a seguir.

Introdução do Absurdo (⊥Intro)

A regra de introdução do absurdo (⊥Intro) permite-nos obter o símbolo do absurdo sempre que tivermos estabelecido uma contradição explicita na forma das sentenças P e sua negação $\neg P$.



Em geral, você aplicará a regra (⊥Intro) apenas no contexto de alguma subprova, para mostrar que a suposição da subprova leva a uma contradição. O único caso em que você será capaz de derivar $\bot$ em sua prova principal[1] é quando as premissas do argumento que está querendo provar forem elas próprias inconsistentes.

Provas Formais de Inconsistência

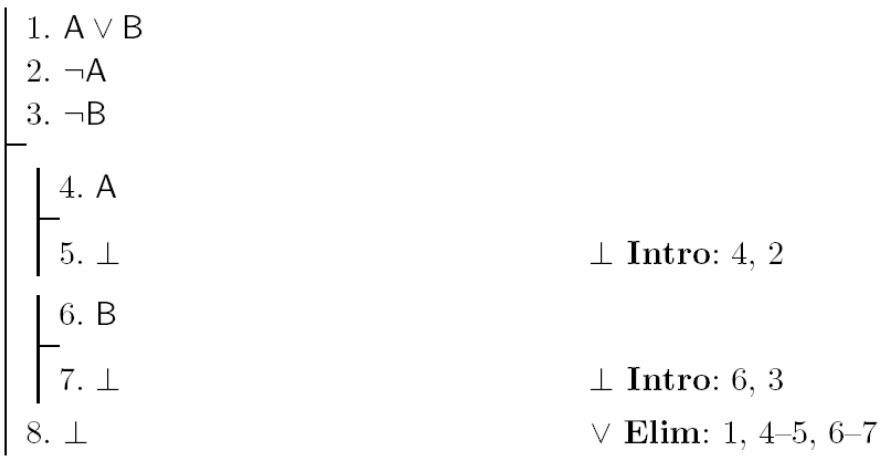
De fato, é assim que fazemos uma prova formal de que um determinado conjunto de premissas é inconsistente. Uma prova formal de inconsistência é, portanto, uma prova que deriva $\bot$ em seu nível principal.

Faça o Tente Isto (7) da Lista de Exercícios 5-6 para praticar a aplicação das regras (⊥Intro) e (¬Intro).

[1] Nos referimos a "prova principal" ou "nível principal" em oposição a "subprova". Prova principal é a prova mais externa, que contém subprovas.

O símbolo $\bot$ funciona exatamente da mesma forma que qualquer outra sentença em uma prova. Em particular, se você está raciocinando por casos e obtém $\bot$ em cada uma de suas subprovas, então você pode utilizar **(∨Elim)** para derivar $\bot$ em sua prova principal.

Veja, como exemplo, uma prova que mostra que as premissas $A \vee B$, $\neg A$ e $\neg B$ são inconsistentes.



O fato importante a ser notado aqui é o passo 8, onde aplicamos **(∨Elim)** para extrair o símbolo do absurdo de nossas duas subprovas e colocá-lo na prova principal. Este passo está claramente justificado, uma vez que mostramos que assumindo qualquer um dos dois **A** ou **B**, nós imediatamente chegamos a uma contradição. Como as premissas nos dizem que um ou outro deve ser verdadeiro, então as premissas são inconsistentes.

Outras Formas de Introduzir $\bot$

A regra **(⊥Intro)** reconhece apenas as contradições mais descaradas, aquelas onde você estabeleceu uma sentença **P** e sua negação **¬P**. O que acontece se, no decorrer de uma prova, você encontrar uma inconsistência de alguma outra forma?

Por exemplo, suponha que você deriva uma sentença única que é TT-contraditória, como **¬(A ∨ ¬A)**, ou duas sentenças, como **¬A ∨ ¬B** e **A ∧ B** que juntas formam um conjunto TT-contraditório, mas que não são uma a negação da outra?

Ocorre que se você pode provar qualquer conjunto TT-contraditório de sentenças ou qualquer sentença TT-contraditória, as regras que já apresentamos a você lhe permitirão provar $\bot$. Isso pode até exigir bastante trabalho e criatividade, mas é possível. Mais adiante provaremos este resultado, mas por enquanto, apenas acredite!

Introduzindo $\bot$ com (TautCon)

Uma forma de checar se algumas sentenças são TT-contraditórias é tentar derivar $\bot$ delas usando uma única aplicação do mecanismo **(TautCon)**. Em outras palavras, entre o símbolo $\bot$, escolha **(TautCon)** como regra e cite as sentenças que quer verificar se são contraditórias. Se **(TautCon)** lhe disser que $\bot$ se segue das sentenças citadas (ou seja, se ao clicar ***check step*** a resposta é OK), então pode ter certeza de que é possível provar isso usando apenas as regras de introdução e eliminação para $\wedge$, $\vee$, $\neg$ e $\bot$.

Claro que há outras formas de contradição além das TT-contradições. Por exemplo, suponha que você tenha demonstrado as seguintes três sentenças: **Cube(b)**, **b = c** e **¬Cube(c)**. Estas sentenças não são TT-contraditórias, mas você pode ver que uma simples aplicação de **(=Elim)** lhe dará o par TT-contraditório **Cube(c)** e **¬Cube(c)**.

Introduzindo $\bot$ com (FOCon)

Se você suspeitar que derivou algumas sentenças cuja inconsistência resulta dos conectivos booleanos *mais* o predicado de identidade, você pode verificar isso utilizando o mecanismo **(FOCon)**, uma vez que **(FOCon)** entende o significado de **=**.

Se **(FOCon)** diz que $\bot$ se segue das sentenças citadas (e se estas sentenças não contém quantificadores), então você deve ser capaz de provar $\bot$ utilizando apenas as regras de introdução e eliminação para $=$, $\wedge$, $\vee$, $\neg$ e $\bot$.

Introduzindo ⊥ com (AnaCon)

O único caso em que você pode chegar a uma contradição mas não ser capaz de provar ⊥ com as regras de **F** é quando a inconsistência depende do significado de outros predicados além da identidade. Por exemplo, suponha que você deriva a contradição **n < n**, ou o par contraditório de sentenças **Cube(b)** e **Tet(b)**. As regras de **F** não lhe possibilitarão obter uma contradição da forma **P** e **¬P**. Pelo menos não sem premissas adicionais.

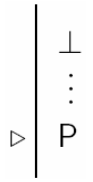
Isto significa que em **Fitch**, o mecanismo **(AnaCon)** permitirá que você estabeleça contradições que não podem ser derivadas em **F**. Claro que o mecanismo **(AnaCon)** só entende os predicados da linguagem dos blocos (e mesmo aí, exclui os predicados **Adjoins** e **Between**).

Mas ele permitirá, por exemplo, que você derive ⊥ das sentenças **Cube(b)** e **Tet(b)**. Você pode ou fazer isso diretamente, digitando ⊥ e citando as duas sentenças, ou indiretamente, utilizando **(AnaCon)** para provar, digamos, **¬Cube(b)** a partir de **Tet(b)**.

Faça o Tente Isto (8) da Lista 5-6 para praticar estas formas alternativas de introduzir ⊥.

Eliminação do Absurdo (⊥Elim)

Conforme comentamos no Capítulo 5, em uma prova (ou subprova), se você foi capaz de estabelecer uma contradição (obter ⊥) então você está justificado a asserir qualquer sentença de FOL **P**. No nosso sistema formal, este passo de inferência é modelado pela regra de eliminação do absurdo **(⊥Elim)**. Esquematicamente:



O Tente Isto (9) da Lista 5-6 ilustra as duas regras para ⊥ e apresenta uma tática de prova que lhe será útil em várias ocasiões no futuro.

Na verdade, nós não precisaríamos de uma regra nova para **(⊥Elim)**. Você consegue provar qualquer sentença a partir de uma contradição sem utilizar esta regra. Suponha, por exemplo, que você estabeleceu uma contradição no passo 17 de alguma prova. Segue um exemplo de como introduzir **P** (e **P** poderia ser qualquer sentença) no passo 21 sem utilizar **(⊥Elim)**.

Mesmo assim, incluímos a regra **(⊥Elim)** para tornar nossas provas menores e mais naturais.

Usos Generosos das Regras ¬ em FITCH

A regra **(¬Elim)** permite a você retirar dois símbolos de negação da frente de uma sentença. Usos repetidos desta regra permitem a retirada de quatro, seis, ou qualquer número par de símbolos de negação. Por esta razão, a implementação de **(¬Elim)** em **Fitch** permite a remoção de qualquer número par de símbolos de negação em um único passo.

Aplicações *Default* das Regras ¬ em FITCH

Em uma aplicação *default* de **(¬Elim)**, **Fitch** removerá tantos símbolos de negação quanto for possível (sempre um número par!) da frente da sentença citada e insere o resultado no passo justificado pela regra **(¬Elim)**.

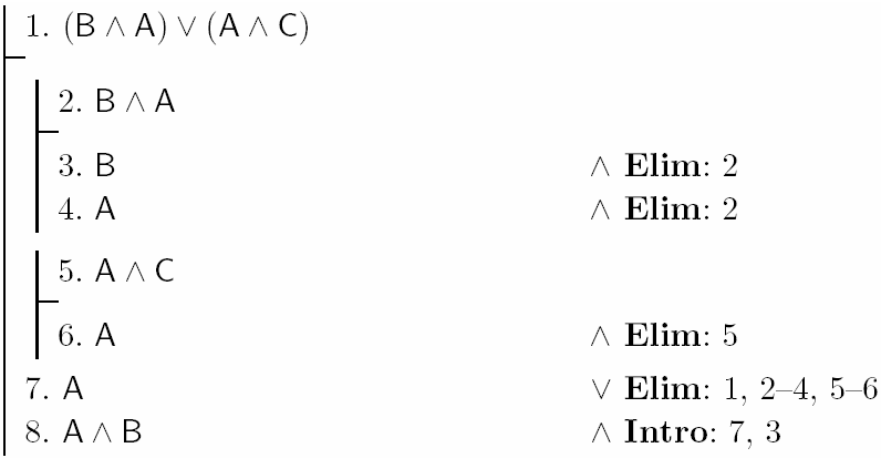
Em uma aplicação *default* da **(¬Intro)** a sentença inserida automaticamente por **Fitch** será exatamente a negação da suposição da subprova citada.

Faça o Tente Isto (10) da Lista 5-6 para praticar estes usos das regras da negação.

[4] O Uso Correto das Subprovas

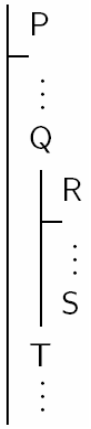
Subprovas são a característica principal dos sistemas de dedução do estilo Fitch. É importante que você entenda como utilizá-las corretamente.

A prova formal abaixo, parece que foi construída de acordo com nossas regras, mas ela pretende provar que **A ∧ B** se segue de **(B ∧ A) ∨ (A ∧ C)**, o que, claramente, não está correto.



O problema está no passo 8. Neste passo, citamos como suporte para aplicação da regra **(∧Intro)** o passo 3, que ocorre dentro de uma subprova anterior. Mas acontece que este tipo de justificação – as que se utilizam de sentenças de uma subprova que já foi concluída – não é legítimo. Para entender porque isso não é legítimo, é necessário que você pense um pouco sobre a função que as subprovas têm em uma inferência.

Uma subprova, tipicamente, se parece com isto:



Descartando Suposições ao Finalizar Subprovas

As subprovas iniciam pela introdução de uma nova suposição. **R** no exemplo acima. As inferências dentro da subprova dependem desta nova suposição, junto com qualquer outra premissa ou suposição da prova principal. Então, no nosso exemplo, a derivação de **S** pode depender de ambos, **P** e **R**.

Quando a subprova termina, o que é indicado pelo fim da linha vertical que limita a subprova, as inferências subseqüentes não podem mais utilizar a suposição da subprova nem nada que dependa desta suposição. Nós dizemos que a suposição foi *descartada* ou que a subprova foi *finalizada*.

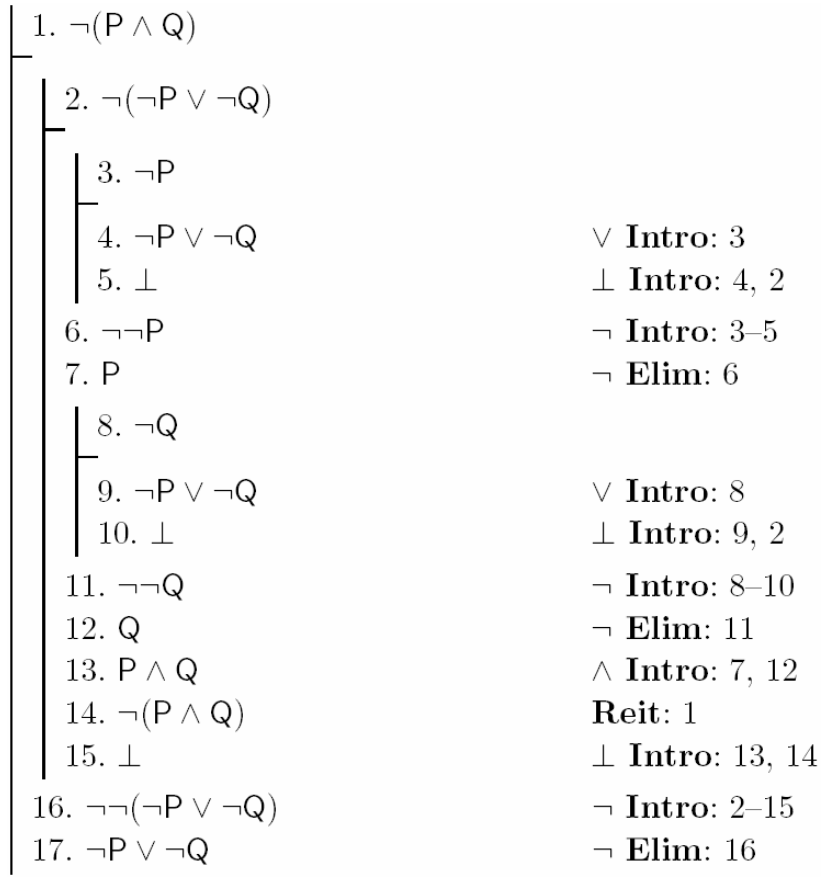
Quando uma suposição foi descartada, os passos individuais em sua subprova não são mais acessíveis. É apenas a subprova como um todo que pode ser citada como justificação para um passo posterior. Isto significa que ao justificar **T** no exemplo acima, nós podemos citar **P**, **Q** e a subprova como um todo, mas não podemos citar itens individuais

internos à subprova, tais como **R** ou **S**. Pois estes passos dependem de suposições que não estão mais disponíveis (foram descartadas). Uma vez que a subprova tenha sido finalizada, elas não são mais acessíveis.

Este foi o erro que cometemos no passo 8 da prova falaciosa que apresentamos anteriormente. Citamos um passo interno a uma subprova que já tinha sido finalizada (o passo 3). A sentença naquele passo, **B**, tinha sido provada com base na suposição $B \wedge A$, uma suposição que havíamos feito apenas temporariamente para resolver um dos casos de um argumento por casos. A suposição $B \wedge A$ não está mais em vigor no passo 8 e, portanto, não pode ser utilizada neste ponto.

Esta proibição não nos impede de citar, a partir do interior de uma subprova, passos anteriores fora da subprova, desde que não ocorram no interior de outra subprova já terminada. Por exemplo, na prova esquemática apresentada acima, a justificação de **S** poderia muito bem incluir o passo que contém **Q**. Já a justificação de **T** não pode incluir o passo que contém **S**.

Esta observação se torna mais sugestiva quando você está trabalhando em uma subprova de uma subprova. Ainda não vimos nenhum exemplo em que precisássemos de subprovas dentro de subprovas, mas tais exemplos são bastante fáceis de encontrar. Aqui está um em que provamos uma das direções da primeira lei de DeMorgan.



Note que a subprova 2-15 contém duas subprovas, 3-5 e 8-10. No passo 5 da subprova 3-5 nós citamos o passo 2 da subprova parental 2-15. Similarmente, no passo 10 da subprova 8-10, nós citamos o passo 2. Isto é legítimo, pois a subprova 2-15 ainda não havia terminado no passo 10. Apesar de não termos precisado fazer isso neste exemplo, poderíamos ter citado o passo 1 em qualquer das subprovas.

Note também que no passo 14 utilizamos a regra de reiteração **(Reit)**. Não era necessária a sua utilização aqui, só a utilizamos para ilustrar o seguinte ponto:

Com relação a subprovas, a reiteração é como qualquer outra regra. Quando você a usa, você pode citar passos fora da subprova imediata, se as provas que contém os passos citados ainda não terminaram. Mas você não pode citar um passo no interior de uma subprova que já foi finalizada. Por exemplo, se substituíssemos a justificação do passo 15 por **(Reit: 10)**, então nossa prova não estaria mais correta.

Subprovas Aninhadas (Subprovas Dentro de Subprovas)

Como você verá, a maioria das provas em **F** exigirá subprovas dentro de subprovas, o que nós chamamos de subprovas aninhadas. Para criar uma tal subprova em **Fitch**, você precisa apenas selecionar ***New Subproof*** no menu ***Proof*** quando já estiver dentro da primeira subprova.

LEMBRE-SE

- Ao justificar um passo em uma subprova, você pode citar qualquer passo anterior contido na prova principal ou em qualquer subprova cuja suposição ainda esteja em vigor. Você não pode jamais citar um passo individual interno a uma subprova que já tenha sido finalizada.
- Fitch lhe impõe estas restrições automaticamente ao não permitir a citação de passos individuais internos a subprovas que já tenham sido finalizadas.

[5] Estratégias e Táticas

Alguns estudantes tentam construir provas formais tentando cegamente, ao acaso, juntar seqüências de passos permitidos pelas regras de introdução e eliminação. Um processo não mais racional do que jogar paciência. Esta abordagem pode até ocasionalmente funcionar, mas em geral falha, ou de qualquer modo, torna mais difícil a tarefa de construir uma prova.

Nesta seção apresentaremos alguns conselhos sobre como agir quando estiver tentando fazer uma prova e a solução não estiver pulando diante de seus olhos. Os conselhos consistem em duas estratégias importantes e uma máxima essencial.

Máxima Essencial

Tenha sempre, bastante claro em sua mente o significado das sentenças de sua prova.

Ao prestar atenção no significado das sentenças com que está trabalhando, você evitará MUITAS armadilhas, entre elas a de tentar provar uma sentença que na verdade não é conseqüência das premissas.

Seu primeiro passo ao tentar construir uma prova deverá ser ***sempre*** se convencer de que a conclusão é conseqüência das premissas.

Isso porque no processo de entendimento do significado das sentenças e de reconhecimento da validade do argumento, você freqüentemente já terá alguma idéia sobre como fazer a prova.

(Estratégia 1) Tente uma Prova Informal

Após se convencer que o argumento é de fato válido, a primeira estratégia é tentar fazer uma prova informal, do tipo que você usaria para convencer um colega de classe.

Freqüentemente, a estrutura básica de seu raciocínio informal pode ser diretamente formalizada utilizando-se as regras de **F**. Por exemplo, se você se viu usando o método da prova indireta em sua prova informal, então esta parte de seu raciocínio provavelmente exigirá o uso da introdução da negação (**¬Intro**). Se você usou prova por casos, então você quase que certamente usará a eliminação da disjunção (**∨Elim**) em sua prova formal.

(Estratégia 2) Trabalhando de Trás para Frente

Suponha que você já se convenceu de que o argumento é válido, mas está com dificuldades em encontrar uma prova informal para ele. Ou ainda que você não consegue ver como sua prova informal poderia ser convertida para uma prova que utilize apenas as regras de **F**. A segunda estratégia ajuda em qualquer um destes casos. Ela é conhecida como "trabalhando de trás para frente."

O que você faz é olhar para a conclusão e pensar qual sentença ou sentenças adicionais poderiam permitir que inferíssemos esta conclusão. Então você simplesmente insere estes passos (estas sentenças) em sua prova, não se preocupando sobre como exatamente eles serão justificados, e cite-os como suporte da sentença final (a conclusão). Agora você considera estes passos intermediários, que você acabou de inserir na prova, como as sentenças que precisam ser provadas e veja se consegue prová-los. Se conseguir, sua prova estará completa.

Vamos desenvolver um exemplo que aplica estas duas estratégias. Suponha que você seja requisitado a fazer uma prova formal do seguinte argumento:

$$\neg P \vee \neg Q \vdash \neg(P \wedge Q)$$

A primeira coisa a fazer é entender o significado das sentenças. Ao fazer isso, você verá que este argumento é a aplicação em uma direção de uma das leis de DeMorgan. Portanto ele é um argumento válido. Reconhecer que é um argumento válido já representou mais um passo de nossa tarefa.

Quando você pensa sobre o argumento, você pode descobrir que o que lhe convence da validade do argumento é a seguinte observação, que é difícil de formalizar:

- Se a premissa é verdadeira, então ou **P** ou **Q** é falsa, o que torna $P \wedge Q$ falsa; e então a conclusão é verdadeira.

Embora este seja um argumento completamente convincente, não é imediatamente claro como ele deveria ser traduzido em termos das regras de introdução e eliminação do sistema **F**.

Vamos tentar trabalhar de trás para frente para ver se obtemos uma prova informal que seja mais fácil de formalizar.

Uma vez que a conclusão é uma negação, podemos prová-la assumindo $P \wedge Q$ e derivando uma contradição (método da prova indireta). Então vamos supor $P \wedge Q$ e tomar $\perp$ como nossa nova meta. Agora as coisas parecem um pouco mais claras; pois das premissas sabemos que ou $\neg P$ ou $\neg Q$ é verdadeira, mas qualquer destes casos contradiz um dos conjuntos da conjunção que assumimos ($P \wedge Q$). Então, uma prova por casos vai nos permitir derivar uma contradição.

Apenas para registrar mais claramente estes raciocínios, segue abaixo uma prova informal que representa esta discussão.

> **PROVA**: Temos como premissa $\neg P \vee \neg Q$ e queremos provar $\neg(P \wedge Q)$. Objetivando fazer uma prova indireta, vamos assumir $P \wedge Q$ e tentar derivar daí uma contradição. Há dois casos a considerar, uma vez que, de nossa premissa, ou $\neg P$ ou $\neg Q$ é verdadeira. Mas cada uma destas sentenças contradiz a suposição que fizemos $P \wedge Q$: $\neg P$ contradiz o primeiro conjunto e $\neg Q$ contradiz o segundo. Conseqüentemente, nossa suposição leva a uma contradição e, por absurdo, concluímos sua negação $\neg(P \vee Q)$, o que termina a prova.

O Tente Isto (11) da Lista 5-6 ajuda você a construir a prova formal que modela esta argumentação informal.

Armadilhas ao Trabalhar de Trás para Frente

Apesar de ser uma ótima técnica, não pense que basta aplicar mecanicamente a estratégia de trabalhar de trás para frente que todos os problemas serão resolvidos. É essencial que você pare a cada passo e verifique se as novas metas que precisa provar são razoáveis. Se elas não parecerem plausíveis, você deve tentar alguma outra coisa.

Aqui está um exemplo de por que esta checagem constante é tão importante. Suponha que você seja solicitado a provar que a sentença $A \vee C$ é conseqüência de $(A \wedge B) \vee (C \wedge D)$. Trabalhando de trás para frente você poderia pensar que se pudesse provar **A**, então você poderia inferir a conclusão desejada com uma aplicação de $(\vee\textbf{Intro})$. Esquematizada, sua prova parcial se pareceria com esta:

1. $(A \wedge B) \vee (C \wedge D)$	
2. A	**Rule?**
3. $A \vee C$	$\vee$ **Intro**

O problema em fazer isso é que **A** não é conseqüência da premissa dada, e nenhuma quantidade de trabalho fará você provar **A** a partir de $(A \wedge B) \vee (C \wedge D)$. Se você não perceber isso de início, você poderia gastar uma grande quantidade de tempo tentando construir uma prova impossível! Mas se você percebe este fato, você pode tentar uma outra abordagem mais promissora. (Neste caso, a eliminação da disjunção é claramente o caminho certo para esta prova.)

Trabalhar de trás para frente, apesar de ser uma tática valiosa, não substitui o bom e honesto pensamento!

Fazendo Verificações com os Mecanismos Con

Quando você está construindo uma prova formal em Fitch, você pode evitar ficar tentando provar alguma conclusão intermediária incorreta, ao fazer uma verificação do passo com o mecanismo **(TautCon)**.

No caso ilustrado acima, por exemplo, se você usar **(TautCon)** no passo 2, citando a premissa como suporte, você imediatamente descobriria que não há esperanças para a tentativa de provar **A** a partir da premissa dada.

Muitos dos exercícios deste livro pedem a você que determine se um argumento é válido e justifique sua resposta apresentando ou uma prova de conseqüência ou contraexemplo (uma prova de não-conseqüência). Você deve abordar estes problemas da forma que estamos descrevendo nesta seção. Primeiro tente entender as sentenças envolvidas e decidir se a conclusão se segue das premissas. Se você acha que a conclusão não se segue das premissas, ou não tem nenhum palpite a esse respeito, tente encontrar um contraexemplo. Se você tiver sucesso, mostrou que o argumento é inválido. Se você não conseguir encontrar um contraexemplo, esta própria tentativa freqüentemente lhe dá *insights* sobre porque o argumento é válido, *insights* que podem ajudá-lo a construir a prova requerida.

Podemos resumir nossos conselhos sobre estratégia em um procedimento de sete passos para abordar problemas deste tipo.

LEMBRE-SE

Ao avaliar a validade de um argumento, use o seguinte método:

1. Entenda o que as sentenças estão dizendo.
2. Decida se você acha que a conclusão se segue das premissas ou não.
3. Se você acha que ela não se segue, ou não tem certeza, tente encontrar um contraexemplo.
4. Se você acha que ela se segue, tente fazer uma prova informal.
5. Se o exercício pede uma prova formal, use a prova informal para guiá-lo na sua busca por uma prova formal.
6. Ao fazer provas de conseqüência (tanto formais quanto informais), não se esqueça da tática de trabalhar de trás para frente.
7. Ao trabalhar de trás para frente, sempre verifique se suas metas intermediárias são conseqüência da informação disponível.

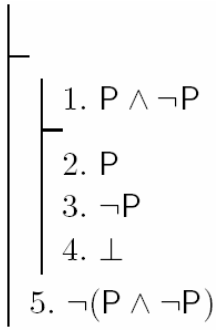
[6] Provas Sem Premissas

Nem todas as provas começam com a suposição de premissas. Isso pode parecer estranho, mas de fato, é com esse tipo de prova que nosso sistema dedutivo demonstra que uma sentença é uma necessidade lógica (verdade lógica).

Demonstrando Necessidades Lógicas

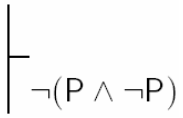
Uma sentença que pode ser provada sem nenhuma premissa é logicamente necessária. Veja um exemplo trivial de uma prova deste tipo que mostra que **a = a ∧ b = b** é uma verdade lógica.

O primeiro passo desta prova não é uma premissa, mas uma aplicação da regra **(=Intro)**. Você poderia pensar que qualquer prova sem premissas deveria começar com esta regra, uma vez que esta é a única regra que não necessita citar passos anteriores de uma prova. Mas de fato, este nem é um exemplo muito representativo de provas sem premissas. Um exemplo mais típico e interessante é o que apresentamos abaixo, que mostra que a sentença **¬(P ∧ ¬P)** é uma necessidade lógica.



Note que não há nenhuma suposição acima da primeira barra de Fitch horizontal, o que indica que a prova principal não tem premissas. O primeiro passo da prova é a suposição de uma subprova. A subprova segue e deriva uma contradição baseada em sua suposição, o que nos permite concluir que a negação da suposição da subprova se segue sem a necessidade de premissas. Em outras palavras, é uma verdade lógica.

Quando queremos que você prove que uma sentença é uma necessidade lógica, usaremos a notação Fitch para indicar que você deve prová-la sem assumir qualquer premissa. Por exemplo, a prova acima mostra que o seguinte "argumento" é válido:



LEMBRE-SE

Uma prova sem premissas mostra que sua conclusão é uma verdade lógica.

Capítulo 7 - Condicionais

Existem muitas construções lógicas importantes em português além dos conectivos booleanos ($\wedge$, $\vee$ e $\neg$):

1. Max está em casa **se** Claire está na biblioteca.
2. Max está em casa **apenas se** Claire está na biblioteca.
3. Max está em casa **se e somente se** Claire está na biblioteca.
4. **Nem** Max está em casa, **nem** Claire está na biblioteca.
5. Max está em casa, **a menos que** Claire esteja na biblioteca.
6. Max está em casa, **ainda que** Claire esteja na biblioteca.
7. Max está em casa **a despeito do fato de** Claire estar na biblioteca.
8. Max está em casa **sempre que** Claire está na biblioteca.
9. Max está em casa **porque** Claire está na biblioteca.

E isso é apenas a ponta do iceberg. Há MUITO mais!! Algumas destas construções são funções de verdade, outras não.

Lembre-se que um conectivo é uma **função de verdade** (ou verofuncional) se a verdade ou falsidade da sentença composta é completamente determinada pelos valores de verdade de suas partes constituintes. Em outras palavras, um conectivo é uma função de verdade se pode ser definido em uma tabela de verdade.

Conectivos Não-Verofuncionais

FOL não inclui conectivos que não sejam funções de verdade. Isto não significa que estes conectivos não sejam importantes, mas seus significados tendem a ser vagos, sujeitos a interpretações conflitantes.

A decisão de não incluir conectivos não-verofuncionais em FOL é análoga à exigência que fizemos no Capítulo 1, de que todos os predicados de FOL devem ter interpretações precisas. Queremos com isso ser tão precisos quanto possível!

Dos conectivos listados acima um, o "**porque**", claramente não é uma função de verdade. Isso não é difícil de provar.

PROVA: Para provar que **porque** não é uma função de verdade, basta apresentarmos duas circunstâncias nas quais uma sentença cujo conectivo principal é **porque** tenha valores de verdade diferentes, mesmo quando os valores de verdade de suas partes constituintes são os mesmos. Ou seja, considere **P**, **Q** e **P porque Q** três sentenças. Se existem duas situações (1 e 2) em que os valores de verdade de **P porque Q** são diferentes (por exemplo, na situação 1 **P porque Q** é verdadeira e na situação 2 **P porque Q** é falsa) e, no entanto, em ambas as situações os valores de verdade de **P** e **Q** são os mesmos (na situação 1, **P** tem o mesmo valor de verdade que na situação 2 e o mesmo se dá com **Q**). Se isso ocorre, é porque os valores de verdade de **P** e **Q** não determinam completamente o valor de verdade de **P porque Q**. Isso significa que **porque** não é uma função de verdade. Em uma tabela de verdade não saberíamos preencher o valor de verdade da sentença **P porque Q** apenas olhando para os valores de verdade de **P** e **Q**. Vamos então apresentar as duas situações.

Considere as sentenças:

P : Max está em casa
Q : Claire está na biblioteca
P porque Q : Max está em casa porque Claire está na biblioteca.

Vejamos agora duas situações em que os valores de **P** e **Q** são constantes, mas o valor de **P porque Q** se altera.

Situação 1: Max soube que Claire está na biblioteca e, portanto, não poderá dar comida a Miau. Então Max corre para casa para alimentar Miau.

Situação 2: Max está em casa e espera que Claire chegue, como de costume, nos próximos minutos. No entanto ela, inesperadamente, precisou ir à biblioteca pegar a referência bibliográfica de um texto.

Em ambas as situações temos: **P** (Max está em casa) e **Q** (Claire está na biblioteca) verdadeiras. No entanto, na situação 1, **P porque Q** (Max está em casa porque Claire está na biblioteca) é verdadeira, mas na situação 2 **P porque Q** é falsa.

Portanto, **porque** não é uma função de verdade. Quando **P** e **Q** são ambos verdadeiros, não temos como saber se **P porque Q** será verdadeiro ou falso.

A razão pela qual **porque** não é função de verdade é que este conectivo tipicamente afirma uma ***conexão causal*** entre os fatos descritos pelas sentenças constituintes.

Na prova acima, os valores de verdade de **P** e **Q** não se alteraram entre as situações. O que se alterou foi que na primeira situação uma conexão causal estava presente entre **P** e **Q**, e na segunda não estava.

Os conectivos lógicos e a relação de conseqüência lógica, ao contrário do que muita gente pensa, **não estabelecem conexões causais entre as sentenças**.

A lógica é uma disciplina **analítica**, ela não pode nos dizer nada sobre as conexões causais dos fatos do mundo!!

Neste capítulo, vamos introduzir dois conectivos verofuncionais novos, a *implicação material* (ou condicional material) e a *biimplicação material* (ou bicondicional material). Como veremos no final do capítulo, estes novos símbolos de fato não aumentam o poder expressivo de FOL. Eles, no entanto, tornam muito mais fácil dizer e provar certas coisas e, portanto, são adições válidas à nossa linguagem.

Implicação Material (ou Condicional): $\rightarrow$

O símbolo $\rightarrow$ é utilizado para combinar duas sentenças **P** e **Q** para formar uma nova sentença $P \rightarrow Q$, chamada de *condicional* (ou *implicação material*). **P** é chamado antecedente da implicação (ou do condicional) e **Q** é chamado conseqüente da implicação (ou do condicional).

Antes de discutirmos as contrapartidas do português para este conectivo, vamos explicar seu significado.

A Semântica do Condicional

A sentença $P \rightarrow Q$ é verdadeira se e somente se **P** é falsa ou **Q** é verdadeira (ou ambas as coisas).

A seguinte tabela de verdade sumariza esta definição:

Tabela de Verdade para o Condicional

P	Q	$P \rightarrow Q$
T	T	T
T	F	F
F	T	T
F	F	T

Regra para o Condicional no Jogo Henkin-Hintikka

A definição e a tabela acima mostram que, na verdade, $P \rightarrow Q$ é apenas uma outra forma de dizer: $\neg P \vee Q$.

É por isso que, no jogo Henkin-Hintikka, o **Mundo de Tarski** sempre substitui sentenças da forma $P \rightarrow Q$ para esta forma equivalente: $\neg P \vee Q$.

LEMBRE-SE

1. Se **P** e **Q** são sentenças de FOL, então $P \rightarrow Q$ também é.
2. A sentença $P \rightarrow Q$ é falsa em apenas um caso: se o antecedente **P** é verdadeiro e o conseqüente **Q** é falso. Caso contrário, ela é verdadeira.

As Formas da Implicação Material no Português

A sentença do português "*Se P então Q*" aproxima-se bastante deste significado de implicação material. Pois não há dúvidas de que este condicional em português é falso quando **P** é verdadeiro e **Q** é falso. Portanto, traduziremos, por exemplo a sentença: "*Se Max está em casa então Claire está na biblioteca*" por:

$$\text{Home(max)} \rightarrow \text{Library(claire)}$$

Nesta disciplina sempre traduziremos *"se...então..."* usando o símbolo $\rightarrow$, mas há, de fato, muitos usos desta expressão no português que não são bem traduzidos pela implicação material. Considere, por exemplo, a sentença:

Se Max estivesse em casa, então Miau também deveria estar.

Esta sentença pode ser falsa mesmo se Max não está em casa. Basta supor que quem a proferiu pensou, erroneamente, que Miau estivesse com Max, quando, na verdade, Miau está com Claire no veterinário. Mesmo com Max fora de casa, sob esta circunstância a sentença é falsa, pois Miau não estava com Max, então não é verdade que se Max estivesse em casa, Miau também deveria estar.

Mas a sentença na linguagem de primeira-ordem (FOL)

$$\text{Home(max)} \rightarrow \text{Home(miau)}$$

é automaticamente verdadeira se Max não está em casa! Basta ver a tabela de verdade.

Já vimos que o conectivo **porque** não é uma função de verdade (verofuncional), uma vez que ele expressa uma conexão causal entre o antecedente e o conseqüente. A construção em português *se...então...* também pode expressar um tipo de conexão causal entre antecedente e conseqüente. É isso que parece estar ocorrendo no exemplo acima. Como conseqüência, muitos usos de *se...então...* em português simplesmente não são verofuncionais. A verdade da sentença completa, nestes casos, depende de algo mais do que apenas a verdade de suas partes. Ela depende de haver ali alguma conexão genuína entre o assunto do antecedente e o do conseqüente.

No entanto, ainda que a implicação material seja inadequada para capturar algumas sutilezas dos condicionais em português (não é uma tradução perfeita), ela é o que de melhor conseguimos fazer com um conectivo que seja função de verdade. Saibam, porém, que este assunto é bastante controverso entre os lógicos.

"Se" e "Apenas Se"

Outras expressões do português que podemos traduzir utilizando a implicação material $P \rightarrow Q$ incluem: "*P apenas se Q*", "*Q dado que P*" e "*Q se P*".

Para entender por que "*P apenas se Q*" se traduz por $P \rightarrow Q$ e "*P se Q*" se traduz por $Q \rightarrow P$ precisamos pensar cuidadosamente sobre a diferença entre "*apenas se*" e "*se*".

Condição Necessária

Em português, a expressão "*apenas se*" introduz o que é chamado de uma ***condição necessária***, uma condição que precisa ser satisfeita para obtermos alguma outra coisa. Por exemplo, suponha que seus amáveis professores de lógica anunciem, no início da disciplina, que você será aprovado *apenas se* fizer todas as listas de exercícios. O que seus professores estão dizendo é que fazer as listas de exercícios é uma *condição necessária* para a aprovação: se você não fizer, não é aprovado. Mas os professores não estão garantindo que você será aprovado caso faça as listas de exercícios: claramente há outras formas de ser reprovado, tais como faltar às provas ou entregar todos os exercícios das listas errados.

A afirmação de que você será aprovado *apenas se* entregar todas as listas de exercícios, exclui apenas uma possibilidade: a de você ser aprovado sem entregar todas as listas de exercícios. Em outras palavras, *P apenas se Q* é falsa só quando *P* é verdadeira e *Q* é falsa, e este é exatamente o caso em que $P \rightarrow Q$ é falsa.

Condição Suficiente

Compare isto com a afirmação de que você será aprovado na disciplina *se* entregar todas as listas de exercícios. É uma situação completamente diferente. Um professor que faça esta promessa (e seus amáveis professores de lógica não a fizeram) está estabelecendo uma política de avaliação bastante frouxa: basta entregar as listas de exercícios que você terá uma nota que aprova (maior ou igual a 7), independentemente de quão bem você tenha feito os exercícios ou mesmo de se você se preocupou ou não em fazer as provas!

Em português, a expressão *se* introduz o que é chamado de uma ***condição suficiente***, uma condição que garante que alguma coisa será obtida (neste caso, a aprovação na disciplina). Por causa disso, uma sentença do português com a forma *P (será provado) **se** Q (fizer os exercícios)* deve ser traduzida como $Q \rightarrow P$. Esta sentença será falsa apenas quando Q é verdadeira (entregar as listas de exercícios) e P é falsa (ser aprovado na disciplina).

Outros Usos de $\rightarrow$ ("...*a menos que*...")

Em FOL, também usamos $\rightarrow$ combinado com $\neg$ para traduzir sentenças com a forma "*a menos que P, Q*" ou "*Q a menos que P*". Considere, por exemplo, a sentença:

"*Claire está na biblioteca a menos que Max esteja em casa*".

Compare-a com a sentença

"*Claire está na biblioteca se Max não está em casa*".

Repare que estas duas sentenças serão falsas exatamente nas mesmas circunstâncias, nomeadamente, quando Claire não está na biblioteca e, no entanto, Max está em casa. De uma maneira mais geral, "***Q** a menos que **P***" é verdadeira nas mesmas circunstâncias em que "***Q** se **não-P***", e portanto é traduzida por:

$$\neg P \rightarrow Q$$

Uma boa maneira de se lembrar disso é sussurrar "*se não*" sempre que você ver "*a menos que*".

No entanto, o uso mais importante de $\rightarrow$ em lógica de primeira ordem não é em conexão com as expressões acima, mas em conexão com sentenças *quantificadas universalmente*, ou seja, sentenças da forma *Todos os **A**s são **B**s* ou *Cada **A** é um **B***. As sentenças de primeira ordem (de FOL) análogas a esta terão a forma:

$$\text{Para todo objeto } x\ (A(x) \rightarrow B(x))$$

Esta sentença afirma que qualquer objeto que você escolher ou não será um **A** ou será um **B**. Mas isto é um assunto para mais tarde, que aprenderemos na Parte II deste livro.

Reduzindo o Conceito de Conseqüência Lógica ao de Verdade Lógica

Há um outro fator que nos ajuda a entender a importância da implicação material para a lógica. Ela nos permitirá reduzir a noção de *conseqüência lógica* à de *verdade lógica*, pelo menos nos casos em que o número de premissas do argumento é finito.

Sabemos que uma sentença **Q** é conseqüência lógica de um conjunto de premissas $P_1, \ldots, P_n$ se e somente se é impossível que as premissas sejam todas verdadeiras e a conclusão falsa. Outra forma de expressar isso é dizer que é impossível que a sentença **Q** seja falsa quando $(P_1 \wedge \ldots \wedge P_n)$ for verdadeira.

Dado o significado do conectivo $\rightarrow$ (sua tabela de verdade), podemos ver, pela definição acima, que **Q** é uma conseqüência de $P_1, \ldots, P_n$ se e somente se é impossível que a seguinte sentença seja falsa.

$$(P_1 \wedge \ldots \wedge P_n) \rightarrow Q$$

Assim, uma forma de verificar a validade tautológica de um argumento com um conjunto finito de premissas é construir uma tabela verdade para a sentença acima e verificar se a coluna final contém apenas **TRUE**.

LEMBRE-SE

1. As seguintes construções em português são todas traduzidas para $P \rightarrow Q$:
   - *Se **P** então **Q**.*
   - *Q se P.*
   - *P apenas se Q.*
   - *Q dado que P* - ou - *Dado que P, Q.*
2. A seguinte construção é traduzida para $\neg P \rightarrow Q$:
   - *Q a menos que P* - *ou* - *A menos que P, Q.*
3. **Q** é uma conseqüência lógica de $P_1, \ldots, P_n$ se e somente se a sentença $(P_1 \wedge \ldots \wedge P_n) \rightarrow Q$ for uma verdade lógica.

Biimplicação Material (ou Bicondicional): $\leftrightarrow$

Dadas duas sentenças **P** e **Q**, podemos ainda formar outra sentença com elas, através do uso do operador **bicondicional** (ou *biimplicação material*): $P \leftrightarrow Q$. Uma sentença da forma $P \leftrightarrow Q$ é verdadeira se e somente se as sentenças **P** e **Q** têm o mesmo valor de verdade, ou seja, ou ambas são verdadeiras ou ambas são falsas.

"*Se e Somente Se*"

Em português o bicondicional é comumente traduzido pela expressão: "*se e somente se*". Por exemplo, podemos traduzir a sentença "*Max está em casa se e somente se Claire está na biblioteca*" por:

Home(max) ↔ Library(claire)

A Semântica do Bicondicional

A interpretação semântica do bicondicional é dada pela seguinte tabela de verdade.

Tabela de Verdade para o Bicondicional

P	Q	P ↔ Q
T	T	T
T	F	F
F	T	F
F	F	T

Note que a coluna final da tabela de verdade é exatamente a mesma do que a da sentenças $(P \rightarrow Q) \wedge (Q \rightarrow P)$. Faça como exercício (exercício 7.3) esta tabela de verdade e confira.

Por esta razão, os lógicos costumam tratar a sentença $P \leftrightarrow Q$ como uma abreviação para $(P \rightarrow Q) \wedge (Q \rightarrow P)$.

Regra para o Bicondicional no Jogo Henkin-Hintikka

O programa **Mundo de Tarski** também utiliza esta abreviação no jogo Henkin-Hintikka. Assim, a regra do jogo para $P \leftrightarrow Q$ é simples. Sempre que uma sentença desta forma for encontrada, ela será substituída por $(P \rightarrow Q) \wedge (Q \rightarrow P)$.

LEMBRE-SE

1. Se P e Q são sentenças de FOL, então $P \leftrightarrow Q$ também é.
2. A sentença $P \leftrightarrow Q$ é verdadeira se e somente se P e Q têm o mesmo valor de verdade.

Insinuações Sociais

Há muitos casos problemáticos em traduções do português para FOL. Por exemplo, muitos estudantes resistem em traduzir uma sentença como

Max está em casa a menos que Claire esteja na biblioteca

por $\neg$Library(claire) $\rightarrow$ Home(max). Eles usualmente pensam que o significado desta sentença em português seria mais acuradamente capturado por uma afirmação bicondicional como:

$\neg$Library(claire) $\leftrightarrow$ Home(max)

A razão para que esta última forma pareça mais natural é que quando afirmamos a sentença em português, há uma certa sugestão de que se Claire *está* na biblioteca, então Max *não está* em casa.

Para resolver casos problemáticos como este, é freqüentemente útil distinguir entre as *condições de verdade* de uma sentença, e outras coisas que, em algum sentido, se seguem **da asserção** da sentença (mas não da própria sentença). Vejamos um caso óbvio. Suponha que alguém afirme a sentença

Está um dia adorável

Uma coisa que você pode concluir disso é que a pessoa que disse a sentença entende português. Isto, porém, não é uma parte do que foi dito, mas uma parte do que pode ser inferido do fato da sentença ter sido dita. A verdade ou falsidade da sentença não tem nada a ver com as habilidades lingüísticas de quem a proferiu.

O filósofo H. P. Grice desenvolveu uma teoria do que ele chamou de ***insinuações sociais*** (*conversational implicatures*) para ajudar a separar as condições de verdade genuínas de uma sentença de outras conclusões que podem ser tiradas do ato de sua asserção (do fato dela ter sido dita). Não veremos esta teoria em detalhes, mas conhecer um pouco

sobre ela pode ajudar bastante em nossas traduções do português para FOL, portanto, apresentaremos uma pequena introdução à teoria de Grice.

Teste do Cancelamento de Insinuações

Suponha que alguém tenha proferido, em português, uma sentença **S**, e que estejamos tentando decidir se uma conclusão particular que tiramos dela é uma parte do significado de **S** ou, ao invés, é apenas uma de suas *insinuações sociais*. Grice apontou que se a conclusão é parte do significado, então ela não pode ser "cancelada" por alguma elaboração adicional do falante.

Então, por exemplo, concluímos que 'Max está em casa' é parte do significado de uma asserção de '*Max e Claire estão em casa*', porque não conseguimos cancelar esta conclusão. Dizer que *Max e Claire estão em casa, mas Max não está em casa*, apenas nos levaria a uma contradição.

Compare isto com alguém que diga '*Está um dia adorável*'. Suponha que o falante continue e diga, talvez lendo hesitantemente de um livro de frases: *'Você fala um pouco de Francês'*? Neste caso, a sugestão (insinuação) de que o falante entende português foi efetivamente cancelada.

Um uso mais esclarecedor do teste da '*cancelabilidade'* de Grice diz respeito à expressão *'ou...ou...*'. Lembre-se de que afirmamos que esta expressão deveria ser traduzida em FOL por uma disjunção inclusiva, utilizando $\vee$. Podemos agora ver que a sugestão de que esta frase expressaria uma disjunção exclusiva é geralmente apenas uma *insinuação social*. Por exemplo, quando um garçom lhe sugere: '*Você pode ou tomar uma sopa, ou comer uma salada*', existe uma forte sensação de que você não pode escolher as duas. Mas isso é claramente apenas uma insinuação social, uma vez que ele poderia, sem contradizer-se, continuar e dizer: '*E você pode escolher as duas, se quiser*'. Se *'ou...ou..*' expressasse uma disjunção exclusiva, isto seria como se o garçom tivesse dito: *'Você pode tomar uma sopa, ou comer uma salada, mas não ambas, e você pode escolher as duas, se quiser*', o que é claramente contraditório.

Voltemos agora para a sentença '*Max está em casa a menos que Claire esteja na biblioteca*'. Nós acima negamos que sua tradução correta fosse

$$\neg \text{Library(claire)} \leftrightarrow \text{Home(max)}$$

que é equivalente à conjunção da tradução correta

$$\neg \text{Library(claire)} \rightarrow \text{Home(max)}$$

com a seguinte afirmação adicional, que é a forma contrapositiva de $\text{Home(max)} \rightarrow \neg\text{Library(claire)}$)

$$\text{Library(claire)} \rightarrow \neg\text{Home(max)}$$

Será que esta última sentença é uma parte do significado da sentença em português, ou é apenas uma insinuação social? O teste da cancelabilidade de Grice mostra que ela é apenas uma insinuação. Afinal de contas, é perfeitamente aceitável que o falante continue e diga *'Por outro lado, se Claire está na biblioteca, eu não faço idéia de onde Max esteja*'. Esta elaboração retira a sugestão de que se Claire está na biblioteca, então Max não está em casa.

Outra insinuação comum surge com a frase '*apenas se*', que as pessoas freqüentemente usam, com o mesmo significado que a frase logicamente mais forte *'se e somente se*'. Por exemplo, suponha que um pai diga a seu filho, '*Você pode comer a sobremesa apenas se comer todo o feijão*'. Já vimos que isto não é garantia de que se a criança comer o feijão ela terá a sobremesa, uma vez que *'apenas se*' introduz uma condição necessária mas não suficiente. No entanto está claro que a asserção do pai sugere que a criança poderá comer a sobremesa se ela comer todo o terrível feijão. Mas, novamente, esta sugestão pode ser cancelada sem produzir nenhuma contradição. Suponha que o pai continue e diga: '*Se você comer todo o feijão eu vou verificar se ainda tem sorvete no freezer*' Esta sentença cancela a implicação de que a sobremesa está garantida.

LEMBRE-SE

Se a asserção de uma sentença carrega consigo uma sugestão que pode ser cancelada (sem contradição) por alguma elaboração adicional do falante, então a sugestão é apenas uma *insinuação social*, mas não parte do conteúdo da sentença original.

Completude Verofuncional

Temos agora, à nossa disposição, cinco conectivos verofuncionais (que são funções de verdade), um unário ($\neg$), e quatro binários ($\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$). Será que deveríamos introduzir mais algum? Embora tenhamos visto algumas poucas expressões do português que não podem ser expressas em FOL (como '*porque*'), isso ocorre porque elas não são funções de verdade. Há também, algumas expressões, tais como '*nem...nem...*' que são funções de verdade, mas que,

mesmo não possuindo conectivos exclusivos para elas em FOL, podem ser traduzidas facilmente utilizando os conectivos que conhecemos.

A questão que nos colocamos nesta seção é se há algum conectivo verofuncional que precisamos adicionar à nossa linguagem. Será possível que encontremos alguma construção em português que seja verofuncional mas que, com os símbolos que temos, não consigamos expressá-la em FOL? Se isso vier a ocorrer, estaremos diante de uma infeliz limitação de nossa linguagem de primeira ordem!

Como podemos responder a esta questão? Bem, vamos começar pensando a respeito dos conectivos binários, aqueles que se aplicam a duas sentenças para construir uma terceira. Qual o número total de conectivos verofuncionais binários possíveis? Se pensarmos sobre as tabelas de verdade possíveis para tais conectivos, podemos descobrir este número total. Primeiro, uma vez que estamos lidando com conectivos binários, haverá quatro linhas na tabela de cada um deles. Cada linha pode receber o valor **TRUE** ou **FALSE**, então há $2^4 = 16$ formas possíveis de fazer isso. Por exemplo, aqui está uma tabela que captura a função de verdade expressa pela expressão '*nem...nem...*'.

P	Q	Nem P nem Q
T	T	F
T	F	F
F	T	F
F	F	T

Uma vez que há apenas 16 maneiras diferentes de preencher a coluna final de uma tabela com duas colunas de referência, há apenas 16 funções de verdade binárias, e, portanto, o número de conectivos verofuncionais binários possíveis também é 16. Poderíamos olhar para cada uma destas tabelas e mostrar como expressar sua função de verdade com os conectivos que temos a nossa disposição. Por exemplo, expressão $\neg(P \vee Q)$ resulta em uma tabela idêntica à tabela acima, o que mostra que $\neg(P \vee Q)$ e *'nem P nem Q'* correspondem à mesma função de verdade. Portanto, o significado da expressão *'nem P nem Q'* é expresso em FOL por $\neg(P \vee Q)$.

Mas ao invés de termos todo este trabalho com estas 16 tabelas, há uma forma mais geral e sistemática de mostrarmos isso. Suponha que estejamos pensando em introduzir um conectivo verofuncional binário novo, digamos $\clubsuit$. Ele terá uma tabela de verdade como a que se segue, com um dos dois valores de verdade em cada linha.

P	Q	P ♣ Q
T	T	1º valor
T	F	2º valor
F	T	3º valor
F	F	4º valor

Se todos os quatro valores forem falsos, então podemos claramente expressar $P \clubsuit Q$ através da sentença $P \wedge \neg P \wedge Q \wedge \neg Q$.[1] Suponha agora que pelo menos um dos valores seja **TRUE**. Como deveríamos expressar $P \clubsuit Q$? Uma forma poderia ser esta. Considere $C_1, \ldots, C_4$ as seguintes quatro conjunções:

$$C_1 = (P \wedge Q)$$
$$C_2 = (P \wedge \neg Q)$$
$$C_3 = (\neg P \wedge Q)$$
$$C_4 = (\neg P \wedge \neg Q)$$

Note que a sentença C_1 será verdadeira se os valores de verdade de P e Q são como os valores especificados na primeira linha da tabela de verdade de $P \clubsuit Q$ (ambos verdadeiros) e falsa nos outros casos. Similarmente, C_2 será verdadeira apenas quando P e Q têm os valores de verdade da segunda linha da tabela para $P \clubsuit Q$. De forma análoga, C_3 e C_4 correspondem à terceira e quarta linhas da tabela.

Para construir uma sentença que tenha uma tabela de verdade exatamente igual à de $P \clubsuit Q$, tudo o que temos que fazer é tomar uma disjunção dos Cs apropriados. Por exemplo, se $P \clubsuit Q$ é verdadeira apenas nas linhas 2 e 4, então $C_2 \vee C_4$ é equivalente a esta sentença.

Isto mostra que todas as funções de verdade binárias são expressáveis utilizando apenas os conectivos $\neg$, $\wedge$ e $\vee$.

[1] Apenas $P \wedge \neg P$ já seria suficiente, mas queremos uma sentença que use tanto P quanto Q. Também seria possível expressar esta função de verdade pela sentença $P \wedge \neg P \wedge Q$. Se tiver dúvidas, faça as tabelas de verdade e confira!

É fácil ver que um procedimento similar nos permite expressar todas as funções de verdade unárias possíveis. Um conectivo unário, digamos $\otimes$, terá uma tabela de verdade como a seguinte:

P	$\otimes$P
T	1º valor
F	2º valor

Se ambos os valores abaixo de $\otimes$P são **FALSE**, então podemos expressar isto usando a sentença $P \wedge \neg P$. Por outro lado, podemos expressar $\otimes$P como uma disjunção de uma ou mais das seguintes sentenças:

$$C_1 = P$$
$$C_2 = \neg P$$

C_1 será incluído como um dos disjuntos se o valor da primeira linha de $\otimes$P for **TRUE**, e C_2 será incluído se o valor da segunda linha for **TRUE**.

Uma vez que tenhamos entendido como este procedimento funciona, veremos que ele pode ser aplicado igualmente para conectivos verofuncionais de qualquer aridade. Suponha, por exemplo, que queremos expressar o conectivo verofuncional ternário definido pela seguinte tabela de verdade:

P	Q	R	♥(P, Q, R)
T	T	T	**T**
T	T	F	**T**
T	F	T	**F**
T	F	F	**F**
F	T	T	**T**
F	T	F	**F**
F	F	T	**T**
F	F	F	**F**

Uma tradução bastante boa de ♥(P, Q, R) é '*se P então Q senão R*'. Quando nós aplicamos o método acima para expressar este conectivo, obtemos a seguinte sentença:

$$(P \wedge Q \wedge R) \vee (P \wedge Q \wedge \neg R) \vee (\neg P \wedge Q \wedge R) \vee (\neg P \wedge \neg Q \wedge R)$$

Ou seja, a tabela de verdade da disjunção acima é exatamente a mesma de ♥(P, Q, R) e, por isso, expressam a mesma função de verdade. Então, a disjunção acima é uma tradução adequada em FOL da expressão '*se P então Q senão R*'.

De maneira mais geral, se ♠ expressa um conectivo *n*-ário, então podemos usar este procedimento para obter uma sentença que seja tautologicamente equivalente a ♠(P_1, ..., P_n). Primeiro, nós definimos as conjunções C_1, ..., C_{2^n} que correspondem às 2^n linhas da tabela de verdade que definem o conectivo. Então, formamos uma disjunção D que conterá C_k como disjunto se e somente se a *k-ésima* linha da tabela de verdade tiver o valor **TRUE**. (Se todas as linhas contém **FALSE**, então fazemos D ser $P_1 \wedge \neg P_1$.) Por razões que já explicamos esta disjunção será tautologicamente equivalente a ♠(P_1, ..., P_n).

O que fizemos foi o esboço de uma prova de que qualquer função de verdade, de qualquer aridade, pode ser expressa usando apenas os conectivos booleanos ($\neg$, $\wedge$ e $\vee$). Este é um fato suficientemente interessante para merecer ser destacado. Nós diremos que um conjunto de conectivos é '*verofunciomalmente' completo*' se os conectivos do conjunto são suficientes para nos permitir expressar qualquer função de verdade. Podemos destacar este fato importante como um *teorema*. Um teorema não é nada mais do que uma conclusão que o autor acha suficientemente interessante para ser destacada.

Teorema da Completude Verofuncional Booleana

TEOREMA: os conectivos booleanos $\neg$, $\wedge$ e $\vee$ são verofuncionalmente completos.

Há outras coleções de operadores que são verofuncionalmente completos. Na verdade, podemos nos livrar de $\wedge$ ou de $\vee$ sem perdermos a completude verofuncional. Por exemplo, $P \vee Q$ pode ser expressa da seguinte forma usando apenas $\neg$ e $\wedge$:

$$\neg(\neg P \wedge \neg Q)$$

Isto significa que podemos nos livrar de todas as ocorrências de $\vee$ em nossas sentenças em favor de $\neg$ e $\wedge$. Alternativamente, podemos nos livrar de $\wedge$ em favor de $\neg$ e $\vee$, conforme você verá no Exercício 7.25 da Lista de Exercícios 7-8. É claro que em qualquer um destes casos as sentenças resultantes serão muito maiores e mais difíceis de entender.

Um Único Conectivo Verofuncionalmente Completo

Nós, de fato, poderíamos ser mais econômicos ainda em nossas escolhas de conectivos. Suponha que usássemos a forma $\mathbf{P \downarrow Q}$ para expressar '*nem **P** nem **Q***'. Este conectivo, assim definido, é, sozinho, verofuncionalmente completo. Para ver isso note que $\mathbf{\neg P}$ pode ser expressa como:

$$\mathbf{P \downarrow P}$$

que representa a afirmação: '*nem **P** nem **P***'. Analogamente, $\mathbf{P \wedge Q}$ pode ser expressa como:

$$\mathbf{(P \downarrow P) \downarrow (Q \downarrow Q)}$$

que representa a afirmação: '*nem não-**P** nem não-**Q***'. Assim, em teoria, poderíamos usar apenas um conectivo verfofuncional e com ele expressar qualquer coisa que podemos expressar usando os nossos cinco conectivos atuais ($\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$).[2]

As Desvantagens da Economia

Há duas desvantagens em economizar nos conectivos. Primeiro, conforme já dissemos, quanto menos conectivos utilizarmos, mais difícil será de entender nossas sentenças. Mas ainda pior que isso é que nossas provas se tornarão muito mais complicadas.

Por exemplo, sempre expressarmos $\wedge$ em termos de $\neg$ e $\vee$, uma simples aplicação da regra **($\wedge$Intro)** deveria ser substituída por dois usos de **($\perp$Intro)**, um uso de **($\vee$Intro)** e um uso de **($\neg$Intro)** (veja o exercício 7.26). É por isso que não estamos economizando nos conectivos.

LEMBRE-SE

2. Um conjunto de conectivos é *verofuncionalmente completo* se tais conectivos nos permitirem expressar cada uma das funções de verdade.
3. Vários conjuntos de conectivos, incluindo os conectivos booleanos são verofuncionalmente completos.

Há, em outros textos de lógica, várias notações alternativas para os símbolos conectivos verofuncionais. O quadro a seguir resume as mais freqüentes destas notações.

Nossa Notação	Notações Alternativas Equivalentes
$\neg P$	~P, $\overline{P}$, !P
$P \wedge Q$	P & Q, P &&Q, P . Q, PQ
$P \vee Q$	P | Q, P || Q
$P \rightarrow Q$	$P \supset Q$
$P \leftrightarrow Q$	$P \equiv Q$

[2] Se não estiver convencido disso, faça as tabelas de verdade para $\mathbf{P \downarrow P}$ e $\mathbf{(P \downarrow P) \downarrow (Q \downarrow Q)}$ tomando por base a tabela de verdade para a expressão '*nem...nem...*' que apresentamos na página 7 acima.

Capítulo 8 - A Lógica dos Condicionais

Uma conseqüência do *Teorema da Completude Verofuncional Booleana* (página 8 do Capítulo 7) é que ao introduzirmos os símbolos do condicional e do bicondicional, não aumentamos o poder expressivo de FOL. Uma vez que $\rightarrow$ e $\leftrightarrow$ podem ser definidos através dos conectivos booleanos, sempre poderemos eliminá-los de todas as asserções, substituindo-os por suas definições.

Assim, por exemplo, se queremos provar $P \rightarrow Q$ podemos apenas provar $\neg P \vee Q$, usar a definição e obtermos $P \rightarrow Q$. Na prática, no entanto, esta é uma péssima idéia. É muito mais natural utilizar as regras que envolvem estes símbolos diretamente. Além disso, as provas resultantes são mais simples e mais fáceis de entender.

A implicação material, em particular, é um símbolo extremamente útil e bastante utilizado em vários tipos de inferências nas mais diversas áreas, desde à matemática, até o direito e a filosofia. É por isso que precisamos aprender mais a respeito sobre como provar sentenças que tenham a forma $P \rightarrow Q$.

Como fizemos anteriormente com os conectivos booleanos, vamos primeiro olhar para provas informais que envolvem os condicionais e depois incorporaremos estes métodos de prova em nosso sistema **F**.

Passos Válidos de Inferência

Entre os métodos informais, separaremos os que são meros passos válidos de inferência dos que se configuram em importantes métodos de prova.

Modus Ponens ou Eliminação do Condicional

O passo de inferência mais comum envolvendo $\rightarrow$ leva o nome latino de ***modus ponens***, e também é chamado de *eliminação do condicional*.

A regra diz que se você estabeleceu as sentenças $P \rightarrow Q$ e P, então você pode inferir Q. Esta regra é obviamente válida. Uma rápida olhada para a tabela de verdade de $\rightarrow$ nos mostra isso, pois em todas as linhas que $P \rightarrow Q$ e P são ambas verdadeiras, Q também é verdadeira.

Eliminação do Bicondicional

Há um passo de inferência, similar ao anterior, para o bicondicional, uma vez que o bicondicional é logicamente equivalente a uma conjunção de dois condicionais.

Se você tiver estabelecido uma das sentenças $Q \leftrightarrow P$ ou $P \leftrightarrow Q$ e conseguir estabelecer P, então você pode inferir Q. Este passo de inferência é chamado de *eliminação do bicondicional*.

Contraposição

Além destas regras simples, há várias equivalências úteis envolvendo nossos símbolos novos. Uma das mais importantes é conhecida como a *Lei da Contraposição*. Ela estabelece que $P \rightarrow Q$ é logicamente equivalente a $\neg Q \rightarrow \neg P$. Esta segunda sentença condicional é chamada de *contraposição* da sentença condicional original.

Uma tabela de verdade conjunta mostra claramente por que estas formas são logicamente equivalentes.

Esta é uma equivalência lógica particularmente útil, porque muitas vezes é mais fácil provar a contraposição de um condicional do que a forma original.

Mais Equivalências Lógicas Envolvendo Condicionais

Aqui estão algumas equivalências lógicas úteis de ter em mente. Examine-as cuidadosamente e, se tiver dúvida em algumas delas, faça suas tabelas de verdade com o programa **Boole** para entendê-las.

$P \rightarrow Q$	$\Leftrightarrow$	$\neg Q \rightarrow \neg P$
$P \rightarrow Q$	$\Leftrightarrow$	$\neg P \vee Q$
$\neg(P \rightarrow Q)$	$\Leftrightarrow$	$P \wedge \neg Q$
$P \leftrightarrow Q$	$\Leftrightarrow$	$(P \rightarrow Q) \wedge (Q \rightarrow P)$
$P \leftrightarrow Q$	$\Leftrightarrow$	$(P \wedge Q) \vee (\neg P \wedge \neg Q)$

LEMBRE-SE

Sejam **P** e **Q** sentenças quaisquer de FOL.

1. Modus Ponens: De $P \rightarrow Q$ e **P**, infere-se **Q**.
2. Eliminação do Bicondicional: De **P** e ou $P \leftrightarrow Q$ ou $Q \leftrightarrow P$ infere-se **Q**.
3. Contraposição: $P \rightarrow Q \Leftrightarrow \neg Q \rightarrow \neg P$.

O Método da Prova Condicional

Um dos mais importantes métodos de prova é o método da prova condicional, que permite provar sentenças condicionais (com a forma $P \rightarrow Q$).

Suponha que você queira provar uma declaração condicional $P \rightarrow Q$. O que você faz é assumir temporariamente o antecedente da condicional (**P**). Então, se com esta suposição adicional você conseguir provar **Q**, o método da prova condicional permite a você inferir $P \rightarrow Q$ a partir das premissas originais.

Vejamos um exemplo simples que usa tanto o *modus ponens* quanto a prova condicional. Mostraremos que $Tet(a) \rightarrow Tet(c)$ é uma conseqüência lógica das duas premissas seguintes: $Tet(a) \rightarrow Tet(b)$ e $Tet(b) \rightarrow Tet(c)$. Em outras palavras, vamos mostrar que o operador $\rightarrow$ é transitivo.

PROVA: nossas premissas são $Tet(a) \rightarrow Tet(b)$ e $Tet(b) \rightarrow Tet(c)$, e queremos provar $Tet(a) \rightarrow Tet(c)$. Mantendo em mente que vamos usar o método da prova condicional, vamos assumir temporariamente, em adição às nossas premissas, que $Tet(a)$ é verdadeiro. Então, aplicando *modus ponens* em $Tet(a)$ e nossa primeira premissa ($Tet(a) \rightarrow Tet(b)$) concluímos $Tet(b)$. Utilizando novamente *modus ponens* em $Tet(b)$ (que acabamos de concluir) e nossa segunda premissa ($Tet(b) \rightarrow Tet(c)$), obtemos $Tet(c)$. Assim, admitindo condicionalmente $Tet(a)$ como uma suposição adicional, concluímos $Tet(c)$. Portanto, o método da prova condicional nos assegura que $Tet(a) \rightarrow Tet(c)$ é conseqüência de nossas premissas originais.

A prova condicional é claramente uma forma válida de inferência, uma que usaremos o tempo todo. Se **Q** se segue logicamente de algumas premissas acrescidas da suposição adicional **P**, então podemos ter certeza que se as premissas forem verdadeiras, a sentença condicional $P \rightarrow Q$ deve também ser verdadeira. Afinal de contas uma implicação só pode ser falsa se **P** for verdadeira e **Q** for falsa, mas a prova condicional mostra justamente que isto jamais pode acontecer, pois se obtivemos uma prova de **Q** a partir de algumas premissas e de **P**, então esta prova mostra justamente que sempre que estas premissas e **P** forem verdadeiras, **Q** também será.

Vejamos um exemplo mais interessante, que utiliza tanto a prova condicional quanto a prova por contradição. Provaremos que:

$$Par(n^2) \rightarrow Par(n)$$

PROVA: O método da prova condicional nos permite fazer esta prova assumindo a suposição de que $Par(n^2)$ e provando, a partir disso, que $Par(n)$. Então, assuma que n^2 é par. Para provar que n é par usaremos o método da prova por contradição. Assumiremos, então, que n não é par, ou seja, que é impar, e mostraremos que isso leva a uma contradição.

Se n é impar, podemos expressá-lo como $2m + 1$ para algum m. [1] Temos então:

$$n = 2m + 1$$
$$n^2 = (2m + 1)^2$$
$$= 4m^2 + 4m + 1$$
$$= 2(2m^2 + 2m) + 1$$

Mas isso mostra que n^2 é impar, contradizendo nossa primeira suposição. Esta contradição mostra que n não é impar, ou seja, que é par. Portanto, pelo método da prova condicional provamos que $Par(n^2) \rightarrow Par(n)$.

[1] Todo número que pode ser expresso pela forma $2m + 1$ não é divisível por 2. Portanto, todo número ímpar pode ser expresso desta forma, para algum m.

Você se perdeu tentando entender esta prova? Ela, de fato, tem uma estrutura bastante complicada. Nós, primeiramente assumimos $\mathbf{Par}(n^2)$ tendo em vista o método da prova condicional. Depois, assumimos $\neg\mathbf{Par}(n)$ tendo em vista provar $\mathbf{Par}(n)$ pelo método da prova indireta (por absurdo). Em seguida obtivemos $\neg\mathbf{Par}(n^2)$, que contradiz nossa primeira suposição.

Provas deste tipo são muito comuns e esta é a razão pela qual freqüentemente é mais fácil provar a forma contrapositiva de um condicional. A contrapositiva de nossa declaração inicial é:

$$\neg\mathbf{Par}(n) \rightarrow \neg\mathbf{Par}(n^2)$$

Vejamos uma prova deste condicional.

> PROVA: Para provar $\neg\mathbf{Par}(n) \rightarrow \neg\mathbf{Par}(n^2)$ nós iniciamos assumindo $\neg\mathbf{Par}(n)$, ou seja, que n é ímpar. Então podemos expressar n como $2m + 1$, para algum m. Então podemos ver que:
>
> $$n^2 = (2m + 1)^2$$
> $$= 4m^2 + 4m + 1$$
> $$= 2(2m^2 + 2m) + 1$$
>
> Mas isso mostra que n^2 também é ímpar, ou seja, $\neg\mathbf{Par}(n^2)$.

Ao provarmos a contrapositiva, evitamos a necessidade de uma prova indireta (por absurdo) dentro da prova condicional. Isto torna a prova fácil de entender e, uma vez que a forma contrapositiva é logicamente equivalente a nossa declaração original, nossa segunda prova também serve como uma prova da declaração original.

O método da prova condicional é usado extensivamente em nossos raciocínios do dia a dia. Alguns anos atrás, Bill estava tentando decidir entre duas disciplinas optativas, Português IV ou Pós-modernismo. Sua amiga Sarah declarou que *se Bill fizesse Pós-modernismo, ele não entraria em medicina*. O argumento de Sarah, quando foi questionada por Bill, tomou a forma de uma prova condicional, combinada com uma prova por casos.

> Suponha que você faça Pós-modernismo. Então ou você adotará o desdém pós-moderno em relação à racionalidade, ou você não o adotará. Se não o adotar, você será reprovado na disciplina, o que diminuirá tanto seu IRA que você não entrará em medicina. Mas se você adotar o desprezo pós-moderno para com a racionalidade, você não será capaz de passar em química orgânica, e, em conseqüência, não entrará em medicina. Portanto, em qualquer caso, você não entrará em medicina. Logo, se fizer Pós-modernismo, não entrará em medicina.

Desafortunadamente para Bill, ele já tinha sucumbido ao pós-modernismo e, portanto, rejeitou o argumento de Sarah. Ele foi adiante e fez a disciplina sobre pós-modernismo, foi reprovado em química e não entrou em medicina. Ele, hoje, é um rico lobista em Brasília. Sarah é uma executiva da indústria da Informática em São Paulo.

Provando Bicondicionais

Também podemos utilizar o método da prova condicional para provar bicondicionais. A diferença é que temos que trabalhar duas vezes mais. Para provar $\mathbf{P} \leftrightarrow \mathbf{Q}$ pelo método da prova condicional, precisamos fazer duas coisas: primeiro, assumir $\mathbf{P}$ e provar $\mathbf{Q}$; depois, assumir $\mathbf{Q}$ e provar $\mathbf{P}$. Isso nos dá ambas as sentenças: $\mathbf{P} \rightarrow \mathbf{Q}$ e $\mathbf{Q} \rightarrow \mathbf{P}$, cuja conjunção é equivalente a $\mathbf{P} \leftrightarrow \mathbf{Q}$.

Há uma outra forma de provas envolvendo $\leftrightarrow$ que é muito comum em matemática. Os matemáticos são aficionados em encontrar resultados que mostram que diversas condições diferentes são equivalentes. Assim, você encontrará teoremas que fazem declarações como esta: "As seguintes condições são todas equivalentes: $\mathbf{Q_1}$, $\mathbf{Q_2}$, $\mathbf{Q_3}$." O que eles querem dizer com isso é que todos os seguintes bicondicionais são verdadeiros:

$$\mathbf{Q_1} \leftrightarrow \mathbf{Q_2}$$
$$\mathbf{Q_2} \leftrightarrow \mathbf{Q_3}$$
$$\mathbf{Q_1} \leftrightarrow \mathbf{Q_3}$$

Provando um Ciclo de Condicionais

Para provar estes três bicondicionais da forma padrão que vimos acima, você precisará de fazer seis provas condicionais, duas para cada bicondicional. Mas podemos diminuir nosso trabalho pela metade se notarmos que é suficiente provar algum ciclo de resultados como o seguinte:

$$Q_1 \to Q_2$$
$$Q_2 \to Q_3$$
$$Q_3 \to Q_1$$

Uma prova destas declarações necessitaria de apenas três provas condicionais. Uma vez que você tenha provado estas declarações, não há necessidade de provar as direções reversas das implicações, uma vez que elas se seguem por transitividade de $\to$. Por exemplo, nós não precisamos provar explicitamente que $Q_2 \to Q_1$, a direção reversa do primeiro condicional, uma vez que esta implicação se segue de $Q_2 \to Q_3$ e $Q_3 \to Q_1$, nossos outros dois condicionais.

Vejamos um exemplo bastante simples. Provaremos que as seguintes condições sobre um número natural n são todas equivalentes:

1. n é par;
2. n^2 é par;
3. n^2 é divisível por 4

PROVA: no lugar de provar todos os seis condicionais, provaremos que $(3) \to (2) \to (1) \to (3)$. Assuma (3). Se n^2 é divisível por 4, então, é claro que é divisível por 2. Portanto, $(3) \to (2)$. Em seguida provaremos que $(2) \to (1)$ através da contrapositiva. Assim, assumiremos que n é ímpar e provaremos que n^2 é ímpar. Uma vez que n é ímpar, podemos escrevê-lo da forma $2m + 1$. Mas então, como já mostramos, $n^2 = 2(2m^2 + 2m) + 1$, que também é impar. Finalmente, provaremos que $(1) \to (3)$. Se n é par, pode ser expresso como $2m$. Então, $n^2 = (2m)^2 = 4m^2$, que é divisível por 4. Isto completa o ciclo, mostrando que as três condições são, de fato, equivalentes.

LEMBRE-SE

1. **Método da Prova Condicional**: para provar $P \to Q$, assuma P e prove Q.
2. Para provar um número de bicondicionais, tente arranjá-los em um ciclo de condicionais.

Regras Formais para $\to$ e $\leftrightarrow$

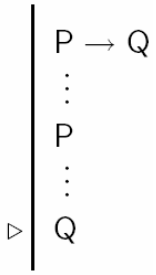
Vamos agora estudar os análogos formais dos métodos de prova envolvendo o condicional e o bicondicional. Novamente, vamos incorporar no Sistema F regras de introdução e eliminação para cada conectivo.

Regras para o Condicional (implicação)

A regra de *modus ponens* ou eliminação do condicional **($\to$Elim)** é facilmente formalizada.

Eliminação do Condicional ($\to$Elim)

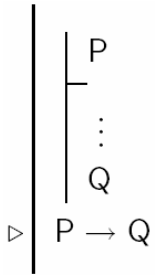
Se você provou ambos $P \to Q$ e P, então você pode asserir Q, citando os dois passos anteriores como justificativa. O seguinte esquema simboliza esta regra:



Introdução do Condicional ($\to$Intro)

A regra de introdução correspondente é a contrapartida formal do método da prova condicional. Ela vai exigir a utilização de subprovas. Para provar uma sentença da forma $P \to Q$ iniciamos uma subprova com P como suposição e tentamos

provar **Q**. Se conseguirmos, fechamos a subprova e concluímos a sentença condicional desejada, citando a subprova como justificativa. Esquematicamente :



Estratégia e Tática: Trabalhando de Trás para Frente

A estratégia de trabalhar de trás para frente usualmente funciona extremamente bem em provas que envolvem condicionais, particularmente quando a conclusão desejada é, ela própria, um condicional. Assim, se o objetivo da prova é mostrar que a sentença $P \rightarrow Q$ é conseqüência de um conjunto de premissas, você deve fazer o esboço de uma subprova que tenha **P** como suposição e **Q** como passo final.

Faça o Tente Isto (1) da Lista de Exercícios 7-8 para praticar.

Transformando uma Prova Com Premissas em uma Prova Sem Premissas de um Condicional

Uma vez que temos a introdução do condicional à nossa disposição, podemos converter qualquer prova com premissas em uma prova, sem premissas, de um condicional correspondente.

Por exemplo, no Tente Isto (7) da Lista de Exercícios 5-6 apresentamos a seguinte prova

que demonstra $\neg\neg A$ a partir da premissa **A**. Podemos agora usar esta prova para construir uma outra prova para a sentença (logicamente necessária) $A \rightarrow \neg\neg A$. Veja:

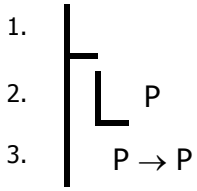
Note que a subprova, aqui, é idêntica à prova original (acima). Nós apenas encapsulamos aquela prova em nossa prova nova e aplicamos, ao final, a regra de introdução do condicional **(→Intro)** para derivar $A \rightarrow \neg\neg A$.

Usos Generosos das regras → em FITCH

A regra **(→Elim)** não se importa com a ordem em que você cita as sentenças de suporte.

A regra (→Intro) não exige que o conseqüente do condicional deduzido seja o último passo da subprova citada, embora ele usualmente seja.

Além disso, o passo da suposição pode ser o único passo de uma subprova, como em uma prova de uma sentença com a forma $P \rightarrow P$.



→Intro: 2

Aplicações *Default* das Regras → em Fitch

As aplicações *default* das regras do condicional funcionam exatamente como se poderia esperar. Ao você citar as sentenças ou subprovas de suporte para a aplicação da regra, o programa **Fitch** preenche automaticamente a conclusão para você.

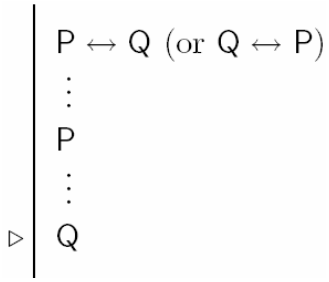
Faça o Tente Isto (2) da Lista de Exercícios 7-8 para praticar os usos generosos e aplicações *default* das regras para o condicional.

Regras para o Bicondicional (biimplicação)

As regras para o bicondicional são exatamente o que você deveria esperar, dadas as regras para o condicional.

Eliminação do Bicondicional (↔Elim)

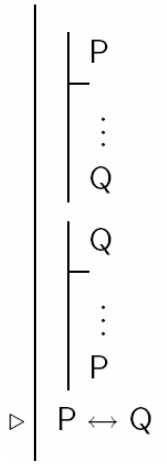
A regra de eliminação do bicondicional pode ser formulada esquematicamente como segue:



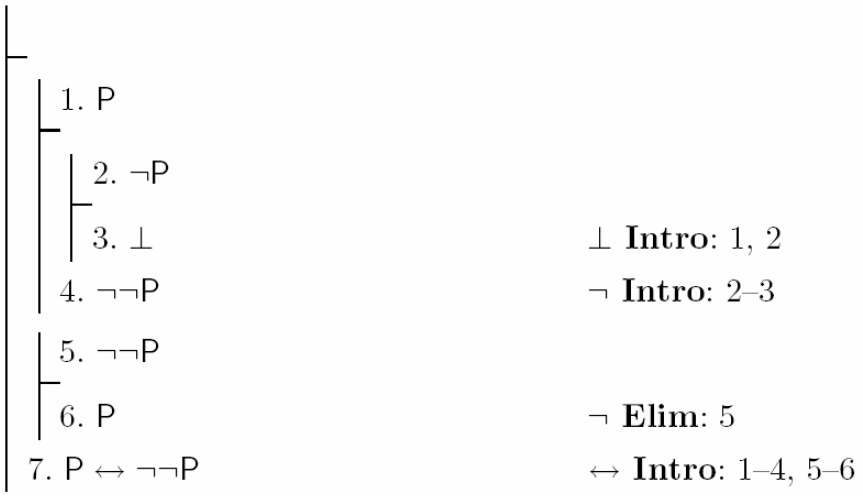
Isto significa que você pode concluir **Q** se tiver estabelecido, anteriormente na prova, **P** e um dos dois bicondicionais indicados.

Introdução do Bicondicional (↔Intro)

A regra de introdução para o bicondicional $P \leftrightarrow Q$ requer que você faça duas subprovas, uma mostrando que **Q** se segue de **P**, e uma outra mostrando que **P** se segue de **Q**. Esquematicamente:



Eis um exemplo simples de uma prova que usa a introdução do bicondicional. Ele mostra como provar a lei da dupla negação dentro do Sistema **F**.



Estratégia e Tática: Trabalhando de Trás para Frente

Quando você estiver construindo uma prova cuja conclusão é um bicondicional, é particularmente importante esboçar antecipadamente a prova. Adicione as duas subprovas necessárias e a conclusão desejada, citando as subprovas como suporte. Só após isso tente preencher as subprovas. Isto é uma boa idéia porque estas provas, algumas vezes, tornam-se bastante longas e envolventes. O esboço irá ajudá-lo a lembrar-se do que está tentando fazer.

Faça o Tente Isto (3) da Lista de Exercícios 7-8 para praticar.

Correção e Completude

Uma vez que introduzimos todas as regras formais para os conectivos verofuncionais, vamos agora nos perguntar sobre duas propriedades desejáveis para os sistemas dedutivos, as quais os lógicos denominam *correção* e *completude*.

Mas não se confunda com estes nomes. Os usos de *correto* e *completo* que veremos aqui são bastante diferentes das noções de *argumento correto* e *conjunto de conectivos verofuncionalmente completo*.

Correção

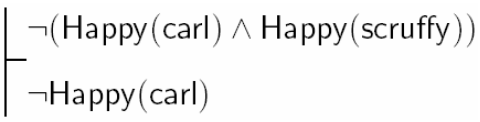
Pretendemos que nosso sistema formal **F** seja um sistema de dedução **correto**, no sentido de que qualquer argumento que possa ser provado em **F** deve ser genuinamente válido.

A primeira questão que nos perguntaremos, então, é se fomos bem sucedidos neste objetivo. Será que o sistema **F** nos permite construir provas apenas de argumentos genuinamente válidos?

Tal indagação é conhecida como a questão da correção para o sistema dedutivo **F**.

A resposta a esta questão pode até parecer obvia, mas ela merece ser olhada mais de perto. Quem nos garante que não há nenhuma imperfeição em alguma de nossas regras oficiais? Ou talvez haja problemas que estão além das regras individuais. Algo sobre a forma como elas interagem.

Considere, por exemplo, o seguinte argumento:



Sabemos que este argumento não é válido, dado que é claramente possível a premissa ser verdadeira e a conclusão falsa. Mas como sabemos que as regras que introduzimos não permitem alguma prova bastante complicada e engenhosa da conclusão a partir da premissa? Afinal de contas não há como examinarmos todas as provas possíveis para nos certificarmos de que não há uma com esta premissa e esta conclusão. Isto porque existe uma quantidade infinita de provas possíveis.

Para responder nossa questão sobre a correção, precisamos torná-la mais precisa. Já vimos que a noção de conseqüência lógica é um tanto vaga. O conceito de conseqüência tautológica, por sua vez, é uma definição precisa que se aproxima desta noção informal.

Uma maneira de tornar nossa questão mais precisa é perguntar se as regras dos conectivos verofuncionais nos permitem provar apenas argumentos que sejam tautologicamente válidos. Esta questão deixa de considerar se as regras da identidade são legítimas, mas nós vamos tratar disso mais adiante neste livro.

Vamos introduzir alguns símbolos novos para tornar mais fácil expressar a declaração que queremos investigar.

Subsistema Formal F_T

Usaremos F_T para nos referirmos à porção do sistema formal **F** que contém apenas as regras de introdução e eliminação para os conectivos $\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$, $\perp$.

Você pode pensar no subscrito $_T$ de F_T como se referindo a "tautologia" ou "*truth-functional*"[2]

Também escreveremos $P_1, \ldots, P_n \vdash_T S$ para indicar que existe uma prova forma em F_T de **S** p partir das premissas $P_1, \ldots, P_n$.[3]

Teorema da Correção de F_T

Podemos agora formular nossa declaração com precisão.

Teorema (Correção de F_T) Se $P_1, \ldots, P_n \vdash_T S$ então **S** é uma conseqüência tautológica de $P_1, \ldots, P_n$.

Prova: suponha que *p* seja uma prova construída no sistema formal F_T. Vamos mostrar que a seguinte afirmação é verdadeira:

(1) Qualquer sentença que ocorra em qualquer passo na prova *p* é uma conseqüência tautológica das suposições em vigor naquele passo.

Isto se aplica não apenas às sentenças no nível principal de *p*, mas também às sentenças que ocorrem nas subprovas, não importando quão profundamente aninhadas elas sejam. As suposições em vigor em um dado passo sempre incluem as premissas principais, mas se estamos lidando com um passo interno a subprovas aninhadas, estão em vigor, também, todas as suposições destas subprovas.

O teorema da correção é conseqüência da afirmação **(1)** porque se **S** está no nível principal de *p*, então as únicas suposições em vigor são as premissas $P_1, \ldots, P_n$. Logo, **S** é conseqüência tautológica de $P_1, \ldots, P_n$.

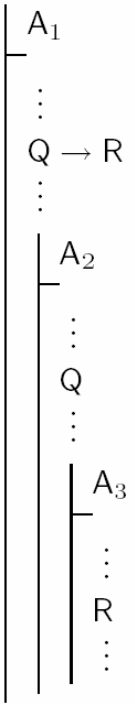
Para provar a afirmação **(1)** usaremos o método da prova por absurdo. Suponha (por absurdo) que haja um passo em *p* contendo uma sentença que não é uma conseqüência tautológica das suposições em vigor naquele passo. Chamaremos este de um *passo inválido*. A idéia de nossa prova é olhar para o primeiro passo inválido em *p* e mostrar que nenhuma das 12 regras de F_T poderia justificá-lo. Em outras palavras, aplicaremos o método da prova por casos para mostrar que haverá uma contradição, seja qual for a regra de F_T tenha sido aplicada para justificar o passo inválido. Isso nos permitirá concluir que nossa suposição (hipótese do absurdo) [de que há um passo em *p* contendo uma sentença que não é conseqüência tautológica das suposições em vigor naquele passo] tem que ser falsa. Logo, não há passos inválidos nas provas de F_T, e, portanto, F_T é correto (**S** é conseqüência tautológica de $P_1, \ldots, P_n$).

CASO 1: **($\rightarrow$Elim)** Suponha que o primeiro passo inválido justifique a sentença **R** através de uma aplicação de **($\rightarrow$Elim)** às sentenças $Q \rightarrow R$ e **Q** que ocorrem anteriormente na prova *p*. Seja $A_1, \ldots, A_k$ uma lista de todas as suposições em vigor no passo de **R**. Se este é um passo inválido, **R** não é conseqüência tautológica de $A_1, \ldots, A_k$. Mas nós mostraremos que isso nos leva a uma contradição.

Uma vez que **R** é o primeiro passo inválido na prova *p*, sabemos que $Q \rightarrow R$ e **Q** são ambos passos válidos. Ou seja, são conseqüências tautológicas das suposições em vigor nestes passos. A observação crucial é que, uma vez que F_T nos permite citar sentenças apenas da prova principal ou de subprovas cujas suposições estejam ainda em vigor (subprovas ainda não fechadas), então nós sabemos que todas as suposições em vigor nos passos $Q \rightarrow R$ e **Q** ainda estão em vigor no passo **R**. Portanto, as suposições em vigor nestes dois passos estão entre as suposições da lista $A_1, \ldots, A_k$. Uma ilustração pode ajudar. Suponha que nossa prova tenha a seguinte forma:

[2] que é a expressão inglesa para verofuncional.

[3] O símbolo $\vdash$ é comumente utilizado em lógica para indicar a provabilidade do que está à direita a partir do que está à esquerda. Se você pensar na barra de Fitch (nossa notação para argumentos) ficará fácil de lembrar o significado desta notação.

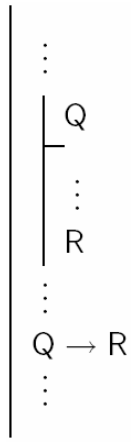


Deve estar claro que as restrições a citações de passos anteriores em provas garantem que todas as suposições em vigor nos passos citados continuam em vigor no passo contendo **R**. No exemplo mostrado, a suposição A_1 está em vigor no passo contendo $Q \rightarrow R$, as suposições A_1 e A_2 estão em vigor no passo contendo **Q**, e as suposições A_1, A_2 e A_3 estão em vigor no passo contendo **R**.

Suponha agora que construamos uma tabela de verdade conjunta para as sentenças $A_1, \ldots, A_k$, **Q**, $Q \rightarrow R$ e **R**. Devido a hipótese que assumimos de que **R** é um passo inválido (hipótese do absurdo), deve haver uma linha *h* nesta tabela na qual $A_1, \ldots, A_k$ sejam todas verdadeiras, mas **R** seja falsa. No entanto, uma vez que **Q** e $Q \rightarrow R$ são conseqüências tautológicas de $A_1, \ldots, A_k$, ambas estas sentenças são verdadeiras na linha *h*. Mas isto contradiz a tabela de verdade para $\rightarrow$, pois sabemos que se **Q** e $Q \rightarrow R$ são verdadeiras, não é possível que **R** seja falsa (**Q** verdadeira e **R** falsa tornariam $Q \rightarrow R$ falsa !!)

Ou seja, chegamos a uma contradição. Logo, não é possível que a regra que justifique o primeiro passo inválido de uma prova *p* seja a regra **(→Elim)**. Para terminar a prova temos que fazer casos semelhantes para todas as outras 11 regras formais de F_T. Não faremos isso. Provaremos mais dois casos e deixaremos os demais como exercício.

CASO 2: **(→Intro)** Suponha que o primeiro passo inválido derive a sentença $Q \rightarrow R$ a partir de uma aplicação da regra **(→Intro)** citando uma subprova anterior cuja suposição é **Q** e a conclusão é **R**.



Novamente, considere $A_1, \ldots, A_k$ como as suposições em vigor no passo $Q \rightarrow R$. Note que as suposições em vigor no passo **R** são $A_1, \ldots, A_k$ e **Q**. Uma vez que o passo **R** ocorre antes do primeiro passo inválido, (*a*) **R** deve ser uma conseqüência tautológica de $A_1, \ldots, A_k$ e **Q**.

Suponha que tenhamos construído uma tabela de verdade conjunta para as sentenças $A_1, \ldots, A_k$, **Q**, $Q \rightarrow R$ e **R**. Como $Q \rightarrow R$ é o primeiro passo inválido na prova p, ele não é conseqüência tautológica das suposições em vigor. Portanto, deve haver uma linha *h* nesta tabela onde $A_1, \ldots, A_k$ sejam todas verdadeiras, mas $Q \rightarrow R$ seja falsa. Como $Q \rightarrow R$ é falsa nesta linha, então **Q** tem que ser verdadeira e

R falsa. Mas isto contradiz nossa observação (*a*) do parágrafo acima, de que **R** é uma conseqüência tautológica de $A_1, \ldots, A_k$ e **Q**. Novamente, não é possível que a regra que justifique o primeiro passo inválido de uma prova *p* seja a regra **(→Intro)**.

CASO 3: **(⊥Elim)** Suponha que o primeiro passo inválido em *p* derive a sentença **Q** a partir de ⊥. Uma vez que este é o primeiro passo inválido, ⊥ deve ser uma conseqüência tautológica das suposições em vigor no passo de ⊥. Devido às mesmas considerações feitas no primeiro caso, as suposições em vigor par ⊥ continuam em vigor para **Q**. Então, como ⊥ ocorre antes do primeiro passo inválido, ⊥ é uma conseqüência tautológica das suposições $A_1, \ldots, A_k$ em vigor no passo **Q**. Mas isso só ocorre se $A_1, \ldots, A_k$ forem TT-contraditórias. Em outras palavras, não nenhuma linha em uma tabela de verdade conjunta para $A_1, \ldots, A_k$ em que $A_1, \ldots, A_k$ sejam todas verdadeiras. Mas então, **Q** é vacuamente uma conseqüência tautológica de $A_1, \ldots, A_k$. E isso contradiz a suposição do caso 3, de que **Q** é o primeiro passo inválido de *p*. Portanto, também não é possível que a regra que justifique o primeiro passo inválido de uma prova *p* seja **(⊥Elim)**.

CASOS das demais regras: Vimos três 3 casos das 12 regras. Os demais casos são similares a estes e, portanto, deixamo-los como exercícios.

Como em todos os 12 casos uma contradição é demonstrada, podemos concluir que nossa suposição original (hipótese do absurdo), de que é possível uma prova em F_T conter um passo inválido deve ser falsa. Isto finaliza a prova da correção.

Tendo provado o Teorema da Correção, podemos ficar absolutamente seguros de que não importa o quão duro alguém tentar, não importa quão inteligente e criativo seja quem estiver tentando, será impossível produzir uma prova de **¬Happy(carl)** a partir da premissa **¬(Happy(carl) ∧ Happy(scruffy))**. Por que? Não há tal prova porque a primeira sentença não é conseqüência tautológica da segunda.

Um **corolário** é um resultado que se segue com pouco esforço de um teorema anterior. Podemos formular o seguinte corolário, que simplesmente aplica o Teorema da Correção a casos em que não há premissas.

Corolário Se $\vdash_T$ **S**, ou seja, se há uma prova sem premissas de **S** em F_T, então, **S** é uma tautologia.

A importância deste corolário é ilustrada na **Figura 8.1**. O corolário nos diz que se uma sentença é provável em F_T, então ela é uma tautologia. O Teorema da Correção nos assegura isso. Basta que o apliquemos a uma prova sem premissas, pois sabemos que tautologia e conseqüência tautológica de um conjunto vazio de premissas são a mesma coisa.

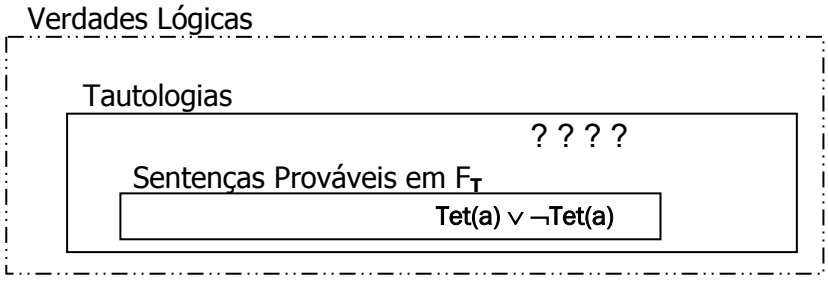


Figura 8.1: O Teorema da Correção para F_T nos diz que apenas tautologias são prováveis sem premissas em F_T.

Repare nos pontos de interrogação na figura acima. Até agora não sabemos se há alguma tautologia (ou argumentos tautologicamente válidos) que não são prováveis em F_T. Esta é a segunda questão que precisamos nos dirigir: a questão da *completude*.

Sistemas Dedutivos Corretos x Argumentos Corretos

Antes de prosseguirmos, vale a pena reforçar que o uso do termo *correção* nesta seção tem muito pouco a ver com nosso uso anterior do termo para descrever argumentos válidos com premissas verdadeiras. O que *correção* de fato significa, quando aplicado a sistemas dedutivos é que o sistema permite a você provar apenas argumentos *válidos*. Seria mais apropriado usarmos o termo "validação" (Teorema da Validação). Mas não é assim que ele é tradicionalmente chamado.

Completude

Algumas vezes, ao fazer um exercício, você pode ter se desesperado tentando encontrar uma prova para algum argumento que sabia ser válido. Nossa segunda questão se dirige a este problema. Será que nossos sistemas dedutivos nos permitem provar tudo o que deveríamos ser capazes de provar?

Claro que isso levanta a questão sobre o que nós "deveríamos" ser capazes de provar, o que, novamente, nos confronta com a vaguidade da noção de conseqüência lógica.

Mas, com o Teorema da Correção, que já provamos, sabemos que o máximo que F_T nos permitirá provar são conseqüências tautológicas. Então, podemos reformular nossa questão com mais precisão:

Seria possível nos convencermos de que dado qualquer conjunto de premissas $P_1, \ldots, P_n$ e qualquer conseqüência tautológica S destas premissas, nosso sistema dedutivo F_T nos permite construir uma prova de S a partir de $P_1, \ldots, P_n$? Ou será que há algumas conseqüências tautológicas de certos conjuntos de premissas que simplesmente estão fora do alcance do nosso sistema dedutivo F_T? O Teorema da Completude nos assegura que a segunda alternativa não ocorre.

Teorema da Completude de F_T

Formulando o ponto acima com precisão temos:

Teorema (Completude de F_T) Se S é uma conseqüência tautológica de $P_1, \ldots, P_n$, então $P_1, \ldots, P_n \vdash_T S$.

A prova deste resultado é bastante mais complicada do que a prova do Teorema da Correção, e requer material que ainda não foi introduzido. Conseqüentemente, não a apresentaremos aqui, mas apenas no Capítulo 17.

Este resultado é chamado *Teorema da Completude* porque nos diz que as regras de introdução e eliminação são completas para a lógica dos conectivos verofuncionais. Qualquer argumento tautologicamente válido (válido simplesmente em virtude do significado dos conectivos verofuncionais) pode ser provado em F_T.

Correção e Incompletude

Note, no entanto, que o Teorema da Correção implica um tipo de *incompletude*, uma vez que as regras de F_T nos permitem provar apenas conseqüências *tautológicas* de nossas premissas. Elas não nos permitem provar qualquer conseqüência lógica que não seja conseqüência tautológica das premissas. Por exemplo, o Teorema da Correção de F_T mostra que não há maneira de provar $\mathbf{Dodec(c)}$ de $\mathbf{Dodec(b) \wedge b = c}$ em F_T, uma vez que a primeira sentença não é conseqüência tautológica da segunda, embora seja claramente conseqüência lógica da segunda. Para provar coisas como estas, precisamos das regras da identidade. Também não conseguimos provar $\neg\mathbf{Larger(c, b)}$ a partir $\mathbf{Larger(b, c)}$. Para fazer isso precisaríamos de regras sobre o predicado $\mathbf{Larger}$ e da utilização de *quantificadores*. O que só vamos começar a estudar a partir do próximo capítulo.

Usos da Correção e Completude

O Teorema da Completude nos oferece um método para mostrar que um argumento tem uma prova sem a necessidade de fazermos de fato tal prova: basta mostrar que a conclusão é conseqüência tautológica das premissas.

Por exemplo, é óbvio que $\mathbf{A \rightarrow (B \rightarrow A)}$ é uma tautologia, então, pelo Teorema da Completude, sabemos que deve haver uma prova sem premissas de $\mathbf{A \rightarrow (B \rightarrow A)}$. Similarmente, a sentença $\mathbf{B \wedge \neg D}$ é uma conseqüência tautológica de $\mathbf{\neg((A \wedge B) \rightarrow (C \vee D))}$ então, sabemos que deve ser possível provar a primeira a partir da segunda.

O Teorema da Correção, por outro lado, nos oferece um método para dizer que um argumento não tem uma prova em F_T: basta mostrar que a conclusão não é uma conseqüência tautológica das premissas.

Por exemplo, $\mathbf{A \rightarrow (A \rightarrow B)}$ não é uma tautologia, então não é possível construir uma prova desta sentença em F_T, não importa o quanto você tente. Similarmente a sentença $\mathbf{B \wedge \neg D}$ não é uma conseqüência tautológica de $\mathbf{\neg((A \vee B) \rightarrow (C \wedge D))}$, então nós sabemos que não há prova disso em F_T.

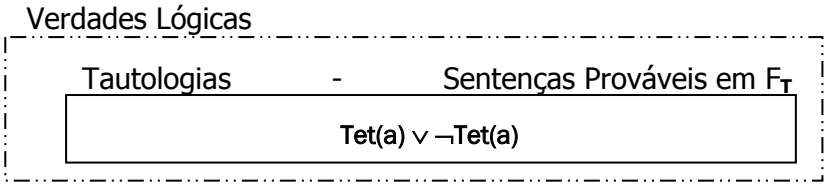


Figura 8.2: Completude e correção para F_T nos dizem que todas e apenas as tautologias são prováveis sem premissas em F_T.

Tirando Proveito do Mecanismo TautCon

Sabemos que o mecanismo **(TautCon)** verifica se uma sentença é conseqüência tautológica das sentenças citadas em suporte. Conforme o que vimos acima podemos, então, utilizar este mecanismo para nos ajudar a saber se é possível fazer uma determinada prova usando apenas as regras de F_T. Se **(TautCon)** diz que uma determinada sentença é uma conseqüência tautológica das sentenças citadas, então sabemos que será possível fazer uma prova da sentença, utilizando as sentenças citadas como premissas, ainda que não tenhamos percebido exatamente como tal prova será desenvolvida. Por outro lado, se **(TautCon)** diz que a sentença não é conseqüência tautológica das sentenças citadas, então não há razão em tentar encontrar uma prova em F_T, pelo simples fato de que não existe tal prova.

LEMBRE-SE

1. **(Completude de F_T)** Se **S** é uma conseqüência tautológica de $P_1, ..., P_n$, então existe uma prova de **S** a partir das premissas $P_1, ..., P_n$ que utiliza apenas as regras de introdução e eliminação para $\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$ e $\perp$.
2. **(Correção de F_T)** Se **S** não é uma conseqüência tautológica de $P_1, ..., P_n$, então não existe prova de **S** a partir das premissas $P_1, ..., P_n$ que utiliza apenas as regras de introdução e eliminação para $\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$ e $\perp$.
3. O Mecanismo **(TautCon)** nos ajuda a determinar qual destas alternativas ocorre.
4. Note que os itens (1) e (2) afirmam que:

   S é conseqüência tautológica de $P_1, ..., P_n$ *se e somente se* existe uma prova em F_T de **S** a partir de $P_1, ..., P_n$

Argumentos Válidos: alguns exercícios de revisão

Há sabedoria no velho ditado: "Não deixe que as árvores o atrapalhem de ver a floresta"[4].

A floresta no nosso caso é um entendimento dos argumentos válidos. As árvores são os vários métodos de prova, formais e informais, e as noções de contra-exemplo, tautologia e demais noções afins. Os problemas nesta seção (Exercícios 8.44 a 8.53) são planejados para lembrar-nos da relação entre a floresta e as árvores, além de ajudar-nos a rever as principais idéias discutidas até aqui.

Uma vez que agora já sabemos que nossas regras de introdução e eliminação são suficientes para provar qualquer argumento tautologicamente válido, estamos liberados para usar o mecanismo **(TautCon)** em nossas provas formais, para justificar passagens simples que, em uma prova informal nem seriam mencionadas. Assim, por exemplo, se para fazer uma prova você precisar supor uma instância princípio do terceiro excluído ($P \vee \neg P$), então ao invés de repetir a prova desta verdade lógica (Exercício 6.33), você pode simplesmente inseri-la em sua prova e justificá-la com **(TautCon)**.

Mas não esqueça, porém, de usar o bom senso. **(TautCon) só ajudará se você utilizá-lo para pular passos simples, aqueles que em provas informais nem mencionamos, ou se mencionamos nem justificamos**. Caso contrário, se abusar do **(TautCon)**, você não estará estudando lógica, mas apenas confirmando que o mecanismo implementado no programa **Fitch** funciona. E isso você já sabe.

***** Este é o FIM da PARTE I do livro. Na PARTE II estudaremos os Quantificadores !!! *****

[4] Tradução livre de: "*Don't lose sight of the forest for the trees*"

Resumo Geral dos Capítulos

Capítulo 1 - Sentenças Atômicas

Constantes Individuais

Em FOL

- Cada constante individual deve nomear um objeto (existente).
- Nenhuma constante individual pode nomear mais de um objeto.
- Um objeto pode ter mais de um nome ou mesmo nenhum nome.

Predicados

Em FOL

- Cada símbolo de predicado tem uma única (e fixa) "aridade", um número que diz a você quantos nomes ele necessita para formar uma sentença atômica.
- Cada predicado é interpretado por uma propriedade ou relação determinável da mesma aridade que o predicado.

Sentenças Atômicas

Em FOL,

- Sentenças atômicas são formadas colocando-se um predicado de aridade *n* na frente de *n* nomes (cercados por parênteses e separados por vírgulas).
- Sentenças atômicas também são construídas a partir do predicado de identidade, =, usando notação infixa: os argumentos são colocados em cada um dos lados do predicado.
- A ordem dos nomes é crucial na formação de sentenças atômicas.

Símbolos de Função

Em uma linguagem com símbolos de função,

- Termos complexos são tipicamente formados colocando-se um símbolo de função de aridade *n* na frente de *n* termos (simples ou complexos).
- Termos complexos são usados exatamente como os nomes (termos simples) na construção de sentenças atômicas.
- Em FOL, assumimos que os termos complexos também se referem a um e apenas um objeto.

Capítulo 2 - A Lógica das Sentenças Atômicas

Validade de Argumentos e Conseqüência Lógica

1. Um ***argumento*** é uma série de afirmações na qual uma, chamada de ***conclusão***, é tomada como ***conseqüência*** das outras, chamadas de ***premissas***.
2. Um argumento é ***válido*** se a conclusão for verdadeira em qualquer circunstância na qual as premissas são verdadeiras. Diremos que a *conclusão* de um argumento logicamente válido é uma ***conseqüência lógica*** de suas premissas.
3. Um argumento é ***correto*** se for válido e suas premissas forem todas verdadeiras.

Provas (Demonstração)

1. Uma prova para uma sentença **S** a partir das premissas P_1 ,..., P_n é uma demonstração passo a passo que mostra que **S** deve ser verdadeira em quaisquer circunstâncias nas quais as premissas P_1, ..., P_n sejam todas verdadeiras.
2. Provas formais e informais diferem apenas no estilo, não no rigor.

Princípios Sobre a Identidade

Há quatro princípios importantes válidos para a relação de identidade:

1. **(=Elim)**: Se **b = c**, então tudo o que é válido para **b** é valido para **c**. Este princípio também é conhecido como indiscernibilidade de idênticos.
2. **(=Intro)**: Sentenças da forma **b = b** são sempre verdadeiras em FOL. Isto também é conhecido como reflexividade da identidade.
3. **Simetria da Identidade**: se **b=c**, então **c=b**.
4. **Transitividade da Identidade**: Se **a=b** e **b=c**, então **a=c**.

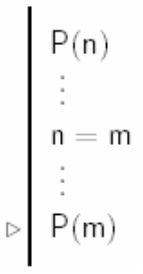
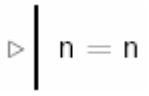
Os últimos dois princípios se seguem dos dois primeiros.

Dependência de Predicados

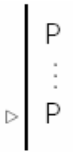
A tabela abaixo apresenta as dependências entre predicados mais comumente encontradas em linguagens formalizadas, juntamente com os predicados da linguagem dos blocos em que cada uma delas é satisfeita. Não há regras lógicas para estas propriedades. mas o mecanismo **(AnaCon)** permite inferir a maioria delas[1]

NOME	DEPENÊNCIA	EXEMPLO NA LINGUAGEM DOS BLOCOS
Reflexividade	Vale **P(a, a)**	SameSize, SameShape, SameCol, SameRow
Simetria	Se vale **P(a, b)**, então vale **P(b, a)**	SameSize, SameShape, SameCol, SameRow, Adjoins
Transitividade	Se valem **P(a, b)** e **P(b, c)**, então vale **P(a, c)**	SameSize, SameShape, Larger, Smaller, SameCol, SameRow, LeftOf, RIghtOf, FrontOf, BackOf,
Não-reflexividade	Não vale **P(a, a)**	Larger, Smaller, Adjoins, LeftOf, RightOf, FrontOf, BackOf, Between
Antissimetria	Se vale **P(a, b)**, então não vale **P(b, a)**	Larger, Smaller, LeftOf, RightOf, FrontOf, BackOf
Não-transitividade	Não é verdade que: "Se valem **P(a, b)** e **P(b, c)**, então vale **P(a, c)**"	Adjoins
Relações Inversas	**P(a, b)** vale, se e somente se **Q(b, a)** também vale (**P – Q** são inversos)	Larger – Smaller, LeftOf – RightOf, FrontOf – BackOf

Regras Formais para a Identidade =

Eliminação da Identidade (= Elim)	Introdução do Condicional (= Intro)
	

Regra Estrutural de Reiteração

Eliminação da Identidade (= Elim)


[1] Exceção aos predicados Between e Adjoins, que não são tratados pelo mecanismo **(AnaCon)**.

Provas Formais

- O sistema dedutivo que você está aprendendo é um sistema dedutivo do estilo Fitch, chamado $\mathcal{F}$.
- A aplicação de computador que auxilia você na construção de provas em $\mathcal{F}$ é, por isso, chamada de programa **Fitch**.
- Quando você escreve suas provas no papel, você está utilizando o sistema $\mathcal{F}$, quando o faz no computador, você está utilizando o programa Fitch.

Demonstrando Não-Conseqüência

Para demonstrar a invalidade de um argumento com premissas $P_1, \ldots, P_n$ e conclusão Q, encontre um contra-exemplo: uma circunstância possível em que $P_1, \ldots, P_n$ sejam todas verdadeiras, mas Q seja falsa. Tal contra-exemplo mostra que Q não é conseqüência de $P_1, \ldots, P_n$.

Capítulo 3 - Os Conectivos Booleanos

Negação

1. Se P é uma sentença de FOL, então $\neg P$ também é.
2. A sentença $\neg P$ é verdadeira se e somente se P não for.
3. Uma sentença que é ou atômica ou a negação de uma sentença atômica é chamada de ***literal***.

P	¬P
TRUE	FALSE
FALSE	TRUE

Conjunção

1. Se P e Q são sentenças de FOL, então $P \wedge Q$ também é.
2. A sentença $P \wedge Q$ é verdadeira se e somente se P e Q forem ambas verdadeiras.

P	Q	P ∧ Q
T	T	T
T	F	F
F	T	F
F	F	F

Disjunção

1. Se P e Q são sentenças de FOL, então $P \vee Q$ também é.
2. A sentença $P \vee Q$ é verdadeira se e somente se P é verdadeira ou Q é verdadeira (ou ambas são verdadeiras).

P	Q	P ∨ Q
T	T	T
T	F	T
F	T	T
F	F	F

Rergras do Jogo Henkin-Hintikka para os Conectivos Booleanos

Forma da Sentença	Seu Compromisso	Quem Joga	A Jogada
$P \vee Q$	TRUE	Você	Escolher uma das sentenças **P** ou **Q** que seja verdadeira.
	FALSE	o programa	
$P \wedge Q$	TRUE	o programa	Escolher uma das sentenças **P** ou **Q** que seja falsa.
	FALSE	Você	
$\neg P$	qualquer que seja	-	Substituir $\neg P$ por **P** e trocar o compromisso.

Uso dos Parênteses

Os parênteses devem ser utilizados sempre que possa haver ambigüidade caso sejam omitidos. Na prática, isto significa que conjunções e disjunções precisam ser envolvidas por parênteses sempre que estiverem combinadas com outros conectivos.

Formas Equivalentes de Dizer Coisas

(Dupla Negação e Leis de DeMorgan) Para quaisquer sentenças **P** e **Q**:

1. Dupla negação: $\neg\neg P \Leftrightarrow P$
2. DeMorgan: $\neg(P \wedge Q) \Leftrightarrow (\neg P \vee \neg Q)$
3. DeMorgan: $\neg(P \vee Q) \Leftrightarrow (\neg P \wedge \neg Q)$

Tradução Correta português-FOL

Para que uma sentença de FOL seja uma boa tradução de uma sentença do português, é suficiente que as duas sentenças tenham os mesmos valores de verdade em todas as circunstâncias possíveis, ou seja, que elas tenham as mesmas ***condições de verdade***.

Sutilezas das Traduções – Limitações de FOL

1. A expressão "***e***" do português algumas vezes sugere ordem temporal. A expressão $\wedge$ de FOL nunca dá este tipo de sugestão.
2. As expressões do português "***mas***", "***porém***", "***todavia***", "***contudo***", "***entretanto***", "***no entanto***" são todas variantes estilizadas da expressão "***e***".
3. As expressões do português "***ou ... ou***" e "***ambos***" são frequentemente usadas como parênteses para evitar possíveis ambigüidades das sentenças.

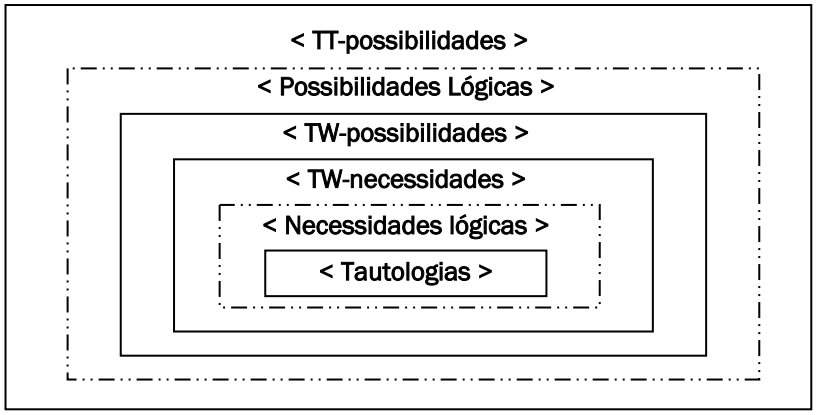
Capítulo 4 - A Lógica dos Conectivos Booleanos

Necessidade e Possibilidade Lógicas - Noções Vagas

Possibilidade Lógica: uma afirmação é *logicamente possível* se há alguma circunstância logicamente possível (ou situação, ou mundo) na qual a afirmação é verdadeira.

Necessidade Lógica: similarmente, uma sentença é *logicamente necessária* se ela for verdadeira em todas as circunstâncias logicamente possíveis.

Diagrama – Necessidade / Possibilidade)



Tautologias ⊂ Necessidades Lógicas

Seja **S** uma sentença de FOL construída a partir de sentenças atômicas usando apenas conectivos verofuncionais. Uma tabela de verdade para **S** mostra como o valor de verdade de **S** depende somente dos valores de verdade de suas partes atômicas.

1. **S** é uma tautologia se e somente se toda linha da tabela de verdade para **S** leva o valor **TRUE**.
2. Se **S** é uma tautologia, então **S** é uma verdade lógica (ou seja, é logicamente necessária)
3. Algumas verdades lógicas não são tautologias

Equivalências Tautológicas ⊂ Equivalências Lógicas

Sejam **S** e **S'** sentenças de FOL construídas a partir de sentenças atômicas utilizando-se apenas conectivos verofuncionais. Para verificar se são equivalências tautológicas, construímos tabelas de verdade conjuntas para as duas sentenças.

1. **S** e **S'** são tautologicamente equivalentes se e somente se cada linha da tabela de verdade conjunta atribui os mesmos valores para **S** e **S'**.
2. Se **S** e **S'** são tautologicamente equivalentes, então eles são logicamente equivalentes.
3. Algumas sentenças logicamente equivalentes não são tautologicamente equivalentes.

Conseqüência Lógica como Conceito Fundamental da Lógica

Podemos definir os conceitos que temos tratados como casos particulares do conceito fundamental de conseqüência lógica:

Necessidade Lógica: uma sentença é *logicamente necessária* (ou uma necessidade lógica) quando for *conseqüência lógica* de qualquer conjunto de premissas.

Equivalência Lógica: duas sentenças são *logicamente equivalentes* se cada uma for *conseqüência lógica* da outra.

Possibilidade Lógica: uma sentença **S** é *logicamente possível* (ou uma possibilidade lógica) quando sua negação **¬S** não for conseqüência lógica de qualquer conjunto de premissas. Ou seja, quando **¬S** não for uma necessidade lógica.

Conseqüência Tautológica ⊂ Conseqüência Lógica

Sejam P_1, ..., P_n e **Q** sentenças de FOL construídas a partir de sentenças atômicas utilizando-se apenas conectivos verofuncionais. Em uma tabela de verdade conjunta para todas estas sentenças, podemos verificar os seguintes fatos:

1. **Q** é conseqüência tautológica de $P_1, ..., P_n$ se e somente se toda linha que atribui **T** para cada $P_1, ..., P_n$, também atribui **T** para **Q**.

2. Se **Q** é uma conseqüência tautológica de $P_1, ..., P_n$, então **Q** também é uma conseqüência lógica de $P_1, ..., P_n$.
3. Algumas conseqüências lógicas não são conseqüências tautológicas. (exemplo **a=c** é conseqüência lógica, mas não conseqüência tautológica de $a=b \wedge b=c$).

Capítulo 5 - Métodos de Prova para a Lógica Booleana

Passos de Inferência Válidos

4. Em uma prova informal a partir de algumas premissas, se sabemos que **Q** é uma conseqüência lógica das sentenças $P_1, ..., P_n$ e se cada $P_1, ..., P_n$ já foi provado a partir dessas mesmas premissas, então podemos afirmar **Q** em nossa prova.
5. Cada passo de uma prova informal deve ser ***significativo*** mas ***de fácil entendimento***.
6. Se um passo é significativo ou de entendimento fácil, isso depende da audiência para quem a prova é endereçada.
7. Os seguintes padrões de inferência válidos são geralmente utilizados em prova informais sem serem mencionados:
    - De $P \wedge Q$ infere-se **P**.
    - De **P** e **Q** infere-se $P \wedge Q$.
    - De **P** infere-se $P \vee Q$.

Método da Prova Por Casos

Para provar **S** a partir de $P_1 \vee ... \vee P_n$ usando este método, prove **S** a partir de cada um dos $P_1, ..., P_n$.

Método da Prova Indireta (ou por Absurdo)

Para provar $\neg S$ usando este método, assuma **S** e prove uma contradição $\perp$.

Do Absurdo ($\perp$) Tudo se Segue

A prova de uma contradição $\perp$ a partir das premissas $P_1, ..., P_n$ (sem suposições adicionais) mostra que estas premissas são inconsistentes. Um argumento com premissas inconsistentes é sempre ***válido***, mas, mais importante ainda, é sempre ***incorreto***.

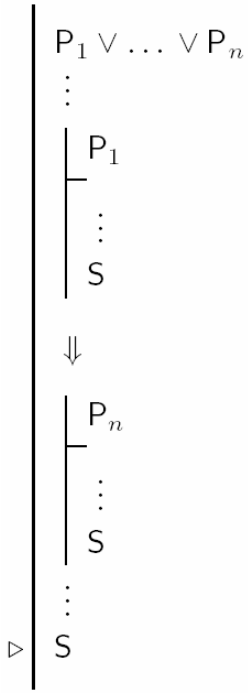
- Como um argumento com premissas inconsistentes é sempre válido, então do absurdo ($\perp$) tudo se segue.

Capítulo 6 - Provas Formais e Lógica Booleana

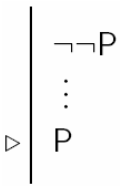
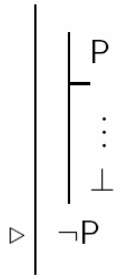
Regras Formais para a Conjunção $\wedge$

Eliminação da Conjunção ($\wedge$ Elim)	Introdução da Conjunção ($\wedge$ Intro)
$P_1 \wedge \ldots \wedge P_i \wedge \ldots \wedge P_n$ $\vdots$ $\triangleright$ P_i	P_1 $\Downarrow$ P_n $\vdots$ $\triangleright$ $P_1 \wedge \ldots \wedge P_n$

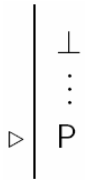
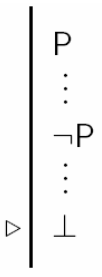
Regras Formais para a Disjunção $\vee$

Eliminação da Dijsunção ($\vee$ Elim)	Introdução da Disjunção ($\vee$ Intro)
	

Regras Formais para a Negação $\neg$

Eliminação da Negação ($\neg$ Elim)	Introdução da Negação ($\neg$ Intro)
	

Regras Formais para o Absurdo $\bot$

Eliminação do Absurdo ($\bot$ Elim)	Introdução do Absurdo ($\bot$ Intro)
	

Subprovas

- Ao justificar um passo em uma subprova, você pode citar qualquer passo anterior contido na prova principal ou em qualquer subprova cuja suposição ainda esteja em vigor. Você não pode jamais citar um passo individual interno a uma subprova que já tenha sido finalizada.
- Fitch lhe impõe estas restrições automaticamente ao não permitir a citação de passos individuais internos a subprovas que já tenham sido finalizadas.

Método para Avaliar a Validade de Argumentos (fazer provas)

Ao avaliar a validade de um argumento, use o seguinte método:

1. Entenda o que as sentenças estão dizendo.
2. Decida se você acha que a conclusão se segue das premissas ou não.
3. Se você acha que ela não se segue, ou não tem certeza, tente encontrar um contraexemplo.
4. Se você acha que ela se segue, tente fazer uma prova informal.
5. Se o exercício pede uma prova formal, use a prova informal para guiá-lo na sua busca por uma prova formal.
6. Ao fazer provas de conseqüência (tanto formais quanto informais), não se esqueça da tática de trabalhar de trás para frente.
7. Ao trabalhar de trás para frente, sempre verifique se suas metas intermediárias são conseqüência da informação disponível.

Provas sem Premissas

Uma prova sem premissas mostra que sua conclusão é uma verdade lógica.

Capítulo 7 - Condicionais

Definição da Implicação Material (ou Condicional)

1. Se P e Q são sentenças de FOL, então $P \rightarrow Q$ também é.
2. A sentença $P \rightarrow Q$ é falsa em apenas um caso: se o antecedente P é verdadeiro e o conseqüente Q é falso. Caso contrário, ela é verdadeira.

Traduções do Condicional

1. As seguintes construções em português são todas traduzidas para $P \rightarrow Q$:
    - *Se **P** então **Q**.*
    - *__Q__ se __P__.*
    - *__P__ apenas se __Q__.*
    - *__Q__ dado que __P__* - ou - *Dado que __P__, __Q__.*
2. A seguinte construção é traduzida para $\neg P \rightarrow Q$:
    - *__Q__ a menos que __P__* - ou - *A menos que __P__, __Q__.*

Interpretando Conseqüência Lógica via Implicação Material

1. Q é uma conseqüência lógica de $P_1, \ldots, P_n$ se e somente se a sentença $(P_1 \wedge \ldots \wedge P_n) \rightarrow Q$ for uma verdade lógica.

Definição da Biimplicação Material (ou Bicondicional)

1. Se P e Q são sentenças de FOL, então $P \leftrightarrow Q$ também é.
2. A sentença $P \leftrightarrow Q$ é verdadeira se e somente se P e Q têm o mesmo valor de verdade.

Insinuações Sociais (*Conversational Implicatures*)

Se a asserção de uma sentença carrega consigo uma sugestão que pode ser cancelada (sem contradição) por alguma elaboração adicional do falante, então a sugestão é apenas uma *insinuação social*, mas não parte do conteúdo da sentença original.

Completude Verofuncional

1. Um conjunto de conectivos é *verofuncionalmente completo* se tais conectivos nos permitirem expressar cada uma das funções de verdade.
2. Vários conjuntos de conectivos, incluindo os conectivos booleanos são verofuncionalmente completos.

Notações Alternativas para os Conectivos

Nossa Notação	Notações Alternativas Equivalentes
$\neg P$	~P, $\overline{P}$, !P
$P \wedge Q$	P & Q, P &&Q, P . Q, PQ
$P \vee Q$	P | Q, P || Q
$P \rightarrow Q$	$P \supset Q$
$P \leftrightarrow Q$	$P \equiv Q$

Capítulo 8 - A Lógica dos Condicionais

Equivalências Lógicas entre Condicionais e Conectivos Booleanos

Os seguintes pares de sentenças são logicamente equivalentes e ajudam a entender significado do condicional e do bicondicional:

1. $P \rightarrow Q \quad \Leftrightarrow \quad \neg Q \rightarrow \neg P$
2. $P \rightarrow Q \quad \Leftrightarrow \quad \neg P \vee Q$
3. $\neg(P \rightarrow Q) \quad \Leftrightarrow \quad P \wedge \neg Q$
4. $P \leftrightarrow Q \quad \Leftrightarrow \quad (P \rightarrow Q) \wedge (Q \rightarrow P)$
5. $P \leftrightarrow Q \quad \Leftrightarrow \quad (P \wedge Q) \vee (\neg P \wedge \neg Q)$

Passos de Inferência Válidos

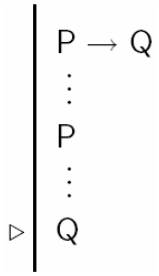
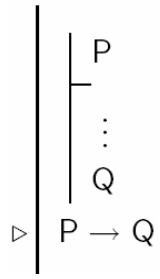
Sejam **P** e **Q** sentenças de FOL.

1. Modus Ponens: De $P \rightarrow Q$ e **P**, infere-se **Q**.
2. Eliminação do Bicondicional: De **P** e ou $P \leftrightarrow Q$ ou $Q \leftrightarrow P$ infere-se **Q**.
3. Contraposição: $P \rightarrow Q \Leftrightarrow \neg Q \rightarrow \neg P$

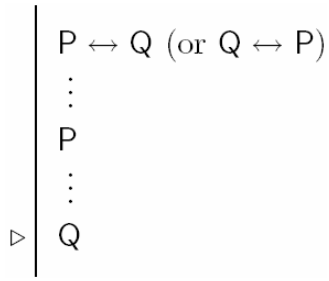
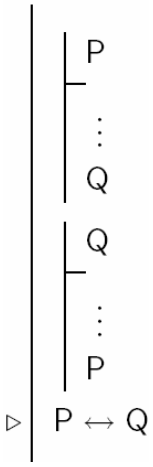
Métodos de Prova

1. O Método da Prova Condicional: Para provar $P \rightarrow Q$, assuma **P** e prove **Q**.
2. O Método dos Ciclos de Condicionais: Para provar um número de bicondicionais, tente arranjá-los de modo a formar um ciclo de condicionais (ex: $P_1 \rightarrow P_2 \rightarrow \ldots \rightarrow P_n \rightarrow P_1$).

Regras Formais para o Condicional $\rightarrow$

Eliminação do Condicional ($\rightarrow$ Elim)	Introdução do Condicional ($\rightarrow$ Intro)
	

Regras Formais para o Bicondicional $\leftrightarrow$

Eliminação do Bicondicional ($\leftrightarrow$ Elim)	Introdução do Bicondicional ($\leftrightarrow$ Intro)
	

Correção e Completude de F_T

Dado um argumento com premissas $P_1, \ldots, P_n$ e conclusão S:

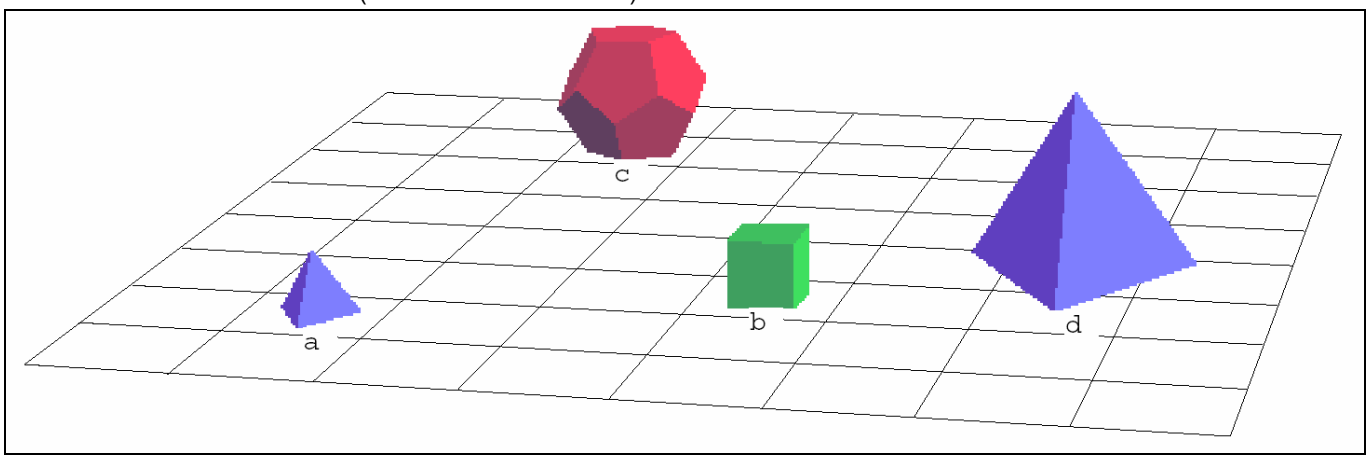
1. **(Completude de F_T)** Se S é uma conseqüência tautológica de $P_1, \ldots, P_n$, então existe uma prova de S a partir das premissas $P_1, \ldots, P_n$ que utiliza apenas as regras de introdução e eliminação para $\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$ e $\perp$.
2. **(Correção de F_T)** Se S não é uma conseqüência tautológica de $P_1, \ldots, P_n$, então não existe prova de S a partir das premissas $P_1, \ldots, P_n$ que utiliza apenas as regras de introdução e eliminação para $\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$ e $\perp$.
3. Note que os itens (1) e (2) acima afirmam que:

S é conseqüência tautológica de $P_1, \ldots, P_n$ *se e somente se* existe uma prova em F_T de S a partir de $P_1, \ldots, P_n$.

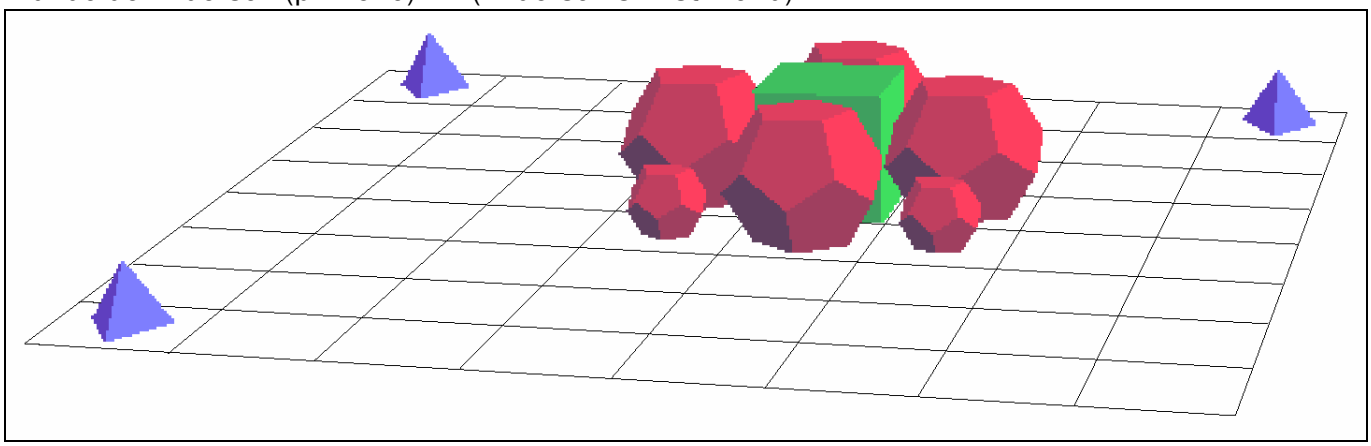
Todos os Mundos
do Programa Mundo de Tarski

Todos os Mundos

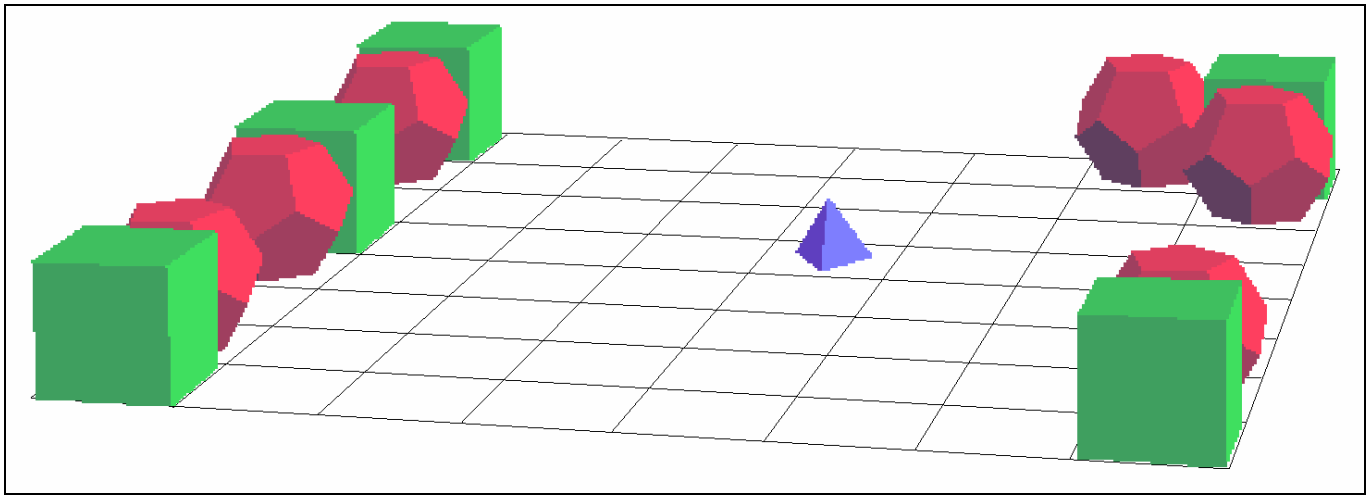
Mundo de Ackermann - (Ackermann's World)



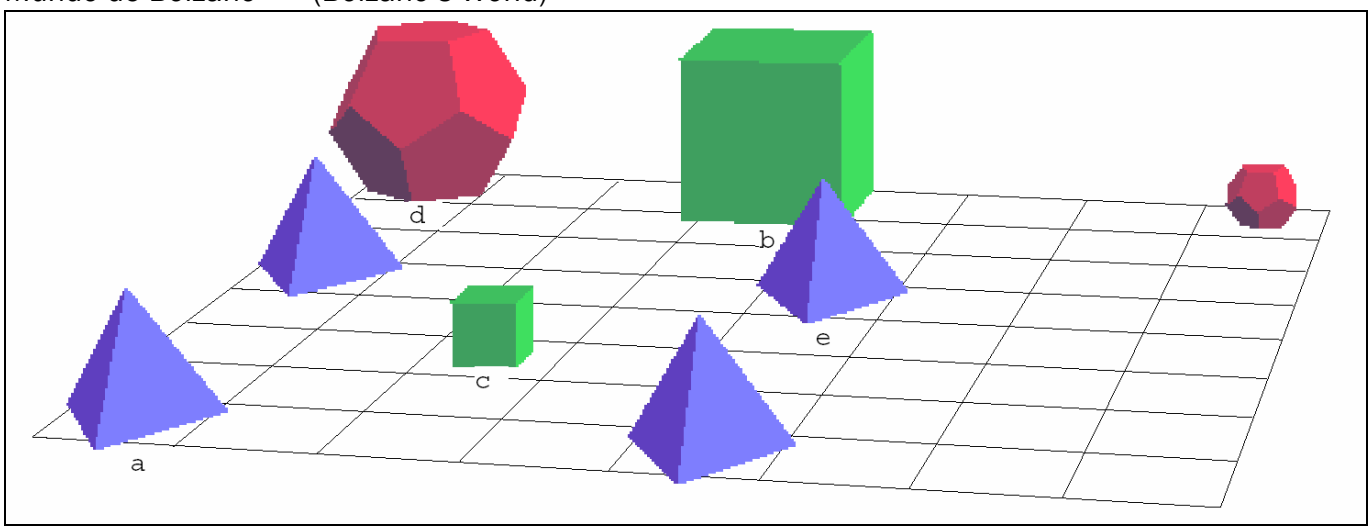
Mundo de Anderson (primeiro) - (Anderson's First World)



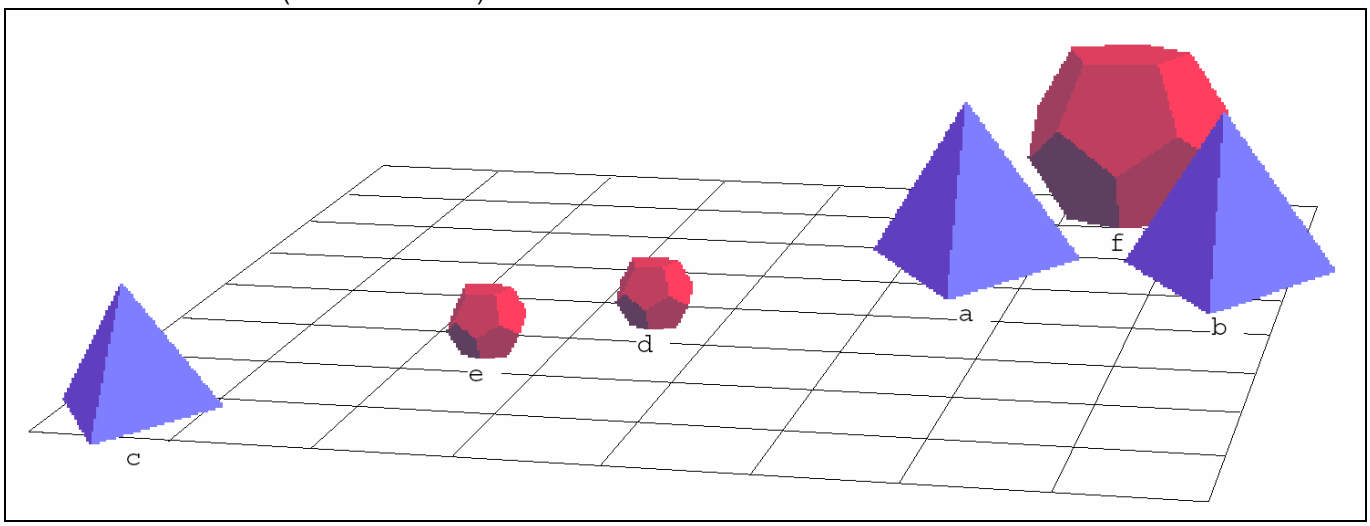
Mundo de Anderson (segundo) - (Anderson's First World)



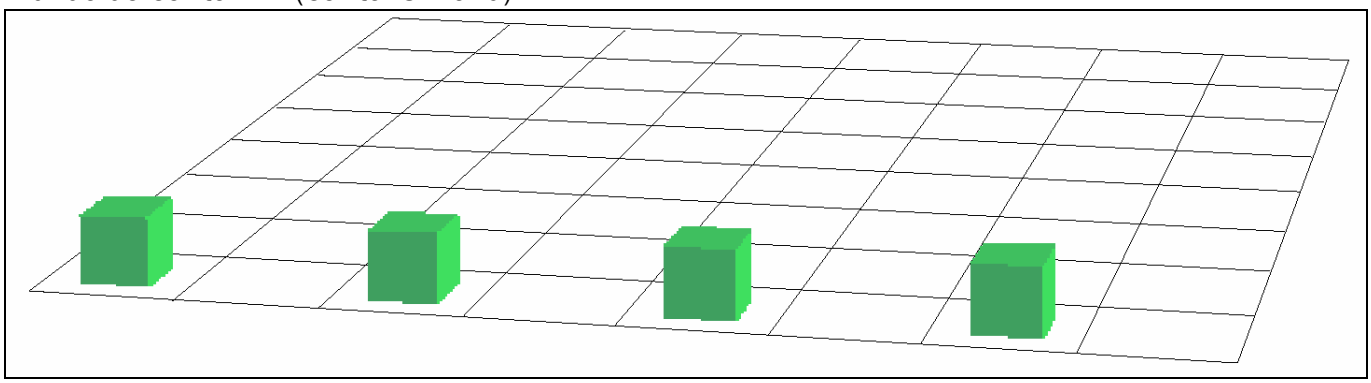
Mundo de Bolzano - (Bolzano's World)



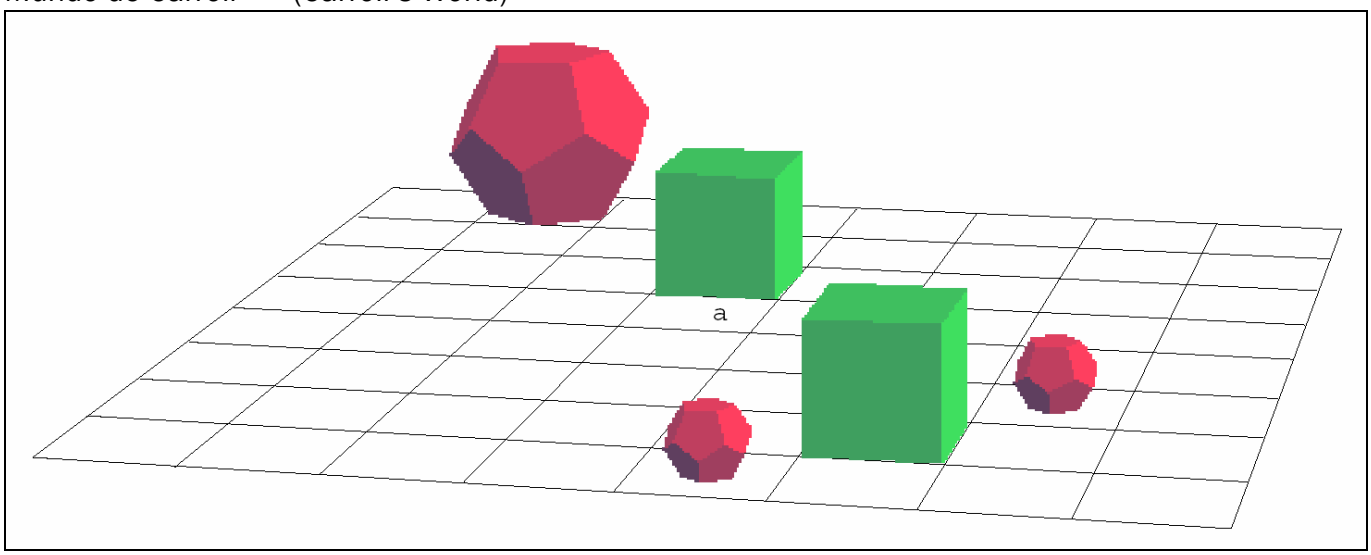
Mundo de Boole - (Boole's World)



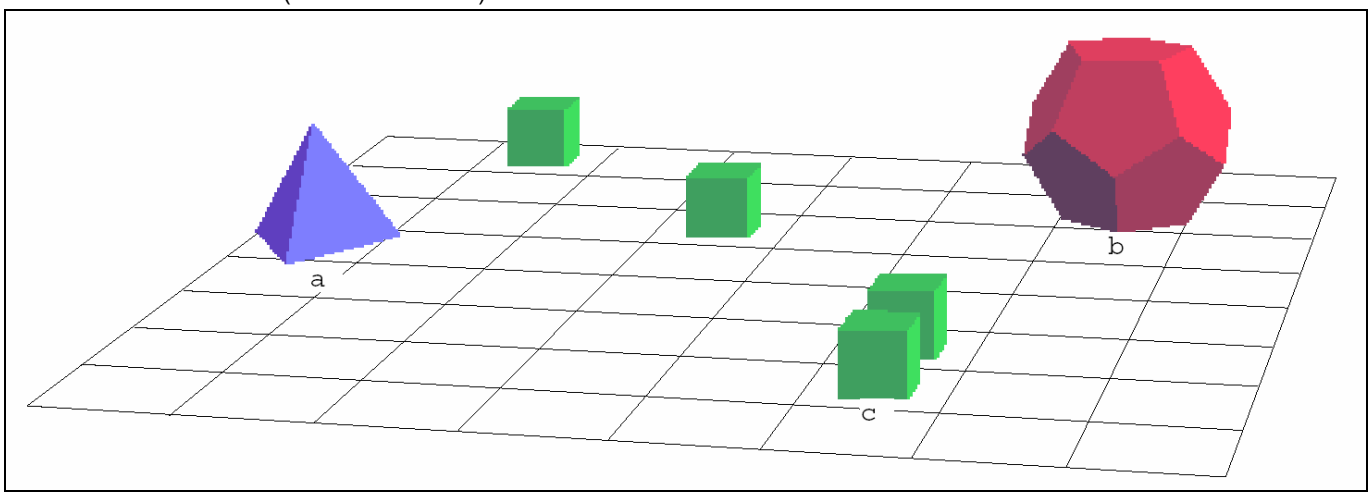
Mundo de Cantor - (Cantor's World)



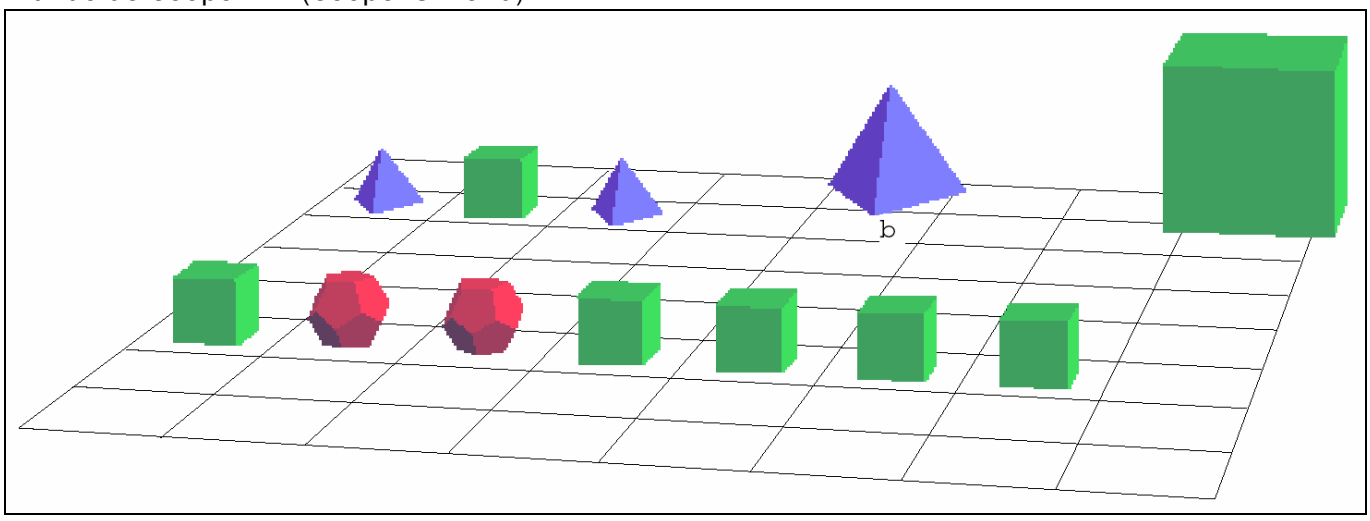
Mundo de Carroll - (Carroll's World)



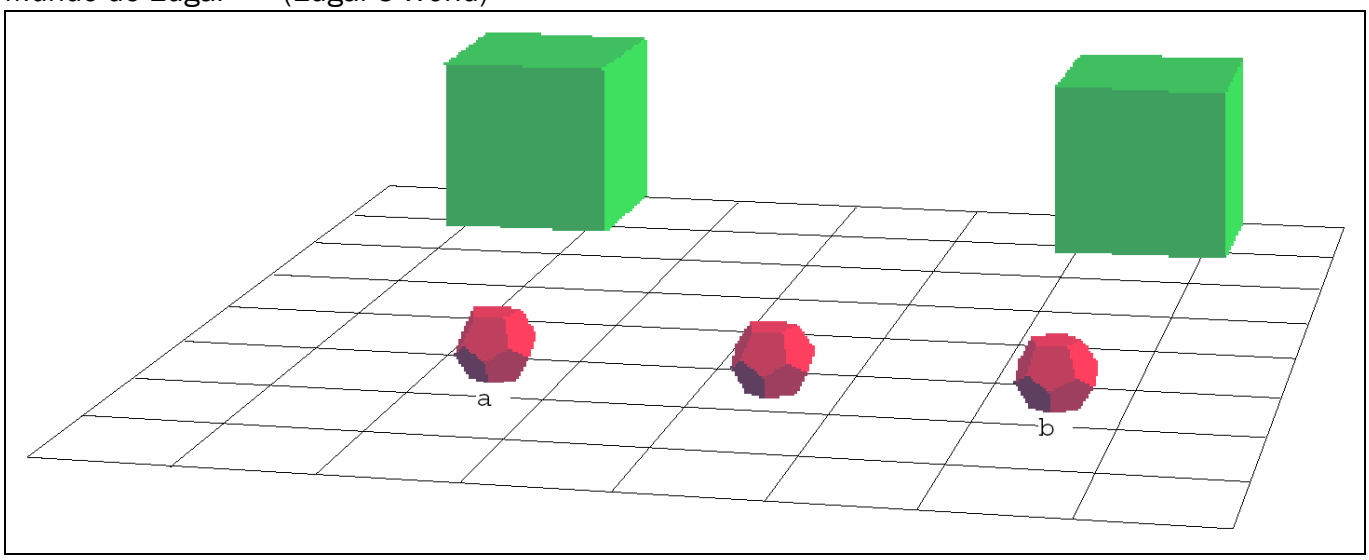
Mundo de Claire - (Claire's World)



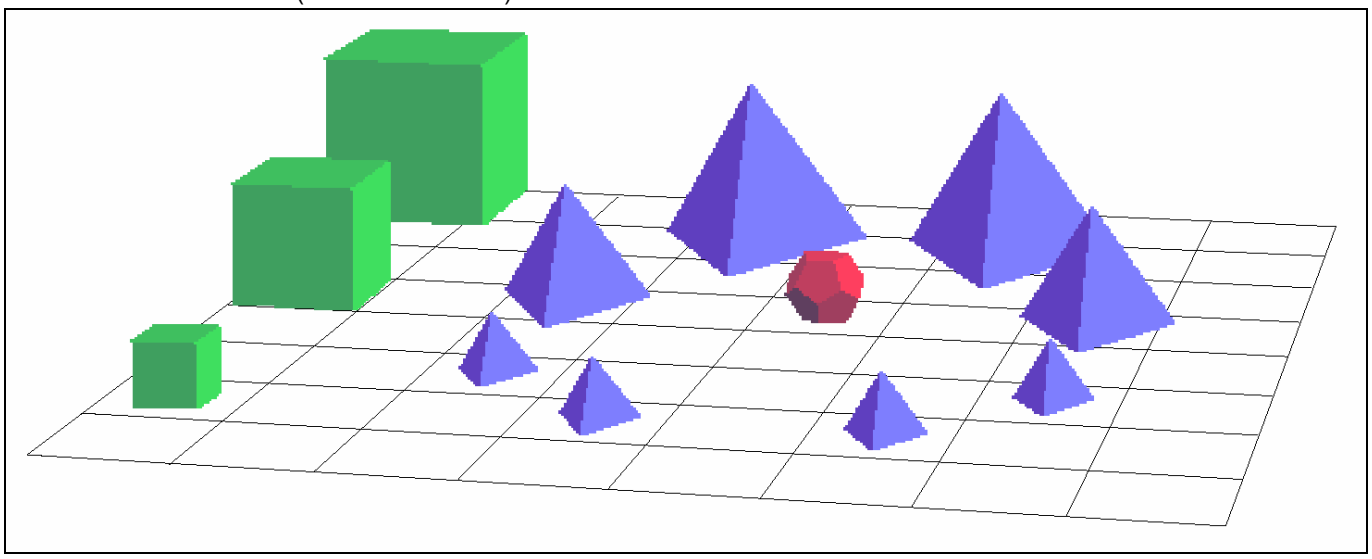
Mundo de Cooper - (Cooper's World)



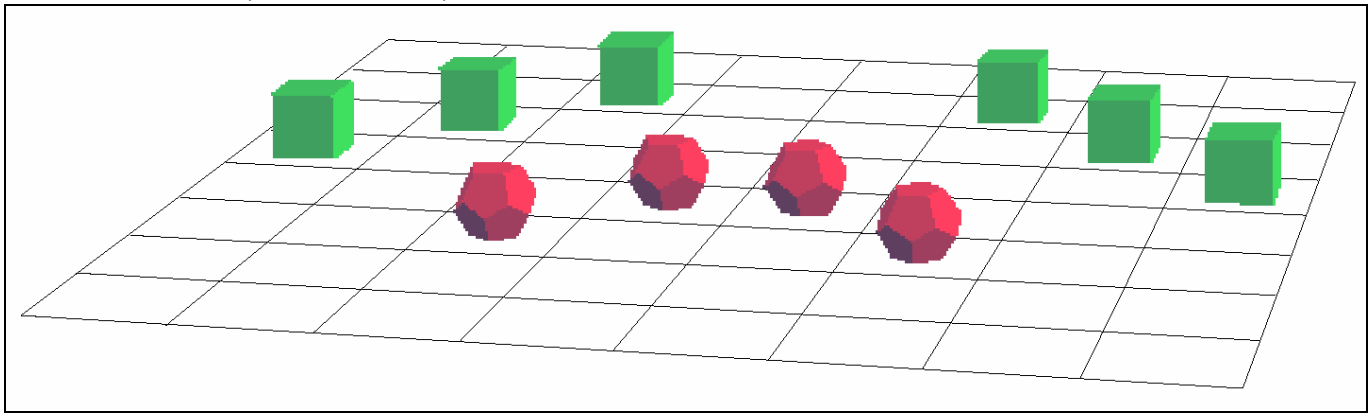
Mundo de Edgar - (Edgar's World)



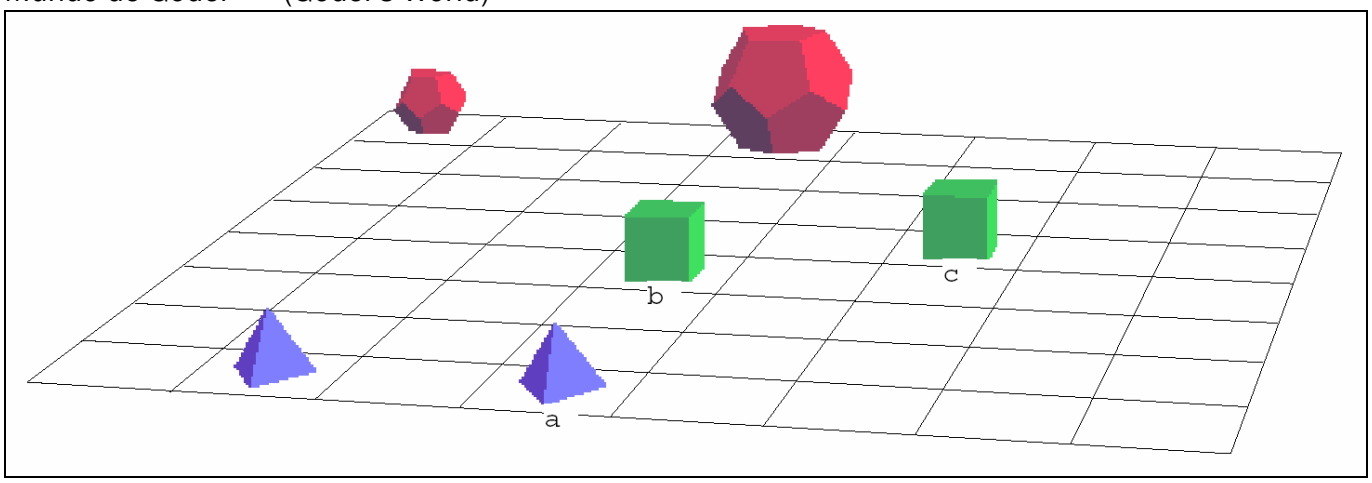
Mundo de Finsler - (Finsler's World)



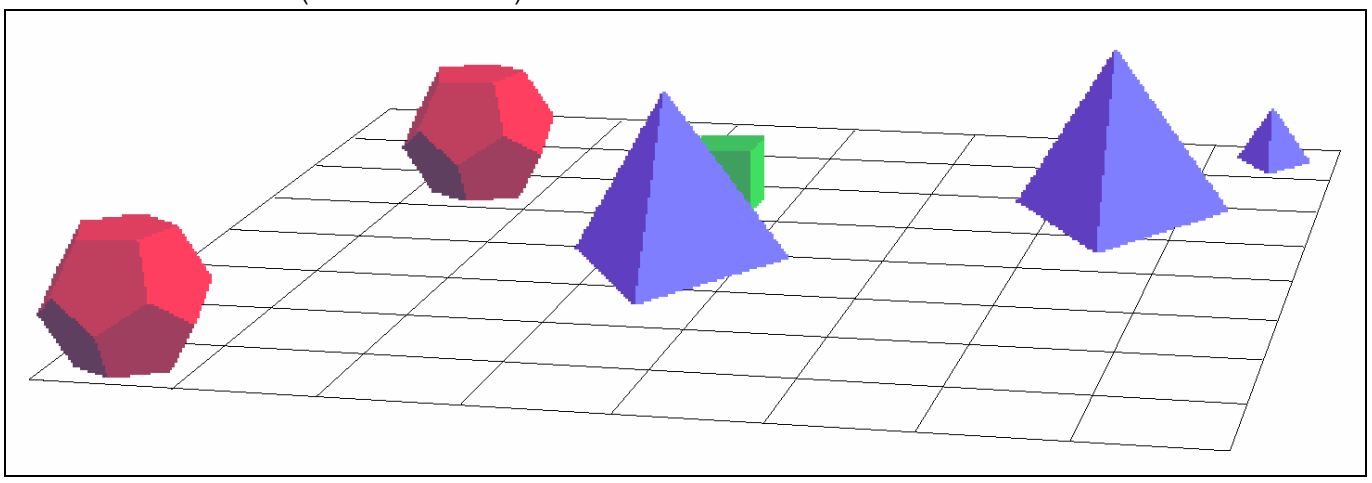
Mundo Game - (Game's World)



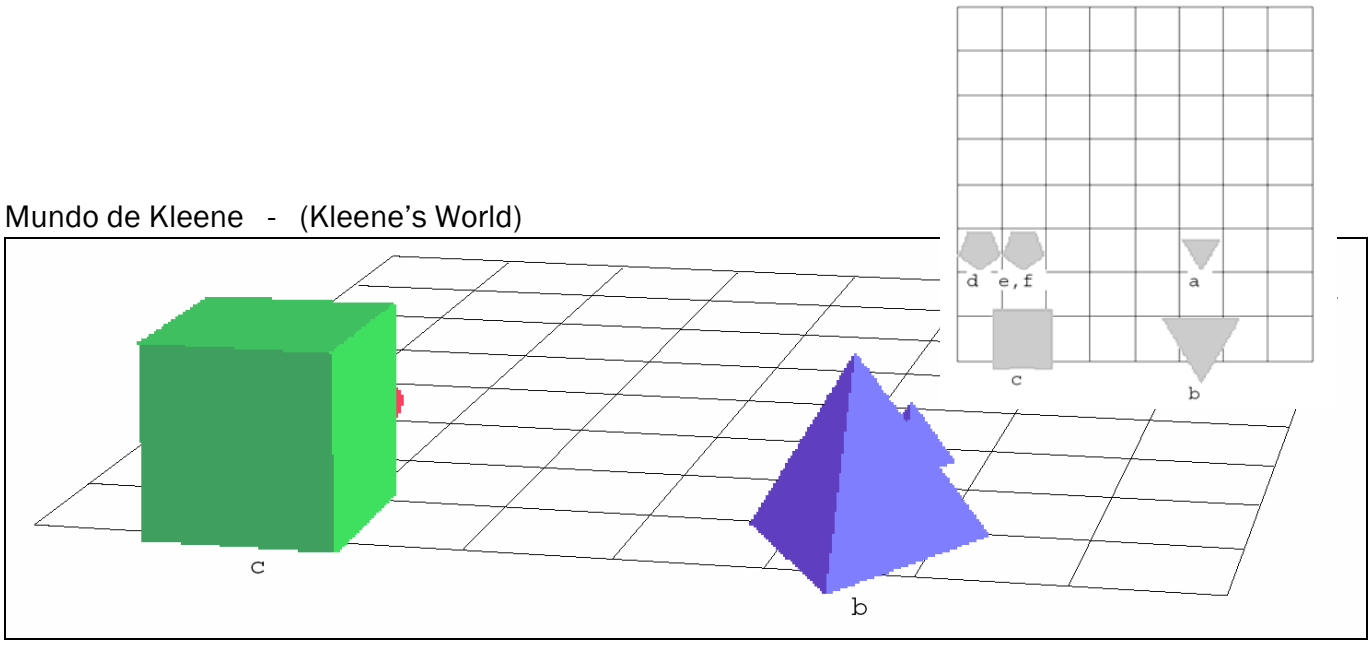
Mundo de Gödel - (Gödel's World)



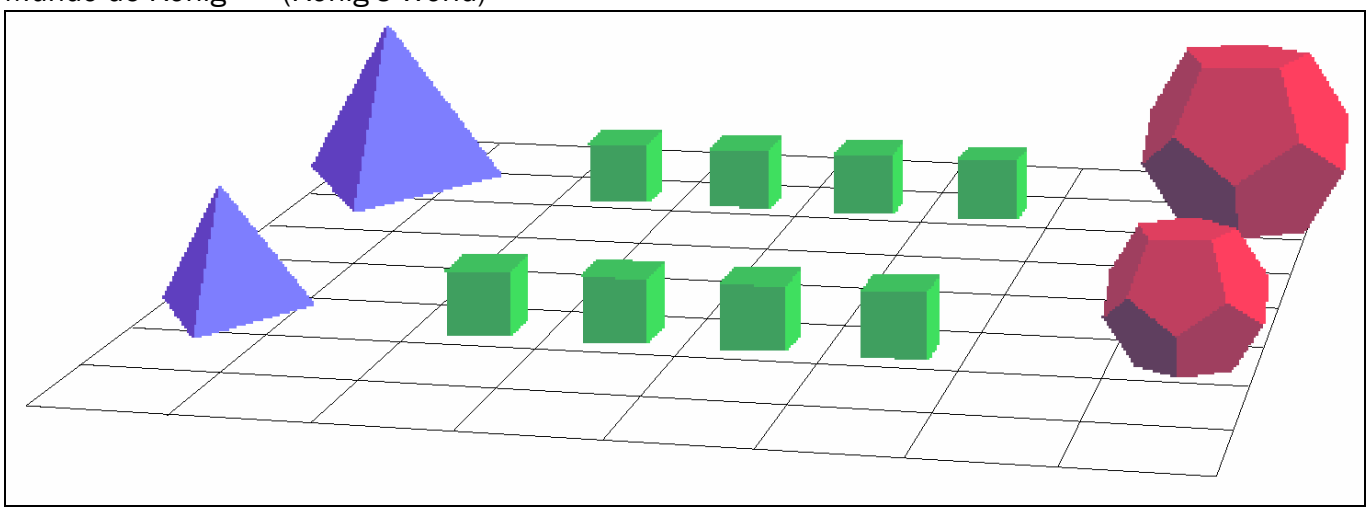
Mundo de Henkin - (Henkin's World)



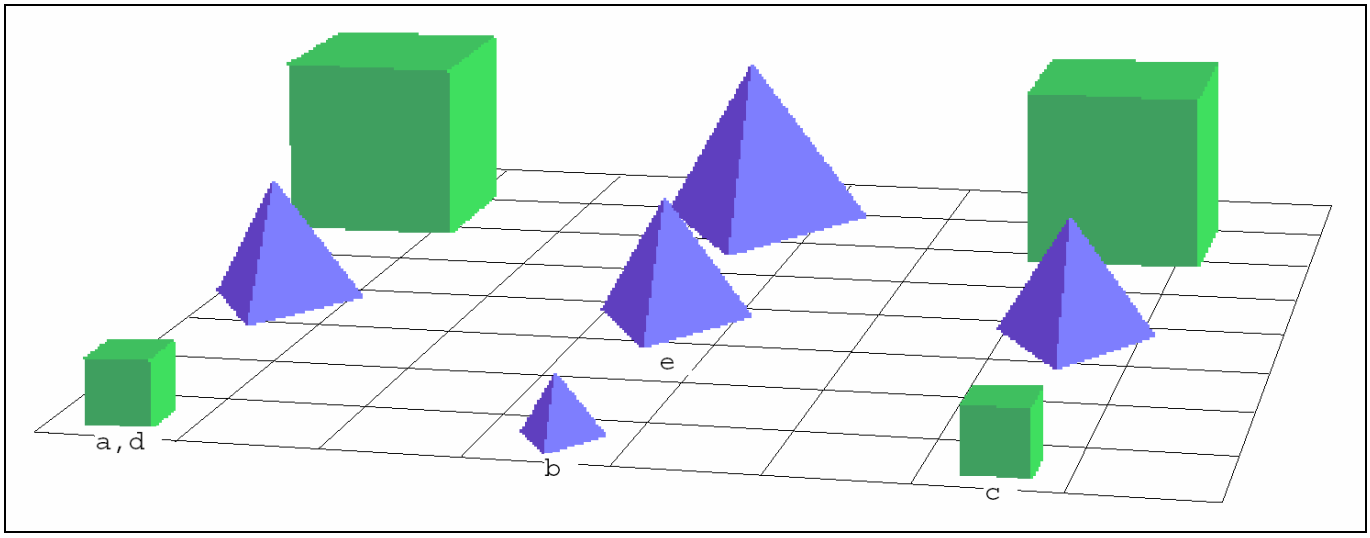
Mundo de Kleene - (Kleene's World)



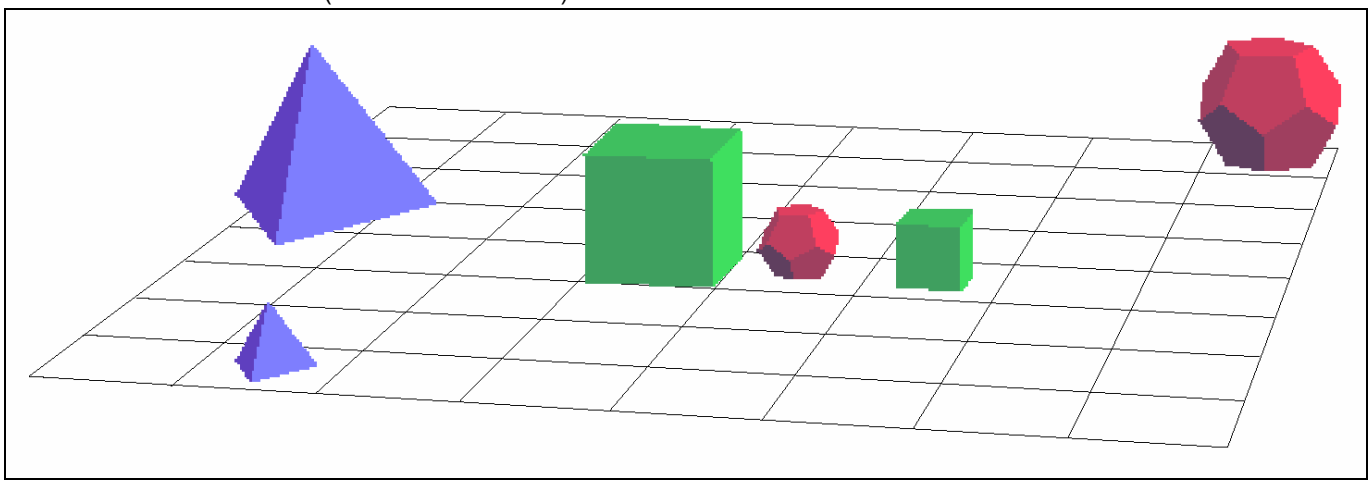
Mundo de König - (König's World)



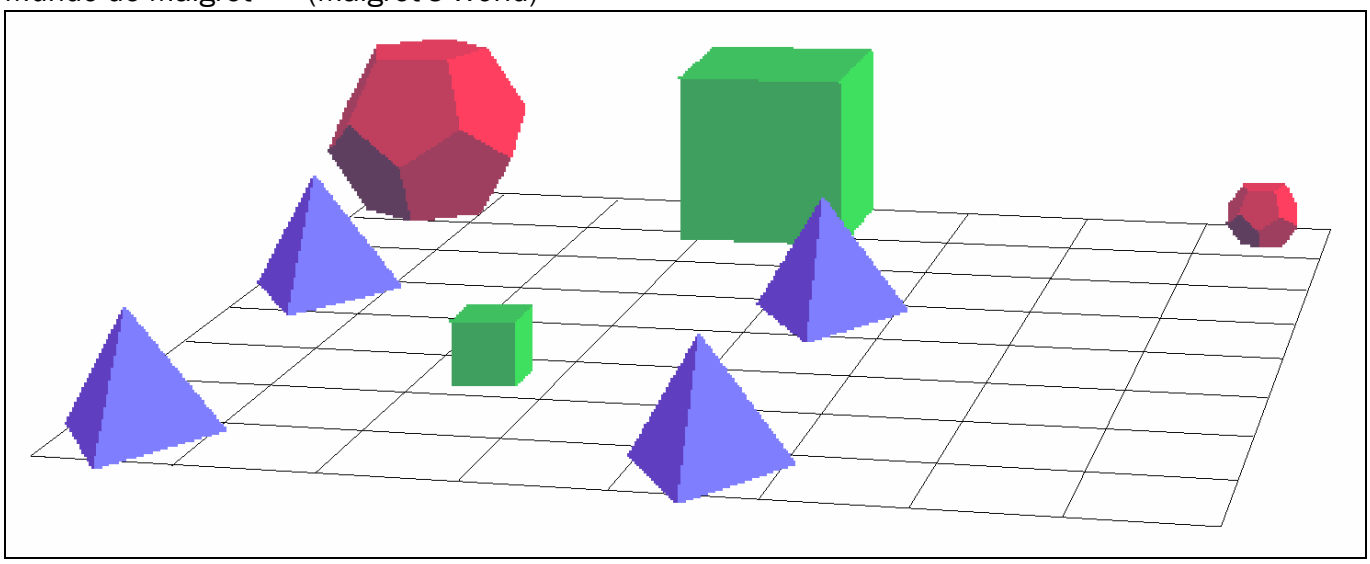
Mundo de Leibniz - (Leibniz's World)



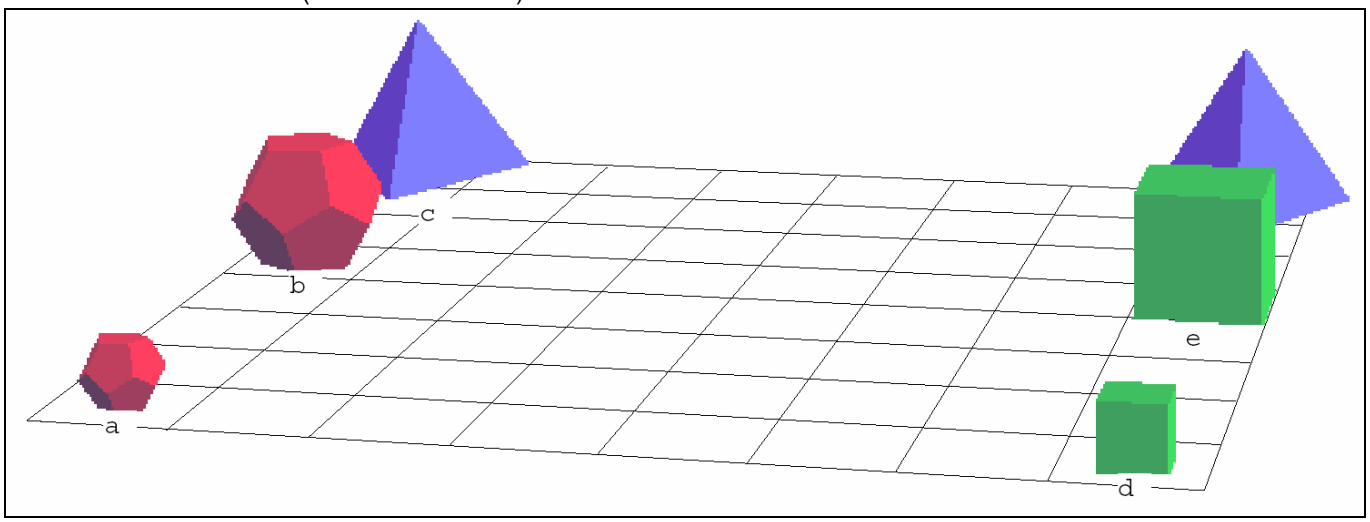
Mundo de Lestrade - (Lestrade's World)



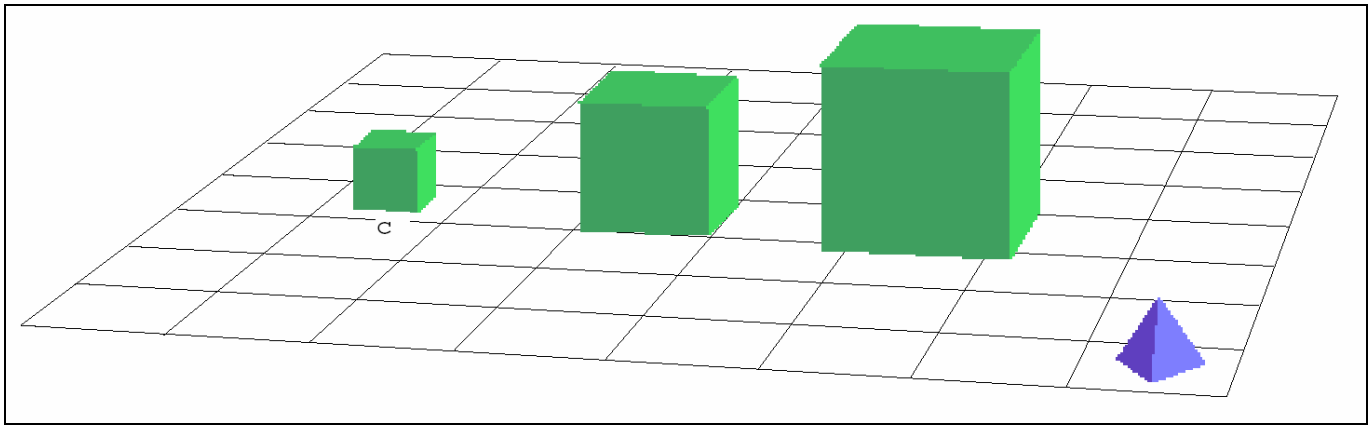
Mundo de Maigret - (Maigret's World)



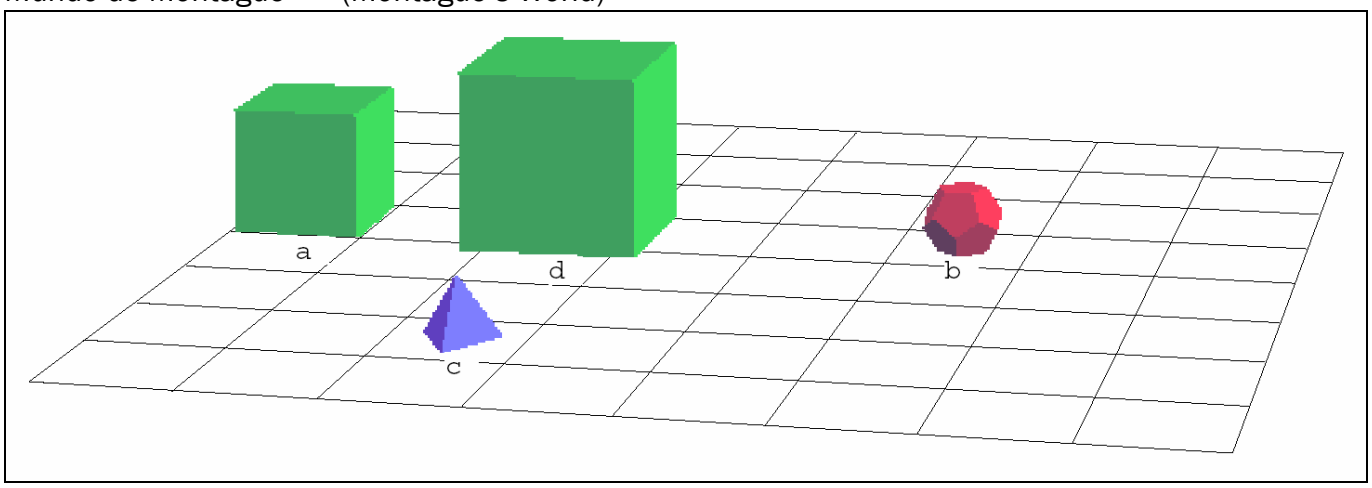
Mundo de Malcev - (Malcev's World)



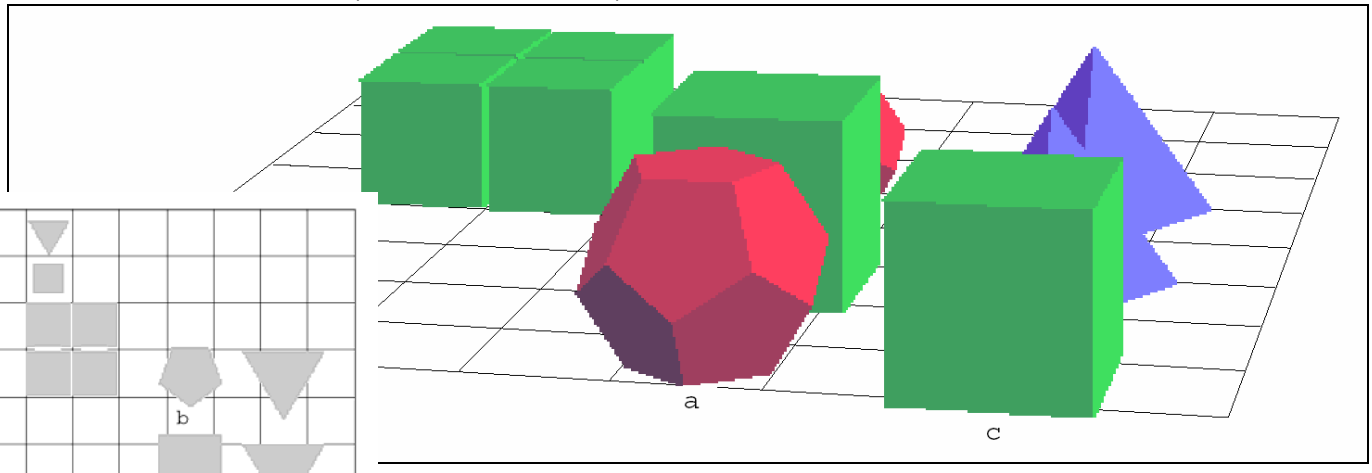
Mundo de Mary Ellen - (Mary Ellen's World)



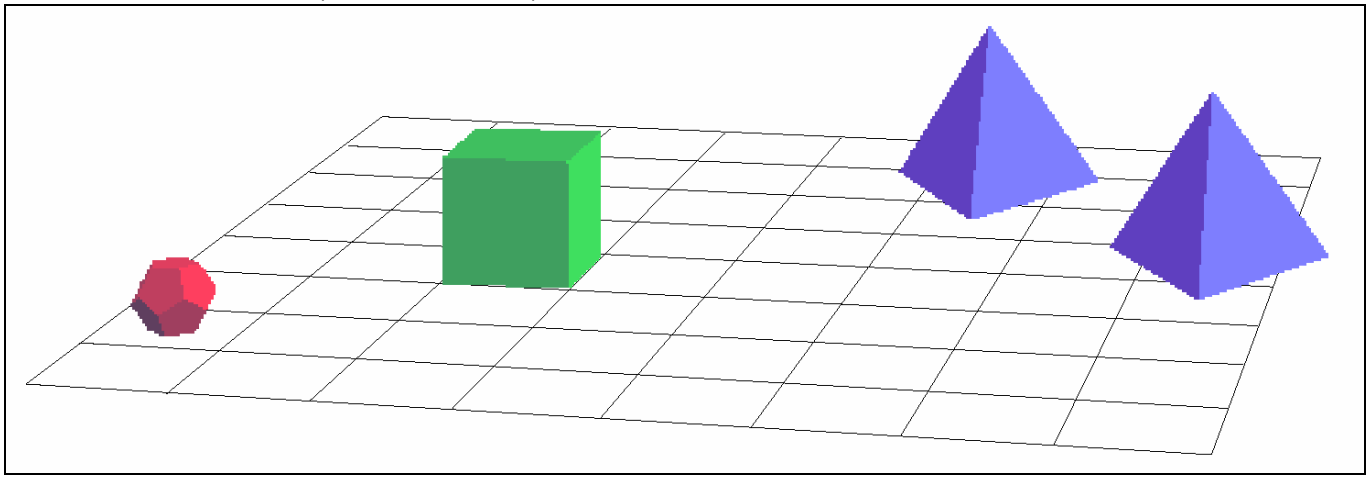
Mundo de Montague - (Montague's World)



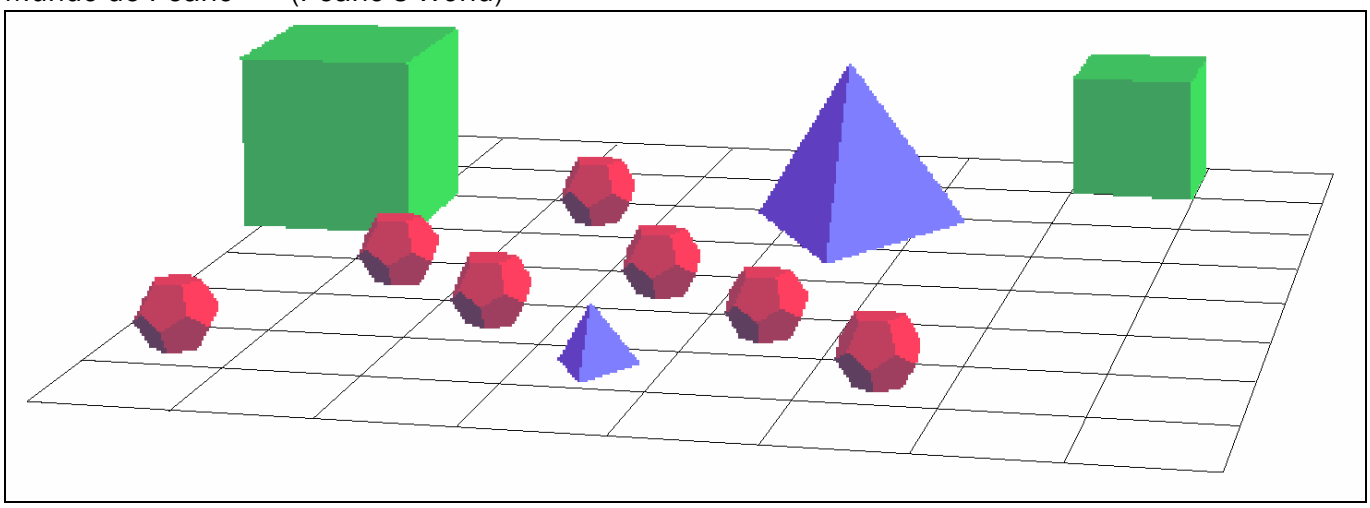
Mundo de Mostowski - (Mostowski'a World)



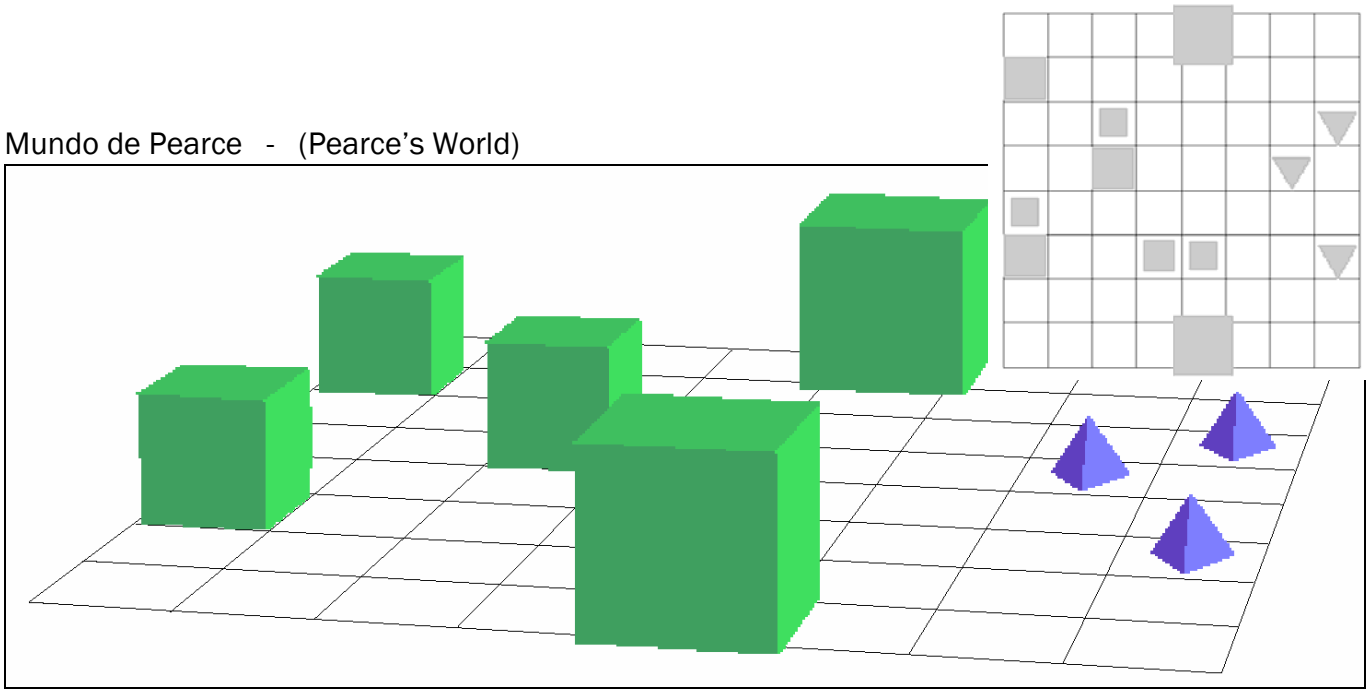
Mundo de Ockham - (Okham's World)



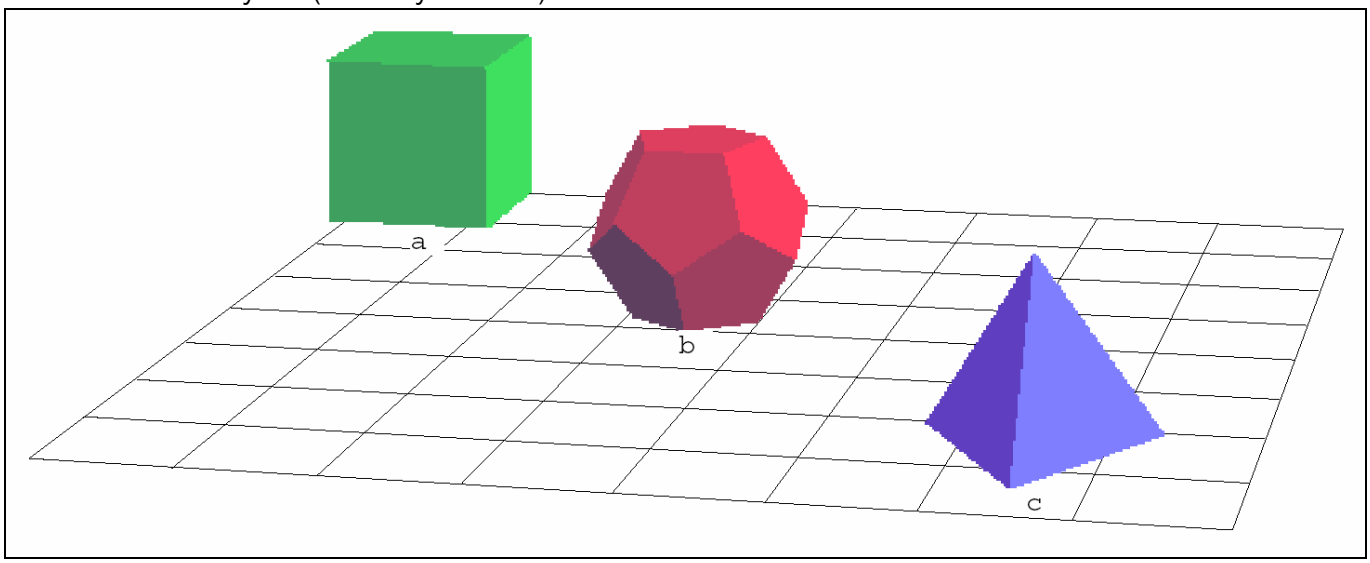
Mundo de Peano - (Peano's World)



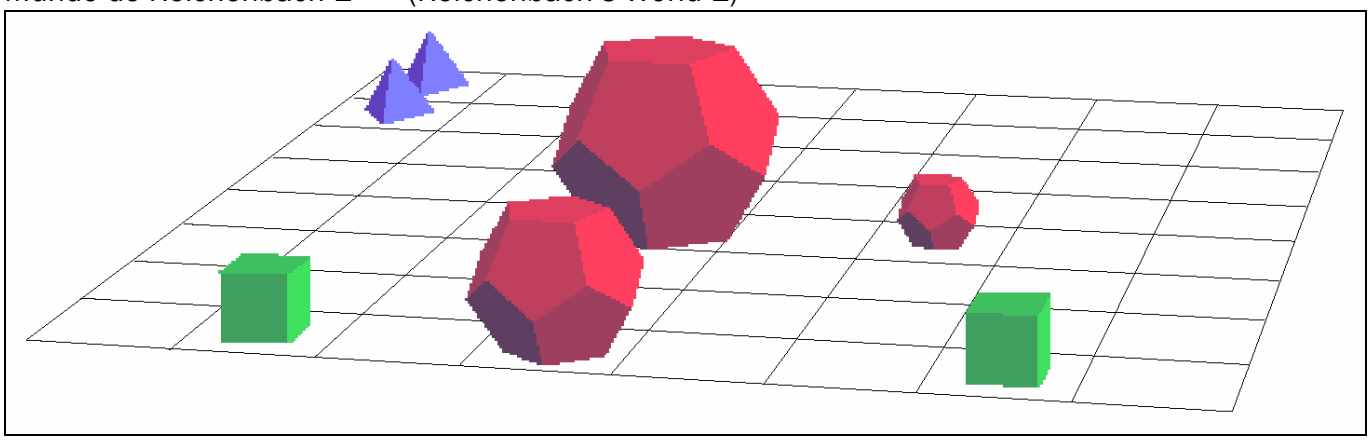
Mundo de Pearce - (Pearce's World)



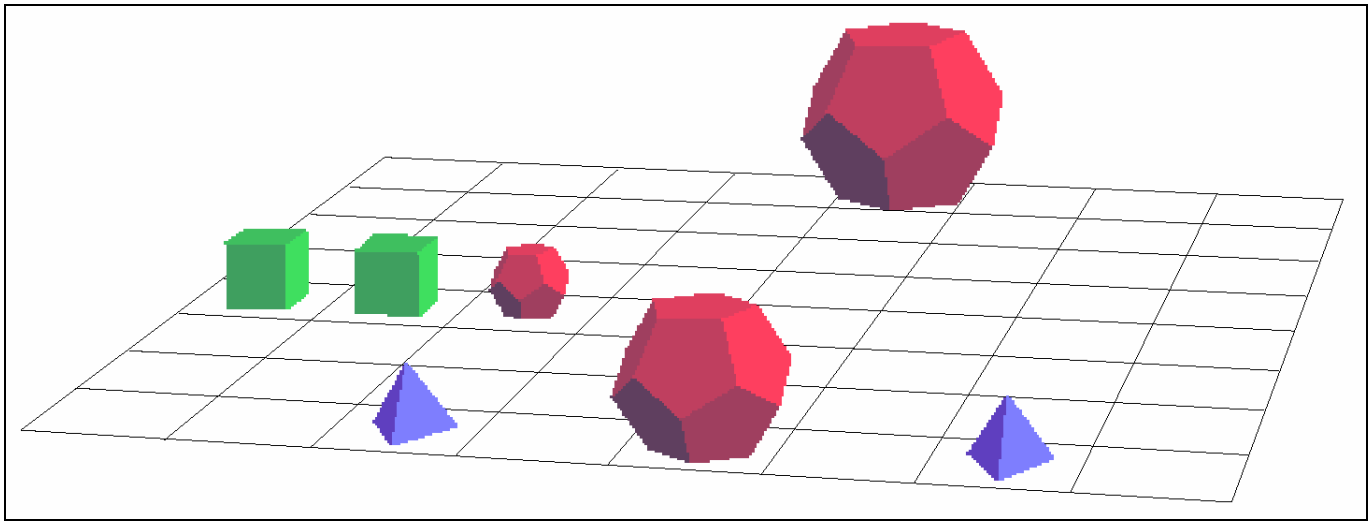
Mundo de Ramsey - (Ramsey's World)



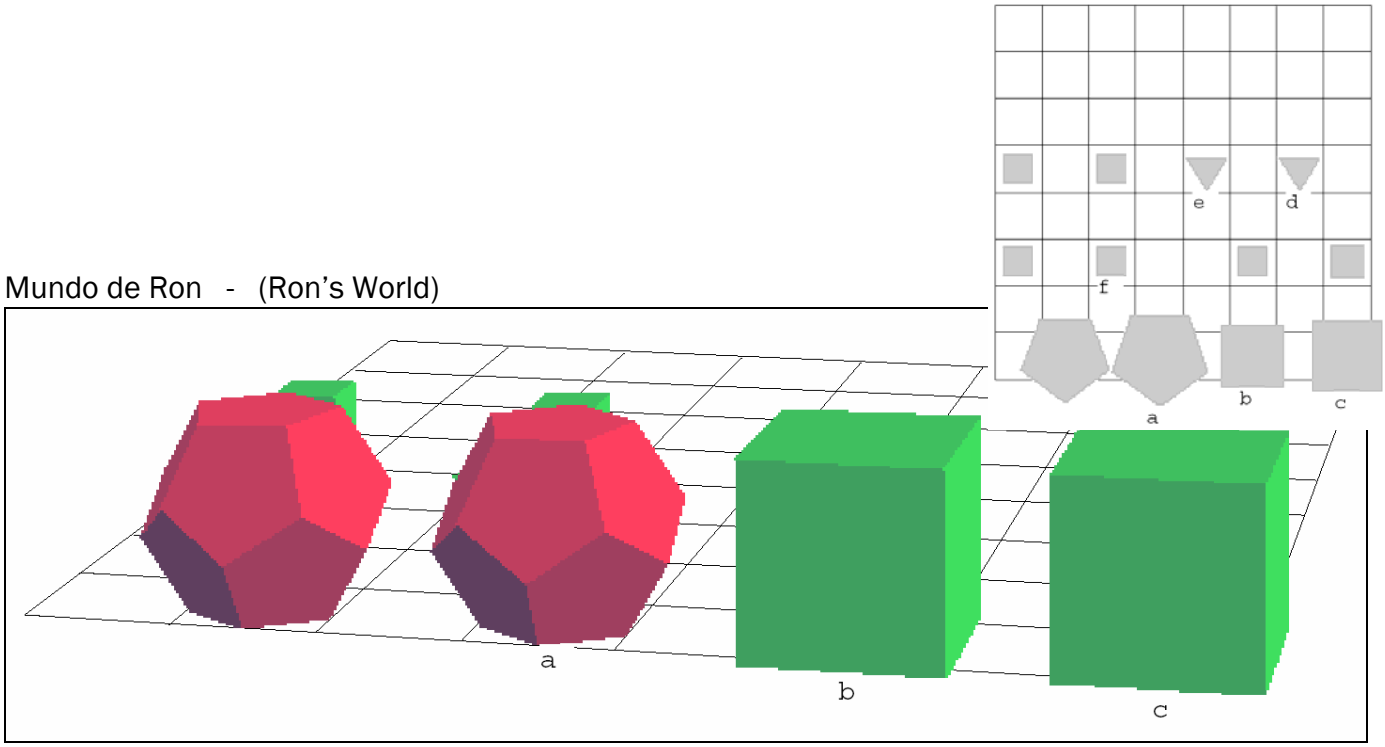
Mundo de Reichenbach 1 - (Reichenbach's World 1)



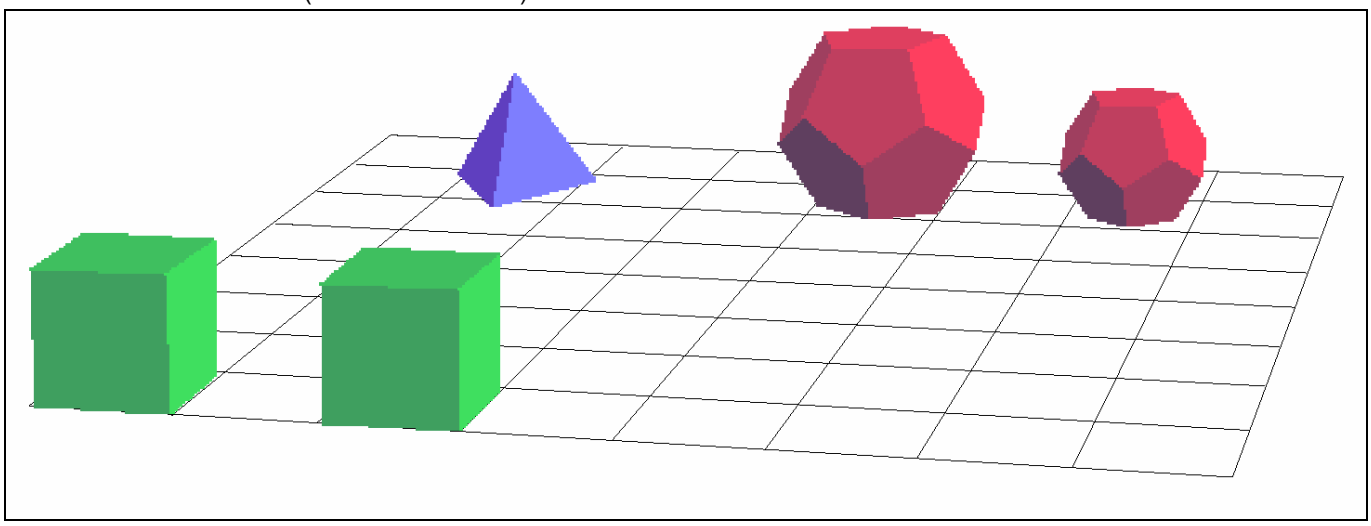
Mundo de Reichenbach 2 - (Reichenbach's World 2)



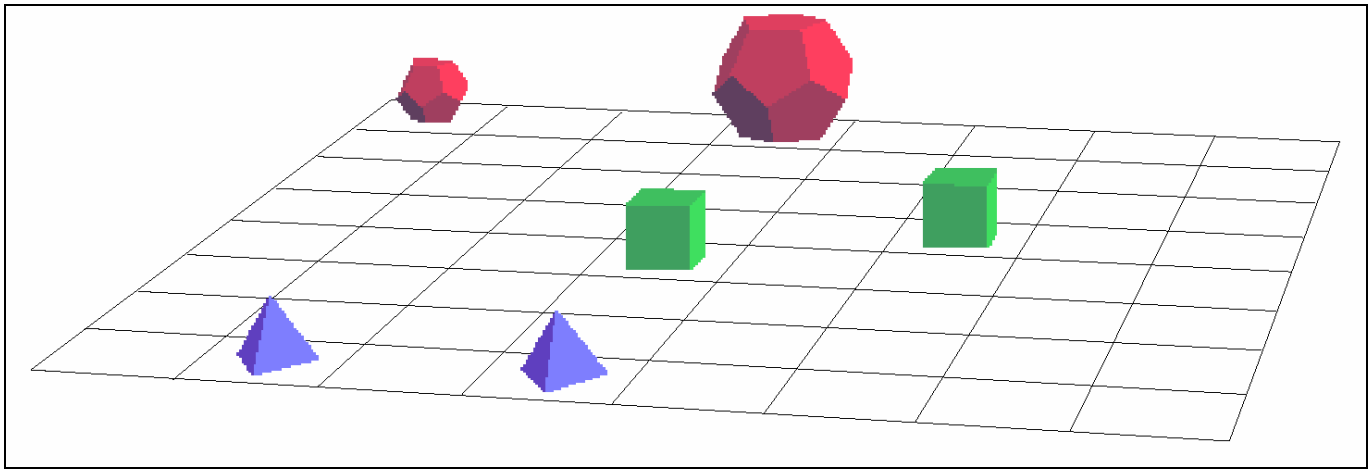
Mundo de Ron - (Ron's World)



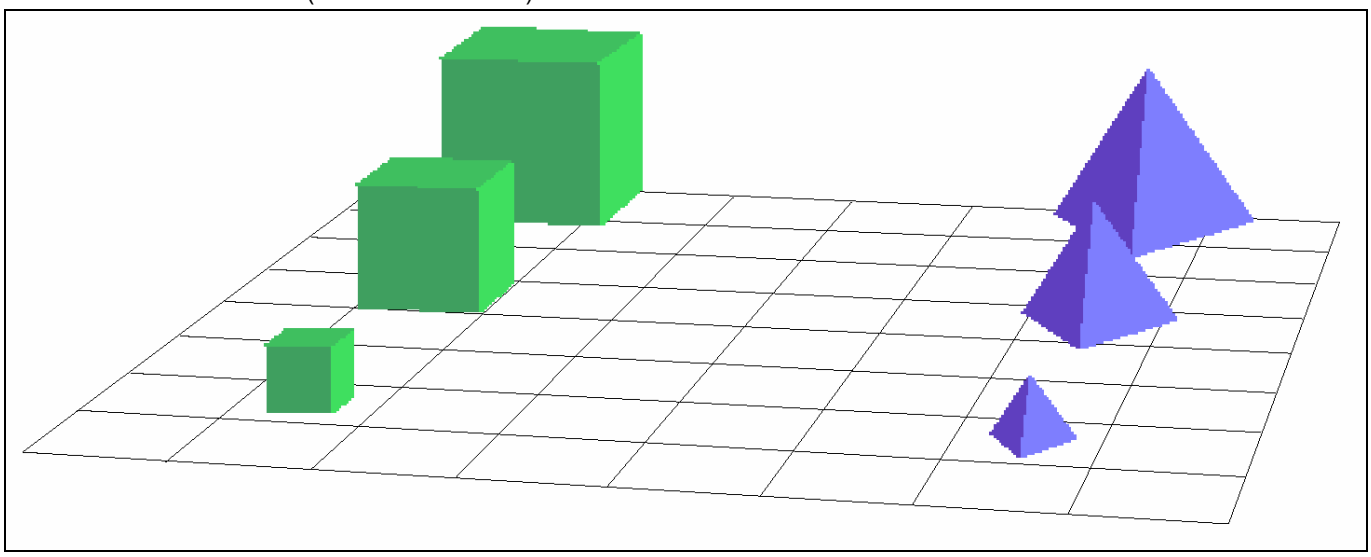
Mundo de Russell - (Russell's World)



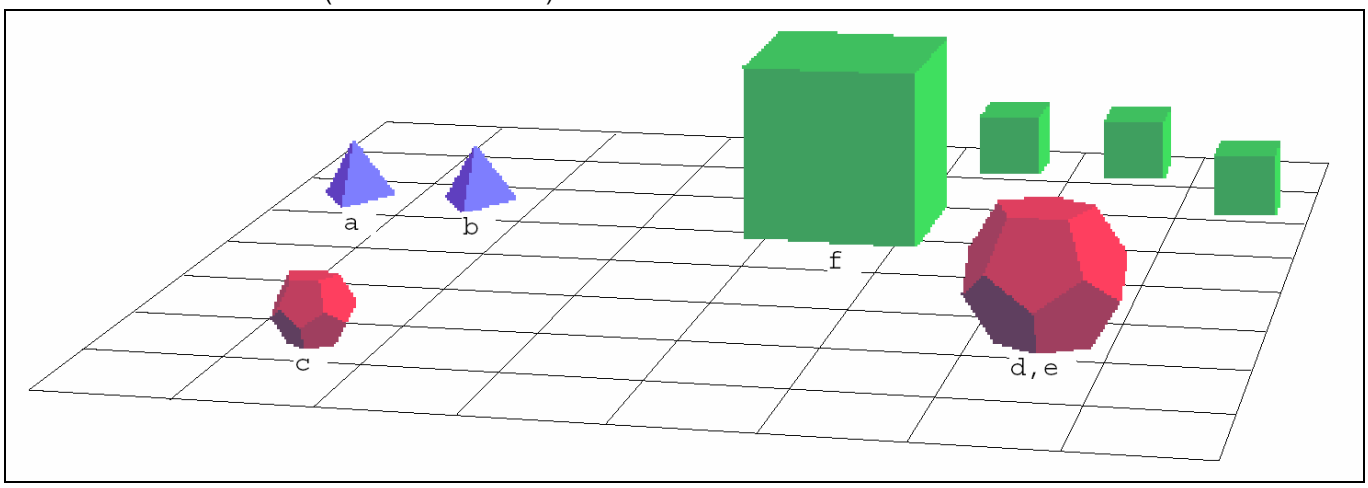
Mundo de Sherlock - (Sherlock's World)



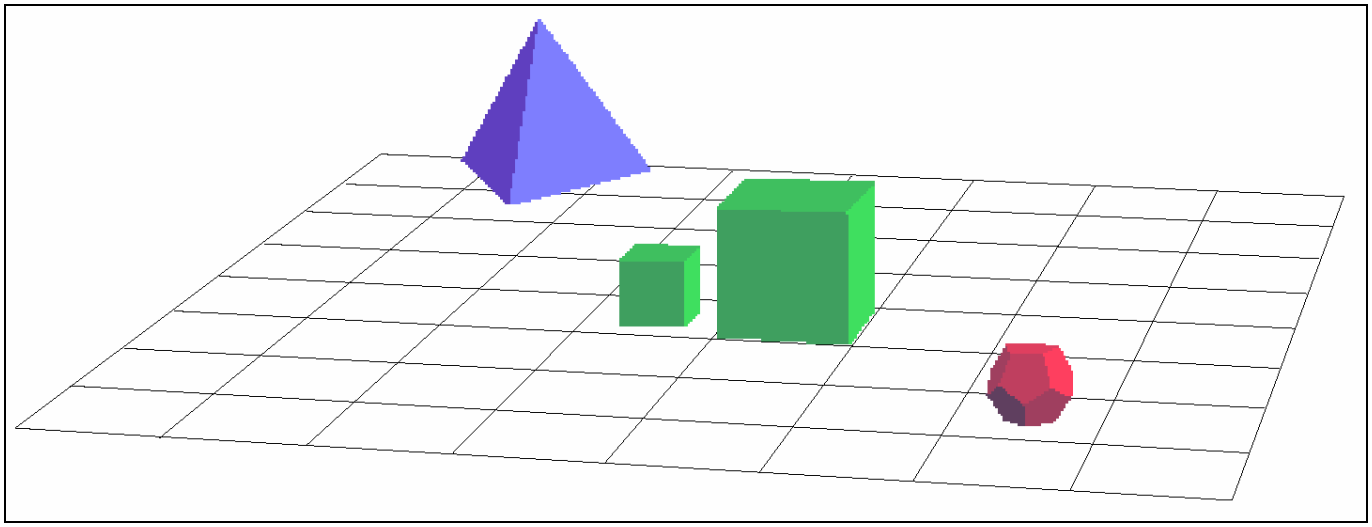
Mundo de Skolem - (Skolem's World)



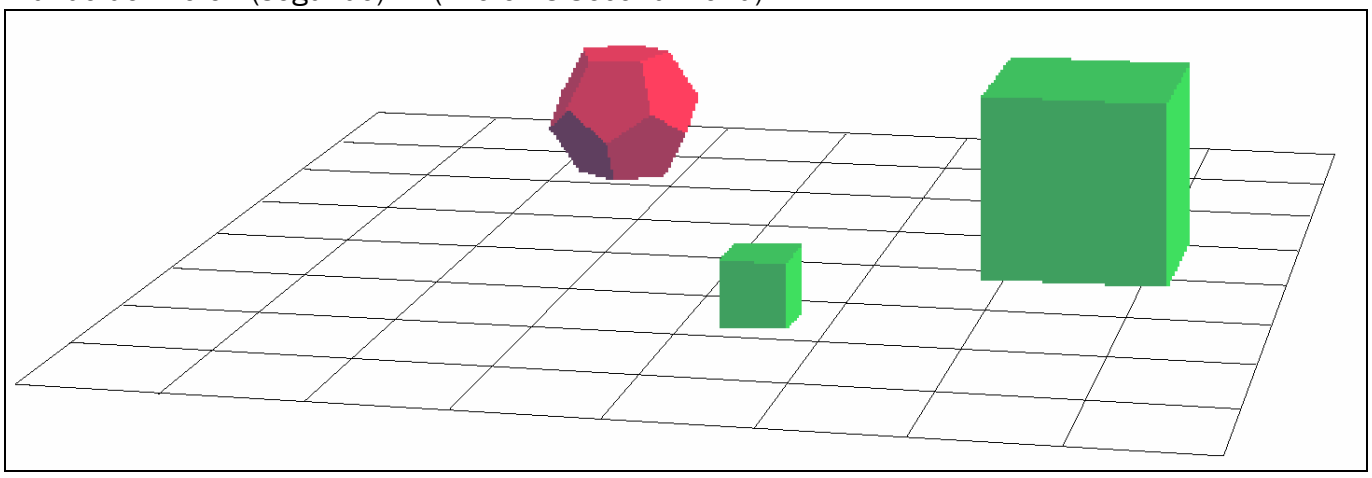
Mundo de Socrates - (Socrates' World)



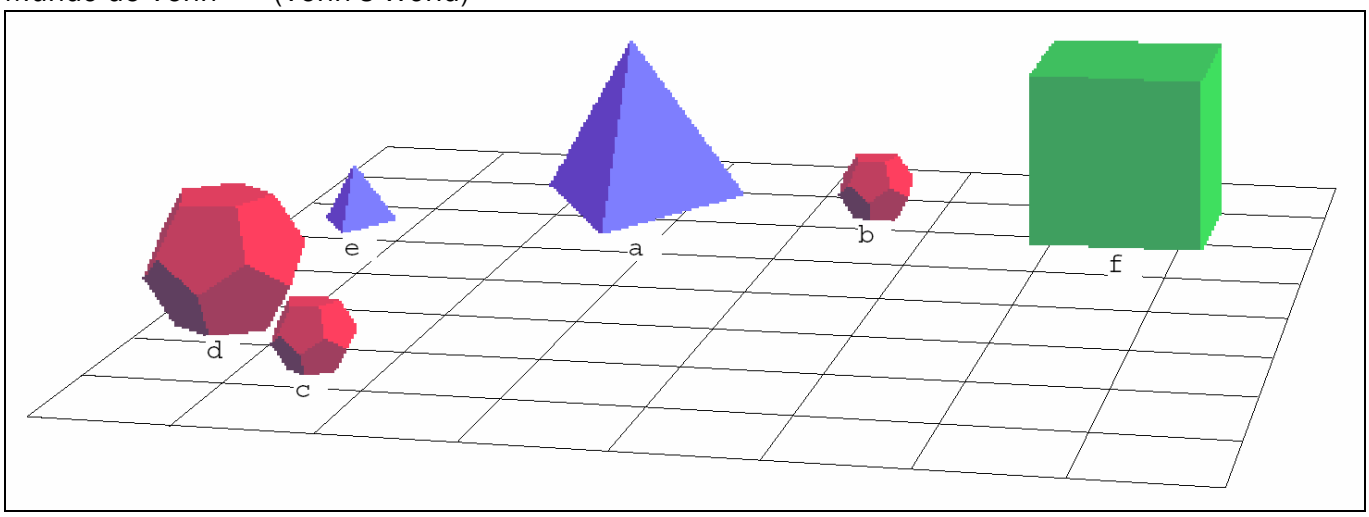
Mundo de Thoralf (primeiro) - (Thoralf's First World)



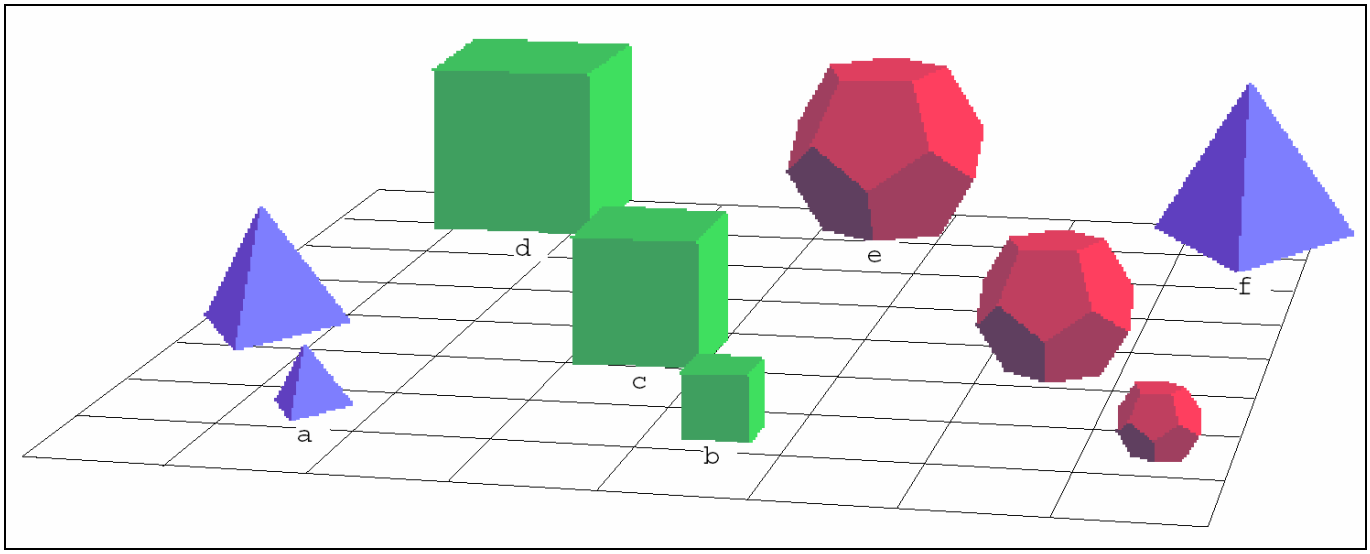
Mundo de Thoralf (segundo) - (Thoralf's Second World)



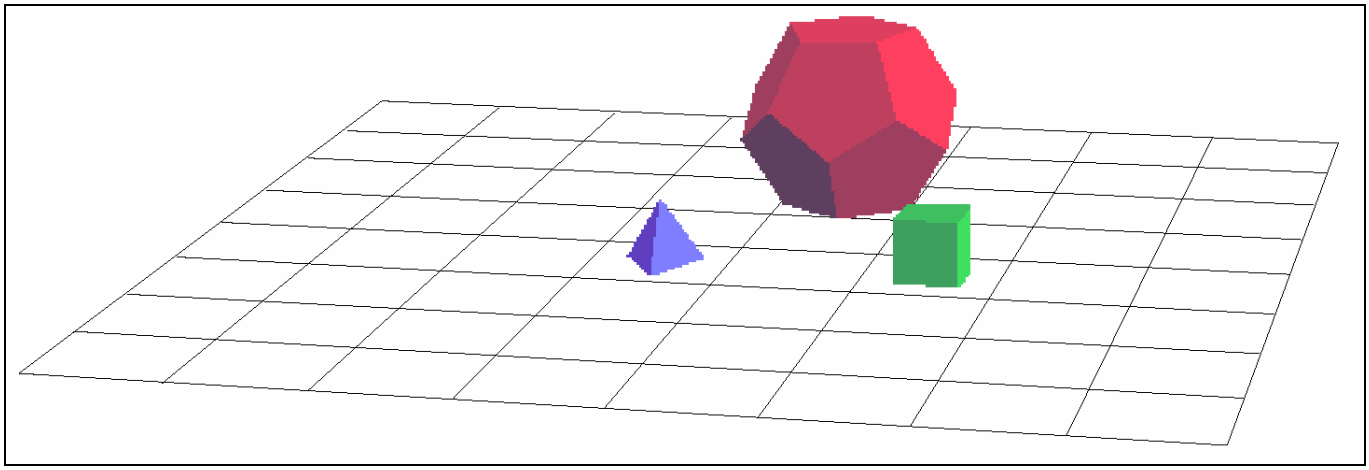
Mundo de Venn - (Venn's World)



Mundo de Wittgenstein - (Wittgenstein's World)



Mundo Submeta-me 1 - (World Submit Me 1)



Tabelas:

a) Predicados da Linguagem dos Blocos

b) Predicados Coloquiais (Tabela 1.2)

Predicados da Linguagem de Blocos		
Sentença atômica	*Interpretação*	*Aridade*
Tet(a)	**a** é um tetraedro	1 propriedades
Cube(a)	**a** é um cubo	
Dodec(a)	**a** é um dodecaedro	
Small(a)	**a** é pequeno	
Medium(a)	**a** é médio	
Large(a)	**a** é grande	
SameSize(a, b)	**a** tem o mesmo tamanho que **b**	2 relações binárias
SameShape(a, b)	**a** tem o mesmo formato que **b**	
Larger(a, b)	**a** é maior que **b**	
Smaller(a, b)	**a** é menor que **b**	
SameCol(a, b)	**a** está na mesma coluna que **b**	
SameRow(a, b)	**a** está na mesma linha que **b**	
Adjoins(a, b)	**a** e **b** estão em quadros adjacentes (não diagonais) da grade	
LeftOf(a, b)	**a** está mais próximo da borda esquerda que **b**	
RightOf(a, b)	**a** está mais próximo da borda direita que **b**	
FrontOf(a, b)	**a** está mais próximo da borda da frente que **b**	
BackOf(a, b)	**a** está mais próximo da borda de traz que **b**	
a = b	**a** e **b** nomeiam o mesmo objeto	
Between(a, b, c)	**a**, **b** e **c** estão na mesma linha, coluna, ou diagonal, e **a** está entre **b** e **c**	3 - relação ternária

Português	FOL	Comentário
Nomes:		
Max	**max**	
Claire	**claire**	
Folly	**folly**	O nome de um certo cachorro.
Carl	**carl**	O nome de outro cachorro.
Scruffy	**scruffy**	O nome de um certo gato.
Pris	**pris**	O nome de outro gato.
2:00 pm, 2 de janeiro de 2005	**2:00**	O nome de um instante do tempo.
2:01 pm, 2 de janeiro de 2005	**2:01**	Um minuto mais tarde
:	**:**	Similarmente para outros tempos
Predicados:		
x é um animal de estimação	**Pet(x)**	
x é uma pessoa	**Person(x)**	
x é um estudante	**Student(x)**	
t é mais cedo que t'	**t < t'**	Mais cedo que, para instantes de tempo
x estava com fome no instante t	**Hungry(x, t)**	
x estava brava no instante t	**Angry(x, t)**	
x possuia y no instante t	**Owned(x, y, t)**	
x deu y a z no instante t	**Gave(x, y, z, t)**	
x alimentou y no instante t	**Fed(x, y, t)**	

Tabela 1.2: Nomes e predicados de uma linguagem

Regras do Sisitema *F*

Regras do Sistema Formal F de Lógica Clássica

Introdução

(∧Intro)	(∨Intro)	(¬Intro)	(⊥Intro)	(→Intro)	(↔Intro)	(=Intro)	(∃Intro)	(∀Intro) *	(∀Intro2) *
P_1 ⇓ P_n : $P_1 \wedge \ldots \wedge P_n$	P_i : $P_1 \vee \ldots \vee P_i \vee \ldots \vee P_n$	P : ⊥ ¬P	P : ¬P : ⊥	P : Q P→Q	P : Q Q : P P ↔ Q	n=n	S(c) : ∃xS(x)	c : P(c) ∀xP(x)	c P(c) : Q(c) ∀x(P(x)→Q(x))

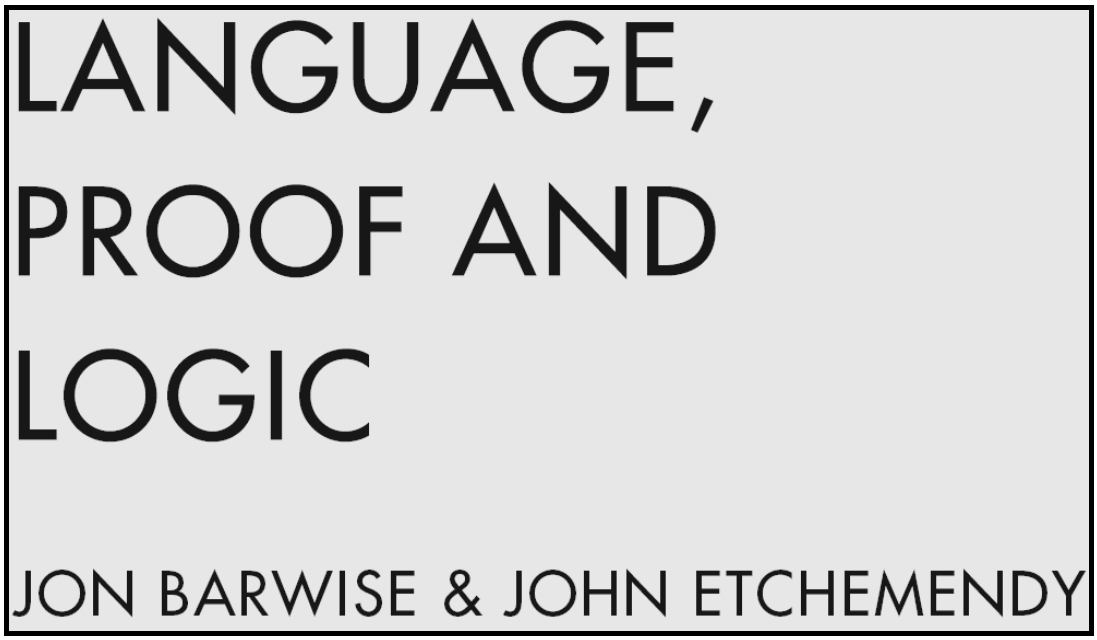
Eliminação

(∧Elim)	(∨Elim)	(¬Elim)	(⊥Elim)	(→Elim)	(↔Elim)	(=Elim)	(∃Elim) *	(∀Elim)	(Reit)
$P_1 \wedge \ldots \wedge P_i \wedge \ldots \wedge P_n$: P_i	$P_1 \vee \ldots \vee P_n$: P_1 : S ⇓ P_n : S : S	¬¬P : P	⊥ : P	P→Q : P : Q	P↔Q (ou Q↔P) : P : Q	P(n) : n = m : P(m)	∃xS(x) : c S(c) : Q Q	∀xS(x) : S(c)	P : P

(Taut Con)	Não é uma regra geral da lógica. Representa apenas um teste rápido para saber se a sentença em questão é uma conseqüência tautológica das sentenças citadas. Quando for (o que poderia ser verificado em uma tabela de verdade conjunta) a checagem da regra no programa Fitch resulta OK. Quando não for, a checagem resulta em erro.
(FO Con)	Não é uma regra geral da lógica. Ela depende do significado do predicado =. Ela representa apenas um teste para saber se a sentença em questão é conseqüência lógica das sentenças citadas (em virtude do significado dos conectivos verofuncionais, quantificadores e =). Como tal teste não é um procedimento mecanicamente decidível, como o teste para conseqüências tautológicas, esta regra pode se enganar em alguns casos.
(Ana Con)	Não é uma regra geral da lógica. Ela depende do significado dos predicados que estamos utilizando. Para os predicados da linguagem de blocos acima, ela pode ser utilizada para justificar uma conclusão baseada no fato de que certos predicados são simétricos, transitivos, reflexivos, inversos,... Por exemplo, de Larger(a, b) e Larger(b, c), é claro que Larger(a, c), pois Larger (maior que) é uma relação transitiva. A regra (Ana Con) é utilizada para justificar este tipo de passo em uma prova.
(*)	Restrição à aplicação das regras (∀Intro), (∀Intro2) e (∃Elim). ➔ A constante **c** não deve ocorrer fora da subprova em que é introduzida.

Lógica - EXERCÍCIOS

MÓDULO 2



Tradução Resumida da 1ª Parte

1. Tradução dos Exercícios dos Capítulos 1 a 8

2. Questões Importantes (Conceituais e Gerais)

3. Todos as Sentenças (do programa Mundo de Tarski)

4. Alguns Cabeçalhos de Provas das Seções Experimente

5. Diagramas para Desenho de Mundos

Todos os Exercícios da Parte I

Exercícios do Capítulo 1 – (Sentenças Atômicas)

Experimente . . . . . .

(1) Está na hora de testar-se usando o **Mundo de Tarski**. Neste exercício, você usará o Mundo de Tarski para familiarizar-se com as interpretações das sentenças atômicas da linguagem de blocos. Antes de começar, entretanto, você precisa aprender como iniciar o Mundo de Tarski e executar algumas operações básicas. Leia as seções apropriadas do manual do usuário que descreve o Mundo de Tarski antes de continuar.

(2) Inicie o **Mundo de Tarski** e abra os arquivos **Wittgenstein's World** (Mundo de Wittgenstein) e **Wittgenstein's Sentences** (Sentenças de Wittgenstein). Você vai encontrá-los na pasta de **TW Exercice Files**. Nestes arquivos, você verá alguns blocos no mundo e uma lista de sentenças atômicas. (Nós acrescentamos comentários a algumas das sentenças. Comentários são prefaciados por um ponto-e-vírgula (**;**) que diz para o Mundo de Tarski ignorar o resto da linha.)

(3) Mova-se pelas sentenças usando as teclas de seta em seu teclado, enquanto avalia o valor de verdade de cada sentença mentalmente no mundo determinado. Use o botão ***Verify*** para conferir suas avaliações. (Como as sentenças são todas sentenças atômicas, o botão de ***Game*** não será útil.) Se você estiver surpreso com quaisquer das avaliações, tente descobrir porque sua interpretação diverge da correta.

(4) Em seguida, modifique o Mundo de Wittgenstein de muitos modos de diferentes, vendo o que acontece com o valor de verdade das várias sentenças. O ponto principal disto é o ajudá-lo a descobrir como o Mundo de Tarski interpreta os vários predicados. Por exemplo, o que torna **BackOf(d,c)** falso? Duas coisas têm que estar na mesma coluna para uma estar atrás (BackOf) da outra?

(5) Brinque com o mundo e as sentenças tanto quanto você precisar até que esteja seguro de ter entendido os significados das sentenças atômicas deste arquivo. Por exemplo, no mundo original, nenhuma das sentenças com o predicado **Adjoins** é verdadeira. Você deveria tentar modificar o mundo para fazer algumas delas verdadeiras. Quando fizer isso, você notará que blocos grandes não podem ficar juntos de outros blocos.

(6) Fazendo este exercício, notará você indubitavelmente que **Between** (entre) não significa exatamente o mesmo que em seu uso na linguagem natural. Isto se deve à necessidade de interpretar **Between** como um predicado específico. Para simplicidade, insistimos nós que para que **b** esteja entre **c** e **d**, todos os três devem estar na mesma linha, coluna, ou diagonal.

(7) Quando você terminar, não salve as alterações que fez nos arquivos Wittgenstein.

Exercícios. . . . . .

(1.1) Não deixe de fazer a seção **Experimente** acima. Os exercícios destas seções são em geral fáceis mas cruciais. Neste caso a idéia é você se familiarizar com as sentenças atômicas da linguagem dos blocos. Se estiver usando o computador, siga rigorosamente as instruções acima. Caso esteja fazendo os exercícios manualmente, abra as apostilas de mundos e sentenças nas páginas do **Mundo de Wittgenstein** e das **Sentenças de Wittgenstein** e responda quais destas sentenças que são verdadeiras neste mundo.

(1.2) (Copiando algumas sentenças atômicas) Este exercício lhe dará alguma prática com a janela ***Keyboard*** do programa **Mundo do Tarski**, como também com a sintaxe de sentenças atômicas. Na lista abaixo há apenas sentenças atômicas de nossa linguagem de blocos. Inicie uma nova janela de sentenças e insira nela a lista abaixo. Use o botão ***Verify*** em cada fórmula, depois que você inseri-la, para confirmar que é uma sentença. Se você cometer um erro, corrija-o antes de continuar. Certifique-se de utilizar o comando ***Add Sentence*** entre as sentenças, e não a tecla ***ENTER***. Se você fizer isto corretamente, as sentenças serão numeradas e separadas por traços horizontais.

1. **Tet(a)**
2. **Medium(a)**
3. **Dodec(b)**
4. **Cube(c)**
5. **FrontOf(a, b)**
6. **Between(a, b, c)**
7. **a = d**
8. **Larger(a, b)**
9. **Smaller(a, c)**
10. **LeftOf(b, c)**

Lembre-se de gravar estas sentenças em um arquivo com o nome **Sentenças 1.2**.

(1.3) (Construindo um mundo) Construa um mundo no qual todas as sentenças do Exercício 1.2 sejam simultaneamente verdadeiras. Lembre-se de gravar este mundo com o nome "Mundo 1.3"

(1.4) (Traduzindo sentenças atômicas) Aqui estão algumas sentenças simples do português. Inicie um arquivo de sentenças novo e traduza cada uma delas em FOL.

*1. **a** é um cubo.*
*2. **b** é menor que **a**.*
*3. **c** está entre **a** e **d**.*
*4. **d** é grande.*
*5. **e** é maior que **a**.*
*6. **b** é um tetraedro.*
*7. **e** é um dodecaedro.*
*8. **e** está à direita de **b**.*
*9. **a** é menor que **e**.*
*10. **d** está atrás de **a**.*
*11. **b** está na mesma linha que **d**.*
*12. **b** é do mesmo tamanho que **c**.*

Depois de traduzir as sentenças, construa um mundo no qual todas suas traduções sejam verdades. Grave seu trabalho com os nomes **Sentenças 1.4** e **Mundo 1.4**.

(1.5) (Nomeando objetos) Abra os arquivos **Lestrade's Sentences** (Sentenças de Lestrade) e **Lestrade's World** (Mundo de Lestrade). Você notará que nenhum dos objetos neste mundo tem um nome. Sua tarefa é nomear os objetos de tal modo que todas as sentenças da lista se tornem verdadeiras. Lembre-se de gravar sua solução no arquivo **Mundo 1.5**. (use o comando ***Save World As…*** e não ***Save World***, para não modificar o arquivo Mundo de Lestrade).

(1.6)* (Nomeando objetos, continuação) As escolhas que você fez no Exercício 1.5 não foram impostas a você. Quer dizer, você poderia ter nomeado diferentemente os objetos e ainda assim tornar as sentenças verdadeiras. Faça outra atribuição de nomes, diferente da que fez no exercício anterior, em que as sentenças sejam também verdadeiras. Grave seu trabalho no arquivo **Mundo 1.6**.

(1.7) (Predicados são sensíveis a contexto) Enfatizamos o fato de que FOL assume que todo predicado é interpretado por uma relação determinada, enquanto que este não é o caso em linguagens naturais como português. Realmente, até mesmo quando as coisas parecerem totalmente determinadas, há freqüentemente alguma forma de sensibilidade a contexto. Na realidade, nós colocamos um pouco desta sensibilidade a contextos no programa **Mundo de Tarski**. Por exemplo, considere a diferença entre os predicados Maior (**Larger**) e AtrásDe (**BackOf**). Se um cubo **a** é maior ou não que um cubo **b**, isto é uma questão determinada que não varia de acordo com a nossa perspectiva sobre o mundo. Se um cubo **a** está atrás (**BackOf**) de um cubo **b**, também é uma questão determinada, mas que, neste caso, depende de nossa perspectiva sobre o mundo. Se você girar o mundo 90 graus, a resposta poderia mudar.

Abra os arquivos **Sentenças de Austin** e **Mundo de Wittgenstein**. Avalie as sentenças deste arquivo e tabule os valores de verdade resultantes em uma tabela como abaixo. Nós já temos os valores na primeira coluna, com relação ao mundo original. Gire o mundo 90 graus à direita, avalie as sentenças novamente e acrescente os resultados à tabela. Repita o processo até dar uma volta completa no mundo.

	Original	**Girado 90º horário**	**Girado 180º**	**Girado 270º**
1.	F			
2.	F			
3.	V			
4.	F			
5.	V			
6.	F			

Você poderia imaginar uma sentença atômica da linguagem de blocos que produzisse uma linha da tabela acima com o seguinte padrão: **V**, **F**, **V**, **F** ?
Acrescente uma sétima sentença às Sentenças de Austin que exiba o padrão anterior.
Há alguma sentença atômica na linguagem que produziria uma linha com este padrão? (**F**, **V**, **F**, **F**)
Se houver, insira esta sentença em uma oitava linha. Se não houver, deixe a linha 8 do arquivo de sentenças em branco.
Há alguma sentença atômica que produziria uma linha com exatamente três valores **V**? Se houver, insira esta sentença como a nona sentença do arquivo.Caso contrário, deixe a linha 9 em branco.

(1.8) (Diferentes Linguagens FOL) Suponha que você tenha duas linguagens de primeira ordem: a primeira contém os predicados binários **GaveScruffy(x, y)** e **GaveCarl(x, y)**, e os nomes **max** e **claire**; a segunda contém o predicado ternário **Gave(x, y, z)** e os nomes **max**, **claire**, **scruffy**, e **carl**.

1. Liste todas as sentenças atômicas que podem ser expressadas na primeira linguagem. (Algumas delas podem dizer coisas estranhas, como **GaveScruffy(claire, claire)**, mas não se preocupe com isso.)
2. Quantas sentenças atômicas podem ser expressas na segunda linguagem? (Conte todas elas, inclusive as estranhas como **Gave(scruffy, scruffy, scruffy)**.)

3. Quantos nomes e predicados binários uma linguagem como a primeira precisaria ter para poder dizer tudo o que conseguimos dizer com a segunda?

Português	FOL	Comentário
Nomes:		
Max	**max**	
Claire	**claire**	
Folly	**folly**	O nome de um certo cachorro.
Carl	**carl**	O nome de outro cachorro.
Scruffy	**scruffy**	O nome de um certo gato.
Pris	**pris**	O nome de outro gato.
2:00 pm, 2 de janeiro de 2005	**2:00**	O nome de um instante do tempo.
2:01 pm, 2 de janeiro de 2005	**2:01**	Um minuto mais tarde
:	**:**	Similarmente para outros tempos
Predicados:		
x é um animal de estimação	**Pet(x)**	
x é uma pessoa	**Person(x)**	
x é um estudante	**Student(x)**	
t é mais cedo que t'	**t < t'**	Mais cedo que, para instantes de tempo
x estava com fome no instante t	**Hungry(x, t)**	
x estava brava no instante t	**Angry(x, t)**	
x possuia y no instante t	**Owned(x, y, t)**	
x deu y a z no instante t	**Gave(x, y, z, t)**	
x alimentou y no instante t	**Fed(x, y, t)**	

Tabela 1.2: Nomes e predicados de uma linguagem

(1.9) Teremos vários exercícios que usam os símbolos explicados na Tabela 1.2. Inicie um arquivo de sentenças novo no programa **Mundo de Tarski** e traduza as sentenças abaixo para FOL, usando os nomes e predicados listados na tabela. (Você terá que digitar os nomes e predicados pelo teclado. Certifique-se de digitá-los exatamente como aparecem na tabela; por exemplo, digite **2:00**, e não **2:00 pm** ou **2 pm**.) Todas as referências a tempo são assumidas serem instantes de tempo do dia 2 de janeiro de 2005.

1. *Claire possuia Folly as 2 pm.*
2. *Claire deu Pris a Max as 2:05 pm.*
3. *Max é um estudante.*
4. *Claire alimentou Carl as 2 pm.*
5. *Folly pertencia a Max as 3:05 pm.*
6. *2:00 pm é mais cedo que 2:05 pm.*

Grave seu arquivo com o nome **Sentences 1.9**.

(1.10) Traduza as sentenças abaixo para o português, construindo sentenças que soem natural. Consulte a Tabela 1.2 acima.

1. **Owned(max, scruffy, 2:00)**
2. **Fed(max, scruffy, 2:30)**
3. **Gave(max, scruffy, claire, 3:00)**
4. **2:00 < 2:00**

(1.11)* Para cada sentença da lista abaixo, sugira uma tradução para uma sentença atômica de FOL. Além de apresentar a tradução, explique a quais tipos de objetos os nomes que você utilizar se referem e o significado que tem em mente para cada símbolo de predicado.

1. *Max cumprimentou Claire. (com um aperto de mão)*
2. *Max cumprimentou Claire ontem.*
3. *AIDS é menos contagiosa do que tuberculose.*
4. *A Espanha está entre a França e Portugal em tamanho.*
5. *A miséria adora companhia.*

(1.12) (Símbolos de Função) Expresse em português do modo mais claro que conseguir as afirmações feitas pelas seguintes sentenças de FOL. Você deve tentar construir sentenças tão naturais quanto possível. A propósito, todas as sentenças são verdadeiras.

1. **Taller(father(claire), father(max))**
2. **john = father(max)**
3. **Taller(claire, mother(mother(claire)))**
4. **Taller(mother(mother(max)), mother(father(max)))**
5. **mother(melanie) = mother(claire)**

(1.13) (Símbolos de Função) Assuma que expandimos a linguagem de blocos para incluir as funções **fm**, **bm**, **lm** e **rm** (*frontmost*, *backmost*, *leftmost*, *rightmost* – conforme explicado no texto do Capítulo 1). Então, as seguintes fórmulas deveriam todas ser sentenças da liguagem:

1. **Tet(lm(e))**
2. **fm(c) = c**
3. **bm(b) = bm(e)**
4. **FrontOf(fm(e), e)**
5. **LeftOf(fm(b), b)**
6. **SameRow(rm(c), c)**
7. **bm(lm(c)) = lm(bm(c))**
8. **SameShape(lm(b), bm(rm(e)))**
9. **d = lm(fm(rm(bm(d))))**
10. **Between(b, lm(b), rm(b))**

Complete a tabela seguinte com **TRUE** ou **FALSE** conforme a sentença indicada na linha seja verdadeira ou falsa no mundo indicado na coluna. Uma vez que o programa Mundo de Tarski não entende símbolos de função, você não será capaz de verificar suas respostas (com o botão **Verify**).

	Mundo de Leibniz	M. de Bolzano	M. de Boole	M. de Wittgenstein
1.	TRUE			
2.				
3.				
4.				
5.	FALSE			
6.		TRUE		
7.				
8.			FALSE	
9.				
10.				

(1.14) Como você provavelmente notou ao fazer o Exercício 1.13, três das sentenças são verdadeiras em todos os quatro mundos. Uma destas não pode ser falsificada em nenhum mundo, por causa dos significados dos predicados e símbolos de função que ela contém. O objetivo deste exercício é construir um mundo em que todas as outras sentenças do Exercício 1.13 são falsas. Grave o mundo com o nome **World 1.14**.

(1.15) Suponha que você tenha duas linguagens de primeira ordem para falar sobre pais. A primeira, que chamaremos de linguagem funcional, contém os nomes **claire**, **melanie**, e **jon**, o símbolo de função **father**, e os símbolos de predicado = e **Taller**. A segunda linguagem, que chamaremos de linguagem relacional, tem os mesmos nomes, nenhum símbolo de função e os predicados binários =, **Taller** e **FatherOf**, onde **FatherOf(c, b)** significa que ***c*** é o pai de ***b***. Traduza as sentenças atômicas seguintes da linguagem relacional para a linguagem funcional. Tenha cuidado. Algumas sentenças atômicas, tais como **claire = claire,** são sentenças das duas linguagens.

1. **FatherOf(jon, claire)**
2. **FatherOf(jon, melanie)**
3. **Taller(claire, melanie)**

Quais das sentenças atômicas seguintes da linguagem funcional podem ser traduzidas em sentenças atômicas da linguagem relacional? Traduza aquelas que podem ser traduzidas e explique o problema que ocorre com as que não podem.

4. **father(melanie) = jon**
5. **father(melanie) = father(claire)**
6. **Taller(father(claire), father(jon))**

Quando adicionarmos conectivos e quantificadores a FOL, seremos capazes de traduzirmos livremente, nos dois sentidos, entre as linguagens funcionais e relacionais.

(1.16) Vamos supor que todos tenham um astro de cinema favorito. Dada esta hipótese, construa uma linguagem de primeira ordem para falar sobre pessoas e suas estrelas de cinema favoritas. Utilize um símbolo de função que permita a você se referir ao astro de cinema favorito de um indivíduo, e mais um símbolo de relação que permita a você dizer que uma pessoa é um ator melhor do que outra. Explique a interpretação de seus símbolos de função e relação, e então utilize sua linguagem para expressar as seguintes afirmações:

1. *Harrison é o astro favorito de Nancy.*
2. *O astro favorito de Nancy é melhor do que Sean.*
3. *O astro favorito de Nancy é melhor do o astro favorito de Max.*
4. *O astro favorito do astro favorito de Claire é Brad.*
5. *Sean é seu próprio astro favorito.*

(1.17)* Construa uma linguagem de primeira ordem para falar sobre pessoas e suas respectivas alturas. Contudo, ao invés de usar símbolos de relação como **Taller**, utilize um símbolo de função que o permita a se referir à altura das pessoas, em conjunto com os símbolos de relação = e <. Explique a interpretação de seu símbolo de função, e então utilize sua linguagem para expressar as seguintes afirmações:

1. *George é mais alto do que Sam.*
2. *Sam e Mary têm a mesma altura.*

Você vê algum problema com este símbolo de função? Se vê, explique o problema. [Dica: O que acontece se você aplicar o símbolo de função duas vezes?]

(1.18)* Para cada uma das sentenças da lista a seguir, construa uma sentença atômica em FOL que seja uma tradução. Além de apresentar a tradução, explique a quais tipos de objetos os nomes que você utilizou se referem e apresente o significado pretendido para os predicados e símbolos de função que você usou. Para cada sentença você terá que inventar constantes individuais, predicados e símbolos de funções.

1. *A capital de Indiana é maior do que a capital da Califórnia.*
2. *A amante de Hitler morreu em 1945.*
3. *Max apertou a mão do pai de Claire.*
4. *Max é o pai de seu filho.*
5. *O filho mais velho de John e Nancy é mais novo que o filho mais velho de Jon e Mary Ellen.*

(1.19) Quais das seguintes sentenças atômicas da linguagem de primeira ordem da teoria dos conjuntos são verdadeiras e quais são falsas? Considere ***a*** como o nome do número 2, ***b*** como o nome do conjunto { 2, 4, 6}, ***c*** como o nome do número 6 e ***d*** como o nome do conjunto {2, 7, {2, 4, 6}}.

1. **a ∈ c**
2. **a ∈ d**
3. **b ∈ c**
4. **b ∈ d**
5. **c ∈ d**
6. **c ∈ b**

(1.20) Mostre que as seguintes expressões são termos na linguagem de primeira ordem da aritmética. Faça isso explicando quais as clausas da definição se aplicam e em qual ordem. A quais números cada uma das sentenças se refere?

1. **(0 + 0)**
2. **(0 + (1 x 0))**
3. **((1 + 1) + ((1 + 1) x (1 + 1)))**
4. **(((1 x 1) x 1) x 1)**

(1.21) Encontre uma forma de expressar o fato de que três é menor do que quatro em uma sentença da linguagem de primeira ordem para a aritmética.

(1.22)* Mostre que há infinitamente muitos termos da linguagem da aritmética de primeira ordem que se referem ao número um.

Exercícios do Capítulo 2 - (A Lógica das Sentenças Atômicas)

Exercícios . . . . . .

(2.1) (Validade e Correção – Classificando argumentos) Abra o arquivo **Socrates' Sentences** (Sentenças de Sócrates). Este arquivo contém oito argumentos separados por linhas tracejadas, com rótulos indicando as premissas e conclusões de cada um deles.

(1) Na primeira coluna da tabela abaixo, classifique cada um dos argumentos como válido ou inválido. Ao avaliá-los, você pode pressupor quaisquer das características dos mundos que podem ser construídos com o programa **Mundo de Tarski** (por exemplo, que dois blocos não podem ocupar o mesmo quadro na grade, etc.).

Argumento	Válido?	Correto no Mundo de Sócrates?	Correto no Mundo de Wittgenstein?
1.			
2.			
3.			
4.			
5.			
6.			
7.			
8.			

(2) Agora abra o arquivo **Mundo de Sócrates** e avalie cada uma das sentenças, se é verdadeira ou falsa. Use o resultado de sua avaliação para preencher a coluna de cada argumento como correto ou incorreto no Mundo de Sócrates. (Lembre-se que apenas argumentos válidos podem ser corretos. Argumentos inválidos são automaticamente incorretos.)

(3) Abra o arquivo **Mundo de Wittgenstein** e faça o mesmo, preenchendo a terceira coluna da tabela.

(4) Para cada argumento que você avaliou como **inválido**, construa um mundo no qual as premissas do argumento são todas verdadeiras, mas a conclusão é falsa. Grave estes arquivos como **World 2.1.x**, onde **x** é o número do argumento. (Se você tiver dificuldade em construir tal mundo, talvez você deva repensar sua avaliação do argumento como **inválido**).

- Este exercício (2.1) reforça um ponto muito importante, que muitos estudantes de lógica costumam esquecer: **a validade** de um argumento **depende apenas do argumento**. Ela independe dos fatos específicos do mundo sobre o qual as sentenças se referem. A **correção** de um argumento, por outro lado, **depende** de ambos, **do argumento e do mundo**.

(2.2) (Classificando Argumentos) Para cada um dos argumentos abaixo, identifique as premissas e conclusões colocando-o no Formato Fitch. Então diga se o argumento é válido ou não. Para os primeiros cinco argumentos, também dê sua opinião sobre se eles são corretos. (Lembre-se que apenas argumentos válidos podem ser corretos.) Se algumas de suas avaliações dos argumentos depender de interpretações específicas sobre os predicados, explique estas dependências.

1. *Qualquer um que ganha um Oscar é famoso. Meryl Streep ganhou um Oscar. Então, Meryl Streep é famosa.*
2. *Harrison Ford não é famoso. Afinal de contas, atores que ganham Oscar são famosos, e ele nunca ganhou um.*
3. *O direito a portar armas é a liberdade mais importante. Charlton Heston disse isso, e ele jamais está errado.*
4. *Al Gore deve ser desonesto. Afinal de contas, ele é um político e dificilmente qualquer político é honesto.*
5. *Mark Twain morou em Hannibal, no Missouri, pois San Clemens nasceu lá e Mark Twain é Sam Clemens.*
6. *Ninguém com menos de 21 anos comprou cerveja aqui a noite passada, oficial. Afinal, estávamos fechados, então ninguém comprou nada aqui noite passada.*
7. *Claire deve morar na mesma rua que Laura, uma vez que ela mora na mesma rua que Max e ele e Laura moram na mesma rua.*

(2.3) Para cada um dos argumentos abaixo, identifique as premissas e conclusões, colocando-o no formato Fitch, e decida se o argumento é válido. Se sua avaliação depender de interpretações particulares dos predicados, explique estas dependências.

1. *Muitos dos estudantes do curso de cinema participam de testes para filmes. Conseqüentemente, deve haver muitos estudantes no curso de cinema.*
2. *Há poucos estudantes no curso de cinema, mas muitos deles participam de testes para filmes. Então, há muitos estudantes no curso de cinema.*
3. *Há muitos estudantes no curso de cinema. Afinal de contas, muitos estudantes participam de testes para filmes e apenas estudantes no curso de cinema participam destes testes.*
4. *Há trinta estudantes na minha disciplina de lógica. Alguns dos estudantes entregaram hoje seus trabalhos a tempo. A maioria dos estudantes foi à festa ontem à noite. Então, algum estudante que foi à festa conseguiu entregar o trabalho a tempo.*
5. *Há trinta estudantes na minha disciplina de lógica. Algum estudante que foi à festa ontem à noite deve ter entregado o trabalho a tempo. Pois alguns estudantes entregaram seus trabalhos a tempo e todos eles foram à festa.*
6. *Há trinta estudantes em minha disciplina de lógica. A maioria dos estudantes entregou o trabalho a tempo hoje. A maioria dos estudantes foi à festa ontem à noite. Então, algum estudante que foi à festa entregou o trabalho a tempo*

(2.4) (Validade e Verdade) Pode um argumento válido ter:

(a) premissas falsas e conclusão falsa?
(b) premissas falsas e conclusão verdadeira?
(c) premissas verdadeiras e conclusão falsa?
(d) premissas verdadeiras e conclusão verdadeira?

Para os itens que você respondeu **sim**, de um **exemplo** de um tal argumento. Para os itens que você respondeu **não**, explique **por que**.

(2.5) (Transitividade da Identidade) Escreva uma prova informal do argumento seguinte usando apenas **(=Elim)** – (Indiscernibilidade dos idênticos). Certifique-se de apontar qual nome está sendo substituído por qual e em qual sentença.

$$\begin{array}{|l} b = c \\ a = b \\ \hline a = c \end{array}$$

(2.6) Faça uma prova informal de que o argumento seguinte é válido. Se você provou a transitividade da identidade no Exercício 2.5, então você pode utilizar nesta prova este princípio. Caso contrário, utilize apenas a indiscernibilidade de idênticos **(=Elim)**.

$$\begin{array}{|l} \mathrm{SameRow}(a, a) \\ a = b \\ b = c \\ \hline \mathrm{SameRow}(c, a) \end{array}$$

(2.7) Considere as seguintes sentenças e responda:

(1) *Max e Claire não são parentes.*
(2) *Nancy é a mãe de Max.*
(3) *Nancy não é a mãe de Claire.*

(a) A sentença (3) é conseqüência de (1) e (2)?
(b) A sentença (2) é conseqüência de (1) e (3)?
(c) A sentença (1) é conseqüência de (2) e (3)?

Em cada caso, se sua resposta foi não, descreva uma possível circunstância na qual as premissas são verdadeiras e a conclusão falsa.

Para cada um dos argumentos a seguir, avalie sua validade. Se o argumento for válido, apresente uma prova informal. Se for inválido, construa um mundo em que as premissas sejam verdadeiras e a conclusão falsa.

2.8

Large(a)
Larger(a, c)

Small(c)

2.9

LeftOf(a, b)
b = c

RightOf(c, a)

2.10

SameSize(b, c)
SameShape(b, c)
---
b = c

2.11

LeftOf(a, b)
RightOf(c, a)

LeftOf(b, c)

2.12

BackOf(a, b)
FrontOf(a, c)

FrontOf(b, c)

2.13

SameSize(a, b)
Larger(a, c)
Smaller(d, c)

Smaller(d, b)

2.14

Between(b, a, c)
LeftOf(a, c)

LeftOf(a, b)

Experimente (1) . . . . . .

(1) Vamos usar o programa **Fitch** para construir uma prova formal de **SameRow(b, a)** a partir das premissas **SameRow(a, a)** e **b = a**. Inicie o programa Fitch e abra o Arquivo **Identity 1**. Aqui temos o início da prova formal. As premissas aparecem sobre a barra de Fitch. A prova pode parecer ligeiramente diferente das provas do livro, uma vez que em Fitch as linhas não têm que ser numeradas, por razões que descobriremos logo. (Se você quiser numerar as linhas, selecione a opção ***show step numbers*** no menu ***Proof***. Mas não faça isso agora.)

(2) Antes de nós começarmos a construir a prova, note que abaixo da janela de prova de há uma faixa separada chamada "faixa de metas" (***Goal strip***), contendo a meta da prova. Neste caso a meta é provar a sentença **SameRow(b, a)**. Se atingirmos esta meta com sucesso, poderemos fazer com que o programa Fitch mostre uma 'checkmark' à direita da meta.

(3) Vamos construir a prova. O que nós precisamos fazer é escrever as linhas necessárias para completar a prova, da mesma maneira que fizemos ao término da última seção. Acrescente uma linha nova na prova selecionando a opção ***Add Step After*** no menu ***Proof***. Na linha criada, entre a sentença **a = b**, ou digitando-a ou usando a 'barra de ferramentas' acima da janela de prova. Nós inicialmente iremos usar esta linha para obter nossa conclusão e então voltaremos e provaremos esta linha.

(4) Uma vez que você entrou **a = b**, insira outra linha abaixo desta e entre a sentença da meta **SameRow(b, a)**. Clique com o mouse na palavra ***Rule?*** que aparece à direita de **SameRow(b, a)**. No menu que surgiu, vá para as Regras de Eliminação (***Elim***) e selecione =. Se você fizer isso direito, o nome de regra deve estar indicando **(=Elim)**. Se não estiver, tente novamente.

(5) A seguir, "cite" a primeira premissa e a sentença intermediária que você acabou de entrar. Isso é feito clicando nestas duas sentenças, em qualquer ordem. Se você clicar em uma sentença errada, apenas clique de novo e ela será "des-citada". Com as sentenças certas citadas, clique no botão ***Verify Proof***. A última linha deve agora ter recebido a marca de confirmação (azul), uma vez que ele é uma instância válida da regra **(=Elim)**. A linha que contém **a = b** não recebeu marca de confirmação (mas uma marca vermelha, de erro), porque que nós ainda não indicamos de quais linhas ela se segue. Nem a meta recebeu a marca de confirmação, porque ainda não completamos a prova de **SameRow(b, a)**. Calma! Tudo a seu tempo.

(6) Agora acrescente uma linha antes da primeira linha que você introduziu (a que contém **a = b**), e insira a sentença **b = b**. Faça isto movendo o indicador de foco (o triângulo na margem esquerda) para a linha que contém **a = b** e selecionando a opção ***Add Step Before*** no menu ***Proof***. (Se a linha nova aparecer no lugar errado, selecione a opção ***Delete Step***, também no menu ***Proof***.) Entre a sentença **b = b** e justifique-a usando a regra **(=Intro)**. Clique no botão ***Check Step*** e verifique se a marca de confirmação aparece.

(7) Finalmente, justifique a linha que contém **a = b** usando a regra **(=Elim)**. Você precisa mover o indicador de foco para esta linha, e então "citar" a segunda premissa e a sentença **b = b**. Agora, a prova inteira, inclusive a meta, deve receber a marca de confirmação. Para verificar isso, clique no botão ***Verify Proof***. A prova deveria se parecer à prova completada na página 57, com exceção da ausência de números nas linhas. (Selecione ***Show Step Numbers*** no menu ***Proof***. Os números das linhas e das justificativas devem aparecer e a prova ficará igual à do livro.)

(8) Nós mencionamos anteriormente, que o programa Fitch o deixa tomar alguns atalhos, permitindo a você fazer em uma única linha coisas que necessitariam de várias linhas, caso nós aderíssemos estritamente ao **Sistema *F***. Esta prova é um caso particular disso. Nós construímos uma prova adequada a ***F***, mas Fitch, na verdade, possui a regra de simetria de identidade internamente (intrínseca) a **(=Elim)**. Assim nós poderíamos provar a conclusão diretamente a partir das duas premissas, usando uma única aplicação da regra **(=Elim)**. É o que faremos em seguida.

(9) Acrescente outra linha bem ao final de sua prova com a sentença de meta **SameRow(b, a)**. Justifique esta linha usando **(=Elim)** e "citando" apenas as duas premissas. Você verá que o passo recebe marca de confirmação quando você verifica a prova.

(10) Grave sua prova em um arquivo com o nome **Proof Identity 1**.

Experimente (2) . . . . . .

(1) No programa **Fitch**, abra o arquivo a **Ana Con 1**. Neste arquivo você verá nove premissas seguida por seis conclusões que são conseqüências destas premissas. Realmente, cada uma das conclusões segue de três ou menos das premissas.

(2) Posicione o cursor de foco (o pequeno triângulo) na primeira conclusão após a Barra de Fitch (que é **SameShape(c, b)**). A regra **(AnaCon)** foi invocada como justificativa desta conclusão, mas nenhuma sentença foi citada. Esta conclusão segue de **Cube(b)** e **Cube(c)**. Cite estas sentenças e confira a linha (Clique em ***Check Step***).

(3) Agora mova o cursor de foco para a linha que contém **SameRow(b, a)**. Uma vez que a relação de estar na mesma linha (SameRow) é simétrica e transitiva, **SameRow(b, a)** segue de **SameRow(b, c)** e **SameRow(a, c)**. Cite estas duas sentenças e confira a linha.

(4) A terceira conclusão, **BackOf(e, c)**, segue de três das premissas. Veja se você pode descobrir quais são. Cite-as. Se você errar, Fitch responderá com um X (vermelho) quando você conferir a linha.

(5) Agora preencha as citações necessárias para justificar a quarta e a quinta conclusões. Nestes dois casos é você que terá que invocar a regra **(AnaCon)**. (Você encontrará a regra no submenu ***Con*** do menu popup ***Rule?***)

(6) A conclusão Final, **SameCol(b, b)**, não requer que qualquer premissa seja citada em seu apoio. Ela simplesmente é uma verdade analítica, quer dizer, verdadeira em virtude de seu significado. Especifique a regra e confira este linha.

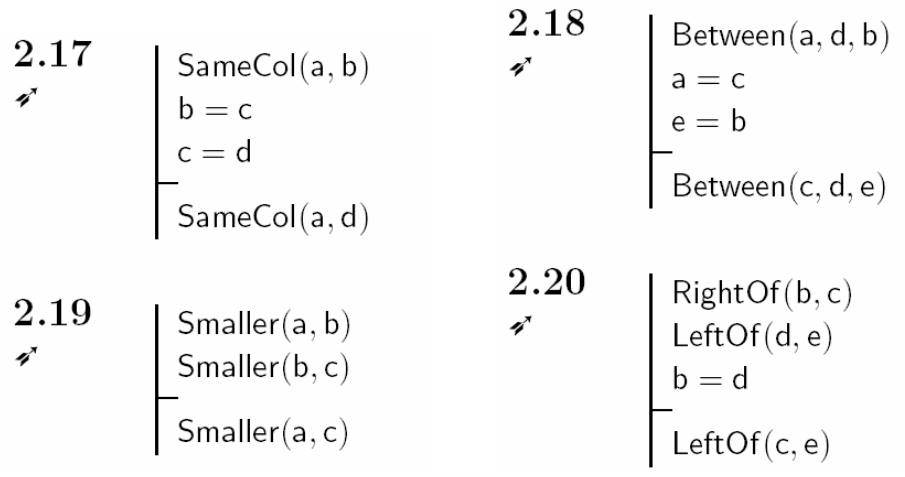
(7) Quando você terminar, clique em ***Verify Proof*** para ver se todas as metas foram recebem a marca de confirmação. Grave seu trabalho em um arquivo com o nome **Proof Ana Con 1**.

Exercícios . . . . . .

(2.15) Se você pulou as seções *Experimente 1 e 2*, volte e faça-as agora.

(2.16) Use o programa **Fitch** para produzir uma versão formal da prova informal que você fez no *Exercício 2.5* (transitividade da identidade). Lembre-se que as informações iniciais deste exercício estão no arquivo **Exercise 2.16** e que você deverá salvar seu trabalho com o nome **Proof 2.16**.

Para resolver os exercícios a seguir, você precisa rodar o programa Fitch e abrir o arquivo **Exercise 2.x** (onde 2.x corresponde ao número de cada exercício abaixo). Ao terminar cada exercício, salve seu trabalho em um arquivo com o nome **Proof 2.x**. Seu objetivo é construir provas formais das conclusões a partir das premissas.



Observação: apenas nos exercícios **2.19** e **2.20** é permitido o uso da regra **Ana Con**.

Experimente (3)

(1) Inicie o **Mundo de Tarski** e abra o arquivo de sentenças **Bill's Argument** (Argumentos de Bill). Este argumento afirma que **Between(b, a, d)** se segue das premissas: **Between(b, c, d)**, **Between(a, b, d)**, e **Left(a, c)**. O que você acha?

(2) Inicie um mundo novo e insira, em uma das linhas, 4 blocos rotulados como **a**, **b**, **c** e **d**.

(3) Posicione os blocos de modo que a conclusão seja falsa. Verifique as premissas. Se alguma não for verdadeira, reposicione-os até que todas sejam verdadeiras. A conclusão continua falsa? Se não, continue tentando.

(4) Se você não estiver conseguindo, tente colocar os blocos na ordem **d**, **a**, **b**, **c**. Agora, verifique todas as sentenças **(CTRL-F)** para certificar-se de que todas as premissas são verdadeiras e a conclusão falsa. Este mundo é um contra-exemplo do argumento. Então nós demonstramos, com ele, que a conclusão não se segue das premissas.

(5) Grave seu trabalho nos arquivos **World Counterexample 1** e **Sentences Counterexample 1**.

Exercícios

(2.21) Se você pulou a seção *Experimente 3*, volte e faça-a agora.

(2.22) Será que o argumento abaixo é válido? Correto? Se for válido, faça uma prova informal de que é. Se não for válido apresente um contra-exemplo informal.

> *Todos os cientistas da computação são ricos. Qualquer um que sabe como programar um computador é um cientista da computação. Bill Gates é rico. Portanto, Bill Gates sabe como programar um computador.*

(2.23) Será que o argumento abaixo é válido? Correto? Se for válido, faça uma prova informal de que é. Se não for válido apresente um contra-exemplo informal.

> *Filósofos têm inteligência suficiente para serem cientistas da computação. Qualquer um que se torne um cientista da computação certamente ficará rico. Qualquer um com inteligência suficiente para ser um cientista da computação se tornará um. Portanto, todo filósofo ficará rico.*

Cada um dos problemas seguintes apresenta um argumento formal na linguagem dos blocos. Se o argumento for válido, escreva uma prova formal para ele usando o programa **Fitch**. (Você encontrará os arquivos para os exercícios no diretório de exercícios "**Exercicies 2.24**",...) *Importante*: cada aplicação da regra **(AnaCon)** (conseqüência analítica) pode citar, no máximo, duas sentenças. Se o argumento não for válido, construa um contra-exemplo usando o programa **Mundo de Tarski**.

2.24

Larger(b, c)
Smaller(b, d)
SameSize(d, e)

Larger(e, c)

2.25

FrontOf(a, b)
LeftOf(a, c)
SameCol(a, b)

FrontOf(c, b)

2.26

SameRow(b, c)
SameRow(a, d)
SameRow(d, f)
LeftOf(a, b)

LeftOf(f, c)

2.27

SameRow(b, c)
SameRow(a, d)
SameRow(d, f)
FrontOf(a, b)

FrontOf(f, c)

Exercícios do Capítulo 3 - (Os Conectivos Booleanos)

Experimente (1) . . . . . .

(1) Abra o arquivo **Wittgenstein's World**. Inicie um arquivo de sentenças novo e escreva a seguinte sentença: **¬¬¬¬¬Between(e, d, f)**

(2) Use o botão ***Verify*** para checar o valor de verdade da sentença.

(3) Agora jogue o jogo (clique no botão ***Game***) comprometendo-se com o valor que você quiser. O que acontece com o número de símbolos de negação conforme você avança no jogo? O que com os compromissos de suas escolhas?

(4) Agora jogue outra vez o jogo comprometendo-se com o valor contrário do que fez anteriormente. Se você ganhou na primeira vez, deverá perder este jogo, e vice-versa. Não se sinta mal pela derrota.

Exercícios . . . . . .

(3.1) Não deixe de fazer a seção *Experimente 1* acima.

(3.2) (Avaliando sentenças negadas) Abra os arquivos **Boole's World** (Mundo de Boole) e **Brouwer's Sentences** (Sentenças de Brouwer). No arquivo de sentenças há uma lista de sentenças construídas a partir de sentenças atômicas usando apenas o símbolo de negação. Para cada sentença, decida se ela é verdadeira ou falsa. Verifique se acertou. Se a sentença for falsa, torne-a verdadeira adicionando ou retirando um símbolo de negação. Quando tiver feito todas as sentenças verdadeiras, verifique **(CTRL F)** e grave seu trabalho com o nome **Sentences 3.2**.

(3.3) (Construindo um mundo) Inicie um arquivo de sentenças novo. Escreva as sentenças abaixo no arquivo e grave-o com o nome **Sentences 3.3**.

1. **¬Tet(f)**
2. **¬SameCol(c, a)**
3. **¬¬SameCol(c, b)**
4. **¬Dodec(f)**
5. **c ≠ b**
6. **¬(d ≠ e)**
7. **¬SameShape(f, c)**
8. **¬¬SameShape(d, c)**
9. **¬Cube(e)**
10. **¬Tet(c)**

Agora inicie um arquivo de Mundo novo e construa um mundo onde todas estas sentenças sejam verdadeiras. Conforme você modifica o mundo para fazer as últimas sentenças verdadeiras, certifique-se de não falsificar acidentalmente as sentenças anteriores. Quando terminar, grave o arquivo com o nome **World 3.3**.

(3.4) Seja **P** uma sentença verdadeira e **Q** uma sentença formada colocando-se um determinado número de símbolos de negação na frente de **P**. Mostre que se você colocar um número par de negações, então **Q** será verdadeira, mas se você colocar um número ímpar, **Q** será falsa.

[**Dica**: uma prova completa deste fato simples requer uma técnica conhecida como *indução matemática*, que aprenderemos apenas no Capítulo 16. Aqui basta escrever uma explicação, o mais claramente que conseguir, sobre porque isso ocorre]

Agora assuma que **P** é atômica, com valor de verdade desconhecido, e **Q** é formada como explicado acima. Não importa quantos símbolos de negação **Q** tenha, ela sempre terá o mesmo valor de verdade que um literal, ou terá o mesmo valor que **P** ou o mesmo valor que **¬P**. Descreva um método simples para saber qual dos dois valores de verdade **Q** terá.

Experimente (2). . . . . .

(1) Abra o arquivo **Claire's World**. Inicie uma nova janela de sentenças e entre com a sentença:

¬Cube(a) ∧ ¬Cube(b) ∧ ¬Cube(c)

(2) Note que esta sentença é falsa neste mundo, uma vez que **c** é um cubo. Jogue o jogo comprometendo-se (erroneamente) com a verdade da sentença. Você verá que o programa **Mundo de Tarski** imediatamente aponta

a subfórmula falsa. Seu compromisso com a verdade da sentença assegura que você perderá o jogo, mas durante o percurso, a razão da falsidade da sentença se tornará aparente.

(3) Agora jogue novamente, comprometendo-se com a falsidade da sentença. Quando o Mundo de Tarski lhe pede para escolher uma subfórmula conjuntiva que você considera falsa, escolha a primeira sentença. Ela não é falsa, mas selecione-a assim mesmo e veja o que acontece depois de sua escolha.

(4) Siga com o jogo até que o Mundo de Tarski diga-lhe que você perdeu. Então clique no botão ***Back*** (voltar) algumas vezes, até que esteja de volta ao ponto onde teve que escolher uma das sub-fórmulas conjuntivas como falsa. Desta vez, escolha a sub-fórmula falsa e continue o jogo. Desta vez você vencerá.

(5) Note que você pode perder o jogo mesmo quando seu palpite original sobre a verdade da sentença foi correto, se você fizer alguma má escolha no decorrer do jogo. O Mundo de Tarski sempre permite que você volte e refaça diferentemente suas escolhas. Se seu palpite original for correto, sempre haverá uma forma de você ganhar o jogo. Se não for possível para você ganhar o jogo, então seu palpite original estava errado.

(6) Grave seu arquivo com o nome **Sentences Game 1** quando tiver terminado.

Exercícios . . . . . .

(3.5) Não deixe de praticar a seção *Experimente 2* acima.

(3.6) Inicie um arquivo de sentenças novo e abra o arquivo **Wittgenstein's World**. Escreva as seguintes sentenças no arquivo de sentenças:

1. **Tet(f) ∧ Small(f)**
2. **Tet(f) ∧ Large(f)**
3. **Tet(f) ∧ ¬Small(f)**
4. **Tet(f) ∧ ¬Large(f)**
5. **¬Tet(f) ∧ ¬Small(f)**
6. **¬Tet(f) ∧ ¬Large(f)**
7. **¬ (Tet(f) ∧ Small(f))**
8. **¬ (Tet(f) ∧ Large(f))**
9. **¬ (¬Tet(f) ∧ ¬Small(f))**
10. **¬ (¬Tet(f) ∧ ¬Large(f))**

Com as sentenças digitadas, decida quais você considera verdadeiras. Anote seus palpites para lembrar. Agora, utilize o **Tarski's World** para avaliar as sentenças (**CTRL-F**) e verifique seus palpites. Sempre que tiver errado, jogue o jogo para ver em que ponto você errou.
Caso não tenha errado nenhuma, o jogo não será muito instrutivo. Mas jogue algumas vezes mesmo assim, apenas por diversão. Em particular, experimente jogar comprometendo-se com a falsidade da sentença 9. Como esta sentença é verdadeira no **Mundo de Wittgenstein**, o programa irá ganhar de você. Certifique-se de entender TUDO o que acontece em todo o processo do jogo.
Em seguida, modifique o tamanho do bloco, **f** preveja como isto irá afetar o valor de verdade das 10 sentenças, e veja se sua predição está correta. Qual é o número máximo destas sentenças que você consegue tornar verdadeiras em um único mundo? Construa um mundo no qual o máximo número de sentenças sejam verdadeiras. Grave ambos os arquivos como **World 3.6** e **Sentences 3.6**.

(3.7) (Construindo um Mundo) Abra o arquivo **Max's Sentences**. Construa um mundo em que todas as sentenças sejam verdadeiras. Você deve iniciar com um mundo com seis blocos e ir modificando-o, tentando tornar todas as sentenças verdadeiras. Certifique-se de que, conforme você faz as últimas sentenças verdadeiras, você não falsifica inadvertidamente as primeiras. Grave seu trabalho como **World 3.7**.

Experimente (3). . . . . .

(1) Abra o arquivo **Ackermann's World**. Inicie um arquivo de sentenças novo e entre a seguinte sentença, certificando-se de não errar os parênteses:

Cube(c) ∨ ¬(Cube(a) ∨ Cube(b))

(2) Jogue o jogo comprometendo-se (erroneamente) com a verdade desta sentença. Como a sentença é uma disjunção, e você está comprometido com sua verdade, você será solicitado a escolher um dos disjuntos que considera falso. Como o primeiro é obviamente falso, escolha o segundo.

(3) Neste ponto, você se encontrará comprometido com a falsidade de uma disjunção (verdadeira). Então você está comprometido com a falsidade de cada um dos disjuntos. O programa apontará para você que você está

comprometido com a falsidade de **Cube(b)**. Mas isto está claramente errado, uma vez que **b** é um cubo. Continue até o **Mundo de Tarski** dizer-lhe que você perdeu.

(4) Jogue novamente, desta vez comprometendo-se com a falsidade da sentença. Você será capaz de ganhar o jogo desta vez. Se não conseguir, volte e tente novamente.

(5) Grave sua janela de sentenças com o nome de: **Sentences Game 2**.

Exercícios . . . . . .

(3.8) Se você pulou a seção *Experimente 3*, volte e faça-a agora.

(3.9) Abra o arquivo **Wittgenstein's World** e o arquivo **Sentences 3.6** que você criou para o *Exercício 3.6*. Modifique suas sentenças substituindo todos os $\wedge$ por $\vee$. Grave o arquivo modificado com o nome **Sentences 3.9**. Decida agora quais destas sentenças modificadas são verdadeiras. Novamente, anote suas escolhas. Então utilize o programa para avaliar as sentenças e checar suas escolhas (**CTRL F**). Em todos os casos que tenha errado, use o jogo para ver onde cometeu o erro. Caso não tenha errado nada, jogue algumas vezes mesmo assim. Como no *Exercício 3.6*, encontre o número máximo de sentenças que você pode tornar verdadeiras através da mudança do tamanho ou da forma (ou ambos) do bloco **f**. Grave o mundo modificado com o nome **World 3.9**.

(3.10) Abra o arquivo **Ramsey's World** e inicie uma janela de sentenças nova. Digite as seguintes sentenças no arquivo:

1. **Between(a, b, c)** $\vee$ **Between(b, a, c)**
2. **FrontOf(a, b)** $\vee$ **FrontOf(c, b)**
3. $\neg$**SameRow(b, c)** $\vee$ **LeftOf(b, a)**
4. **RightOf(b, a)** $\vee$ **Tet(a)**

Atribua valor de verdade para cada uma destas sentenças, de acordo com o mundo de Ramsey. Verifique suas escolhas verificando o valor de verdade que o programa atribui (**CTRL F**). Faça agora uma simples modificação no mundo que transforma todas as sentenças em falsidades. Grave o mundo modificado com o nome **World 3.10** e as sentenças com o nome **Sentences 3.10**.

(3.11) Certifique-se de que o programa **Mundo de Tarski** esteja configurado para apresentar os mundos em 3D (opção ***3-D View*** do menu ***Display***). Então abra os arquivos **Kleene's World** e **Kleene's Sentences**. Alguns objetos estão escondidos atrás de outros, tornando impossível avaliar a verdade de algumas sentenças. Cada um dos seis nomes **a**, **b**, **c**, **d**, **e** e **f** nomeiam objetos do arquivo. Mesmo sem ver todos os objetos, algumas das sentenças da lista podem ser avaliadas mesmo com as informações incompletas que temos. Avalie a verdade ou falsidade de cada afirmação, se puder, sem recorrer à visão 2-D do mundo. Então jogue o jogo. Se seu compromisso inicial for correto, mas você perder o jogo, volte e jogue outra vez. Feito isso, adicione comentários em cada sentença explicando se você pode avaliar a seu valor de verdade no mundo em 3-D e porque. Finalmente, selecione a opção ***2-D View*** do menu ***Display*** e verifique seu trabalho. A primeira sentença já está comentada, para dar a você uma idéia da tarefa solicitada. (O ponto-e-vírgula ";" diz ao programa **Mundo de Tarski** que o que se segue é um comentário.)

Experimente (4). . . . . .

(1) Vamos experimentar fazer a avaliação de algumas sentenças construídas a partir de sentenças atômicas usando os três conectivos $\wedge$, $\vee$, $\neg$. Abra o arquivos **Boole's Sentences** e **Wittgenstein's World**.

(2) Avalie (se verdadeira ou falsa) cada uma das sentenças do arquivo e, verifique sua avaliação (botão ***Verify***). Se a sua avaliação estiver errada, use o Jogo (botão ***Game***) para entender o porquê. Não mude para a sentença seguinte até que tenha entendido porque a que está avaliando é verdadeira ou falsa.

(3) Você consegue perceber a importância dos parênteses? Após ter entendido todas as sentenças, tente verificar quais das sentenças falsas você consegue tornar verdadeiras através apenas de mudanças nos parênteses, sem qualquer outra troca. Grave seu trabalho com o nome **Sentences Ambiguity 1**.

Exercícios . . . . . .

Para dominar realmente uma linguagem nova, você precisa utilizá-la, não apenas ler sobre ela. Os exercícios e problemas seguintes são justamente para isto.

(3.12) Não deixe de fazer a seção *Experimente 4*.

(3.13) (Construindo um Mundo) Abra o arquivo **Schroder's Sentences** e construa um mundo no qual todas estas sentenças sejam verdadeiras. Conforme trabalha, você se verá constantemente modificando o mundo que está criando. Sempre que fizer uma mudança no mundo, tenha cuidado para não tornar falsa nenhuma sentença anterior, que já estava verdadeira. Quando terminar, verifique se todas as sentenças ficaram realmente verdadeiras. Grave seu trabalho no arquivo **World 3.13**.

(3.14) (Parênteses) Mostre que a sentença **¬(Small(a) ∨ Small(b))** não é conseqüência da sentença **¬Small(a) ∨ Small(b)**. Você fará isto construindo um mundo com um contra-exemplo, em que a primeira sentença (conclusão) é falsa e a segunda (premissa) é verdadeira.

(3.15) (Mais Parênteses) Mostre, da mesma forma que no exercício anterior, que a sentença **Cube(a) ∧ (Cube(b) ∨ Cube(c))** não é conseqüência da sentença **(Cube(a) ∧ Cube(b)) ∨ Cube(c)**.

(3.16) (Equivalências de *DeMorgan*) Abra o arquivo **DeMorgan's Sentences**. Construa um mundo onde todas as sentenças ímpares sejam verdadeiras. Note que não importa como você faça isto, as sentenças com números pares também se tornaram verdadeiras. Grave este mundo com o nome **World 3.16.1**. Em seguida construa um mundo em que todas as sentenças ímpares sejam falsas. Novamente, não importa como você faça isto, as sentenças pares também se tornarão falsas. Grave este mundo com o nome **World 3.16.2**.

(3.17) No exercício 3.16 você percebeu um importante fato sobre as sentenças pares e ímpares do arquivo **DeMorgan's Sentencesn**. Tente explicar por que cada sentença par tem sempre o mesmo valor de verdade que a sentença ímpar que a precede.

(3.18) (Equivalências na Linguagem dos Blocos) Na linguagem dos blocos usada no programa **Mundo de Tarski**, há um número de formas equivalentes de expressar alguns predicados. Abra o arquivo **Bernays' Sentences**. Você encontrará uma lista de sentenças atômicas onde todas as sentenças pares estão em branco. Em cada um destes espaços escreva uma sentença que é equivalente à sentença imediatamente acima, mas não use o predicado usado na sentença. (Ao fazer isso, você poderá pressupor todos os fatos gerais que sabe sobre o programa **Mundo de Tarski**, como por exemplo, que os blocos têm apenas três formas, três tamanhos,...) Se suas respostas estiverem corretas, as sentenças pares terão exatamente os mesmos valores de verdade das sentenças ímpares imediatamente anteriores a elas, em qualquer mundo. Verifique se isso ocorre também nos mundos **Ackermann's World**, **Bolzano's World**, **Boole's World** e **Leibniz's World**. Grave o arquivo se sentenças que você modificou com o nome **Sentences 3.18**.

(3.19) (Equivalências em Português) Há também formas equivalentes de expressar predicados em português. Para cada uma das seguintes sentenças de FOL, encontre uma sentença atômica em português que expresse a mesma coisa. Por exemplo, a sentença **Man(max) ∧ ¬Married(max)** poderia ser expressa em português através da sentença atômica *Max é solteiro*.

1. **FatherOf(chris, alex) ∨ MotherOf(chris, alex)**
2. **BrotherOf(chris, alex) ∨ SisterOf(chris, alex)**
3. **Human(chris) ∧ Adult(chris) ∧ ¬Woman(chris)**
4. **Number(4) ∧ ¬Odd(4)**
5. **Person(chris) ∧ ¬Odd(chris)**
6. **mother(mother(alex)) = mary ∨ mother(father(alex)) = mary** [Note que **mother** e **father** são símbolos de função]

(3.20) (Descrevendo um Mundo Simples) Abra o arquivo **Boole's World**. Inicie um arquivo de sentença novo onde você descreverá algumas características deste mundo. Verifique cada uma das sentenças (botão ***Verify***) para certificar-se de que seja de fato uma sentença e verdadeira neste mundo.

1. *Note que* **f** *(o dodecaédro largo no fundo) não está na frente de* **a**. *Use a primeira sentença para afirmar isto.*
2. *Note que* **f** *está à direita de* **a** *e a esquerda de* **b**. *Use sua segunda sentença para dizer isso.*
3. *Use sua terceira sentença para dizer que* **f** *está ou atrás ou é menor do que* **a**.
4. *Expresse que o fato de que ambos,* **e** *e* **d** *estão entre* **c** *e* **a**.
5. *Note que nem* **e** *nem* **d** *é maior do que* **c**. *Use sua quinta sentença para dizer isso.*
6. *Note que* **e** *não é nem maior nem menor do que* **d**. *Afirme isso com sua sexta sentença.*
7. *Note que* **c** *é menor do que* **a** *mas maior do que* **e**. *Declare este fato.*
8. *Note que* **c** *está na frente de* **f**; *e, além disso, é menor do que* **f**. *Diga isso com sua oitava sentença.*

(3.21) (Algumas traduções) O Mundo de Tarski oferece uma forma bastante útil de verificar se sua tradução de uma dada sentença do português está correta. Se ela estiver correta, então ela terá sempre o mesmo valor de verdade que a sentença em português, não importa em que mundo as duas sentenças sejam avaliadas. Então, quando você estiver em dúvida sobre uma de suas traduções, simplesmente construa alguns mundos onde a sentença em português é verdadeira e alguns onde ela é falsa, e verifique se sua sentença traduzida tem os mesmos valores de verdade para estes mundos. Você deve sempre usar esta técnica quando tiver dúvida, em qualquer exercício de tradução.
Inicie um arquivo de sentenças novo e traduza, nele, as seguintes sentenças do português em FOL, utilizando apenas os conectivos $\neg$, $\wedge$ e $\vee$, e os predicados da Linguagem dos Blocos.

1. *Ou **a** é pequeno ou **b** e **c** são ambos grandes.*
2. ***d** e **e** estão ambos atrás de **b**.*
3. ***d** e **e** estão ambos atrás de **b** e são maiores que ele.*
4. ***d** e **c** são ambos cubos, no entanto, nenhum deles é pequeno.*
5. *Nem **e** nem **a** está à direita de **c** e à esquerda de **b**.*
6. *Ou **e** não é grande ou ele está atrás de **a**.*
7. ***c** não está entre **a** e **b** nem na frente deles.*
8. *Ou **a** e **e** são ambos tetraedro ou **a** e **f** que são.*
9. *Nem **d** nem **c** está na frente de **c** ou **b**.*
10. ***c** está entre **d** e **f** ou é menor que ambos.*
11. *Não é o caso que **b** está na mesma linha que **c**.*
12. ***b** está na mesma coluna que **e**, que está na mesma linha que **d**, que, por sua vez, está na mesma coluna que **a**.*

Grave seu arquivo de sentenças com o nome **Sentences 3.21**

(3.22) (Verificando suas traduções) Abra o arquivo **Wittgenstein's World**. Note que todas as sentenças escritas em português do exercício anterior são verdadeiras neste mundo. Então, se você as traduziu corretamente, as sentenças de FOL que você gravou no arquivo **Sentences 3.21** também serão verdadeiras neste mundo. Abra este arquivo de sentenças e verifique se isso ocorre. Se você cometeu algum erro, corrija-o. No entanto, como você já sabe, mesmo que uma de suas sentenças seja verdadeira no mundo de Wittgenstein, isso não significa que ela é uma tradução adequada da sentença do português a ela correspondente. Tudo o que você sabe com certeza é que sua tradução e a sentença original têm o mesmo valor de verdade em um mundo específico. Mas se a tradução está correta, ela terá o mesmo valor de verdade que a sentença em português em ***qualquer*** mundo. Então, para testar melhor suas traduções, iremos examiná-las em um número maior de mundos, para ver se elas têm os mesmos valores de verdade que suas contrapartes escritas em português em todos estes mundos.
Vamos iniciar fazendo algumas modificações no Mundo de Wittgenstein. Em primeiro lugar, transforme todos os objetos grandes ou médios em objetos pequenos, e os objetos pequenos em objetos grandes. Com estas mudanças no mundo, as sentenças 1, 3, 4 e 10 se tornarão falsas, enquanto que as outras continuarão verdadeiras. Verifique se o mesmo ocorre com as suas traduções. Caso isso não ocorra com suas traduções, corrija as que estiverem erradas. Em seguida, gire este mundo de Wittgenstein modificado 90 graus no sentido horário (**CTRL-J**). Agora as sentenças 5, 6, 8, 9 e 11 devem ser as únicas verdadeiras.
Vamos agora verificar suas traduções em um outro mundo. Abra o arquivo **Boole's World**. As únicas sentenças em português verdadeiras neste mundo são as sentenças 6 e 11. Verifique se o mesmo se dá com suas traduções (todas falsas com exceção da 6 e 11). Corrija suas traduções caso haja necessidade.
Agora modifique o Mundo de Boole trocando os objetos **b** e **c** de posição. Com esta troca, as sentenças em português 2, 5, 6, 7 e 11 se tornam verdadeiras, enquanto que o resto é falso. Verifique se o mesmo ocorre com suas traduções.
Este exercício é apenas uma checagem. O único arquivo a gravar é o arquivo **Sentences 3.21** do exercício anterior, eventualmente corrigido pelas verificações que você fez aqui.

(3.23) Inicie um arquivo de sentenças novo e traduza o seguinte em FOL. Use os nomes e predicados presentes na *Tabela 1.2* (nos Exercícios do Capítulo 1).

1. *Max é um estudante, não um animal de estimação.*
2. *Claire alimentou Folly as 2 pm e então, dez minutos depois deu-a para Max.*
3. *Folly pertencia a Max ou Claire as 2:05 pm.*
4. *Nem Max nem Claire alimentaram Folly as 2 pm ou as 2:05 pm.*
5. *2:00 pm está entre 1:55 pm e 2:05 pm.*
6. *Quando Max deu Folly a Claire, as 2 pm, Folly não estava com fome, mas uma hora mais tarde ela estava.*

(3.24) Referindo-se mais uma vez à *Tabela 1.2* (Nos Exercícios do Capítulo 1) traduza as seguintes sentenças para o português natural e coloquial.

1. **Student(claire) ∧ ¬Student(max)**
2. **Pet(pris) ∧ ¬Owned(max, pris, 2:00)**
3. **Owned(claire, pris, 2:00) ∨ Owned(claire, folly, 2:00)**
4. **¬(Fed(max, pris, 2:00) ∧ Fed(max, folly, 2:00))**
5. **((Gave(max, pris, claire, 2:00) ∧ Hungry(pris, 2:00)) ∨ (Gave(max, folly, claire, 2:00) ∧ Hungry(folly, 2:00))) ∧ Angry(claire, 2:05)**

(3.25) Traduza as seguintes sentenças em FOL, introduzindo nomes, símbolos de predicado, e de função conforme a necessidade. Explique o significado de cada símbolo de predicado e função introduzido, amenos que seja completamente óbvio.

1. *AIDS é menos contagiosa do que tuberculose, mas mais mortal.*
2. *Abel enganou Stephen no domingo, mas não na segunda-feira.*
3. *Jean ou Brad admira Meryl e Harrison.*
4. *Daisy é uma moendeira e mora em Salvador.*
5. *O filho mais velho de Poloniu não era nem um prestatário nem um prestamista.*

(3.26) (Superando diferenças de dialeto) Todas as sentenças abaixo são sentenças de FOL. Mas elas estão em diferentes *dialetos*. Inicie um arquivo de sentenças no programa **Mundo de Tarski**, e traduza cada uma destas sentenças para o dialeto que temos utilizado.

1. $\overline{\overline{P} \,\&\, \overline{Q}}$
2. **!(P|(Q && P))**
3. **(~P ∨ Q) . P**
4. **P(~Q ∨ RS)**

(3.27) (Traduzindo da Notação Polonesa) Tente traduzir as seguintes sentenças da notação polonesa para o nosso dialeto FOL.

1. **NKpq**
2. **KNpq**
3. **NAKpqArs**
4. **NAKpAqrs**
5. **NAKApqrs**

(3.28) (Buscas Booleanas) Você provavelmente já ouviu falar, ou usou, ferramentas de busca de dados na Internet que permitem "*buscas booleanas completas*". (por exemplo o site *Google*, ou o *Yahoo*, ou o *Alta Vista*) Isto significa que a linguagem de busca permite a você utilizar os conectivos Booleanos que temos estudado. Antes de fazer a busca, no entanto, você precisa descobrir qual dialeto FOL implementado no mecanismo de busca que você utilizará. Vamos experimentar um site de busca que utiliza os símbolos **sinal de menos "-"** para a negação, **espaço " "** para a conjunção e **barra vertical "|"** para a disjunção.
Utilizando um navegador da Internet abra a página web do Google em http://www.google.com Suponha que você queira encontrar informações sobre a utilização do programa **Mundo de Tarski** em universidades mundo a fora.

1. Digite **tarski** no campo de busca e tecle *ENTER*. Você descobrirá que há centenas de milhares de *sites* que mencionam Tarski, que, afinal de contas, foi um famoso lógico,
2. Digite **"tarski's world"** (com as aspas) no campo de busca e tecle *ENTER*. Agora você encontrará todos os sites que contém as palavras "tarski's world". A quantidade diminui um pouco, mas para um número ainda intratável, com dezenas de milhares de sites. A maioria deles não são sites de universidades, mas notícias, vendedores de livros,... Precisamos exluir de nossa busca os endereços ".com" ou ".org".
3. Digite **"tarski's world" -(domain : com | domain : org)** no campo de busca (Não esqueça dos parenteses e nem dos espaços. Todas as expressões no parênteses são separadas por espaços.) Como resultado, você obterá uma lista de todos os sites que contém a expressão "tarski's world" e cujos endereços não terminam em ".com" ou em ".org" [1]. Você verá que o número de sites da lista diminui para algumas dezenas, tornando possível sua tarefa!!!
4. Digite **"tarski's world" -(domain : com | domain : org) (fitch | proof)** no campo de busca e tecle *ENTER*. Agora você obterá uma lista com todos os sites que contêm "tarski's world", não contêm os domínios ".com" ou ".org" em seus endereços e, além disso, contêm a palavra "fitch" ou a palavra "proof".

[1] Ou seja, os sites da lista não são domínios .com (comerciais) nem .org (organizações). Vale lembrar que mesmo as universidades privadas, não têm domínios .com.

5. Construa uma expressão de busca para encontrar páginas web que contenham referências aos programas "Tarski's World" e "Boole", mas não contenham referências nem a "Fitch" nem a "Submit".

(3.29) (Sólidos Booleanos) Quando fazemos uma busca booleana, nós estamos, de fato, usando uma generalização das funções de verdade booleanas. Nós especificamos uma combinação Booleana de palavras como um critério para encontrar documentos (páginas web,...) que contenham (ou não contenham) aquelas palavras. Uma outra generalização das operações Booleanas se refere a objetos espaciais. Na Figura 3.1 abaixo, mostramos quatro formas de combinar um cilindro vertical (**A**) com um cilindro horizontal (**B**) de modo a formar um novo sólido. Escreva uma explicação intuitiva sobre como os conectivos Booleanos estão sendo aplicados neste exemplo da figura. Então descreva como deveria ser o objeto $\neg(\mathbf{A} \wedge \mathbf{B})$ e explique porque nós não apresentamos a você uma figura deste sólido.

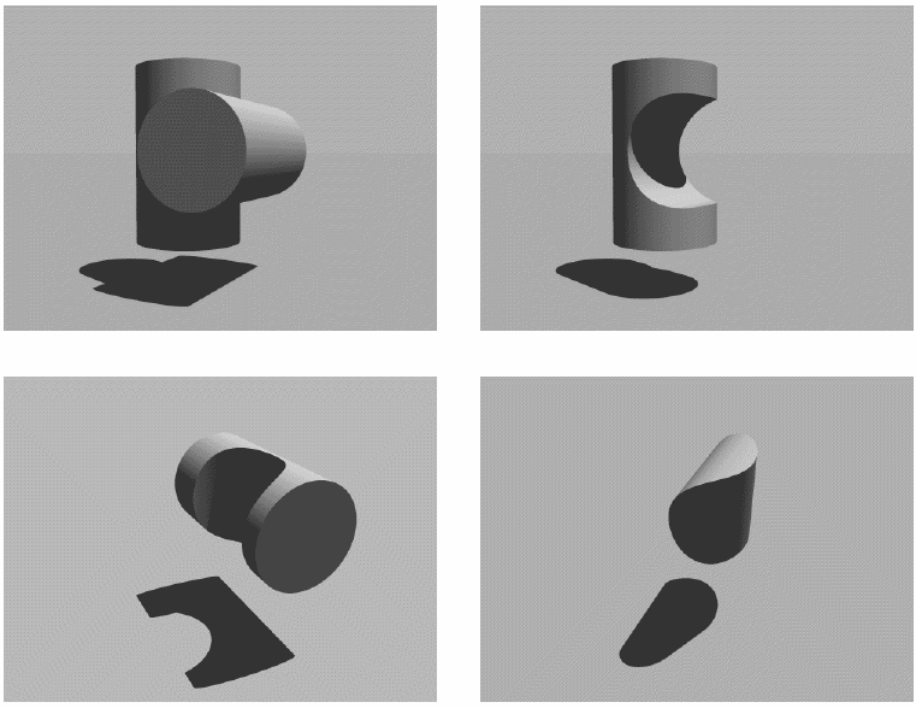


Figura 3.1: combinações Booleanas dos sólidos: $\mathbf{A} \vee \mathbf{B}$, $\mathbf{A} \wedge \neg\mathbf{B}$, $\neg\mathbf{A} \wedge \mathbf{B}$, e $\mathbf{A} \wedge \mathbf{B}$.

Exercícios do Capítulo 4 - (A Lógica dos Conectivos Booleanos)

Experimente (1) . . . . . .

(1) Abra o programa Boole. Vamos usá-lo para construir a tabela de verdade para a sentença **¬(A ∧ (¬A ∨ (B ∧ C)))**. A primeira coisa a fazer é entrar a sentença na parte superior direita da tabela. Para fazer isso, use a barra de ferramentas para entrar os símbolos lógicos e o teclado para digitar as letras **A**, **B** e **C**. (Você também pode digitar os símbolos a partir do teclado, utilizando as teclas **&** para ∧, **|** para ∨ e **~** para ¬. Sempre que entrar os símbolos pelo teclado, separe-os das letras e parênteses com espaços). Se a sentença digitada for bem formada, o pequeno "**(1)**" que aparece acima da sentença se tornará verde.

(2) Para construir as colunas de referência, clique na parte superior esquerda da tabela de modo a mover o ponto de inserção para o topo da primeira coluna de referência. Entre **C** nesta coluna. Então escolha ***Add Column Before*** no menu ***Table*** e entre **B**. Repita este procedimento e adicione a coluna para **A**. Para preencher as colunas de referência, clique abaixo de cada uma delas e digite o padrão desejado de **T**'s e **F**'s. (Lembrese, **V**erdadeiro em inglês é **T**rue. No lugar de **V** você deve colocar **T**)

(3) Clique abaixo dos vários conectivos na sentença alvo (a que queremos avaliar) e note que quadros iluminados cor turquesa aparecem nas colunas cujos valores este conectivo depende. Selecione uma coluna em que os quadros iluminados já estejam preenchidos e preencha a coluna com os valores de verdade apropriados. Continue o processo até que sua tabela esteja completa. Quando tiver terminado, clique no botão ***Verify Table*** para ver se todos os valores estão corretos e a tabela completa.

(4) Uma vez que você tenha uma tabela correta e completa, clique no botão ***Assessment***, logo abaixo da barra de ferramentas. Isto irá permitir-lhe dizer se você acha se a sentença é uma tautologia (***tautology***). Confirme que a sentença é uma tautologia, marcando o quadro correspondente e clique em ***OK*** para fechar esta janela. Agora, clique no botão ***Verify Assess*** para verificar se você interpretou corretamente a tabela. Grave seu trabalho com o nome ***Table Tautology 1***.

Exercícios . . . . . .

Neste capítulo, você usará o programa **Boole** *para construir tabelas de verdade. Apesar de Boole ter a capacidade de construir e preencher as colunas de referência automaticamente para você,* ***não use esta ferramenta*** *para fazer os exercícios deste capítulo. Em capítulos mais avançados, tal utilização será permitida, uma vez que você já tiver aprendido a construir as tabelas por si próprio. A propósito, o professor tem como saber de que forma as colunas de referência foram obtidas nos exercícios que vocês entregarem. Além disso, a prova será feita no papael, ou seja, sem software nenhum para construir colunas automaticamente para você.*

(4.1) Se você pulou a seção *Experimente 1*, acima, volte e faça-a. Ela o ajuda aprender a utilizar o programa Boole.

(4.2) Assuma que **A**, **B** e **C** são sentenças atômicas. Use o programa **Boole** para construir tabelas de verdade para cada uma das seguintes sentenças e, com base nas tabelas indique quais sentenças são tautologias. Grave suas tabelas com os nomes **Table 4.2.x**, onde **x** é o número da sentença.

1. **(A ∧ B) ∨ (¬A ∨ B)**
2. **(A ∧ B) ∨ (A ∧ ¬B)**
3. **¬(A ∧ B) ∨ C**
4. **(A ∨ B) ∨ ¬(A ∨ (B ∧ C))**

(4.3) No *Exercício 4.2* você deve ter descoberto que duas das quatro sentenças são tautologias, e portanto, são verdades lógicas.

1. Suponha que você seja informado de que a sentença **A** é, de fato, uma verdade lógica (por exemplo, **a = a**). Com base nesta informação, descubra se alguma sentença adicional da lista acima (1 – 4) é uma necessidade lógica (verdade lógica).
2. Agora suponha que você seja informado de que a sentença **A** é, de fato, uma sentença logicamente falsa (por exemplo, **a ≠ a**). Com base nesta informação, descubra se alguma sentença adicional da lista acima (1 – 4) é uma necessidade lógica.

Para os exercícios abaixo, use o programa ***Boole*** *para construir tabelas de verdade para as sentenças, indicando se são TT-possíveis ou tautologias. Lembre-se de como deve tratar conjunções e disjunções longas.*

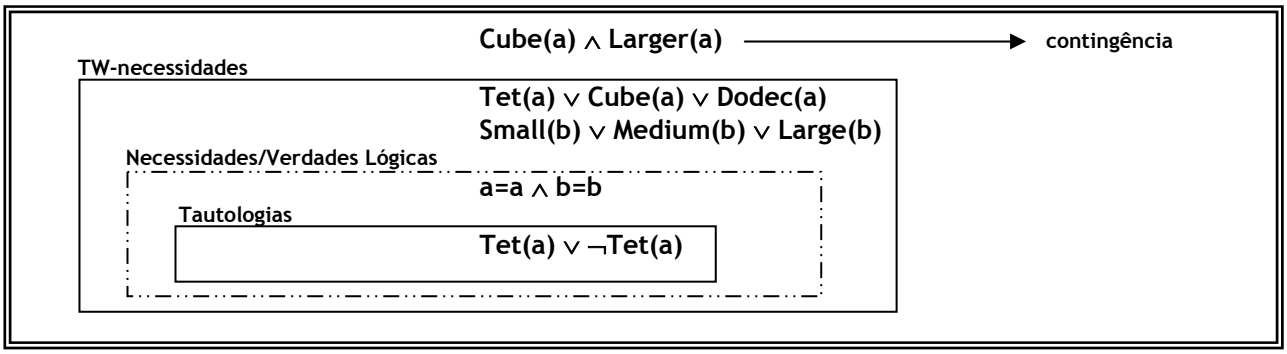
(4.4) $\neg(B \wedge \neg C \wedge \neg B)$

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

(4.6) $\neg[\neg A \vee \neg(B \wedge C) \vee (A \wedge B)]$

(4.7) $\neg[(\neg A \vee B) \wedge \neg(C \wedge D)]$

(4.8) Considere o diagrama abaixo, onde classificamos as sentenças conforme elas sejam contingências (fora de tudo), TW-necessidades (quadro maior), necessidades lógicas (quadro do meio, tracejado) ou tautologias (quadro menor)



Faça uma cópia deste diagrama e coloque, cada uma das sentenças seguintes na região apropriada.

1. **a=a**
2. **a=b ∨ b=b**
3. **a=b ∧ b=b**
4. **¬(Large(a) ∧ Large(b) ∧ Adjoins(a,b))**
5. **Larger(a,b) ∨ ¬Larger(a,b)**
6. **Larger(a,b) ∨ Smaller(a,b)**
7. **¬Tet(a) ∨ ¬Cube(b) ∨ a≠b**
8. **¬(Small(a) ∧ Small(b)) ∨ Small(a)**
9. **SameSize(a,b) ∨ ¬(Small(a) ∧ Small(b))**
10. **¬(SameCol(a,b) ∧ SameRow(a,b))**

(4.9) (Dependências Lógicas) Use o programa **Mundo de Tarski** para abrir o arquivo **Weiner's Sentences**.

1. Para cada uma das dez sentenças neste arquivo, construa uma tabela de verdade no programa **Boole** e indique se ela é uma sentença TT-possível. Grave suas tabelas com os nomes **Table 4.9.x**, onde **x** é o número da sentença em questão. Use os resultados para preencher a primeira coluna da seguinte tabela:

Sentença	TT-possível	TW-possível
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

2. Na segunda coluna da tabela, coloque SIM se você acha que a sentença é TW-possível, ou seja, se é possível construir um mundo com o programa **Mundo de Tarski** em que a sentença seja verdadeira. Caso considere que a sentença não seja TW-possível, coloque NÃO na segunda coluna para ela. Para cada sentença que você considerar TW-possível, construa, de fato, um mundo no qual ela seja verdadeira e grave este mundo com o nome **World 4.9.x**, onde **x** é o número da sentença. As tabelas de verdade que você construiu anteriormente poderão ajudá-lo na construção destes mundos. Experimente.
3. Existe alguma sentença que é TT-possível mas não é TW-possível? Explique por que isso pode acontecer. Existe alguma sentença TW-possível mas não TT-possível? Exlplique por que não.

(4.11) (Este é Difícil !) Suponha que **S** seja uma tautologia construída a partir de três sentenças atômicas (**A**, **B** e **C**). Suponha que substituamos todas as ocorrências de **A** por outra sentença **P**, possivelmente complexa (não atômica). Explique por que a sentença resultante desta substituição continua sendo uma tautologia. (Costumamos expressar isso dizendo que substituição preserva *tautologicidade*). Explique também por que a substituição de sentenças atômicas, mesmo preservando tautologias, nem sempre preserva necessidade lógica.

Ou seja apresente uma sentença que seja logicamente necessária e que, substituindo-se uma de suas sentenças atômicas por outra sentença qualquer, o resultado deixa de ser uma necessidade lógica.

*Nos dois Exercícios 4.12 a 4.18 abaixo, utilize o programa **Boole** para construir tabelas de verdade conjuntas para mostrar que cada par de sentenças é logicamente equivalente (de fato, tautologicamente equivalente). Para adicionar a segunda sentença na sua tabela, escolha **Add Column After** a partir do menu **Table**. Não esqueça de clicar no botão **Assessment** e marcar se as sentenças são ou não tautologicamente equivalentes.*

4.12 (DeMorgan)
$\neg(A \vee B)$ and $\neg A \wedge \neg B$

4.13 (Associativity)
$(A \wedge B) \wedge C$ and $A \wedge (B \wedge C)$

4.14 (Associativity)
$(A \vee B) \vee C$ and $A \vee (B \vee C)$

4.15 (Idempotence)
$A \wedge B \wedge A$ and $A \wedge B$

4.16 (Idempotence)
$A \vee B \vee A$ and $A \vee B$

4.17 (Distribution)
$A \wedge (B \vee C)$ and $(A \wedge B) \vee (A \wedge C)$

4.18 (Distribution)
$A \vee (B \wedge C)$ and $(A \vee B) \wedge (A \vee C)$

(4.19) (TW-equivalências) Suponha que nós introduzimos a noção de ***TW-equivalência***, dizendo que duas sentenças da linguagem dos blocos são TW-equivalentes se e somente se elas têm exatamente os mesmos valores de verdade em cada mundo que pode ser construído com o programa **Mundo de Tarski**.

1. Qual a relação entre TW-equivalência, equivalência tautológica e equivalência lógica?
2. Dê um exemplo de um par de sentenças que sejam TW-equivalentes mas não sejam logicamente equivalentes.

*Para cada um dos argumentos abiaxo, use o método das tabelas de verdade para determinar se a conclusão é uma conseqüência tautológica de suas premissas. A tabela do Exercício 4.24 ficará ENORME. Mas é bom para a alma construir longas tabelas de verdade de vez em quando. Agradeça por poder utilizar o programa **Boole** para ajudá-lo. Se for fazer as tabelas à mão. Nem tente fazer o Exercício 4.24. Você perderá muito tempo e talvez até o juízo.*

4.20
$\mathrm{Tet}(a) \wedge \mathrm{Small}(a)) \vee \mathrm{Small}(b)$
$\therefore$ $\mathrm{Small}(a) \vee \mathrm{Small}(b)$

4.21
$\mathrm{Taller}(\mathrm{claire}, \mathrm{max}) \vee \mathrm{Taller}(\mathrm{max}, \mathrm{claire})$
$\mathrm{Taller}(\mathrm{claire}, \mathrm{max})$
$\therefore$ $\neg\mathrm{Taller}(\mathrm{max}, \mathrm{claire})$

4.22
$\mathrm{Large}(a)$
$\mathrm{Cube}(a) \vee \mathrm{Dodec}(a)$
$\therefore$ $(\mathrm{Cube}(a) \wedge \mathrm{Large}(a)) \vee (\mathrm{Dodec}(a) \wedge \mathrm{Large}(a))$

4.23
$A \vee \neg B$
$B \vee C$
$C \vee D$
$\therefore$ $A \vee \neg D$

4.24
$\neg A \vee B \vee C$
$\neg C \vee D$
$\neg(B \wedge \neg E)$
$\therefore$ $D \vee \neg A \vee E$

(4.25)* Apresente um exemplo de duas sentenças distintas, **A** e **B** da linguagem dos blocos para as quais a sentença $\mathbf{A \wedge B}$ é conseqüência lógica de $\mathbf{A \vee B}$. [**Dica**: Note que $\mathbf{A \wedge A}$ é uma conseqüência lógica de $\mathbf{A \vee A}$, mas aqui nós insistimos que **A** e **B** devem ser sentenças distintas.]

Experimente (2) . . . . . .

(1) Inicie o programa **Fitch** e abra o arquivo **Taut Con 1**. Neste arquivo você encontrará um argumento que tem a mesma forma do argumento do exercício 4.23. (Ignore as duas metas – na janela de baixo – mais tarde nos ocuparemos delas). Posicione o cursor de foco no último passo da prova. No menu ***Rule?***, escolha a regra **(TautCon)**.

(2) Agora cite as três premissas como suporte para este sentença e verifique o passo. A verificação deve ter retornado um erro, uma vez que esta sentença não é uma conseqüência tautológica de suas premissas, conforme você deve ter descoberto ao fazer o *Exercício 4.23*, que tem a mesma forma que esta inferência.

(3) Substitua esta última sentença da prova por: **Home(max) ∨ Hungry(carl).** Esta sentença é uma conseqüência tautológica de duas das premissas. Descubra quais são, cite-as e verifique o passo. Se você acertou, a verificação deve retornar OK. Experimente.

(4) Adicione outro passo à prova e insira nele a seguinte sentença:

Hungry(carl) ∨ (Home(max) ∧ Hungry(pris))

Use **(TautCon)** para ver se esta sentença é uma conseqüência tautológica das três premissas. Pressione ***Verify Proof***. Você descobrirá que mesmo com os passos OK, a meta final falhou. Isto se dá porque foi colocada uma restrição especial para o uso de **(TautCon)** neste exercício.

(5) Pressione o botão ***Goal Constraints***. Você descobrirá que nesta prova é permitido usar **(TautCon)** citando no máximo duas premissas como suporte. Feche a janela de metas e retorne à prova.

(6) A sentença que você digitou se segue da sentença imediatamente acima dela e de apenas uma das três premissas. Desfaça as citações das três premissas, cite a sentença imediatamente acima desta última e tente descobrir qual é a premissa que falta para obtermos a conseqüência tautológica. Quando tiver conseguido, grave seu trabalho com o nome **Proof Taut Con 1**.

Experimente (3) . . . . . .

(1) Com o programa **Fitch** abra o arquivo **Taut Con 2**. Você encontrará uma prova contendo 10 passos cujas regras não foram especificadas.

(2) Coloque o cursor de foco em cada passo. Um de cada vez. Você descobrirá que os passos de suporte a cada um deles já foram citados. Para cada um dos passos, tente se convencer de que ele se segue das sentenças citadas. O passo em questão é uma conseqüência tautológica das sentenças citadas? Se for, indique isso escolhendo **(TautCon)** como a regra que justifica o passo e verifique se você está certo (botão ***Check Step***) Se não for conseqüência tautológica, indique a regra que o justifica como **(AnaCon)** e verifique o passo.[1]

(3) Quando a verificação de todos os passos com as regras **(TautCon)** ou **(AnaCon)** estiver OK, volte e tente descobrir qual é o único passo justificado com **(AnaCon)** que você pode trocar por **(FOCon)**.

(4) Quando cada passo estiver justificado com a regra **Con** mais fraca possível, grave seu trabalho com o nome **Proof Taut Con 2**.

Exercícios . . . . . .

*Para cada um dos argumentos a seguir, decida se a conclusão é uma conseqüência tautológica das premissas. Se for, faça uma prova que estabeleça a conclusão utilizando uma ou mais aplicações de **(TautCon)**. Não cite mais de duas sentenças em cada aplicação de **(TautCon)**. Se a conclusão não for conseqüência das premissas, faça um contraexemplo no **(Mundo de Tarski)** mostrando que o argumento não é válido.*[2]

---

[1] Lembre-se que como **(AnaCon)** é mais forte do que **(TautCon)**, se você colocar em todos os passos **(AnaCon)** como justificativa, a verificação de todos eles será correta. No entanto, o que queremos aqui é que você identifique qual a regra mais fraca de todas que justifica cada um dos passos. Assim, onde for possível justificar com **(TautCon)** é esta regra que você deve usar.

[2] Lembre-se que um contra-exemplo deve ser apresentado através de dois arquivos do programa **Mundo de Tarski**, um arquivo de **Mundo** e um arquivo de **Sentenças**. No arquivo de sentenças devem estar todas as sentenças do argumento, no arquivo de **Mundo** deve ter um mundo em que todas as premissas do argumento são verdadeiras e a conclusão é falsa.

(4.26) Se você pulou as seções *Experimente 2 e 3*, acima, volte e faça-as agora. Elas são passos importantes para que você consiga fazer os próximos exercícios.

4.27

$$\begin{array}{l} \mathsf{Cube(a) \vee Cube(b)} \\ \mathsf{Dodec(c) \vee Dodec(d)} \\ \mathsf{\neg Cube(a) \vee \neg Dodec(c)} \\ \hline \mathsf{Cube(b) \vee Dodec(d)} \end{array}$$

4.28

$$\begin{array}{l} \mathsf{Large(a) \vee Large(b)} \\ \mathsf{Large(a) \vee Large(c)} \\ \hline \mathsf{Large(a) \wedge (Large(b) \vee Large(c))} \end{array}$$

4.29

$$\begin{array}{l} \mathsf{Small(a) \vee Small(b)} \\ \mathsf{Small(b) \vee Small(c)} \\ \mathsf{Small(c) \vee Small(d)} \\ \mathsf{Small(d) \vee Small(e)} \\ \mathsf{\neg Small(c)} \\ \hline \mathsf{Small(a) \vee Small(e)} \end{array}$$

4.30

$$\begin{array}{l} \mathsf{Tet(a) \vee \neg(Tet(b) \wedge Tet(c))} \\ \mathsf{\neg(\neg Tet(b) \vee \neg Tet(d))} \\ \mathsf{(Tet(e) \wedge Tet(c)) \vee (Tet(c) \wedge Tet(d))} \\ \hline \mathsf{Tet(a)} \end{array}$$

Exercícios do Capítulo 5 - (Métodos de Prova para a Lógica Booleana)

Nos exercícios seguintes, há uma lista de padrões de inferência. Apenas alguns deles são válidos. Para cada padrão determine se ele é válido ou não. Caso seja válido, explique por que, apelando para as tabelas de verdade dos conectivos envolvidos. Caso o padrão não seja válido, apresente um exemplo específico de como, partindo de premissas verdadeiras, a inferência sugerida poderia ser usada para levar a uma conclusão falsa.

(5.1) De $P \vee Q$ e $\neg P$, infere-se Q.
(5.2) De $P \vee Q$ e Q, infere-se $\neg P$.
(5.3) De $\neg(P \vee Q)$ infere-se $\neg P$.
(5.4) De $\neg(P \wedge Q)$ e P infere-se $\neg Q$.
(5.5) De $\neg(P \wedge Q)$ infere-se $\neg P$.
(5.6) De $P \wedge Q$ e $\neg P$ infere-se R.

Os dois exercícios seguintes apresentam argumentos válidos. Faça provas informais de suas validades. As provas devem ser escritas em sentenças completas do português, podendo utilizar-se de sentenças em FOL conforme for conveniente, seguindo o estilo que temos feito no texto do capítulo. Sempre que você utilizar prova por casos, indique que está fazendo isto. Você não precisa ser explícito sobre o uso de passos simples. A propósito em geral há mais de uma forma de se provar um resultado.

5.7 ✎

$$\begin{array}{l} \text{Home(max)} \vee \text{Home(claire)} \\ \neg\text{Home(max)} \vee \text{Happy(carl)} \\ \neg\text{Home(claire)} \vee \text{Happy(carl)} \\ \hline \text{Happy(carl)} \end{array}$$

5.8 ✎

$$\begin{array}{l} \text{LeftOf(a, b)} \vee \text{RightOf(a, b)} \\ \text{BackOf(a, b)} \vee \neg\text{LeftOf(a, b)} \\ \text{FrontOf(b, a)} \vee \neg\text{RightOf(a, b)} \\ \text{SameCol(c, a)} \wedge \text{SameRow(c, b)} \\ \hline \text{BackOf(a, b)} \end{array}$$

(5.9) Assuma as mesmas quatro premissas do *Exercício 5.8*. A sentença **LeftOf(b, c)** é uma conseqüência lógica destas premissas? Se for, faça uma prova informal. Se não for, apresente um contraexemplo no **Mundo de Tarski**.

(5.10) Suponha que o time de baskete favorito de Max é o Chicago Bulls e que seu time favorito de futebol americano é o Denver Broncos. John, o pai de Max, está voltando de Indianápolis para São Francisco pela companhia United Airlines, e prometeu que ele compraria para Max um souvenir de um de seus times favoritos no caminho. Explique o raciocínio de John (que o assegura que ele conseguirá comprar um souvenir de um dos times favoritos de Max) apelando para o fato significativo de que todos os vôos da United entre Indianápolis e São Francisco fazem escala em Denver ou Chicago. Torne explícito o papel da prova por casos nesta inferência.

(5.11) Suponha que a polícia esteja investigando um roubo e descubra os seguintes fatos. Todas as portas da casa foram trancadas pelo lado de dentro e não apresentam nenhum sinal de arrombamento. De fato, as únicas formas possíveis de entrar e sair da casa eram através de uma pequena janela do banheiro do térreo, que foi deixada aberta, e uma janela do banheiro do piso superior que estava destrancada. Com base nisso, os detetives descartaram as suspeitas sobre um famoso assaltante, Julius, que pesa 120 kg e tem artrite. Explique a inferência dos policiais.

(5.12) Em nossa prova de que há números irracionais b e c tais que b^c é racional, um dos passos foi afirmar que $\sqrt{2}^{\sqrt{2}}$ é ou racional ou irracional. Qual a justificativa para aceitar esta afirmação em nossa prova?

(5.13) Descreva um raciocínio por casos que você tenha feito ultimamente sobre algum assunto do dia-a-dia.

(5.14) Faça uma prova informal de que se S é conseqüência tautológica de P e também uma conseqüência tautológica de Q, então S é uma conseqüência tautológica de $P \vee Q$. Lembre-se de que a tabela de verdade conjunta para $P \vee Q$ e S pode ter mais linhas do que a tabela conjunta para P e S ou do que a tabela conjunta para Q e S. [*Sugestão*: assuma que você está procurando por uma única linha da tabela de verdade conjunta para $P \vee Q$ e S na qual $P \vee Q$ é verdadeira. Divida a prova em casos conforme se P é verdadeiro ou Q é verdadeiro e prove que S deve ser verdadeiro em ambos os casos.]

Nos exercícios seguintes, decida se cada argumento apresentado é válido. Se for, faça uma prova informal. As provas devem ser escritas em sentenças completas do português, podendo utilizar-se de sentenças em FOL conforme for conveniente, seguindo o estilo que temos feito no texto do capítulo. Sempre que você utilizar prova por casos, ou por contradição, indique que está fazendo isto. Você não precisa ser explícito sobre o uso de passos simples (introdução e eliminação da conjunção e introdução da disjunção). Se o argumento for inválido, construa um contraexemplo com o programa ***Mundo de Tarski.***

(5.15)

b é um tetraédro.
c é um cubo.
Ou ***c*** é maior do que ***b*** ou eles são iguais.
b é menor do que ***c***.

(5.16)

Max ou Claire está em casa mas Scruffy ou Carl está infel
Ou Max não está em casa, ou Carl está infeliz.
Ou Claire não está em casa ou Scruffy está infeliz
Scruffy está infeliz.

5.17

$$\begin{array}{l} \mathrm{Cube(a)} \vee \mathrm{Tet(a)} \vee \mathrm{Large(a)} \\ \neg\mathrm{Cube(a)} \vee \mathrm{a} = \mathrm{b} \vee \mathrm{Large(a)} \\ \neg\mathrm{Large(a)} \vee \mathrm{a} = \mathrm{c} \\ \neg(\mathrm{c} = \mathrm{c} \wedge \mathrm{Tet(a)}) \\ \hline \mathrm{a} = \mathrm{b} \vee \mathrm{a} = \mathrm{c} \end{array}$$

5.18

$$\begin{array}{l} \mathrm{Cube(a)} \vee \mathrm{Tet(a)} \vee \mathrm{Large(a)} \\ \neg\mathrm{Cube(a)} \vee \mathrm{a} = \mathrm{b} \vee \mathrm{Large(a)} \\ \neg\mathrm{Large(a)} \vee \mathrm{a} = \mathrm{c} \\ \neg(\mathrm{c} = \mathrm{c} \wedge \mathrm{Tet(a)}) \\ \hline \neg(\mathrm{Large(a)} \vee \mathrm{Tet(a)}) \end{array}$$

(5.19) Considere as seguintes sentenças e responda:

1. *Folly era animal de estimação de Claire às 2 pm ou às 2:05 pm.*
2. *Folly era animal de estimação de Max às 2 pm.*
3. *Folly era animal de estimação de Claire às 2:05 pm.*

A sentença (3) é conseqüência de (1) e (2)?
A sentença (2) é conseqüência de (1) e (3)?
A sentença (1) é conseqüência de (2) e (3)?

Em cada caso apresente ou uma prova (quando a sentença for conseqüência das outras) ou descreva uma situação que torne as premissas verdadeiras e a conclusão falsa (um contraexemplo). Você deve assumir que Folly é animal de estimação de apenas uma pessoa de cada vez.

(5.20) Suponha que seja sexta-feira a noite e você vai sair com seu namorado (ou namorada). Ele(a) quer ir ao cinema assistir uma comédia romântica, enquanto você quer assistir o último filme da série *O Massacre da Serra Elétrica*. Ele(a) argumenta que se for assistir o *Massacre da Serra Elétrica*, não conseguirá dormir, pois não suporta ver sangue, e ele(a) tem exame do ENEM amanhã cedo. Se não for bem no exame, ele(a) não fará pontos suficientes para entrar em Medicina. Analise o argumento de seu (sua) namorado(a), apontando onde o método de prova indireta está sendo utilizado. Como você poderia rebater este argumento?

(5.21) Descreva um exemplo do dia-a-dia de uso do método de prova indireta (por absurdo) em um argumento que você tenha feito nos últimos dias.

(5.22) Prove que o método de prova indireta (por absurdo) é um método de prova tautologicamente válido. Ou seja, mostre que se $\mathbf{P_1}, \ldots, \mathbf{P_n}, \mathbf{S}$ são TT-contraditórios, então $\neg\mathbf{S}$ é conseqüência tautlógica de $\mathbf{P_1}, \ldots, \mathbf{P_n}$.

Nos três exercícios seguintes você será solicitado a provar fatos simples sobre os números naturais. Você não precisa formalizar sua prova em FOL. Basta escrevê-la em português. Você podera apelar para fatos básicos da aritmética além das definições de números pares e impares. Sempre que fizer isso, aponte explicitamente em sua prova. Também torne explícito qualquer uso dos métodos de prova por casos ou por contradição.

(5.23) Assuma que n^2 é impar. Prove que **n** é impar.

(5.24) Assuma que $n+m$ é impar. Prove que $n \times m$ é par.

(5.25) Assuma que n^2 é divisível por 3. Prove que n^2 é divisível por 9.

(5.26) Uma boa maneira de se certificar de que você entendeu uma prova é tentar generalizá-la. Prove que $\sqrt{3}$ é irracional. [Sugestão: você precisará utilizar alguns fatos sobre a divisibilidade por 3, tais como os expressos no *Exercício 5.25*]

(5.27) Faça duas provas diferentes de que as premissas do argumento abaixo são inconsistentes. Sua primeira prova deve utilizar o método de prova por casos, mas não utilizar nenhuma lei de DeMorgan. Ao passo que sua segunda prova deve utilizar DeMorgan, mas não o método de prova por casos.

$$\begin{array}{l} \mathrm{Home(max)} \vee \mathrm{Home(claire)} \\ \neg\mathrm{Home(max)} \\ \neg\mathrm{Home(claire)} \\ \hline \mathrm{Home(max)} \wedge \mathrm{Happy(carl)} \end{array}$$

Exercícios do Capítulo 6 - (Provas Formais e Lógica Booleana)

Experimente (1) . . . . . .

(1) Abra (com o programa **Fitch**) o arquivo **Conjunction 1**. Existem três sentenças que você será solicitado a provar. Elas são mostradas na faixa de metas, na parte de baixo da janela de provas.

(2) A primeira sentença que você vai provar é **Tet(a)**. Para fazer isso, inicialmente adicione um passo novo na prova (**ctrl A**) e entre a sentença **Tet(a)**.

(3) Em seguida, clique em "*Rule ?*" para abrir o menu de regras, escolha "*Elimination Rules"* e selecione "∧".

(4) Se você tentar verificar este passo (no botão "*check step*"), você notará que a checagem falhará, porque você ainda não citou as sentenças de suporte a este passo. Neste exemplo, você precisa citar apenas a premissa como suporte desta aplicação de regra. Faça isso (clicando na premissa) e verifique este passo (clicando novamente no botão "*check step*").

(5) Você deve, agora, ser capaz de provar, similarmente, cada uma das outras sentenças, cada uma delas através de uma única aplicação da regra ∧Elim. Quando você provar estas sentenças, verifique suas provas (clicando no botão *"Verify Proof"*) e grave seu trabalho com o nome **Proof Conjunction 1**.

Experimente (2) . . . . . .

(1) Abra o arquivo **Conjunction 2**. Vamos provar as duas sentenças metas utilizando as regras de introdução e eliminação da conjunção.

(2) A primeira meta é **Medium(d) ∧ Large(c)**. Adicione um passo à prova (**ctrl-A**) e entre esta sentença. (Lembre-se que você pode copiar e colar a sentença)

(3) Acima do passo que você acabou de criar, adicione 2 passos mais (**ctrl-B**) e entre cada uma das partes dessa conjunção em cada passo adicionado (um deles terá **Medium(d)** e o outro **Large(c)**). Se você conseguir provar estes dois passos, a conclusão se seguirá por (∧**Intro)**. Escolha esta regra para a sentença final e cite, como suporte a aplicação desta regra, as duas sentenças anteriores com as partes da conjunção.

(4) Agora tudo o que você precisa fazer é provar cada uma desta duas sentenças. O que é facilmente obtido utilizando-se a regra (∧**Elim**) em cada um destes passos. Faça isso, citando corretamente as sentenças de suporte e verifique sua prova (no botão *"Verify Proof"*). A primeira meta deve receber uma marca azul de OK.

(5) Prove a segunda meta de modo similar a este. Quando terminar verifique sua prova e grave seu trabalho com o nome **Proof Conjunction 2**.

Experimente (3) . . . . . .

(1) Abra o arquivo **Conjunction 3**. Note que há duas metas a serem provadas. A primeira pede que você prove **Tet(c) ∧ Tet(a)** a partir da premissa. Estritamente falando, isso exigiria que você utilizasse duas vezes a regra (∧**Elim**) seguida por um uso de (∧**Intro**). No entanto, o programa **Fitch** permite que você faça esta prova utilizando apenas uma vez a regra (∧**Elim**). Tente fazer isso e verifique (com o botão *"Check Step"*)

(2) Verifique que a segunda meta também é obtida através de uma única aplicação da regra (∧**Elim**). Quando tiver provado essas sentenças, verifique sua prova (botão *"Verify Proof"*) e grave seu trabalho com o nome **Proof Conjunction 3**.

(3) Em seguida, experimente outras sentenças para ver se elas se seguem da sentença através da regra (∧**Elim**). Por exemplo, a sentença **Tec(c) ∧ Small(a)** se segue? Deveria se seguir?

(4) Quando achar que já entendeu os usos da regra de eliminação da conjunção, feche o arquivo (menu *"File - Close"*) mas não grave as modificações que fez no passo 3.

Experimente (4) . . . . . .

(1) Abra o arquivo **Conjunction 4**. Posicione o cursor no primeiro passo em branco da prova, logo em seguida das premissas. Note que este passo tem uma regra especificada e também uma sentença de suporte a esta regra citada. Verifique este passo (pressionando o botão *"Check Step"*) para ver qual é a sentença default que o programa **Fitch** gera.

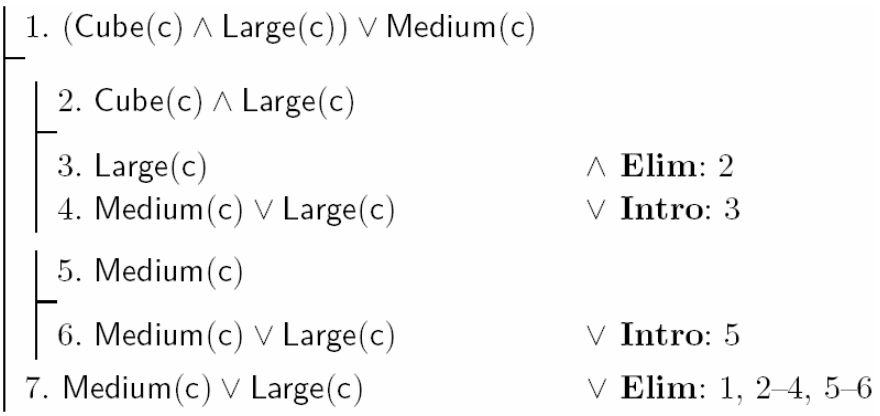
(2) Em seguida, posicione o cursor no passo seguinte, tente predizer qual é a sentença default que será preenchida e verifique o passo. (Os dois últimos passos apresentam diferentes resultados porque as suas sentenças de suporte foram citadas em ordens diferentes.)

(3) Quando tiver verificado todos os passos, grave sua prova com o nome **Proof Conjunction 4**.

(4) Sinta-se a vontade para experimentar um pouco mais com os valores defaults das regras. Assim você perceberá melhor em que ocasiões seu uso será útil.

Experimente (5) . . . . . .

(1) Abra o arquivo **Disjunction 1**. Você será solicitado a provar a sentença **Medium(c)** $\vee$ **Large(c)** a partir da sentença **(Cube(c)** $\wedge$ **Large(c))** $\vee$ **Medium(c)**. Vamos executar passo a passo a construção desta prova, que ao final terá o seguinte aspecto:



(2) Para utilizar a regra ($\vee$**Elim**) neste caso, nós precisaremos de duas subprovas, uma para cada um dos disjuntos da premissa. Uma boa estratégia é iniciar já especificando as duas subprovas necessárias, antes de qualquer outro passo. Para iniciar uma subprova, adicione um passo novo e selecione *"New Subproof"* no menu *"Proof"* (as teclas **ctrl-P**, são um atalho para executar esta tarefa). O programa **Fitch** move o cursor para a posição da hipótese da subprova, permitindo que você entre a sentença que deseja. Entre o primeiro disjunto de sua premissa inicial (ou seja, entre a sentença **Cube(c)** $\wedge$ **Large(c)**). Esta sentença será a hipótese (a premissa) desta subprova.

(3) Antes de trabalhar nesta subprova, vamos agora especificar o segundo caso, (a segunda subprova) para evitar que esqueçamos o que estávamos fazendo. Para fazer isso, precisamos, inicialmente, finalizar a primeira subprova e iniciar a segunda. Finalize a primeira subprova através da opção *"End Subproof"* do menu *"Proof"* (um atalho para fazer isso são as teclas **ctrl-E**). Isso colocará um novo passo fora e imediatamente seguinte à subprova.

(4) Inicie a segunda subprova neste passo escolhendo a opção *"New Subproof"* no menu *"Proof"* (ou digitando o atalho **ctrl-P**). Entre agora o segundo disjunto da premissa inicial (a sentença **Medium(c)**). Temos agora especificadas as duas hipóteses dos dois casos que precisamos considerar. Nosso objetivo é provar que a conclusão desejada (a sentença **Medium(c)** $\vee$ **Large(c)**) se segue destes dois casos.

(5) Retorne o cursor para a primeira subprova e adicione um passo logo em seguida da hipótese (basta posicionar o cursor na hipótese da subprova e digitar **ctrl-A** ou selecionar a opção *"Add Step After"* no menu *"Proof"*). Use, neste passo a regra ($\wedge$**Elim**) para provar **Large(c)**. Então, adicione outro passo a subprova e prove a sentença desejada (nossa meta) utilizando a regra ($\vee$**Intro**). Nestes dois passos você precisa citar adequadamente as sentenças de suporte a aplicação destas regras.

(6) Depois de ter terminado a primeira subprova e de todos os passos terem sido verificados (com o botão *"Check Step"*), mova o cursor para a hipótese da segunda subprova e adicione um novo passo. Utilize a regra (∨**Intro**) para provar a sentença da meta a partir desta hipótese.

(7) Agora que já obtivemos nossa meta nas duas subprovas, já estamos prontos para adicionar o passo final da prova. Com o cursor no último passo da segunda subprova, selecione *"End Subproof"* no menu*"Proof"* (ou, com o cursor no mesmo passo, digite o atalho **ctrl-E**). Entre a sentença da meta no passo novo que apareceu depois da segunda subprova.

(8) Especifique a regra do passo final como (∨**Elim**). Para suporte a esta regra, cite as duas subprovas e a premissa. Verifique sua prova completa (no botão *"Verify Proof"*). Se houver alguma marca de checagem vermelha, compare sua prova com a prova mostrada acima e tente corrigir os erros.

(9) Quando a verificação da prova estiver OK, grave seu trabalho com o nome **Proof Disjunction 1**.

Experimente (6)

(1) Abra o arquivo **Disjunction 2**. O objetivo é provar a sentença **(Cube(b)∧Small(b))∨(Cube(b)∧Large(b))**. A prova final está quase completa, ainda que não pareça.

(2) Posicione o cursor em cada um dos passos vazios, em seqüência, e verifique cada passo (botão *"Check Step"*), para que o **Fitch** preencha-os com as sentenças defaults das regras associadas. No segundo passo vazio, você terá que terminar a sentença digitando o segundo disjunto (**Cube(b)∧Large(b)**) da sentença. (Se o último passo não produzir uma sentença default, é porque você não digitou corretamente o disjunto no passo de (∨**Intro**).)

(3) Quando você terminar, verifique a prova. Você entendeu esta prova? Você conseguiria fazê-la sozinho?

(4) Grave seu trabalho com o nome **Proof Disjunction 2**.

Exercícios

(6.1) Não deixe de fazer as seções *Experimente 1 a 6* acima. Elas são MUITO importantes!

(6.2) Abra o arquivo **Exiercise 6.2**, que contém uma prova formal incompleta. Do jeito que ela está, nenhum dos passos está completo, ou porque a regra não foi especificada, ou porque não há sentenças de suporte às regras citadas, ou porque não há sentenças nos passos. Coloque as peças que estão faltando nesta prova e grave seu trabalho com o nome **Proof 6.2**.

*Utilize o programa **Fitch** para construir provas formais dos seguintes argumentos. Você encontrará arquivos nomeados como **Exercise 6.x** para cada um dos argumentos abaixo. Grave cada uma de suas provas com o nome **Proof 6.x** (onde **x** é o número do argumento abaixo)*

6.3 $a = b \wedge b = c \wedge c = d \vdash a = c \wedge b = d$

6.4 $(A \wedge B) \vee C \vdash C \vee B$

6.5 $A \wedge (B \vee C) \vdash (A \wedge B) \vee (A \wedge C)$

6.6 $(A \wedge B) \vee (A \wedge C) \vdash A \wedge (B \vee C)$

Experimente (7)

(1) Vamos ilustrar as regras de introdução da negação e do absurdo, mostrando como provar ¬¬**A** a partir de **A**. Esta é a outra direção da dupla negação (inversa à da regra (¬**Elim**)). Com o programa **Fitch**, abra o arquivo **Negation 1**.

(2) Vamos examinar passo a passo a construção da seguinte prova:

(3) Para construir esta prova, adicione um passo imediatamente após a premissa. Transforme este passo em uma subprova selecionando *"New Subproof"* a partir do menu *"Proof"*. Entre a hipótese **¬A**.

(4) Adicione um passo novo na subprova, entre ⊥, e selecione a regra (⊥**Intro**). Cite os passos apropriados e verifique este passo (botão *"Check Step"*).

(5) Agora finalize a subprova e entre a sentença final **¬¬A** logo após a subprova. Especifique a regra como (¬**Intro**), cite a subprova como suporte e verifique este passo. Sua prova completa deve ser corretamente verificada (botão *"Verify Proof"*).

(6) Note que na terceira linha de sua prova, você citou um passo fora da subprova (especificamente uma premissa). Isto é legítimo, mas traz a tona uma importante questão. Exatamente, quais passos podem ser citados em um dado ponto de uma prova? Como uma primeira aposta, você poderia pensar que podemos citar qualquer passo anterior. Mas veremos que isto está errado. Veremos por que e também a resposta correta mais adiante.

(7) Grave sua prova com o nome **Proof Negation 1**.

Experimente (8) . . . . . .

(1) Abra, com o **Fitch**, o arquivo **Negation 2**. Você verá uma prova incompleta. Há uma lista grande de premissas, formando vários grupos contraditórios.

(2) Posicione o cursor em cada passo que contém o símbolo ⊥. Você verá que várias sentenças estão citadas como suporte destes passos. Apenas um destes passos é uma aplicação da regra (⊥**Intro**). Qual deles? Especifique a regra deste passo como (⊥**Intro**) e verifique.

(3) Entre os passos restantes, você encontrará um em que as sentenças citadas formam um conjunto de sentenças TT-contraditório (ou Tabela-de-Verdade-Contraditório)[1]. Qual é este passo? Modifique a justificativa deste passo para a regra (**TautCon**) e verifique o passo. Uma vez que a verificação esteja correta, você pode ter a certeza de que poderá deduzir ⊥ a partir destas mesmas premissas utilizando apenas as regras Booleanas.[2]

(4) Nos passos restantes, as sentenças citadas como suporte para 2 deles são contraditórias em virtude do significado do símbolo =. Quais são estes passos? Modifique a justificativa destes passos para a regra (**FOCon**) e verifique-os. Para derivar o ⊥ destas premissas, você precisa das regras da identidade ((**=Elim**) em um caso e (**=Intro**) no outro).

(5) Verifique que os passos restantes não podem ser justificado por nenhuma das seguintes regras: ⊥**Intro**, **TautCon** ou **FOCon**. Troque a justificativa destes passos para a regra (**AnaCon**) e verifique-os.

(6) Grave sua prova com o nome **Proof Negation 2**.

Experimente (9) . . . . . .

(1) Ocorre com freqüência, ao fazermos provas usando a regra (∨**Elim**) de querermos eliminar, de fato, um ou mais disjuntos, porque eles se contradizem com outras hipóteses. A forma da regra (∨**Elim**) não permite isso. A prova que construiremos aqui mostra como contornar esta dificuldade.

---

[1] Um conjunto de sentenças é TT-contraditório quando, em uma tabela-de-verdade conjunta para todas as sentenças não existe nenhuma linha em que todas as sentenças são simultaneamente verdadeiras.

[2] Regras Booleanas são as regras de introdução e eliminação para os operadores ∧, ∨, ¬ e ⊥.

(2) Com o programa **Fitch** abra o arquivo **Negation 3**. Usaremos a regra (∨**Elim**) e duas regras para o ⊥ para provar **P** a partir das premissas **P**∨**Q** e ¬**Q**.

(3) Inicie duas subprovas, a primeira com a hipótese **P**, e a segunda com a hipótese **Q**. Nosso objetivo é estabelecer **P** nas duas subprovas.

(4) Na primeira subprova, podemos simplesmente usar a regra de reiteração **Reit** para repetir a hipótese **P**.

(5) Na segunda subprova, como vamos estabelecer **P**? Em uma prova informal, nós simplesmente eliminamos este caso, porque a hipótese contradiz uma das premissas. Em uma prova formal, no entanto, precisamos estebeleder nossa sentença meta (**P**) nas duas subprovas, e é aqui que a regra (⊥**Elim**) se tornará útil. Primeiro utilize (⊥**Intro**) para mostrar que este caso é contraditório. Você deverá citar a hipótese assumida **Q** e a segunda premissa ¬**Q**. Uma vez que tenha obtido ⊥, como segundo passo da subprova, utilize a regra (⊥**Elim**) para estabelecer **P** nesta subprova.

(6) Com **P** em ambas as subprovas, você pode terminar a prova utilizando (∨**Elim**). Complete a prova.

(7) Salve seu trabalho com o nome **Proof Negation 3**.

Experimente (10) . . . . . .

(1) Abra o arquivo **Negation 4**. Primeiro olhe para a meta para ver qual sentença estamos tentando provar. Então concentre-se em aponte o cursor para cada um dos passos da prova, em seqüência, e verifique o passo (Botão *"Check Step"*). Antes de se mover para o passo seguinte, certifique-se de ter entendido por que a verificação foi OK e, mais importante, por que está sendo feito o que está sendo feito neste passo (a função do passo na prova). Nos passos vazios, tente prever a sentença que o **Fitch** preencherá como default antes de verificar o passo.

(2) Quando tiver terminado, certifique-se de ter entendido a prova completa. Grave a prova com o nome **Proof Negation 4**.

Exercícios . . . . . .

(6.7) Não deixe de fazer as seções *Experimente 6 a 10* !

(6.8) (Substituição) Em provas informais permitimos a você substituir sentenças logicamente equivalentes umas por outras, mesmo quando elas ocorrem no contexto (como parte) de uma sentença maior. Por exemplo, a seguinte inferência é demonstrada através de duas utilizações do princípio da dupla negação aplicado a partes de uma sentença completa.

$$\begin{array}{|l} P \wedge (Q \vee \neg\neg R) \\ \vdash \\ \neg\neg P \wedge (Q \vee R) \end{array}$$

Como você provaria este argumento utilizando o Sistema ***F***, que não tem regra de substituição? Abra o arquivo **Exercise 6.8**, que contém uma prova formal incompleta deste argumento. Dá maneira como a prova está, nenhum de seus passos está correto (experimente verificá-los **CTRL-F**), porque nenhuma regra ou passos de suporte a aplicação de regras foram citados. Coloque as justificações que estão faltando, de modo a completar a prova.

*Avalie cada um dos argumentos a seguir. Se o argumento for válido, use o programa **Fitch** para construir uma prova formal com as regras que já aprendemos. Se não for válido, use o programa **Tarski's World** para construir um contraexemplo (um mundo em que as premissas são verdadeiras e a conclusão falsa). Nas duas últimas provas você precisará usar a regra (**AnaCon**) para mostrar que certas sentenças atômicas são contraditórias umas com as outras e, desta contradição, introduzir ⊥. Use (**AnaCon**) apenas nestes casos. Ou seja, você pode usar a regra (**AnaCon**) apenas nas duas últimas provas, citando somente sentenças atômicas como suporte para a introdução do ⊥.*

6.9 ➶

Cube(b)
¬(Cube(c) ∧ Cube(b))
⊢ ¬Cube(c)

6.10 ➶

Cube(a) ∨ Cube(b)
¬(Cube(c) ∧ Cube(b))
⊢ ¬Cube(c)

6.11 ➶

Dodec(e)
Small(e)
¬Dodec(e) ∨ Dodec(f) ∨ Small(e)
⊢ Dodec(f)

6.12 ➶

Dodec(e)
¬Small(e)
¬Dodec(e) ∨ Dodec(f) ∨ Small(e)
⊢ Dodec(f)

6.13 ➶

Dodec(e)
Large(e)
¬Dodec(e) ∨ Dodec(f) ∨ Small(e)
⊢ Dodec(f)

6.14 ➶

SameRow(b, f) ∨ SameRow(c, f)
∨ SameRow(d, f)
¬SameRow(c, f)
FrontOf(b, f)
¬(SameRow(d, f) ∧ Cube(f))
⊢ ¬Cube(f)

Nos dois exercícios seguintes, determine se as sentenças são consistentes. Se elas forem, utilize o ***Mundo de Tarski*** *para construir um mundo onde ambas as sentenças são verdadeiras. Se elas forem inconsistentes, utilize o* ***Fitch*** *para apresentar uma prova desta inconsistência (ou seja, uma prova de ⊥ tendo as sentenças como premissas). Você pode utilizar (**AnaCon**) em sua prova, mas apenas aplicado a literais (isto é, as sentenças de suporte a qualquer aplicação de (**AnaCon**) que você utilize devem ser atômicas ou negações de sentenças atômicas.*

6.15 ➶ ¬(Larger(a, b) ∧ Larger(b, a))
¬SameSize(a, b)

6.16 ➶ Smaller(a, b) ∨ Smaller(b, a)
SameSize(a, b)

(6.17) Tente recriar a suposta "prova" abaixo com o programa **Fitch**.

1. (Tet(a) ∧ Large(c)) ∨ (Tet(a) ∧ Dodec(b))
⊢
  2. Tet(a) ∧ Large(c)
  ⊢
  3. Tet(a) ∧ **Elim**: 2
  4. Tet(a) ∧ Dodec(b)
  ⊢
  5. Dodec(b) ∧ **Elim**: 4
  6. Tet(a) ∧ **Elim**: 4
7. Tet(a) ∨ **Elim**: 1, 2–3, 4–6
8. Tet(a) ∧ Dodec(b) ∧ **Intro**: 7, 5

Qual passo **Fitch** não lhe permite executar? Por que? A conclusão (sentença 8) é uma conseqüência da premissa (sentença 1)? Explique suas respostas.

Use o programa ***Fitch*** *para apresentar provas formais dos seguintes argumentos. Você precisará utilizar subprovas dentro de subprovas para prová-los.*

6.18 ➶

A ∨ B
⊢ A ∨ ¬¬B

6.19 ➶

A ∨ B
¬B ∨ C
⊢ A ∨ C

6.20 ➶

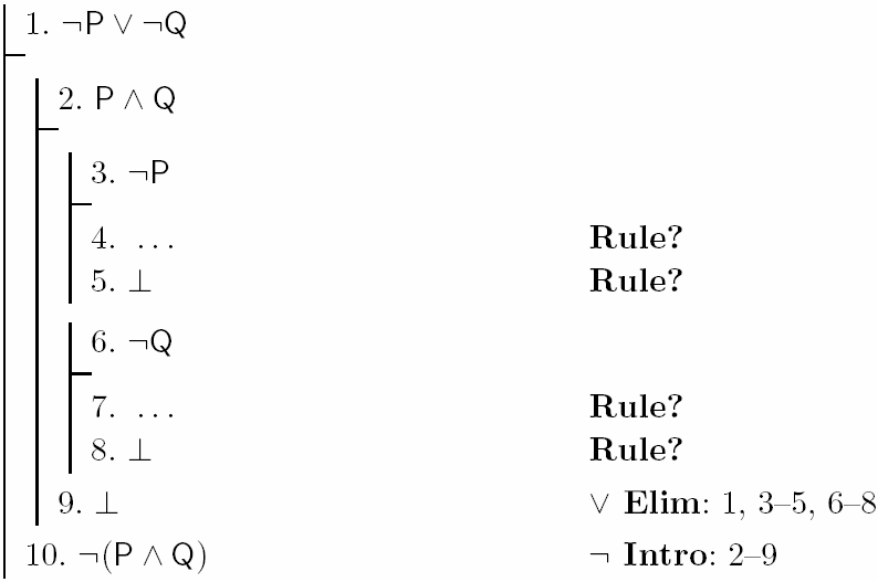
A ∨ B
A ∨ C
⊢ A ∨ (B ∧ C)

Experimente (11) . . . . . .

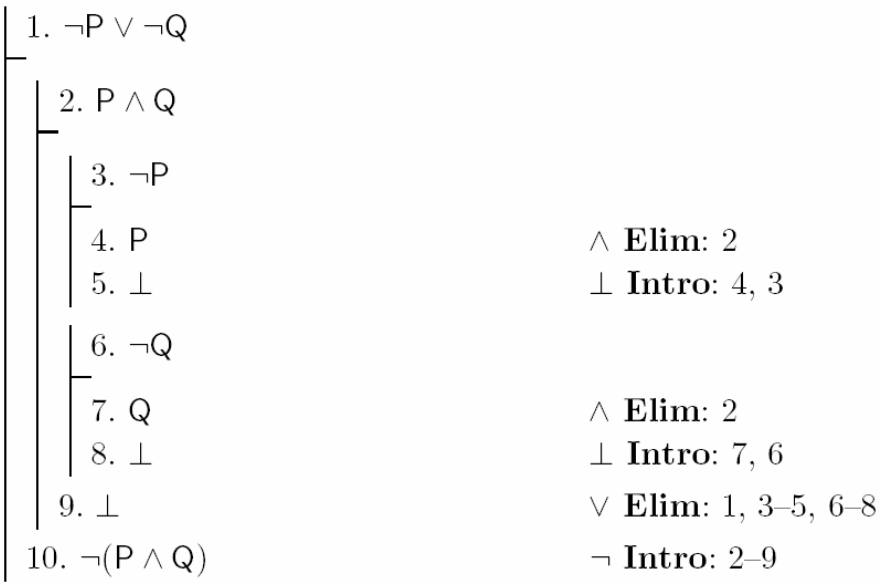
(1) Abra o arquivo **Strategy 1**. Inicie entrando a conclusão desejada em um passo novo da prova. Vamos construir esta prova trabalhando de trás para frente. Adicione um passo anterior à conclusão que você acabou de entrar, de tal modo que sua prova fique com a seguinte forma:

(2) O método principal que utilizaremos é a redução ao absurdo, que corresponde em nosso sistema formal à regra de introdução da negação (**¬Intro**). Transforme então o passo em branco em uma subprova com a hipótese **P∧Q** e o símbolo de contradição no final. Também adicione um passo entre estes dois, para lembrá-lo de que é necessário completar esta subprova. Entre também a justificativa para o passo final, de modo a lembrar-se porque adicionou a subprova. Neste ponto sua prova deve ter a seguinte forma:

(3) Vamos agora mostrar que obtemos uma contradição assumindo os dois casos dos disjuntos da premissa **¬P** e **¬Q**. Como a contrapartida formal para a prova por casos é a eliminação da disjunção, o passo seguinte é iniciar duas subprovas, uma assumindo **¬P** e a outra **¬Q**, sendo que as duas devem terminar (ter como último passo) com ⊥. Não se esqueça de colocar a justificativa para o passo em que você aplica a regra (**∨Elim**). É também uma boa idéia adicionar passos vazios para lembrá-lo de que é preciso continuar trabalhando para completar estas subprovas. Veja como sua prova deve se parecer agora:



(4) Agora ficou fácil preencher os passos que faltam. Termine sua prova como sugerido abaixo:



(5) Grave eu trabalho com o nome **Proof Strategy 1**.

Exercícios . . . . . .

(6.21) Não deixe de fazer a seção *Experimente 11*.

(6.22) Faça uma prova formal, espelhada na prova informal que apresentamos na p.5 do *Capítulo 5*, de **¬(b = c)** a partir das premissas **Cube(c) ∨ Dodec(c)** e **Tet(b)**. Você pode usar (**AnaCon**) apenas aplicada a literais para obter ⊥.

(6.23) Faça uma prova informal que poderia ter sido utilizada pelos autores como base para a construção da prova formal mostrada na página 11 do *Capítulo 6*.

*Em cada um dos exercícios seguintes, faça uma prova informal da validade do argumento indicado. (Você jamais pode usar o princípio que está tentando provar em sua prova informal. Por exemplo, no Exercício 6.24, você não deve usar DeMorgan em sua prova informal.) Então utilize o **Fitch** para construir uma prova formal que espelhe o máximo possível a prova informal que você construiu.*

6.24

$\neg(A \vee B)$

$\neg A \wedge \neg B$

6.25

$\neg A \wedge \neg B$

$\neg(A \vee B)$

6.26

$A \vee (B \wedge C)$

$\neg B \vee \neg C \vee D$

$A \vee D$

6.27

$(A \wedge B) \vee (C \wedge D)$

$(B \wedge C) \vee (D \wedge E)$

$C \vee (A \wedge E)$

*Em cada um dos exercícios seguintes, você deverá decidir se o argumento é válido ou não. Se ele for, utilize o programa **Fitch** para construir uma prova formal. Você poderá utilizar a regra (**AnaCon**), mas apenas envolvendo literais para a introdução de ⊥. Se o argumento não for válido, utilize o **Tarski's World** para construir um contraexemplo.*

6.28

$\mathrm{Cube}(c) \vee \mathrm{Small}(c)$

$\mathrm{Dodec}(c)$

$\mathrm{Small}(c)$

6.29

$\mathrm{Larger}(a, b) \vee \mathrm{Larger}(a, c)$

$\mathrm{Smaller}(b, a) \vee \neg\mathrm{Larger}(a, c)$

$\mathrm{Larger}(a, b)$

6.30

$\neg(\neg Cube(a) \wedge Cube(b))$
$\neg(\neg Cube(b) \vee Cube(c))$
$\vdash$
$Cube(a)$

6.31

$Dodec(b) \vee Cube(b)$
$Small(b) \vee Medium(b)$
$\neg(Small(b) \wedge Cube(b))$
$\vdash$
$Medium(b) \wedge Dodec(b)$

6.32

$Dodec(b) \vee Cube(b)$
$Small(b) \vee Medium(b)$
$\neg Small(b) \wedge \neg Cube(b))$
$\vdash$
$Medium(b) \wedge Dodec(b)$

(6.33) (Terceiro Excluído) Abra o arquivo **Exercise 6.33**. Este arquivo contém uma prova incompleta da lei do terceiro excluído **P∨¬P**. Do jeito que está, a verificação da prova falhará, porque estão faltando algumas sentenças, algumas citações de suporte a aplicação de regras e algumas regras. Preencha as peças que faltam e grave seu trabalho com o nome **Proof 6.33**. Esta prova nos mostra que podemos derivar a lei do terceiro excluído sem a necessidade de nenhuma premissa.

Nos exercícios seguintes, decida se cada sentença indicada é ou não uma verdade lógica na linguagem de blocos. Caso for, utilize o **Fitch** para construir uma prova formal desta sentença sem utilizar premissas (utilize a regra **AnaCon** se necessário, mas aplicada apenas a literais[3]). Caso a sentença não seja uma verdade lógica, utilize o programa **Tarski's World** para construir um contra-exemplo. (Aqui, um contra-exemplo é simplesmente um mundo no qual a sentença seja falsa.)

6.34

$\vdash$
$\neg(a = b \wedge Dodec(a) \wedge \neg Dodec(b))$

6.35

$\vdash$
$\neg(a = b \wedge Dodec(a) \wedge Cube(b))$

6.36

$\vdash$
$\neg(a = b \wedge b = c \wedge a \neq c)$

6.37

$\vdash$
$\neg(a \neq b \wedge b \neq c \wedge a = c)$

6.38

$\vdash$
$\neg(SameRow(a, b) \wedge SameRow(b, c) \wedge FrontOf(c, a))$

6.39

$\vdash$
$\neg(SameCol(a, b) \wedge SameCol(b, c) \wedge FrontOf(c, a))$

As sentenças seguintes são todas tautologias, e portanto devem ser prováveis em nosso sistema formal. Apesar de as suas provas informais serem relativamente simples, suas provas formais não o são, uma vez que é necessário provar passos bastante óbvios. Use o programa **Fitch** para construir provas formais destas sentenças sem a utilização de premissas. A prova da lei do terceiro excluído (exercício 6.33) pode ser um bom modelo para inspirá-lo nestes exercícios.

[3] literais são sentenças atômicas ou negações de sentenças atômicas. Exemplo: **Tet(a)** e **¬Larger(a, b)**.

6.40 ➶⋆ $\vdash A \vee \neg(A \wedge B)$

6.41 ➶⋆ $\vdash (A \wedge B) \vee \neg A \vee \neg B$

6.42 ➶⋆ $\vdash \neg A \vee \neg(\neg B \wedge (\neg A \vee B))$

Exercícios do Capítulo 7 - (Condicionais)

Exercícios . . . . . .

*Use o programa **Boole** para determinar se os seguintes pares de sentenças são tautologicamente equivalentes:*

7.1 $A \rightarrow B$ and $\neg A \vee B$.

7.2 $\neg(A \rightarrow B)$ and $A \wedge \neg B$.

7.3 $A \leftrightarrow B$ and $(A \rightarrow B) \wedge (B \rightarrow A)$.

7.4 $A \leftrightarrow B$ and $(A \wedge B) \vee (\neg A \wedge \neg B)$.

7.5 $(A \wedge B) \rightarrow C$ and $A \rightarrow (B \vee C)$.

7.6 $(A \wedge B) \rightarrow C$ and $A \rightarrow (B \rightarrow C)$.

7.7 $A \rightarrow (B \rightarrow (C \rightarrow D))$ and $((A \rightarrow B) \rightarrow C) \rightarrow D$.

7.8 $A \leftrightarrow (B \leftrightarrow (C \leftrightarrow D))$ and $((A \leftrightarrow B) \leftrightarrow C) \leftrightarrow D$.

(7.9) (Just in case) Prove que a interpretação usual (não-matemática) da expressão inglesa "*just in case*" não é um conectivo verofuncional. Use, como caso de exemplo, a sentença "*Max went home just in case Carl was hungry*".

(7.10) (Avaliando sentenças em um mundo) Com o programa **Mundo de Tarski**, avalie cada uma das **Sentenças de Abelardo** no **Mundo de Wittgenstein**. Se você cometer um erro, jogue o jogo para ver onde você errou. Quando tiver terminado, volte e transforme todas as sentenças FALSAS em VERDADEIRAS através da troca de um ou mais nomes usados na sentença.

(7.11) (Descrevendo um Mundo) Inicie o programa **Tarski's World** e selecione a opção ***Hide Labels*** no Menu ***Display***. Esta opção esconde (não mostra) os nomes dos objetos dos mundos. Abra, então o arquivo **Montague's World**. Neste mundo, cada objeto tem um nome e nenhum objeto tem mais de um nome. (Note que estes nomes não aparecerão, porque a opção *Hide Labels* que você selecionou os esconde) Inicie um arquivo de sentenças novo no qual você descreverá algumas características deste mundo. Verifique (Botão *Verify*) cada uma das sentenças para ver que elas são de fato verdadeiras neste mundo.

1. Note que se **c** é um tetraedro, então **a** não é um tetraedro. (Lembre-se que neste mundo cada objeto tem exatamente um nome) Use sua primeira sentença para expressar este fato.
2. Note também que o mesmo é válido para **b** e **d**. Ou seja, se **b** é um tetraedro, então **d** não é. Use sua segunda sentença para expressar isso.
3. Finalmente, observe que se **b** é um tetraedro, então **c** não é. Expresse isso.
4. Note que se **a** é um cubo e **b** é um dodecaedro, então **a** está a esquerda de **b**. Use sua próxima sentença para expressar este fato.
5. Use sua próxima sentença para expressar o fato de que se **b** e **c** são ambos cubos, então eles estão na mesma linha mas não estão na mesma coluna.
6. Use sua próxima sentença para expressar o fato de que se **b** é um tetraedro então é pequeno.
7. Em seguida, expresse o fato de que se **a** e **d** são ambos cubos, então um está à esquerda do outro. [Note que você precisará de uma disjunção para expressar que um está à esquerda do outro]
8. Note que **d** é um cubo se e somente se ele for médio ou grande. Expresse isso.
9. Observe que se **b** não está nem à direita nem à esquerda de **d**, então um deles é um tetraedro. Expresse esta observação.
10. Finalmente, expresse o fato de que **b** e **c** têm o mesmo tamanho se e somente se um é um tetraedro e o outro um dodecaedro.

Grave suas sentenças em um arquivo com o nome **Sentences 7.11**. Agora selecione ***Show Labels*** no menu ***Display***. Verifique que todas as suas sentenças são de fato verdadeiras. Ao verificar as três primeiras, preste atenção especial no valor de verdades de cada uma de suas partes constituintes. Note que algumas vezes o condicional tem um antecedente falso e algumas vezes um conseqüente verdadeiro. O que ele nunca tem é um antecedente verdadeiro e um conseqüente falso. Em cada um destes três casos, clique no botão ***Game*** e jogue, se comprometendo com a VERDADE. Certifique-se de ter entendido porque o jogo procede da maneira que procede.

(7.12) (Tradução) Traduza as seguintes sentenças do português para FOL. Suas traduções deverão utilizar todos os conectivos proposicionais.

1. *Se **a** é um tetraedro então ele está na frente de **d**.*
2. ***a** está a esquerda ou a direita de **d** apenas se for um cubo.*
3. ***c** está entre **a** e **e** ou **a** e **d**.*
4. ***c** está a direita de **a**, contanto que seja pequeno (**c** pequeno).*
5. ***c** está a direita de **d** apenas se **b** está a direita de **c** e a esquerda de **e**.*
6. *Se **e** é um tetraedro, então ele está a direita de **b** se e somente se está na frente de **b**.*
7. *Se **b** é um dodecaedro, então se ele não está na frente de **d** então ele também não está atrás de **d**.*
8. ***c** está atrás de **a** mas na frente de **e**.*
9. ***e** está na frente de **d** a menos que ele (**e**) seja um tetraedro grande.*
10. *Pelo menos um entre **a**, **b** e **c** é um cubo.*
11. ***a** é um tetraedro apenas se estiver na frente de **b**.*
12. ***b** é maior do que ambos, **a** e **e**.*
13. ***a** e **e** são ambos maiores do que **c**, mas nenhum deles é grande.*
14. ***d** tem a mesma forma que **b** apenas se eles são do mesmo tamanho.*
15. ***a** é grande se e somente se for um cubo.*
16. ***b** é um cubo a menos que **c** seja um tetraedro.*
17. *Se **e** não é um cubo, então ou **b** ou **d** é grande.*
18. ***b** ou **d** é um cubo se **a** ou **c** forem tetraedro.*
19. ***a** é grande apenas se **d** é pequeno.*
20. ***a** é grande apenas se **e** for.*

(7.13) (Verificando suas traduções) Abra o arquivo **Bolzano's World**. Repare que todas as sentenças em português do *Exercício 7.12* acima são verdadeiras neste mundo. Então, se suas traduções forem precisas, elas também deverão ser verdadeiras neste mundo. Verifique (**CTRL F**). Se você cometeu algum erro, volte e corrija-os. Lembre-se que serem verdadeiras no Mundo de Bolzano não garante às suas sentenças que elas sejam traduções adequadas das sentenças em português. Se a tradução for correta, então a sentença original e a tradução terão os mesmos valores de verdade em TODOS os mundos. Então, para melhorar nosso teste, vamos verificar suas traduções em alguns outros mundos.
Abra o **Wittgenstein's World**. Aqui nós podemos ver que as sentenças em português 3, 5, 9, 11, 12, 13, 14 e 20 são falsas, enquanto as outras são verdadeiras. Verifique se o mesmo se dá com suas traduções em FOL (digite **CTRL F**) Caso haja alguma discrepância, corrija a sua tradução. E certifique-se de que a sentença que você corrigir continua verdadeira no Mundo de Bolzano.
Em seguida abra o arquivo **Leibniz's World**. Aqui as sentenças em português 1, 2, 4, 6, 7, 10, 11, 14, 18 e 20 são verdadeiras e as demais falsas. Verifique se o mesmo ocorre com suas traduções em FOL. Corrija o que for necessário.
Finalmente, abra o arquivo **Venn's World**. Neste mundo todas as sentenças em português são falsas. Verifique se o mesmo ocorre com suas traduções e corrija o que for necessário.

(7.14) (Descobrindo tamanhos e formas) Abra o arquivo **Euler's Sentences**. As nove sentenças deste arquivo determinam univocamente as formas e tamanhos dos blocos *a*, *b* e *c*. Tente descobrir quais são essas formas e tamanhos apenas pensando sobre o significado das sentenças e utilizando os métodos de prova informais que já estudamos. Quando descobrir, construa um mundo em que todas as sentenças sejam verdadeiras. Grave seu trabalho com o nome **World 7.14**.

(7.15) (Mais Tamanhos e Formas) Inicie um arquivo de sentenças novo e use-o para traduzir as seguintes sentenças do português para a linguagem dos blocos de FOL.

1. *Se **a** é um tetraedro, então **b** também é.*
2. ***c** é um tetraedro se **b** for.*
3. ***a** e **c** são ambos tetraedros apenas se pelo menos um deles for grande.*
4. ***a** é um tetraedro, mas **c** não é grande.*
5. *Se **c** é pequeno e **d** é um dodecaedro, então **d** não é nem grande nem pequeno.*
6. ***c** é médio se e somente se nenhum entre **d**, **e** e **f** for cubo.*
7. ***d** é um dodecaedro pequeno, a menos que **a** seja pequeno.*
8. ***e** é grande se e somente se é um fato que **d** é grande se e somente se **f** também for.*
9. ***d** e **e** têm o mesmo tamanho.*
10. ***d** e **e** têm a mesma forma.*
11. ***f** é ou um cubo ou um dodecaedro, se for grande.*

12. ***c*** *é maior do que* ***e*** *apenas se* ***b*** *for maior do que* ***c***.

Grave seu trabalho no arquivo **Sentences 7.15**. Então veja se você consegue descobrir os tamanhos e formas dos blocos **a**, **b**, **c**, **d**, **e** e **f**. A tarefa ficará mais fácil se você abordá-la de modo sistemático, tentando preencher a seguinte tabela, conforme raciocina sobre as sentenças.

	a	***b***	***c***	***d***	***e***	***f***
Forma:						
Tamanho:						

Quando tiver terminado de preencher a tabela, use-a como guia para construir um mundo no qual as 12 sentenças em português acima sejam verdadeiras. Verifique que se suas traduções em FOL também são verdadeiras neste mundo. Grave seu trabalho com o nome **World 7.15**

(7.16) (Dando nome aos objetos) Abra os arquivos **Sherlock's World** e **Sherlock's Sentences**. Você notará que nenhum dos objetos no mundo tem nome. Sua tarefa é atribuir os nomes **a**, **b** e **c** de uma forma que todas as sentenças da lista fiquem verdadeiras. Grave o mundo modificado com o nome **World 7.16**.

(7.17) (Construindo um Mundo) Abra o arquivo **Boolos' Sentences** e construa um mundo no qual as cinco sentenças sejam verdadeiras. Este é difícil!

(7.18) Utilizando os símbolos introduzidos na *Tabela 1.2* (*nos exercícios do Capítulo 1*), traduza as seguintes sentenças em FOL.

1. *Se Claire deu Folly para Max as 2:03 então Folly pertencia a ela as 2:00 e a ele as 2:05.*
2. *Max alimentou Folly as 2:00 pm, mas se ele deu-a a Claire, então Folly não estava faminta cinco minutos mais tarde.*
3. *Se nem Max nem Claire alimentaram Folly as 2:00, então ela estava faminta.*
4. *Max estava bravo as 2:05 apenas se Claire alimentou ou Folly ou Sxruffy cinco minutos antes.*
5. *Max é um estudante se e somente se Claire não é.*

(7.19) Utilizando a *Tabela 1.2* (*nos Exercícios do Capítulo 1*) traduza as seguintes sentenças em português coloquial.

1. **(Fed(max, folly, 2:00) $\vee$ Fed(claire, folly, 2:00)) $\rightarrow$ Pet(folly)**
2. **Fed(max, folly, 2:30) $\leftrightarrow$ Fed(claire, scruffy, 2:00)**
3. **$\neg$Hungry(folly, 2:00) $\rightarrow$ Hungry(scruffy, 2:00)**
4. **$\neg$(Hungry(folly, 2:00) $\rightarrow$ Hungry(scruffy, 2:00))**

(7.20) Traduza as seguintes sentenças para FOL da melhor maneira que puder. Explique qualquer símbolo de predicado e de função que você utilizar, bem como, qualquer especificidade na interpretação que está fazendo das sentenças.

1. *Se Abe pode enganar Stephen, certamente ele pode enganar Ulysses.*
2. *Se você me caluniar, eu te caluniarei.*
3. *A França assinará o tratado apenas se a Alemanha assinar.*
4. *Se Tweedledee der uma festa, então Tweedledum também dará, e vice-versa.*
5. *Se John e Mary foram ao concerto juntos, eles devem gostar um do outro.*

(7.21) (O Princípio do Macaco) Um dos usos mais estranhos de '*se...então...*' em português é como uma forma indireta de expressar negação. Suponha que um amigo lhe diga: '*Se Keanu Reeves é um grande ator, então eu sou o tio de um macaco*'. Isto é simplesmente uma forma de negar o antecedente do condicional, neste caso, que Keanu Reeves é um grande ator. Explique por que isto funciona. Sua explicação deve usar a tabela de verdade para $\rightarrow$, mas deve ir além disso. Faça também uma tabela de verdade, com o programa **Boole**, mostrando que $\mathbf{A} \rightarrow \bot$ é equivalente a $\neg\mathbf{A}$.

(7.22) Suponha que Claire sustente a seguinte sentença: *Max conseguiu chegar em casa*. Será que isso implica logicamente ou apenas *insinua socialmente* que é difícil para Carl chegar em casa? Justifique sua resposta.

(7.23) Suponha que Max afirme *Nós podemos caminhar até o cinema ou podemos ir de carro*. Será que esta afirmação implica logicamente, ou meramente insinua que nós não podemos fazer ambos, caminhar e ir de carro? Como este caso difere do exemplo da sopa e salada que vimos no capítulo?

(7.24) (Este é Difícil, mas Muito Interessante) Considere a sentença '*Max está em casa a despeito do fato de Claire estar na biblioteca*'. Qual deveria ser a melhor tradução desta sentença em FOL? Esta sentença dá a impressão de não ser determinada simplesmente pelos valores de verdade de das sentenças atômicas '*Max está*

em casa' e '*Claire está na biblioteca*'. Mas será que esta impressão é devida ao fato de que, como '*porque*', '*a despeito do fato de*' não é um conectivo verofuncional? Ou será que esta impressão é devida ao fato de que, como '*mas*', '*a despeito do fato de*' é um conectivo verofuncional que carrega certas insinuações sociais (*conversational implicatures*) adicionais. Qual destas duas explicações é a correta? Justifique sua resposta.

(7.25) (Substituindo $\wedge$, $\rightarrow$ e $\leftrightarrow$) Use o programa **Tarski's World** para abrir o arquivo **Sheffer's Sentences**. Neste arquivo você encontrará as seguintes sentenças nas linhas ímpares:

1. **Tet(a) $\wedge$ Small(a)**
3. **Tet(a) $\rightarrow$ Small(a)**
5. **Tet(a) $\leftrightarrow$ Small(a)**
7. **(Cube(b) $\wedge$ Cube(c)) $\rightarrow$ (Small(b) $\leftrightarrow$ Small(c))**

Em cada linha par, construa uma sentença que seja equivalente à sentença acima dela, mas que utilize apenas os conectivos $\neg$ e $\vee$. Para ter certeza de que as sentenças que você construiu são logicamente equivalentes às sentenças ímpares, abra alguns dos arquivos de Mundo e verifique, em cada um deles, se elas têm os mesmos valores de verdade.

(7.26) (O Problema da Economia de Conectivos) Tratar um símbolo como *básico*, com suas próprias regras, ou como *símbolo definido*, sem regras próprias, faz uma grande diferença na complexidade das provas. Para ilustrar isso, use o programa **Fitch** e abra o arquivo **Exercise 7.26**. Neste arquivo, você é solicitado a construir uma prova de **$\neg$($\neg$A $\vee$ $\neg$B)** a partir das premissas **A** e **B**. Sua tarefa é fazer esta prova. Repare que uma prova da sentença equivalente **A $\wedge$ B** teria, é claro, um único passo com a regra **($\wedge$Intro)**.

(7.28) (Expressando outro conectivo ternário) Inicie um arquivo de sentenças novo no programa **Mundo de Tarski**. Utilize o método que nós desenvolvemos para expressar um conectivo ternário **♥** definido na tabela de verdade abaixo, e entre digite esta definição como a primeira sentença deste arquivo de sentenças. Então, tente simplificar o resultado tanto quanto puder. Entre a sentença simplificada como a segunda sentença do arquivo. (Esta sentença não deve ter mais do que duas ocorrências de cada **P**, **Q** e **R** e não deve ter mais do que seis ocorrências dos conectivos Booleanos $\wedge$, $\wedge$ e $\neg$.

(7.29) (O símbolo de Sheffer) Outro conectivo binário que é verofuncionalmente completo é o chamado *símbolo de Sheffer*, devido a H. M. Sheffer, um dos lógicos que o estudou. Ele é também conhecido como *nand*, (*não-e*), porque sua tabela de verdade é equivalente à de **$\neg$(P $\wedge$ Q)**. Aqui está ela:

P	Q	P | Q
T	T	**F**
T	F	**T**
F	T	**T**
F	F	**T**

Mostre como expressar as seguintes sentenças utilizando apenas o símbolo de Sheffer:

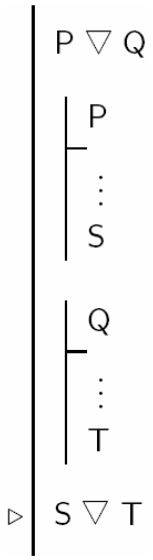
(1) **$\neg$P** (2) **P $\wedge$ Q** (3) **P $\vee$ Q**

(Apenas como advertência, o símbolo **|** têm sido utilizado, atualmente, como uma notação alternativa para $\vee$. Não se confunda com isso !!)

(7.30) (Colocando os Macacos Para Trabalhar) Suponha que nossa linguagem só tenha um único conectivo binário, o condicional $\rightarrow$, e o símbolo do absurdo $\perp$. Utilizando apenas estes dois símbolos, tente encontrar uma forma de expressar as seguintes sentenças: **$\neg$P**, **P $\wedge$ Q**, e **P $\vee$ Q**. [Sugestão: dê uma olhada no Exercício 7.21]

(7.31) (Outro conectivo não-verofuncional) Mostre que o valor de verdade, em um instante particular, da sentença *'Max está em casa sempre que Claire está na biblioteca'* não é determinado pelos valores de verdade de suas sentenças atômica *'Max está em casa'* e *'Claire está na biblioteca'* neste instante específico. Em outras palavras, mostre que *sempre que* não é verofuncional.

(7.32) (Disjunção Exclusiva) Suponha que tivéssemos introduzido o símbolo ∇ para expressar disjunção exclusiva. Descubra se o seguinte método de prova é válido para este conectivo:



Se você acha que sim, que é um método de prova válido, então justifique sua resposta. Se você acha que não, então apresente um exemplo em que o método justificaria uma inferência inválida [premissa (**P ∇ Q**) verdadeira e conclusão (**S ∇ T**) falsa].

Exercícios do Capítulo 8 - (A Lógica dos Condicionais)

Exercícios . . . . . .

(8.1) Na lista seguinte, apresentamos vários padrões de inferência. Alguns são válidos, alguns não são. Para cada padrão, decida se ele é válido ou não. Mais tarde retornaremos a estes padrões através de provas formais e contra-exemplos. Mas por agora, apenas avalie sua validade.

AFIRMAÇÃO DO CONSEQÜENTE

$$\begin{array}{l} A \rightarrow B \\ B \\ \hline A \end{array}$$

MODUS TOLLENS

$$\begin{array}{l} A \rightarrow B \\ \neg B \\ \hline \neg A \end{array}$$

FORTALECIMENTO DO ANTECEDENTE

$$\begin{array}{l} B \rightarrow C \\ \hline (A \wedge B) \rightarrow C \end{array}$$

ENFRAQUECIMENTO DO ANTECEDENTE

$$\begin{array}{l} B \rightarrow C \\ \hline (A \vee B) \rightarrow C \end{array}$$

FORTALECIMENTO DO CONSEQÜENTE

$$\begin{array}{l} A \rightarrow B \\ \hline A \rightarrow (B \wedge C) \end{array}$$

ENFRAQUECIMENTO DO CONSEQÜENTE

$$\begin{array}{l} A \rightarrow B \\ \hline A \rightarrow (B \vee C) \end{array}$$

DILEMA CONSTRUTIVO

$$\begin{array}{l} A \vee B \\ A \rightarrow C \\ B \rightarrow D \\ \hline C \vee D \end{array}$$

TRANSITIVIDADE DO BICONDICIONAL

$$\begin{array}{l} A \leftrightarrow B \\ B \leftrightarrow C \\ \hline A \leftrightarrow C \end{array}$$

(8.2) Abra o arquivo **Conditional Sentences** no programa **Mundo de Tarski**. Suponha que as sentenças neste arquivo são suas premissas. Agora considere as cinco sentenças listadas abaixo. Algumas delas são conseqüência destas premissas, algumas não são. Para as que são, faça uma prova informal. Para as que não são conseqüência, construa um contra-exemplos em que as premissas são verdadeiras e conclusão (a sentença específica da lista abaixo) falsa. Grave seus contra-exemplos com os nomes **World 8.2.x**, onde **x** é o número da sentença.

1. **Tet(e)**
2. **Tet(c) → Tet(e)**
3. **Tet(c) → Larger(f, e)**
4. **Tet(c) → LeftOf(c, f)**
5. **Dodec(e) → Smaller(e, f)**

Os argumento seguintes são válidos. Faça uma prova informal de sua validade. Ou seja, prove sua conclusão assumindo como verdadeiras as suas premissas. Você pode achar mais fácil traduzir os argumentos para FOL antes de tentar fazer as provas. No entanto, isso não é obrigatório. Indique explicitamente qualquer infirência que use *modus ponens*, *eliminação do bicondicional* ou *prova condicional*.

(8.3)

O unicórnio, se não é um animal mítico, é um mamífero, mas se for mítico, é imortal.
Se o unicórnio for imortal ou mamífero, então ele tem chifre.
Se o unicórnio tem chifre, então ele é mágico

O unicórnio é mágico

(8.4)

O unicórnio, se tem chifre, é elusivo e perigoso.
Se for elusivo e perigoso, o unicórnio é raro.
Se for um mamífero, o unicórnio não é raro

O unicórnio, se tem chifre, não é um mamímero.

(8.5)

O unicórnio, se tem chifre, é elusivo e mágico, mas se não tem chifre, não é nenhum dos dois.
Se o unicórnio não tem chifre, ele não é mítico.

O unicórnio tem chifre se e somente se for mágico ou mítico.

(8.6)

a é um tetraedro grande ou um cubo pequeno.
b não é pequeno.
Se ***a*** é um tetraedro ou um cubo, então ***b*** é grande ou pequeno.
a é um tetraedro apenas se ***b*** é médio.
a é pequeno e ***b*** é grande.

(8.7)

b é pequeno a menos que seja um cubo.
Se ***c*** é pequeno, então ou ***d*** ou ***e*** também é.
Se ***d*** é pequeno, então ***c*** não é.
Se ***b*** é um cubo, então ***e*** não é pequeno.
Se ***c*** é pequeno, então ***b*** também é.

(8.8)

d está na mesma linha que ***a***, ***b*** ou ***c***.
Se ***d*** está na mesma linha que ***b***, então ele está na mesma linha que ***a*** apenas se ele não está na mesma linha que ***c***.
d está na mesma linha que ***a*** se e somente se ele está na mesma linha que ***c***
d está na mesma linha que ***a*** se e somente se ele não está na mesma linha que ***b***.

(8.9)

a é ou um cubo ou um dodecaédro ou um tetraédro.
a é pequeno, médio ou grande.
a é médio se e somente se for um dodecaedro.
a é um tetraédro se e somente se for grande.
a é um cubo se e somente se for pequeno.

(8.10) Abra o arquivo **Between Sentences**. Determine se este conjunto de sentenças é satisfativel ou não. Se for, construa um mundo no qual todas as sentenças sejam verdadeiras. Se não, faça uma prova informal de que as sentenças são inconsistentes. Ou seja, assumindo todas elas, derive uma contradição.

(8.11) Analise a estrutura da prova informal abaixo que justifica a seguinte afirmação:

> *Se os EUA não diminuir o consumo de petróleo rapidamente, partes da Califórnia serão inundadas dentro de 50 anos.*

Verifique se há pontos fracos no argumento. Indique também quais premissas estão sendo assumidas implicitamente? Elas são plausíveis?

> ***Prova:*** *Suponha que os EUA não diminua seu consumo de petróleo. Então ele será incapaz de reduzir substancialmente sua emissão de dióxido de carbono nos próximos poucos anos. Mas então, países como China, Índia e Brasil se recusarão a realizar esforços conjuntos para diminuir as emissões de dióxido de carbono. Como, sem estes esforços, estes países se desenvolvem, a emissão de dióxido de carbono piorará e, o efeito estufa se acelerará. Como conseqüência, os oceanos se aquecerão, gelo dos pólos derreterão e o nível do mar subirá. Neste caso, as baixas áreas costeiras da Califórnia sofrerão inundações nos próximos 50 anos. Portanto, se os EUA não diminuir seu consumo de petróleo, partes da Califórnia serão inundadas nos próximos 50 anos.*

(8.12) Descreva uma inferência do dia a dia que utiliza o método da prova condicional.

(8.13) Prove que: **Impar(n+m)** $\rightarrow$ **Par(n x m)**. [Sugestão: compare este caso com o *Exercício 5.24.*]

(8.14) Prove que: **Irracional(x)** $\rightarrow$ **Irracional($\sqrt{x}$)**. [Sugestão: é mais fácil provar a contrapositiva!]

(8.15) Prove que as seguintes condições para os números naturais são todas equivalentes. Use o mínimo possível de provas condicionais.

1. n é divisível por 3
2. n^2 é divisível por 3
3. n^2 é divisível por 9
4. n^3 é divisível por 3
5. n^3 é divisível por 9
6. n^3 é divisível por 27

(8.16) Faça uma prova informal da seguinte propriedade:

> Se **R** é uma conseqüência tautológica de $\mathbf{P}_1, \ldots, \mathbf{P}_n$ e **Q**, então, **Q** $\rightarrow$ **R** é uma conseqüência tautológica de $\mathbf{P}_1, \ldots, \mathbf{P}_n$.

Experimente (1) . . . . . .

(1) Vamos fazer, passo a passo, uma prova de **A→C** partindo da premissa **(A∨B)→C**. Use o programa **Fitch** para abrir o arquivo **Condicional 1**. Repare na premissa e na meta. Adicione um passo à prova e escreva a sentença da meta.

(2) Inicie uma subprova antes da sentença **A→C**. Entre **A** como a hipótese desta subprova.

(3) Adicione um segundo passo na subprova e entre **C**.

(4) Mova o cursor para o passo que tem a sentença **A→C**. Justifique este passo utilizando a regra **(→Intro)** citando a subprova como suporte. Verifique o passo (botão ***Check Step***).
(5) Agora precisamos retornar e preencher a subprova. Adicione um passo entre os dois passos da subprova. Entre a sentença **A∨B**. Justifique este passo com a regra **(∨Intro)**, citando a hipótese da subprova.

(6) Agora mova o cursor para o último passo da subprova. Justifique este passo usando a regra **(→Elim)** citando a premissa e o passo anterior.

(7) Verifique se sua prova está OK (botão ***Verify Proof***) e grave seu trabalho como **Proof Conditional 1**.

Experimente (2) . . . . . .

(1) Abra o arquivo **Condicional 2**. Confira a sentença que queremos provar na janela de meta. Coloque o cursor em cada um dos passos e verifique o passo (botão ***Check Step***). Nos passos que estão vazios, tente prever como o **Fitch** irá preenche-lo com a aplicação *default* da regra.

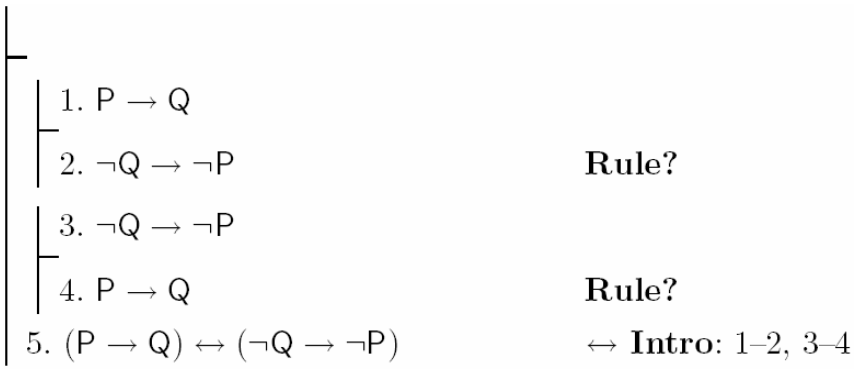
(2) Quando terminar, certifique-se de ter entendido a prova.

Experimente (3) . . . . . .

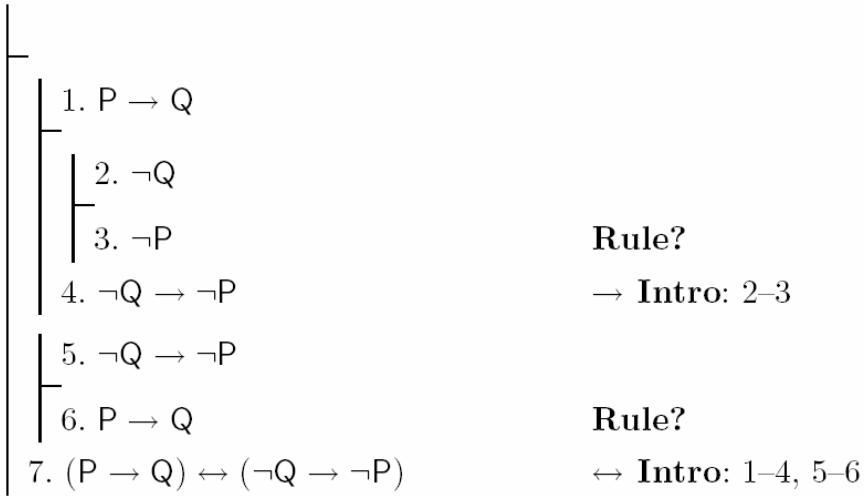
(1) Abra o arquivo **Condicional 3**. Neste arquivo, você será solicitado a provar, sem premissas, a lei da contraposição:

$$(P \rightarrow Q) \leftrightarrow (\neg Q \rightarrow \neg P)$$

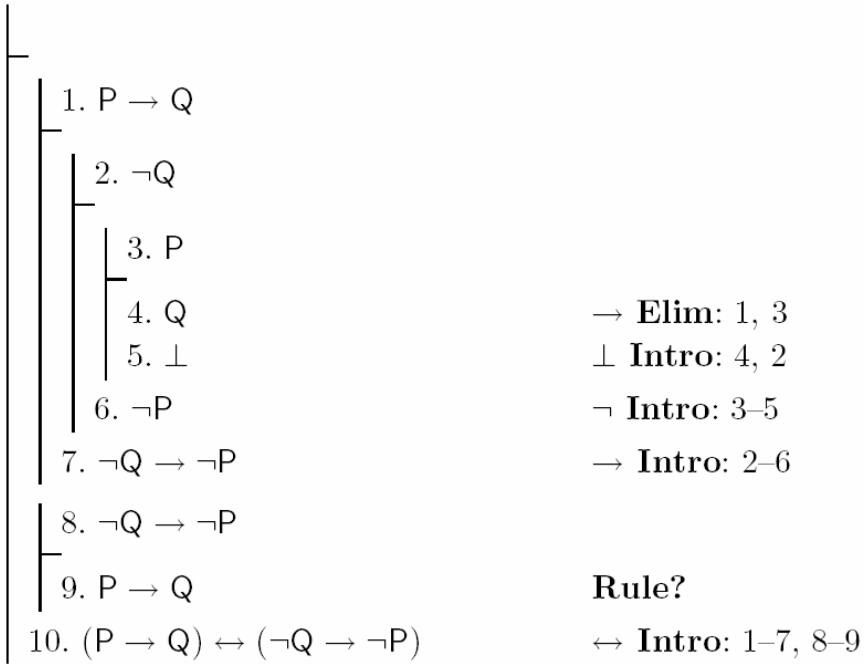
(2) Inicie sua prova esquematizando as duas subprovas que sabe que terá que fazer, mais a conclusão desejada. Sua prova parcial se parecerá com:



(3) Agora que já tem a estrutura geral da prova, inicie preenchendo a primeira subprova. Uma vez que a meta da subprova é uma afirmação condicional (uma implicação), esquematize uma prova condicional que lhe dará tal implicação:



(4) Para derivar **¬P** na subprova, você precisará assumir **P** e derivar uma contradição. Isto é algo bastante direto:



(5) Isto completa a primeira subprova. Felizmente você esquematizou a segunda subprova, então você sabe o que tem que fazer a seguir. Você deve ser capaz de terminar a segunda subrpova sozinho, uma vez que ela é quase idêntica à primeira.

(6) Quando tiver terminado, grave sua seu trabalho com o nome **Proof Conditional 3**.

Exercícios . . . . . .

(8.17) Se pulou as seções *Experimente 1 e 2* acima. Volte e faça-as. Elas são muito importantes.

*Nos exercícios seguintes, retornamos aos padrões de inferências discutidos no Exercício 8.1. Alguns destes padrões, são formas válidas de inferência tradicionalmente conhecidas e utilizadas. Para estes, faça uma prova da conclusão a partir das premissas utilizando o programa **Fitch**. Alguns outros não são padrões válidos. São falácias, dão a impressão de serem válidos, mas não são. Para estes, traduza as sentenças **A**, **B**, **C**,... por sentenças da linguagem de blocos (Por exemplo: **Tet(a)**, **Large(b)**,...) e construa um contra-exemplo, ou seja, construa um mundo no qual as traduções das premissas sejam verdadeiras e a tradução da conclusão seja falsa. Grave suas provas / ou contra-exemplos.*

(8.18) AFIRMAÇÃO DO CONSEQÜENTE

$$\begin{array}{l} A \rightarrow B \\ B \\ \hline A \end{array}$$

(8.19) MODUS TOLLENS

$$\begin{array}{l} A \rightarrow B \\ \neg B \\ \hline \neg A \end{array}$$

(8.20) FORTALECIMENTO DO ANTECEDENTE

$$\begin{array}{l} B \rightarrow C \\ \hline (A \wedge B) \rightarrow C \end{array}$$

(8.21) ENFRAQUECIMENTO DO ANTECEDENTE

$$\begin{array}{l} B \rightarrow C \\ \hline (A \vee B) \rightarrow C \end{array}$$

(8.22) FORTALECIMENTO DO CONSEQÜENTE

$$\begin{array}{l} A \rightarrow B \\ \hline A \rightarrow (B \wedge C) \end{array}$$

(8.23) ENFRAQUECIMENTO DO CONSEQÜENTE

$$\begin{array}{l} A \rightarrow B \\ \hline A \rightarrow (B \vee C) \end{array}$$

(8.24) DILEMA CONSTRUTIVO

$$\begin{array}{l} A \vee B \\ A \rightarrow C \\ B \rightarrow D \\ \hline C \vee D \end{array}$$

(8.25) TRANSITIVIDADE DO BICONDICIONAL

$$\begin{array}{l} A \leftrightarrow B \\ B \leftrightarrow C \\ \hline A \leftrightarrow C \end{array}$$

*Utilize o programa **Fitch** para construir provas formais dos seguintes argumentos. Em dois deles (no 8.28 e no 8.29), você pode se encontrar em uma situação em que tenha que provar alguma instância da Lei do Terceiro Excluído **(P ∨ ¬P)** para conseguir completar sua prova. Se você esqueceu como fazer isso, reveja a solução do Exercício 6.33.*

8.26 $\vdash P \rightarrow (Q \rightarrow P)$

8.27 $\vdash (P \rightarrow (Q \rightarrow R)) \leftrightarrow ((P \wedge Q) \rightarrow R)$

8.28 $P \leftrightarrow \neg P \vdash \bot$

8.29 $\vdash (P \rightarrow Q) \leftrightarrow (\neg P \vee Q)$

8.30 $\vdash \neg(P \rightarrow Q) \leftrightarrow (P \wedge \neg Q)$

*Os argumentos abaixo são traduções formalizadas dos argumentos apresentados nos Exercícios 8.3 a 8.9. Os outros são argumentos para os quais não apresentamos suas versões informais. Você precisará utilizar **(AnaCon)** para introduzir o ⊥ em duas de suas provas.*
Compare a dificuldade que você teve para fazer as provas informais daqueles exercícios com a dificuldade de fazer as provas formais nestes exercícios. Conforme você se acostuma à formalização e também conforme os argumentos vão ficando mais complexos, você notará como se torna muito mais fácil fazer provas formalizadas do que provas informais. Ainda que as provas formalizadas fiquem mais longas.

8.31

$$\begin{array}{l} (\neg \text{Mythical}(c) \rightarrow \text{Mammal}(c)) \\ \quad \wedge (\text{Mythical}(c) \rightarrow \neg \text{Mortal}(c)) \\ (\neg \text{Mortal}(c) \vee \text{Mammal}(c)) \rightarrow \text{Horned}(c) \\ \text{Horned}(c) \rightarrow \text{Magical}(c) \\ \hline \text{Magical}(c) \end{array}$$

8.32

$$\begin{array}{l} \text{Horned}(c) \rightarrow (\text{Elusive}(c) \\ \quad \wedge \text{Dangerous}(c)) \\ (\text{Elusive}(c) \vee \text{Mythical}(c)) \rightarrow \text{Rare}(c) \\ \text{Mammal}(c) \rightarrow \neg \text{Rare}(c) \\ \hline \text{Horned}(c) \rightarrow \neg \text{Mammal}(c) \end{array}$$

8.33

$$\begin{array}{l}(\text{Horned}(c) \rightarrow (\text{Elusive}(c) \wedge \text{Magical}(c))) \\ \quad \wedge (\neg\text{Horned}(c) \rightarrow (\neg\text{Elusive}(c) \\ \quad \wedge \neg\text{Magical}(c))) \\ \neg\text{Horned}(c) \rightarrow \neg\text{Mythical}(c) \\ \hline \text{Horned}(c) \leftrightarrow (\text{Magical}(c) \vee \text{Mythical}(c))\end{array}$$

8.34

$$\begin{array}{l}(\text{Tet}(a) \wedge \text{Large}(a)) \vee (\text{Cube}(a) \\ \quad \wedge \text{Small}(a)) \\ \neg\text{Small}(b) \\ (\text{Tet}(a) \vee \text{Cube}(a)) \rightarrow (\text{Large}(b) \\ \quad \vee \text{Small}(b)) \\ \text{Tet}(a) \rightarrow \text{Medium}(b) \\ \hline \text{Small}(a) \wedge \text{Large}(b)\end{array}$$

8.35

$$\begin{array}{l}\neg\text{Cube}(b) \rightarrow \text{Small}(b) \\ \text{Small}(c) \rightarrow (\text{Small}(d) \vee \text{Small}(e)) \\ \text{Small}(d) \rightarrow \neg\text{Small}(c) \\ \text{Cube}(b) \rightarrow \neg\text{Small}(e) \\ \hline \text{Small}(c) \rightarrow \text{Small}(b)\end{array}$$

8.36

$$\begin{array}{l}\text{SameRow}(d,a) \vee \text{SameRow}(d,b) \\ \quad \vee \text{SameRow}(d,c) \\ \text{SameRow}(d,b) \rightarrow (\text{SameRow}(d,a) \\ \quad \rightarrow \neg\text{SameRow}(d,c)) \\ \text{SameRow}(d,a) \leftrightarrow \text{SameRow}(d,c) \\ \hline \text{SameRow}(d,a) \leftrightarrow \neg\text{SameRow}(d,b)\end{array}$$

8.37

$$\begin{array}{l}\text{Cube}(a) \vee \text{Dodec}(a) \vee \text{Tet}(a) \\ \text{Small}(a) \vee \text{Medium}(a) \vee \text{Large}(a) \\ \text{Medium}(a) \leftrightarrow \text{Dodec}(a) \\ \text{Tet}(a) \leftrightarrow \text{Large}(a) \\ \hline \text{Cube}(a) \leftrightarrow \text{Small}(a)\end{array}$$

(8.38) Use o programa **Fitch** para fazer duas provas formais de necessidade lógica (provar uma sentença a partir de nenhuma premissa). Na primeira prove **(P ∧ Q) → P** e na segunda prove **¬(P ∧Q) ∨ P**. Note que esta segunda sentença é equivalente à primeira! (Os cabeçalhos destas provas estão nos arquivos **Exercise 8.38.1** e **Exercise 8.38.2**) Após fazer estas provas, acho que você não terá mais dúvidas sobre por que é conveniente introduzir → em FOL, ao invés de defini-lo através dos conectivos booleanos.

*Decida se os dois argumentos seguintes são prováveis (demonstráveis) em F_T sem, de fato, tentar fazer suas provas. Faça isso construindo tabelas de verdade no programa **Boole** para avaliar a validade tautológica dos argumentos. Grave suas tabelas e escreve uma explicação clara, com base nos teoremas da correção e completude, de porque o argumento (é ou não) provável.*

8.39

$$\begin{array}{l}A \wedge (B \vee \neg A \vee (C \wedge D)) \\ E \wedge (D \vee \neg(A \wedge (B \vee D))) \\ \hline A \wedge B\end{array}$$

8.40

$$\begin{array}{l}A \wedge (B \vee \neg A \vee (C \wedge D)) \wedge \neg(A \wedge D) \\ \hline \neg(E \wedge (D \vee \neg(A \wedge (B \vee D))))\end{array}$$

Na prova do Teorema da Correção, nós tratamos apenas três das dose regras de F_T. Os próximos três problemas pedem para você tratar mais algumas regras.

(8.41) Faça o caso da prova do Teorama da Correção referente à regra **(¬Elim)**. Seu argumento deverá ser bastante similar ao apresentado para a regra **(→Elim)**.

(8.42) Faça o caso da prova do Teorema da Correção referente à regra **(¬Intro)**. Seu argumento deverá ser similar ao apresentado no texto para a regra **(→Intro)**.

(8.43) Faça o caso da prova do Teorema da Correção referente à regra **(∨Elim)**.

*Nos exercícios seguintes há vários argumentos na linguagem dos blocos. Para cada um deles avalie se é logicamente válido ou não. Se for válido, construa uma prova formal que mostre isto (ATENÇÃO: só é permitido o uso de **AnaCon** para derivar ⊥ a partir de sentenças atômicas. Qualquer outro uso de **AnaCon** é proibido!). Se o argumento for inválido, construa um contraexemplo com o programa **Mundo de Tarski**.*

8.44 ➹

$$\frac{\text{Adjoins}(a,b) \wedge \text{Adjoins}(b,c),\ \text{SameRow}(a,c)}{a \neq c}$$

8.45 ➹

$$\frac{}{\neg(\text{Cube}(b) \wedge b = c) \vee \text{Cube}(c)}$$

8.46 ➹

$$\frac{\text{Cube}(a) \vee (\text{Cube}(b) \rightarrow \text{Tet}(c)),\ \text{Tet}(c) \rightarrow \text{Small}(c),\ (\text{Cube}(b) \rightarrow \text{Small}(c)) \rightarrow \text{Small}(b)}{\neg\text{Cube}(a) \rightarrow \text{Small}(b)}$$

8.47 ➹

$$\frac{\text{Small}(a) \wedge (\text{Medium}(b) \vee \text{Large}(c)),\ \text{Medium}(b) \rightarrow \text{FrontOf}(a,b),\ \text{Large}(c) \rightarrow \text{Tet}(c)}{\neg\text{Tet}(c) \rightarrow \text{FrontOf}(c,b)}$$

8.48 ➹

$$\frac{\text{Small}(a) \wedge (\text{Medium}(b) \vee \text{Large}(c)),\ \text{Medium}(b) \rightarrow \text{FrontOf}(a,b),\ \text{Large}(c) \rightarrow \text{Tet}(c)}{\neg\text{Tet}(c) \rightarrow \text{FrontOf}(a,b)}$$

8.49 ➹

$$\frac{(\text{Dodec}(a) \wedge \text{Dodec}(b)) \rightarrow (\text{SameCol}(a,c) \rightarrow \text{Small}(a)),\ (\neg\text{SameCol}(b,c) \wedge \neg\text{Small}(b)) \rightarrow (\text{Dodec}(b) \wedge \neg\text{Small}(a)),\ \text{SameCol}(a,c) \wedge \neg\text{SameCol}(b,c)}{\text{Dodec}(a) \rightarrow \text{Small}(b)}$$

8.50 ➹

$$\frac{\text{Cube}(b) \leftrightarrow (\text{Cube}(a) \leftrightarrow \text{Cube}(c))}{\text{Dodec}(b) \rightarrow (\text{Cube}(a) \leftrightarrow \neg\text{Cube}(c))}$$

8.51 ➹

$$\frac{\text{Cube}(b) \leftrightarrow (\text{Cube}(a) \leftrightarrow \text{Cube}(c))}{\text{Dodec}(b) \rightarrow a \neq b}$$

8.52 ➹

$$\frac{\text{Cube}(b) \leftrightarrow (\text{Cube}(a) \leftrightarrow \text{Cube}(c))}{\text{Dodec}(b) \rightarrow a \neq c}$$

8.53 ➹

$$\frac{\text{Small}(a) \rightarrow \text{Small}(b),\ \text{Small}(b) \rightarrow (\text{SameSize}(b,c) \rightarrow \text{Small}(c)),\ \neg\text{Small}(a) \rightarrow (\text{Large}(a) \wedge \text{Large}(c))}{\text{SameSize}(b,c) \rightarrow (\text{Large}(c) \vee \text{Small}(c))}$$

Questões Gerais Importantes

Questões Importantes (Capítulos 1 a 8)

Questão 1

1.1. O que é um argumento?

1.2. Quando um argumento é logicamente válido?

1.3. Quando um argumento é correto?

Questão 2

2.1. O que é conseqüência lógica?

2.2. O que é equivalência lógica?

2.3. O que é necessidade lógica?

2.4. O que é possibilidade lógica?

2.5. Como definir equivalência lógica, necessidade lógica e possibilidade lógica através do conceito de conseqüência lógica?

Questão 3

Quais as características principais de um argumento com premissas inconsistentes quanto à validade e correção? Por que?

Questão 4

4.1. Qual a lei do terceiro excluído?

4.2. Que dizem as leis de DeMorgan?

4.3. Como definir $\rightarrow$ e $\leftrightarrow$ com através dos conectivos booleanos?

Questão 5

O que significa dizer que um conectivo é uma função de verdade?

Questão 6

O que significa dizer que um conjunto de conectivos é verofuncionalmente completo?

Questão 7

Como se prova que um conectivo NÀO é verofuncional?

Questão 8

Como distinguir se determinada 'sugestão' que uma sentença carrega é parte efetiva de seu significado, devendo portanto ser levada em consideração quando se traduz a sentença para FOL, ou se é apenas uma insinuação social, algo que não é parte efetiva do significado da sentença, mas apenas algo que seu proferimento sugere?

Questão 9

O que é uma prova?

Questão 10

Explique por que uma prova formal demonstra um argumento!

Questão 11

11.1. O que é uma constante individual?

11.2. O que é um símbolo de predicado?

11.3. O que é um símbolo de função?

11.4. Qual a diferença entre símbolo de predicado e símbolo de função?

Questão 12

12.1. O que é uma sentença atômica?

12.2. O que é um literal?

12.3. O que é um conectivo?

12.4. O que é uma sentença molecular ou complexa?

Questão 13

13.1. Quais as tabelas de verdade para os conectivos $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$?

13.2. Como demonstrar que uma sentença é uma necessidade lógica através do método das tabelas de verdade?

13.3. Como demonstrar que duas sentenças são logicamente equivalentes através do método das tabelas de verdade?

13.4. Como demonstrar que uma sentença é conseqüência lógica de um conjunto de sentenças através do método das tabelas de verdade?

Questão 14

14.1. De que forma o conectivo $\leftrightarrow$ pode relacionar equivalência lógica com necessidade lógica?

14.2. De que forma o conectivo $\rightarrow$ pode relacionar conseqüência lógica com necessidade lógica?

Questão 15

Explique as semelhanças e diferenças entre os conceitos:

15.1. (a) Necessidade Lógica - (b) Tautologia (ou TT-necessidade) - (c) TW-necessidade

15.2. (a) Possibilidade Lógica - (b) TT-possibilidade (ou Possibilidade Tautológica) - (c) TW-possibilidade

15.3. (a) Equivalência Lógica - (b) Equivalência Tautológica

15.4. (a) Conseqüência Lógica - (b) Conseqüência Tautológica

Questão 16

Quais as regras de introdução e eliminação do sistema **F** para os conectivos: **=**, $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\bot$.

Questão 17

17.1. O que significa dizer que um sistema formal (o Sistema **F**, por exemplo) é completo?

17.2. O que significa dizer que um sistema formal (o Sistema **F**, por exemplo) é correto?

17.3. Uma vez que um sistema seja correto e completo, quais as conseqüências desse fato com relação à definição de conseqüência lógica neste sistema?

Todas as Sentenças
do Programa Mundo de Tarski

Todas as Sentenças

Sentenças de Abelardo

1. Cube(d) → Cube(f)
2. Cube(d) → Tet(f)
3. Dodec(d) → Tet(f)
4. Dodec(d) → Cube(f)
5. SameSize(a, b) → SameRow(a, b)
6. SameSize(a, b) → SameSize(a, c)
7. SameSize(a, c) → SameSize(a, b)
8. SameSize(a, c) → SameSize(a, d)
9. (Tet(a) ∧ Cube(c)) → Dodec(d)
10. LeftOf(a, b) → RightOf(b, a)
11. LeftOf(e, d) → RightOf(d, e)
12. (LeftOf(a, c) ∧ LeftOf(c, b)) → Between(c, a, b)
13. Cube(c) → (Large(c) → (Cube(c) ∧ Large(c)))
    ; Make sure you understand why this sentence is
    ; true. Try playing the game committed to False.
14. ¬(Tet(a) → Large(a)) → (Tet(a) ∧ ¬Large(a))
15. Large(d) ↔ Small(a)
16. Dodec(d) ↔ Large(d)
17. Adjoins(a, f) ↔ Adjoins(a, d)
18. ¬(Large(e) ↔ Small(b))
19. (Small(c) ∧ (Cube(a) ∨ Cube(d))) ↔
    ((Small(c) ∧ Cube(a)) ∨ (Small(c) ∧ Cube(d)))
20. Cube(b) ↔ (Cube(c) ↔ Large(c))
    ; This kind of sentence is hard to understand. Play
    ; the game to help you understand it better.

Sentenças de Ackermann

1. Cube(a)
2. Cube(b)
3. Dodec(c)
4. Tet(c)
5. Larger(d, a)
6. Larger(c, b)
7. BackOf(c, a)
8. Between(b, a, c)
9. Medium(c)
10. LeftOf(a, c)

Sentenças de Alan Robinson

1. Small(d) ∨ ¬Small(e) ∨ BackOf(a, b)
2. ¬BackOf(a, b) ∨ Cube(d) ∨ Cube(e)
3. Small(d) ∨ Small(e) ∨ Cube(d)
4. ¬Small(d) ∨ Cube(d)
5. ¬Cube(e)
6. ¬Cube(d)
7. Small(d) ∨ ¬Small(e) ∨ Cube(d) ∨ Cube(e)
8. Small(d) ∨ Cube(d) ∨ Cube(e)
9. Cube(d) ∨ Cube(e)
10. Cube(e)
11. ⊥
    ; This represents the empty clause (box).

Sentenças de Allan

1. ∃x (Dodec(x) ∧ Large(x))
2. ∃x (Dodec(x) → Large(x))
3. ∀x (Tet(x) ∧ Small(x))
4. ∀x (Tet(x) → Small(x))

Sentenças de Aristóteles

1. $\exists x\ (Tet(x) \wedge Large(x))$
2. $\exists x\ (Tet(x) \wedge Medium(x))$
3. $\exists x\ (Cube(x) \wedge \neg Small(x))$
4. $\exists y\ (Dodec(y) \wedge \neg Large(y))$
5. $\forall x\ (Cube(x) \rightarrow Medium(x))$
6. $\forall x\ (Dodec(x) \rightarrow Small(x))$
7. $\forall x\ (Tet(x) \rightarrow \neg Small(x))$
8. $\forall y\ (Cube(y) \rightarrow \neg Tet(y))$

Sentenças de Arnault

1. $\exists x\ \exists y\ \exists z\ (Cube(x) \wedge Dodec(y) \wedge Tet(z))$
2. $\neg\exists x\ Large(x)$
3. $\forall x\ (Dodec(x) \rightarrow \exists y\ (Cube(y) \wedge BackOf(x, y)))$
4. $\forall x\ (Tet(x) \rightarrow \exists y\ \exists z\ (Between(x, y, z)))$
5. $\forall x\ \forall y\ \forall z\ (Between(x, y, z) \rightarrow Larger(x, y))$
6. $\exists x\ \exists y\ (x \# y \wedge \forall w\ ((w = x \vee w = y) \rightarrow \forall z\ \neg BackOf(z, w)))$
7. $\forall x\ (Cube(x) \leftrightarrow \exists y\ (Tet(y) \wedge BackOf(y, x)))$
8. $\forall x\ \forall y\ (Larger(x, y) \rightarrow \exists z\ Between(x, y, z))$
9. $\neg\forall x\ \forall y\ (LeftOf(x, y) \vee RightOf(x, y))$
10. $\exists x\ \exists y\ \neg(FrontOf(x, y) \vee BackOf(x, y))$

Sentenças de Austin

1. Larger(a, b)
2. Between(c, d, b)
3. LeftOf(c, b)
4. FrontOf(c, b)
5. BackOf(e, b)
6. RightOf(a, b)

Sentenças de Bernays

1. SameRow(a, b)
2.
3. SameCol(d, b)
4.
5. a = d
6.
7. SameSize(a, b)
8.
9. SameShape(a, b)
10.
11. BackOf(c, b)
12.
13. LeftOf(a, b)
14.
15. Medium(c)
16.
17. Larger(a, b)
18.
19. Cube(a)
20.

Sentenças de Bernstein

1. $\forall x\ Cube(x) \rightarrow Small(x)$
   ; This still has a free occurrence of x. Fix it with
   ; some parentheses.
2. $\exists a\ Cube(a)$
   ; Don't forget that quantifiers apply only to
   ; variables.
3. $\exists v\ Cube(v) \wedge Medium(v) \wedge Larger(v, c)$
4. $\exists u\ (Small(u) \wedge Cube(u)$
5. $\neg\exists x\ Larger(a, x) \wedge Larger(x, a)$
6. $\forall w\ SameRow(w, b) \rightarrow SameRow(b, w)$
7. $\forall x\ \forall y\ \forall z\ LeftOf(x, y) \wedge LeftOf(y, z) \rightarrow LeftOf(x, z)$
8. $\forall x\ \forall y\ (Larger(a, b) \rightarrow Cube(a) \wedge Dodec(b))$
   ; This one is a bit tricky. The problem is some
   ; missing parentheses.
9. $\forall x\ \forall y\ \ Cube(x) \wedge Cube(y) \rightarrow LeftOf(x, y)$
10. $\forall x\ (Cube(x) \rightarrow \exists x\ Between(x, x, y))$

Sentenças Between

1. Between(a, b, c) → Smaller(b, c)

2. Smaller(b, c) → [RightOf(b, c) ∧ Between(a, b, b)]

3. Between(c, b, a) → [Between(b, a, c) ∨ c = b]

4. Between(a, b, c) ∨ Between(b, a, c) ∨
   Between(c, b, a)

5. ¬Between(b, a, c)

Sentenças do Argumento de Bill

1. Between(b, c, d)	;Premise
2. Between(a, b, d)	;Premise
3. LeftOf(a, c)	;Premise
4. Between(b, a, d)	;Conclusion

Sentenças de Boolos

1. Between(a, c, d) ↔ Between(b, c, e)

2. e # f → (Adjoins(b, c) ∧ Adjoins(e, b) ∧ Adjoins(b, f))

3. SameCol(e, f) → Smaller(e, e)

4. ¬((SameRow(e, b) ∧ SameRow(b, c) ∧
   SameRow(c, f)) → ¬Between(a, c, d))

5. LeftOf(e, b) → RightOf(a, e)

Sentenças de Bozo

1. Small(Cube(a)) FrontOf Tet(e)
   ; It looks like Bozo is trying to say that a small
   ; cube named a is in front of a tetrahedron named e.

2. ¬SameCol(x, b)

3. Cube(a) ∧ Cube(b) ∨ Cube(c)
   ; There are two different ways of adding
   ; parentheses that can turn this into a sentence.

4. Cube(a) ↔ Cube(b) ↔ Cube(e)

5. ∃x ¬(Cube x)

6. ∃a (Cube(a) ∧ Small(a))
   ; Bozo wants to say that there is a small cube.

7. ∃x Cube(x) ∧ Small(x)
   ; Bozo tries again to say that there is a small cube.

8. ∃y (Dodec(y) ∧ Large(y))
   ; Bozo left out a negation sign. There are four
   ; places you could put it, but one would make the
   ; sentence false.

9. ∀y (Cube(x) → ¬Medium(x))

10. ∀x (Tet(x) ∧ Small(x) → FrontOf(x, e))

Sentenças de Boole

1. ¬Medium(c) ∧ Smaller(c, a)

2. ¬(Medium(c) ∧ Smaller(c, a))
   ; Make sure you understand why this sentence and
   ; (1) have different truth values.

3. ¬Cube(d) ∨ Tet(f)

4. ¬(Cube(d) ∨ Tet(f))
   ; Make sure you understand why this sentence and
   ; (3) have different truth values.

5. ¬Large(c) ∧ Larger(c, a)

6. ¬(Large(c) ∧ Larger(c, a))

7. ¬(Cube(d) ∧ Cube(f))
   ; Predict whether this sentence will have the same
   ; truth value as 8 or as 9.

8. ¬Cube(d) ∧ ¬Cube(f)

9. ¬Cube(d) ∨ ¬Cube(f)

10. ¬(Cube(d) ∨ Cube(f))
    ; Predict whether this sentence will have the same
    ; truth value as 8 or as 9.

11. ¬[LeftOf(c, f) ∨ RightOf(c, f)]
    ; Brackets [ ] and braces { } work the same as
    ; parentheses ( ).

12. ¬[LeftOf(a, d) ∨ RightOf(a, d)]

13. Tet(a) ∨ (Tet(f) ∧ Tet(c))

14. (Tet(a) ∨ Tet(f)) ∧ Tet(c)

15. Dodec(a) ∨ Dodec(b) ∨ Dodec(c)

16. LeftOf(a, c) ∧ LeftOf(c, b) ∧ ¬Between(c, a, b)

17. ¬¬(BackOf(e, b) ∧ ¬FrontOf(c, b))

18. BackOf(d, a) ∧ LeftOf(d, e) ∧ FrontOf(b, e) ∧
    Between(c, d, b)

19. Smaller(c, e) ∨ ¬(Cube(a) ∨ Cube(d))

20. ¬(¬Dodec(e) ∨ ¬ ¬Tet(f))

Sentenças de Brouwer

1. Tet(a)

2. Adjoins(a, b)
   ; This claims that a is on a square adjacent to b,
   ; but it is false since there is a square in between.

3. ¬SameRow(e, d)

4. SameCol(a, f)

5. ¬SameRow(a, b)

6. Between(a, d, f)

7. ¬ ¬Larger(f, e)
   ; Two negation signs in a row cancel each other out,
   ; so this claims that f is larger than e.

8. ¬ ¬LeftOf(a, a)

9. ¬ ¬ ¬Cube(c)
   ; Three negation signs in a row are equivalent to
   ; one, so this sentence is equivalent to the claim
   ; that c is not a cube. This is true in Boole's World,
   ; since c is a tet.

10. ¬ ¬ ¬Tet(c)

Sentenças de Buridan

1. a # b ∧ b # c ∧ a # c

2. Larger(a, b) ∧ ¬Larger(a, c)

3. ¬∃x Small(x)

4. ∀x (¬Small(x) → ¬Cube(x))

5. Tet(a) ∧ Dodec(b) ∧ Dodec(c)

6. ∃x (Dodec(x) ∧ ∃y ∃z (Tet(y) ∧ Tet(z) ∧
   Between(x, y, z)))

7. ∀x (Medium(x) → ∃y FrontOf(x, y))

8. ∀x (∃y FrontOf(x, y) → Medium(x))

9. ∀x (LeftOf(x, c) → (Dodec(x) ∨ Large(x)))

10. ∀x (∃y BackOf(x, y) → (x = a ∨ x = b ∨ x = c))

Sentenças de Cantor

1. ∀x ∀y [(Cube(x) ∧ Cube(y)) → (LeftOf(x, y) ∨
   RightOf(x, y))]

2. ∃x ∃y (Cube(x) ∧ Cube(y))

Sentenças de Carnap

1. ∀x (LeftOf(x, b) → RightOf(b, x))

2. ∀x ((Small(x) ∧ BackOf(x, c)) → Dodec(x))

3. ∀x ((Cube(x) ∧ x # b)
   → (Larger(x, b) ∨ Smaller(x, b)))

4. Dodec(d) → ∀x (x = d → Dodec(x))

5. ∃y (Smaller(a, y) ∧ Smaller(y, b)) → Smaller(a, b)

6. ∀x (Larger(x, c) → x # c)

7. ∀x (Between(x, a, d) ∨ ¬Between(x, a, d))

8. ∀x (Between(x, a, d) ∨ ¬Between(x, d, a))

9. ∀x (Dodec(x) → (x = d ∨ Small(x)))

10. ∀x (Cube(x) → LeftOf(x, e))
    → ¬∃y (Cube(y) ∧ ¬LeftOf(y, e))

Sentenças de Church

1. ∀x Large(x)

2. Large(a) ∧ Large(b) ∧ Large(c)

3. ∃y Small(y)

4. Small(a) ∨ Small(b) ∨ Small(c)

Sentenças CNF

1. (LeftOf(a, b) ∨ BackOf(a, b)) ∧ Cube(a)

2.
   ; Write a sentence in disjunctive normal form (DNF)
   ; that is logically equivalent to the CNF sentence
   ; in (1). Do the same thing for the sentences below.

3. Larger(a, b) ∧ (Cube(a) ∨ Tet(a) ∨ a = b)

4.

5. (Between(a, b, c) ∨ Tet(a) ∨ ¬Tet(b)) ∧ Dodec(c)

6.

7. Cube(a) ∧ Cube(b) ∧ (¬Small(a) ∨ ¬Small(b))

8.

9. (Small(a) ∨ Medium(a)) ∧ (Cube(a) ∨ ¬Dodec(a))

10.

Sentenças Conditional

1. $(Small(e) \wedge Dodec(e)) \rightarrow LeftOf(e, f)$
2. $Tet(c) \rightarrow (Tet(e) \vee Dodec(e))$
3. $e \# c \rightarrow Tet(e)$
4. $Small(e) \leftrightarrow Dodec(e)$
5. $LeftOf(e, f) \leftrightarrow Larger(f, e)$

Sentenças DeMorgan 2

1. $\neg\forall x\ (Cube(x) \rightarrow Small(x))$
2. $\neg\exists x\ (Cube(x) \wedge Large(x))$
3. $\neg\forall x\ (Large(x) \leftrightarrow Dodec(x))$
4. $\forall x\ (\neg Large(x) \vee \neg Cube(x))$
5. $\exists x\ (\neg Small(x) \wedge Cube(x))$
6. $\exists x\ ((Large(x) \wedge \neg Dodec(x)) \vee (Dodec(x) \wedge \neg Large(x)))$

Sentenças DeMorgan

1. $\neg(Dodec(a) \vee Tet(a))$
2. $\neg Dodec(a) \wedge \neg Tet(a)$
3. $\neg(Small(b) \wedge Cube(b))$
4. $\neg Small(b) \vee \neg Cube(b)$
5. $\neg FrontOf(c, b) \wedge \neg BackOf(c, b)$
6. $\neg(FrontOf(c, b) \vee BackOf(c, b))$
7. $\neg LeftOf(b, c) \vee \neg LeftOf(c, a)$
8. $\neg(LeftOf(b, c) \wedge LeftOf(c, a))$

Sentenças Distributive

1. $Cube(a) \wedge (LeftOf(a, b) \vee BackOf(a, b))$
2. $(Cube(a) \wedge LeftOf(a, b)) \vee$
   $(Cube(a) \wedge BackOf(a, b))$
3. $Small(c) \vee (a = b \wedge b = c \wedge c = d)$
4. $(Small(c) \vee a = b) \wedge$
   $(Small(c) \vee b = c) \wedge$
   $(Small(c) \vee c = d)$
5. $(Small(d) \wedge Smaller(e, d)) \vee Large(d)$
6. $(Small(d) \vee Large(d)) \wedge (Smaller(e, d) \vee Large(d))$

Sentenças DNF

1. $\neg[\neg((Tet(a) \wedge \neg Large(a)) \vee (Cube(a) \wedge \neg Medium(a)))$
   $\wedge \neg(Dodec(b) \wedge (Large(b) \vee Tet(b)))]$
2. $(Tet(a) \wedge \neg Large(a)) \vee$
   $(Cube(a) \wedge \neg Medium(a)) \vee$
   $(Dodec(b) \wedge Large(b)) \vee$
   $(Dodec(b) \wedge Tet(b))$

Sentenças de Dodgson

1. $\forall x\ (Tet(x) \rightarrow Large(x))$
2. $\forall x\ (Tet(x) \rightarrow Medium(x))$
3. $\forall x\ (Tet(x) \rightarrow Small(x))$
4. $\forall x\ (Tet(x) \rightarrow (Small(x) \wedge Large(x)))$
5. $\forall x\ (Tet(x) \rightarrow Cube(x))$

Sentenças de Edgar

1. $\exists x\ Tet(x)$
2. $\exists x\ (Tet(x) \wedge Large(x))$
3. $\exists x\ (Tet(x) \vee Large(x))$
4. $\exists x\ ((Tet(x) \wedge \neg Tet(x)) \vee Large(x))$
5. $\exists x\ (\neg Tet(x) \vee Large(x))$
6. $\exists x\ (Tet(x) \rightarrow Large(x))$
7. $\exists x\ (\neg Cube(x) \vee Between(x, a, b))$
8. $\exists x\ (Cube(x) \rightarrow Between(x, a, b))$
9. $\exists x\ (Cube(x) \wedge Between(x, a, b))$
10. $\exists x\ Dodec(x) \wedge \exists y\ Large(y)$
11. $\exists x\ Dodec(x) \wedge \exists x\ Large(x)$
12. $\exists x\ (Dodec(x) \wedge Large(x))$

Sentenças de Euler

1. Tet(a) $\vee$ Tet(b) $\vee$ Tet(c)
   ; So at least one of a, b, and c is a Tet. Reason by
   ; cases using the next three sentences to figure out
   ; the shapes of the three blocks.

2. Tet(a) $\rightarrow$ (Tet(b) $\wedge$ Tet(c))

3. Tet(b) $\rightarrow$ (Cube(a) $\wedge$ Tet(c))

4. Tet(c) $\rightarrow$ (Cube(a) $\wedge$ Dodec(b))

5. Larger(a, b) $\leftrightarrow$ Larger(b, c)

6. Larger(c, b) $\leftrightarrow$ Larger(b, a)

7. SameSize(a, c)
   ; Use this with 5 and 6 to determine the relative
   ; sizes of a, b, and c.

8. Small(a) $\rightarrow$ Medium(b)
   ; Using 8 and 9 you can now figure out their absolute
   ; sizes.

9. Medium(a) $\rightarrow$ Small(b)

Sentenças de Frege

1. $\exists$x $\exists$y LeftOf(x, y)

2. $\exists$x $\exists$y (Tet(x) $\wedge$ Cube(y))

3. $\exists$x $\exists$y (Tet(x) $\wedge$ LeftOf(x, y))

4. $\exists$x $\exists$y (Tet(x) $\wedge$ LeftOf(x, y) $\wedge$ Cube(y))

5. $\exists$x $\exists$y (Medium(x) $\wedge$ Tet(y))

6. $\exists$x $\exists$y (Medium(x) $\wedge$ Larger(y, x))

7. $\exists$x $\exists$y (Medium(x) $\wedge$ Tet(y) $\wedge$ Larger(y, x))

8. $\forall$x $\forall$y LeftOf(x, y)

9. $\forall$x $\forall$y (LeftOf(x, y) $\rightarrow$ Cube(x))

10. $\forall$x $\forall$y ((Cube(x) $\wedge$ Tet(y)) $\rightarrow$ LeftOf(x, y))

11. $\forall$x $\forall$y ((Large(x) $\wedge$ Small(y)) $\rightarrow$ LeftOf(x, y))

12. $\forall$x $\forall$y Larger(x, y)

13. $\forall$x $\forall$y (Larger(x, y) $\rightarrow$ Cube(x))

14. $\forall$x $\forall$y ((Cube(x) $\wedge$ Tet(y)) $\rightarrow$ Larger(x, y))

Sentenças Game

1. $\exists$x Cube(x)

2. $\exists$x Tet(x)

3. $\forall$x Small(x)

4. $\forall$x Large(x)

5. $\exists$x Cube(x) $\wedge$ $\exists$x Dodec(x)
   ; Compare the game in 5 and 6.

6. $\exists$x (Cube(x) $\wedge$ Dodec(x))

7. $\exists$y (Dodec(y) $\wedge$ Large(y))
   ; Compare the way the game goes in 7 with 8.

8. $\exists$y (Dodec(y) $\vee$ Large(y))

9. $\forall$z (Dodec(z) $\wedge$ Cube(z))

10. $\forall$z (Dodec(z) $\vee$ Cube(z))

Sentenças Henkin Construction

1. $\forall$x (Cube(x) $\rightarrow$ Small(x))

2. $\exists$x Cube(x)

3. $\exists$x Cube(x) $\rightarrow$ Cube(c)

4. $\exists$x (Dodec(x) $\wedge$ Small(x)) $\rightarrow$ (Dodec(d) $\wedge$ Small(d))

5. Small(c) $\rightarrow$ $\exists$x Small(x)

6. $\neg$(Cube(c) $\rightarrow$ Small(c))
   $\rightarrow$ $\exists$x $\neg$(Cube(x) $\rightarrow$ Small(x))

7. $\neg\forall$x (Cube(x) $\rightarrow$ Small(x))
   $\leftrightarrow$ $\exists$x $\neg$(Cube(x) $\rightarrow$ Small(x))

8. Dodec(d) $\wedge$ $\neg$Small(d)

Sentenças de Henkin

1. $\exists$x Cube(x) $\rightarrow$ Cube(c)

2. $\exists$x (Dodec(x) $\wedge$ Small(x)) $\rightarrow$ (Dodec(d) $\wedge$ Small(d))

3. Small(c) $\rightarrow$ $\exists$x Small(x)

4. $\neg$(Cube(c) $\rightarrow$ Small(c)) $\rightarrow$ $\exists$x $\neg$(Cube(x) $\rightarrow$ Small(x))

5. $\neg\forall$x (Cube(x) $\rightarrow$ Small(x)) $\leftrightarrow$
   $\exists$x $\neg$(Cube(x) $\rightarrow$ Small(x))

Sentenças de Hilbert

1. $\exists x\ \forall y$ Smaller(x, y)
   ; Could this ever be true?

2. $\forall x$ (Dodec(x) $\rightarrow \exists y$ Smaller(x, y))

3. $\exists x$ (Dodec(x) $\wedge \forall y$ Smaller(x, y))

4. $\exists y\ \forall x$ (Dodec(x) $\rightarrow$ Smaller(x, y))
   ; Try to say this in English. It's hard to say it
   ; clearly and unambiguously.

5. $\exists y\ \forall x$ (Dodec(x) $\rightarrow$ Smaller(y, x))

6. $\exists y\ \forall x$ (Dodec(x) $\rightarrow \neg$Smaller(x, y))

7. $\forall x$ ((Cube(x) $\wedge$ Medium(x)) $\rightarrow \neg\exists y$ BackOf(y, x))

8. $\forall x$ ((Cube(x) $\wedge$ Medium(x)) $\rightarrow \exists y \neg$BackOf(y, x))
   ; Do you understand the difference between 7 and
   ; 8? They don't mean the same thing.

9. $\forall x$ ((Cube(x) $\wedge$ Large(x)) $\rightarrow \neg\exists y$ BackOf(y, x))

10. $\forall x$ ((Cube(x) $\wedge$ Large(x)) $\rightarrow \exists y \neg$BackOf(y, x))

11. $\exists x$ (Tet(x) $\wedge \forall y$ (Cube(y) $\rightarrow$ BackOf(y, x)))

12. $\exists x$ (Tet(x) $\wedge \forall y$ (BackOf(y, x) $\rightarrow$ Cube(y)))

13. $\exists x$ (Cube(x) $\wedge \forall y$ (Dodec(y) $\rightarrow$ Smaller(y, x)))

14. $\exists x$ (Cube(x) $\wedge \forall y$ (Smaller(y, x) $\rightarrow$ Dodec(y)))

15. $\forall x$ ((Cube(x) $\wedge \exists y$ LeftOf(x, y)) $\rightarrow$ Large(x))

16. $\forall x$ ((Tet(x) $\wedge \exists y$ FrontOf(x, y)) $\rightarrow$ Small(x))

17. $\forall x$ ($\neg\exists y$ BackOf(y, x) $\rightarrow$ Cube(x))

18. $\forall x$ ($\neg\exists y$ FrontOf(y, x) $\rightarrow$ Tet(x))

19. $\forall x$ (Tet(x) $\rightarrow \exists y\ \exists z$ Between(x, y, z))

20. $\exists y\ \forall x$ (Tet(x) $\rightarrow \exists z$ Between(x, y, z))
   ; How does this differ from the one above? They
   ; mean very different things.

21. $\exists z\ \forall x$ (Tet(x) $\rightarrow \exists y$ Between(x, y, z))

22. $\exists y\ \exists z\ \forall x$ (Tet(x) $\rightarrow$ Between(x, y, z))
   ; Do you see how this can be false, even though 20
   ; and 21 are true?

23. $\exists y\ \forall x$ (Small(x) $\rightarrow$ SameRow(x, y))

24. $\forall x$ (Small(x) $\rightarrow \exists y$ (x # y $\wedge$ SameRow(x, y)))

25. $\forall x$ (Small(x) $\rightarrow \exists y$ (x # y $\wedge$ SameCol(x, y)))

Sentenças de Hercule

1. $\forall x$ ($\neg\exists y$ Smaller(y, x) $\rightarrow$ (x = c $\vee$ x = d $\vee$ x = e))
   ; You might want to skip this sentence until after
   ; you have understood the implications of 2 and 3.

2. $\forall x$ ((x = a $\vee$ x = d) $\leftrightarrow \exists y\ \exists z$ Between(x, y, z))
   ; This will let you determine the identity of d.

3. e = c $\leftrightarrow$ a = d

4. Dodec(b) $\wedge \exists v$ (Dodec(v) $\wedge$ LeftOf(b, v))

5. $\exists v$ (Dodec(f) $\wedge$ Dodec(v) $\wedge$ BackOf(f, v))

6. $\exists x$ (BackOf(x, e) $\wedge \exists y$ (BackOf(y, x) $\wedge$
   $\exists z$ BackOf(z, y)))

7. $\forall y$ (y = b $\rightarrow$ ($\neg\exists x$ Between(x, y, f) $\wedge$ (y = f $\vee$ y = c)))

8. $\exists x$ (Dodec(x) $\wedge$ BackOf(x, a) $\wedge$
   $\forall y$ ((Dodec(y) $\wedge$ BackOf(y, a)) $\rightarrow$ x = y))

9. $\exists x$ Between(x, e, c)

Sentenças de Horn

1. Between(a, b, c)

2. Cube(b) $\vee \neg$Dodec(a) $\vee \neg$Medium(a)

3. Dodec(a) $\vee \neg$Between(a, b, c) $\vee \neg$Medium(a)

4. Medium(a) $\vee \neg$Cube(b)

5. $\neg$Between(a, b, c) $\vee$ Cube(b)

6. Larger(b, a)

Outras Sentenças de Horn

1. $\neg$Large(e) $\vee$ Tet(f) $\vee \neg$Cube(a)

2. Cube(a)

3. $\neg$Tet(f) $\vee \neg$Large(e) $\vee$ Small(d)

4. Large(e)

5. $\neg$Cube(a) $\vee \neg$Large(e) $\vee \neg$Small(d)

Sentenças de Jon Russell

1. $\forall x\ [Tet(x) \vee \exists y\ LeftOf(x, y)]$

2. ;Put a prenex form of 1 in this space.

3. $\forall x\ \forall y[(x \# y \wedge Cube(x) \wedge Cube(y)) \rightarrow \exists z\ Between(z, y, x)]$

4.

5. $\forall x\ [(Cube(x) \wedge \exists y\ Smaller(y, x)) \rightarrow Medium(x)]$

6.

7. $\exists x\ [Cube(x) \rightarrow \forall x\ Small(x)]$

8.

9. $\exists x\ [Cube(x) \rightarrow \forall x\ Cube(x)]$

10.

11. $\neg[\forall x\ Cube(x) \vee \forall x\ Tet(x) \vee \forall x\ Dodec(x)]$

12.

13. $\neg\exists x[Cube(x) \wedge \forall y\ (Tet(y) \rightarrow \exists z\ Between(z, x, y))]$

14.

15. $\exists x\ Cube(x) \leftrightarrow \forall x\ Small(x)$

16.

17. $\neg[\forall x\ Tet(x) \leftrightarrow \forall y\ Small(y)]$

18.

19. $\forall v\ [\exists x\ Larger(x, v) \leftrightarrow \exists x\ LeftOf(x, v)]$

20.

Sentenças de Kleene

1. $LeftOf(c, b) \wedge (d = d)$
   ; We know this is true in Kleene's World since the
   ; first conjunct can be seen to be true and the
   ; second must be true as long as the name 'd' is in
   ; use.

2. $BackOf(b, a) \vee BackOf(a, b)$

3. $Dodec(c) \wedge LeftOf(d, a)$

4. $LeftOf(d, a) \wedge Cube(c)$

5. $Between(e, f, a) \vee Cube(c)$

6. $LeftOf(e, f) \vee RightOf(e, f)$

7. $\neg LeftOf(e, a) \vee \neg RightOf(e, a)$

8. $c = b \vee c \# b$

9. $e = f \vee e \# f$

10. $\neg Large(a)$

Sentenças de Leibniz

1. $\forall x\ (\exists y\ (x = y) \rightarrow x = d)$
   ; Do you see what this is trying to say about the
   ; world? (Notice the names.) You can't express
   ; that in the blocks language. Negate the whole
   ; sentence.

2. $\forall x\ \forall y\ ((Cube(x) \wedge Cube(y)) \rightarrow \exists z\ Between(z, x, y))$
   ; Now this sentence tries to make a reasonable
   ; claim. Why is it false? Play the game and see.
   ; Then fix it so it says what was intended.

3. $\forall x\ (Between(x, d, c) \rightarrow x = b)$

4. $\forall x\ (\exists y\ Between(x, y, c) \rightarrow x = b)$

5. $\forall x\ (\exists y\ Between(x, y, c) \rightarrow \neg Large(x))$

6. $\forall x\ (\exists y\ \exists z\ Between(x, y, z) \rightarrow \neg Large(x))$

7. $\forall x\ (\exists y\ \exists z\ Between(x, y, z) \rightarrow Tet(x))$

8. $\forall x\ (\neg\exists y\ LeftOf(y, x) \rightarrow x = a)$

9. $\forall x\ ((\neg\exists y\ LeftOf(y, x) \wedge \neg\exists y\ FrontOf(y, x)) \rightarrow x = a)$

10. $\forall x\ (\exists y\ \exists z\ (Between(x, y, z) \wedge Tet(y) \wedge Tet(z)) \rightarrow x = e)$

11. $\forall x\ (\exists y\ \exists z\ (Between(x, y, z) \wedge Cube(y) \wedge Cube(z)) \rightarrow x = b)$

12. $\forall y\ (\exists x\ \exists z\ (Between(x, y, z) \wedge x = b) \rightarrow (y = a \vee y = c))$

13. $\forall x\ \forall y\ ((Tet(x) \wedge Small(x) \wedge Tet(y) \wedge Small(y)) \rightarrow x = y)$
    ; Do you see what this says?

14. $\forall x\ \forall y\ ((Dodec(x) \wedge Small(x) \wedge Dodec(y) \wedge Small(y)) \rightarrow x = y)$
    ; If you understood the last one, you see why this is
    ; true as well.

15. $\forall x\ (Dodec(x) \rightarrow x = b)$
    ; This may look like a dumb thing to say, but make
    ; sure you understand it. Why is it true?

16. $\forall x\ (Dodec(x) \leftrightarrow x = b)$
    ; And do you see why this is false? Under what
    ; circumstances would it be true?

17. $\forall x\ ((Tet(x) \wedge Small(x)) \leftrightarrow x = b)$
    ; This time, we got it right.

18. $\exists y\ \forall x\ ((Tet(x) \wedge Small(x)) \leftrightarrow x = y)$
    ; Compare this with the previous sentence. Do
    ; you understand what it says? Play the game
    ; a couple times, committed to both true and false.

19. $\exists y\ \forall x\ ((Cube(x) \wedge Small(x)) \leftrightarrow x = y)$
    ; Play the game here, too. Do you see why can't you
    ; win when committed to true?

20.
    ; In this slot, write a sentence that says there's
    ; exactly one large tetrahedron. Pattern it on
    ; sentence 18.

Sentenças Lestrade

1. Dodec(a)
2. SameShape(b, a)
3. Cube(c)
4. Adjoins(b, c)
5. Between(b, c, f)
6. SameSize(f, e)
7. LeftOf(d, c)
8. Smaller(f, a)
9. b = e
10. Larger(c, b)

Sentenças de Löwenheim

1. ;============ Set One (2-5) =====
2. $\exists x \exists y (x \# y \wedge SameCol(x, y))$
   ; Some of the parties are not lonely.
3. $\exists x \exists y (x \# y \wedge \neg SameCol(x, y))$
4. $\exists x \neg\exists y (x \# y \wedge SameCol(x, y))$
5. $\neg\exists x \exists y (x \# y \wedge SameCol(x, y))$
6. ;============ Set Two (7-10) ====
7. $\forall x \forall y (x \# y \rightarrow SameCol(x, y))$
   ; There is only one party.
8. $\forall x \forall y (x \# y \rightarrow \neg SameCol(x, y))$
9. $\forall x \neg\forall y (x \# y \rightarrow SameCol(x, y))$
10. $\neg\forall x \forall y (x \# y \rightarrow SameCol(x, y))$

Sentenças de Ludwig

1. $\forall x$ Large(x)
2. Large(a) $\wedge$ Large(b) $\wedge$ Large(c)
3. $\exists y$ Small(y)
4. Small(a) $\vee$ Small(b) $\vee$ Small(c)

Sentenças de Maigret

1. Between(c, a, d) $\wedge \neg$Tet(c)
2. $\exists x$ BackOf(x, a) $\wedge \exists x$ FrontOf(x, c)
3. FrontOf(a, c) $\wedge \neg\exists x$ (BackOf(x, a) $\wedge$ FrontOf(x, c))
4. $\exists x$ LeftOf(x, d) $\leftrightarrow$ Large(b)
5. $\exists x \exists y (\neg$Tet(x) $\wedge \neg$Tet(y) $\wedge$ Between(b, x, y))
6. FrontOf(d, b) $\wedge$ LeftOf(d, f)
7. Dodec(e)
8. $\neg\exists x$ LeftOf(x, e) $\vee$ Large(e)

Sentenças de Mary

1. $\exists x \exists y \exists z \exists w \forall u (u = x \vee u = y \vee u = z \vee u = w)$
2. $\exists x$ (Dodec(x) $\wedge \forall y$ (Dodec(y) $\rightarrow$ y = x))
3. $\forall x$ (Dodec(x) $\rightarrow$ Small(x))
4. $\neg\exists v$ Tet(v)
5. $\neg\exists x \exists y$ (Large(x) $\wedge$ Large(y) $\wedge$ x # y)

Sentenças de Max

1. Dodec(f) $\wedge$ Dodec(b) $\wedge$ SameRow(f, b)
2. Tet(a) $\wedge$ Tet(d) $\wedge$ FrontOf(a, d)
3. Cube(c) $\wedge$ Cube(e) $\wedge \neg$SameRow(c, e)
4. Between(b, d, f) $\wedge \neg$Between(b, e, f)
5. $\neg$(Large(d) $\wedge$ Large(e))
   ; Notice that the negation sign applies to the whole
   ; expression that follows it, because of the
   ; parentheses.
6. Larger(e, f) $\wedge$ Larger(b, f)
7. SameSize(d, c) $\wedge$ Large(c)
8. $\neg$BackOf(c, e) $\wedge \neg$BackOf(e, d)
   ; Notice that this sentence is a conjunction of two
   ; sentences, each starting with a negation sign.
9. $\neg$(Small(f) $\wedge$ Small(a))
10. $\neg(\neg$Adjoins(b, f) $\wedge$ SameRow(b, f))
    ; This one is a bit tricky. It is a negation of a
    ; conjunction; the first conjunct starts with a
    ; negation but the second does not.

Sentenças Mixed (misturadas)

1. $\forall x \exists y$ SameRow(x, y)
2. $\exists y \forall x$ SameRow(x, y)

Sentenças de Montague

1. $\forall x$ (Cube(x) $\rightarrow$ ----------)
   ;$\forall x$ (Cube(x) $\rightarrow$ x-is-to-the-left-of-every-tet)

2. $\forall x$ ((Cube(x) $\wedge$ Small(x)) $\rightarrow$
   ----------)
   ;$\forall x$ ((Cube(x) $\wedge$ Small(x)) $\rightarrow$
   ; x-is-in-back-of-a-large-cube)

3. $\exists x$ (Cube(x) $\wedge$ ----------)
   ;$\exists x$ (Cube(x) $\wedge$ x-is-in-front-of-every-tet)

4. $\exists x$ (Cube(x) $\wedge$ Large(x) $\wedge$
   ----------)
   ;$\exists x$ (Cube(x) $\wedge$ Large(x) $\wedge$
   ; x-is-in-front-of-a-small-cube)

5. $\forall x \neg$(----------)
   ;$\forall x \neg$(x-is-larger-than-everything)

6. $\forall x$ ((Cube(x) $\wedge$ ----------) $\rightarrow$
   Large(x))
   ;$\forall x$ ((Cube(x) $\wedge$ x-is-in-front-of-every-tet) $\rightarrow$;Large(x))

7. $\forall x$ (---------- $\rightarrow$ Small(x))
   ;$\forall x$ (x-is-to-the-right-of-a-large cube $\rightarrow$ Small(x))

8. $\forall x$ ((---------- $\wedge$
   ----------) $\rightarrow \neg$Large(x))
   ;$\forall x$ ((x-is-in-back-of-a-cube $\wedge$
   ; x-is-in-front-of-a-cube) $\rightarrow \neg$Large(x))

9. $\forall x$ (---------- $\rightarrow$ Cube(x))
   ;$\forall x$ (there-is-nothing-in-back-of-x $\rightarrow$ Cube(x))

10. $\forall x$ (Dodec(x) $\rightarrow$ ----------)
    ;$\forall x$ (Dodec(x) $\rightarrow$ x-is-smaller-than-some-tet)

Sentenças Null Quantification

1. $\exists$v BackOf(b, c)

2. ; Write a quantifier-free sentence equivalent
   ; to 1 here, and similarly for the next two
   ; sentences.

3. $\forall x$ Dodec(b)

4.

5. $\exists x$ (Cube(c) $\wedge$ Small(c))

6.

7. $\exists x \forall y$ (Tet(y) $\rightarrow$ Small(y))

8. ; Get rid of the null quantifier.

9. $\exists y \forall y$ (Tet(y) $\rightarrow$ Small(y))

10. ; Which quantifier is null?

Sentenças Mais CNF

1. $\neg$[(Cube(a) $\wedge \neg$Small(a)) $\vee$ ($\neg$Cube(a) $\wedge$ Small(a))]

2. ; Put an NNF form of 1 here.

3. ; Put a CNF form of 1 here.

4. $\neg$[(Cube(a) $\vee \neg$Small(a)) $\wedge$ ($\neg$Cube(a) $\vee$ Small(a))]

5. ; Put an NNF form of 4 here. Notice the relationship
   ; with 2.

6. ; Put an CNF form of 5 here. Notice the relationship
   ; with 3.

7. $\neg$(Cube(a) $\wedge$ Larger(a, b)) $\wedge$ Dodec(b)

8.

9.

10. $\neg$($\neg$Cube(a) $\wedge$ Tet(b))

11.

12.

13. $\neg\neg$Cube(a) $\vee$ Tet(b)
    ; Notice the difference between this sentence and
    ;10.

14.

15.

Sentenças de Ockham

1. $\exists x$ (Tet(x) $\wedge$ Large(x))

2. $\exists x \exists y$ (Larger(x, y) $\wedge \neg$Large(x))

3. $\forall x \forall y$ ((Dodec(x) $\wedge$ Dodec(y)) $\rightarrow$ x = y)

4. $\neg \forall y$ (Cube(y) $\rightarrow$ Small(y))

5. $\forall x$ (Large(x) $\leftrightarrow$ Tet(x))

6. $\forall x \forall y$ (Larger(x, y) $\rightarrow$ BackOf(x, y))

7. $\exists x \exists y$ (Cube(x) $\wedge$ Tet(y) $\wedge$ LeftOf(x, y) $\wedge$
   Smaller(x, y))

8. $\exists x \exists y$ (Small(x) $\wedge$ Large(y) $\wedge$
   $\forall z$ (Between(z, x, y) $\leftrightarrow$ Cube(z)))

9. $\forall x$ (Small(x) $\leftrightarrow \forall y$ (y # x $\rightarrow$ LeftOf(x, y)))

Sentenças de Padoa

1. $\forall x \forall y ((\neg \text{RightOf}(x, y) \wedge \neg \text{RightOf}(y, x)) \rightarrow x = y)$

2. $\forall x \forall y ((\text{Cube}(x) \wedge \text{Cube}(y)) \rightarrow \neg(\text{LeftOf}(x, y) \vee \text{LeftOf}(y, x)))$

3. $\exists x (\text{Tet}(x) \vee \text{Dodec}(x)) \wedge \forall x ((\text{Tet}(x) \vee \text{Dodec}(x)) \rightarrow \exists y (\text{BackOf}(x, y) \wedge \forall z (\text{BackOf}(x, z) \rightarrow z = y)))$

4. $\exists x (\text{Cube}(x) \wedge \text{Small}(x) \wedge \exists y \exists z \text{ Between}(x, y, z))$

Sentenças de Peano

1. $\exists x \exists y (\text{Tet}(x) \wedge \text{Larger}(x, y))$
   ; This sentence and 2 say the same thing in
   ; different ways.

2. $\exists x (\text{Tet}(x) \wedge \exists y \text{ Larger}(x, y))$

3. $\exists x \exists y (\text{Cube}(x) \wedge \text{Tet}(y) \wedge \text{Larger}(x, y))$
   ; This sentence and the next two say the
   ; same thing in different ways.

4. $\exists x (\text{Cube}(x) \wedge \exists y (\text{Tet}(y) \wedge \text{Larger}(x, y)))$

5. $\exists x \exists y (\text{Tet}(x) \wedge \text{Cube}(y) \wedge \text{Larger}(x, y))$

6. $\exists x (\text{Tet}(x) \wedge \exists y (\text{Cube}(y) \wedge \text{Larger}(x, y)))$

7. $\exists x \exists y (\text{Dodec}(x) \wedge \text{Tet}(y) \wedge \text{Larger}(x, y))$

8. $\exists x (\text{Dodec}(x) \wedge \exists y (\text{Tet}(y) \wedge \text{Larger}(x, y)))$

9. $\exists x \exists y \exists z \text{ Between}(x, y, z)$

10. $\exists x (\text{Cube}(x) \wedge \exists y \exists z \text{ Between}(x, y, z))$

11. $\exists x (\text{Tet}(x) \wedge \exists y \exists z \text{ Between}(x, y, z))$

12. $\exists x \exists y \exists z (\text{Tet}(x) \wedge \text{Cube}(y) \wedge \text{Between}(x, y, z))$

13. $\forall x \forall y (\text{Cube}(x) \rightarrow \text{Larger}(x, y))$
    ; This sentence and the next say the same thing
    ; in different ways.

14. $\forall x (\text{Cube}(x) \rightarrow \forall y \text{ Larger}(x, y))$

15. $\forall x \forall y ((\text{Cube}(x) \wedge \text{Dodec}(y)) \rightarrow \text{Larger}(x, y))$

16. $\forall x (\text{Cube}(x) \rightarrow \forall y (\text{Dodec}(y) \rightarrow \text{Larger}(x, y)))$

17. $\forall x ((\text{Cube}(x) \wedge \text{Medium}(x)) \rightarrow \forall y \text{ LeftOf}(y, x))$
    ; Be sure to understand the difference between
    ; this sentence and the next one.

18. $\forall x ((\text{Cube}(x) \wedge \text{Medium}(x)) \rightarrow \forall y (y \# x \rightarrow \text{LeftOf}(y, x)))$

19. $\forall x ((\text{Tet}(x) \wedge \text{Small}(x)) \rightarrow \forall y \text{ FrontOf}(x, y))$

20. $\forall x ((\text{Tet}(x) \wedge \text{Small}(x)) \rightarrow \forall y (y \# x \rightarrow \text{FrontOf}(x, y)))$

21. $\forall x ((\text{Tet}(x) \wedge \text{Small}(x)) \rightarrow \forall y \neg \text{FrontOf}(y, x))$

22. $\forall x \forall y (\text{BackOf}(x, y) \rightarrow \neg \text{FrontOf}(x, y))$

23. $\exists x \exists y (\text{BackOf}(x, y) \wedge \neg \text{FrontOf}(x, y))$

24. $\exists x (\text{Tet}(x) \wedge \exists y \exists z (\text{Dodec}(y) \wedge \text{Dodec}(z) \wedge \text{Between}(x, y, z)))$

25. $\exists x \exists y (\text{Tet}(x) \wedge \text{Tet}(y) \wedge \exists z (\text{Dodec}(z) \wedge \text{Between}(z, x, y)))$

26. $\forall x \forall y ((\text{Cube}(x) \wedge \text{Tet}(y)) \rightarrow (\text{FrontOf}(x, y) \vee \text{FrontOf}(y, x)))$

27. $\forall x \forall y ((\text{Cube}(x) \wedge \text{Tet}(y)) \rightarrow (\text{LeftOf}(x, y) \vee \text{LeftOf}(y, x)))$

28. $\forall x \forall y ((\text{Cube}(x) \wedge \text{Cube}(y)) \rightarrow \neg \text{SameRow}(x, y))$
    ; You may need to play the game to see why this
    ; has the truth value it does in Peano's World.
    ; Compare this with 29 and 30.

29. $\forall x \forall y [(\text{Dodec}(x) \wedge \text{Dodec}(y)) \rightarrow \text{SameSize}(x, y)]$

30. $\forall x \forall y [(\text{Tet}(x) \wedge \text{Tet}(y)) \rightarrow \neg \text{SameSize}(x, y)]$
    ; Be sure you understand why this comes out as
    ; it does in Peano's World.

Sentenças de Post

1. $\forall x (\text{Cube}(x) \vee \text{Tet}(x) \vee \text{Dodec}(x))$

2. $\forall x (\text{Small}(x) \vee \text{Medium}(x) \vee \text{Large}(x))$

3. $\forall x \forall y (\text{Larger}(x, y) \rightarrow \text{Large}(x))$

4. $\forall x \forall y (\text{Larger}(x, y) \rightarrow (\text{Large}(x) \vee \text{Small}(y)))$

5. $\neg \exists x \exists y (\text{Larger}(x, y) \wedge \text{Small}(x) \wedge \text{Small}(y))$

6. $\forall x \forall y \forall z ((\text{BackOf}(x, z) \wedge \text{Between}(y, x, z)) \rightarrow \text{BackOf}(x, y))$

Sentenças de Peirce

1. $\forall x\ Cube(x)$
2. $\forall x\ \neg Dodec(x)$
3. $\forall x\ \neg Small(x)$
4. $\neg\forall x\ Small(x)$
5. $\exists x\ Tet(x)$
6. $\exists x\ Dodec(x)$
7. $\exists x\ \neg Large(x)$
8. $\neg\exists y\ Large(y)$
9. $\exists x\ BackOf(x, x)$
10. $\forall x\ (Cube(x) \vee Tet(x))$
11. $\forall x\ (Cube(x) \wedge Tet(x))$
12. $\forall x\ (Large(x) \rightarrow Cube(x))$
13. $\forall x\ (Medium(x) \rightarrow Cube(x))$
14. $\forall x\ (Small(x) \rightarrow Tet(x))$
15. $\exists x\ Tet(x) \rightarrow \exists x\ Large(x)$
16. $\exists x\ (Tet(x) \wedge Large(x))$
17. $\neg\exists x\ (Dodec(x) \wedge Small(x))$
18. $\exists y\ (Cube(y) \wedge Small(y))$
19. $\exists x\ (Medium(x) \wedge Tet(x))$
20. $\forall x\ ((Small(x) \wedge Large(x)) \rightarrow Dodec(x))$
21. $\exists x\ Tet(x) \wedge \exists y\ Tet(y) \wedge \exists z\ Tet(z) \wedge \exists u\ Tet(u)$
22. $\forall x\ (Cube(x) \leftrightarrow \neg Tet(x))$
23. $\forall x\ (Cube(x) \leftrightarrow Large(x))$
24. $\forall x\ Cube(x) \leftrightarrow \forall x\ Large(x)$
25. $\forall x\ Cube(x) \leftrightarrow \forall y\ Cube(y)$
26. $\neg\forall x\ Cube(x) \leftrightarrow \forall x\ \neg Cube(x)$
27. $\neg\exists y\ Tet(y) \leftrightarrow \exists y\ \neg Tet(y)$
28. $\neg\forall x\ Cube(x) \leftrightarrow \exists x\ \neg Cube(x)$
29. $\neg\exists y\ Dodec(y) \leftrightarrow \forall v\ \neg Dodec(v)$
30. $\neg\exists x\ Dodec(x) \leftrightarrow \exists x\ \neg Dodec(x)$

Sentenças Ramsey

1. $\exists x\ \exists y\ (Tet(x) \wedge Tet(y) \wedge SameRow(x, y))$
   ; You only need one block to make this true.
2. $\exists x\ \exists y\ (Tet(x) \wedge Tet(y) \wedge Larger(x, y) \wedge BackOf(x, y))$
   ; How many blocks do you need to add to make this
   ; true?
3. $Cube(c) \wedge Cube(d) \wedge FrontOf(c, d)$
4. $Cube(c) \rightarrow Tet(a)$
   ; Name one of your earlier blocks a.
5. $Cube(d) \wedge Cube(e) \wedge \neg Larger(d, e)$
6. $\exists x\ (Cube(x) \wedge x \# d \wedge x \# e)$
7. $\exists x\ (Tet(x) \wedge Between(x, c, d))$
   ; Reposition things to make this true without new
   ; blocks.
8. $\exists x\ \exists y\ (Between(a, x, y) \wedge Larger(a, x) \wedge Smaller(a, y))$
9. $\exists x\ (Cube(x) \wedge ((x \# c \wedge x \# d) \vee Large(x)))$
10. $\exists x\ \exists y\ \exists z\ (Cube(x) \wedge Dodec(y) \wedge Dodec(z) \wedge Between(y, x, z) \wedge Larger(x, z) \wedge LeftOf(x, z))$
11. $\forall x\ \neg BackOf(x, d)$
12. $\forall x\ ((Cube(x) \wedge Large(x)) \rightarrow x = e)$
13. $\forall x\ \neg LeftOf(x, d)$
14. $\forall x\ (x \# e \rightarrow (FrontOf(x, e) \vee RightOf(x, e)))$
15. $\forall x\ ((Tet(x) \wedge Large(x)) \leftrightarrow x = b)$
16. $\forall x\ ((LeftOf(x, a) \wedge \neg Cube(x)) \rightarrow (Tet(x) \wedge Small(x)))$
17. $\forall x\ ((x \# a \wedge x \# b \wedge x \# d) \rightarrow Small(x))$
18. $\forall x\ \forall y\ ((Medium(x) \wedge Medium(y)) \rightarrow x = y)$
19. $\forall x\ \forall y\ ((Tet(x) \wedge Tet(y) \wedge x \# y) \rightarrow (x = a \vee y = a))$
20. $\forall x\ (Between(x, c, d) \rightarrow (Tet(x) \wedge Medium(x)))$

Sentenças de Robinson

1. $\exists x\ (\exists y\ (y = x \wedge \exists x\ (LeftOf(y, x) \wedge \exists y\ (y = x \wedge \exists x\ (LeftOf(y, x) \wedge \exists y\ (y = x \wedge \exists x\ (LeftOf(y, x) \wedge \exists y\ (y = x \wedge \exists x\ (LeftOf(y, x) \wedge \exists y\ (y = x \wedge \exists x\ (LeftOf(y, x)$
   ))))))))))))
   ;This sentence is interesting in that it uses only
   ;two variables to say something that seems like it
   ;would take more.

Sentenças de Russell

1. $\exists x\ (Cube(x) \wedge \forall y\ (Cube(y) \rightarrow x = y))$

2. $\exists x\ (Cube(x) \wedge \forall y\ (Cube(y) \rightarrow x = y) \wedge Small(x))$

3. $\exists x\ (Tet(x) \wedge \forall y\ (Tet(y) \rightarrow x = y) \wedge Large(x))$

4. $\exists x\ (Tet(x) \wedge \forall y\ (Tet(y) \rightarrow x = y) \wedge x = a)$

5. $\exists x\ ((Dodec(x) \wedge Small(x)) \wedge$
   $\forall y\ ((Dodec(y) \wedge Small(y)) \rightarrow x = y) \wedge$
   $BackOf(x, a))$

6. $\exists x\ ((Dodec(x) \wedge Medium(x)) \wedge$
   $\forall y\ ((Dodec(y) \wedge Medium(y)) \rightarrow x = y) \wedge$
   $FrontOf(x, a))$

7. $\exists x\ ((Dodec(x) \wedge Small(x)) \wedge$
   $\forall y\ ((Dodec(y) \wedge Small(y)) \rightarrow x = y) \wedge$
   $\exists z\ ((Dodec(z) \wedge Medium(z)) \wedge$
   $\forall u\ ((Dodec(u) \wedge Medium(u)) \rightarrow z = u) \wedge$
   $LeftOf(x, z)))$

Sentenças de Schönfinkel

1. $\neg Adjoins(y, y)$
   ; You can make this a sentence by adding a single
   ; quantifier (and variable).

2. $(Tet(w) \rightarrow Large(w))$
   ; Here, too.

3. $Tet(a) \rightarrow Large(w)$
   ; Since you can't add parentheses, you'll have to
   ; put the quantifier in the right place.

4. $Tet(w) \rightarrow Large(w)$
   ; How many quantifiers do you need here?
   ; Remember, you can't add parentheses.

5. $Large(x) \rightarrow \neg Larger(y, x)$

6. $\forall x\ y\ ((Cube(x) \wedge Cube(y)) \rightarrow \neg Larger(x, y))$

7. $\forall\ Cube(a)$
   ; Remember, you are only to add quantifiers or
   ; variables or both. No changing the "a". The
   ; resulting sentence is odd, but we'll explain
   ; what it means later.

8. $\forall x\ Tet(y) \rightarrow Small(x)$
   ; The quantifier here is not doing anything. It
   ; doesn't bind the y (wrong variable) and it
   ; doesn't bind the x (wrong scope).

9. $((Tet(x) \wedge Tet(y)) \rightarrow Between(z, x, y))$

10. $(Tet(x) \wedge Large(x)) \wedge LeftOf(x, y)$

Sentenças de Schröder

1. $Dodec(a) \wedge Cube(b)$

2. $FrontOf(b, a) \wedge LeftOf(d, b) \wedge BackOf(f, d) \wedge RightOf(a, f)$

3. $Tet(c) \wedge Tet(e) \wedge LeftOf(c, d)$

4. $\neg LeftOf(c, e) \wedge \neg LeftOf(e, c)$

5. $Between(d, b, c) \wedge \neg Between(d, c, a)$

6. $\neg Cube(b) \vee (Cube(d) \wedge BackOf(d, b))$

7. $(Small(c) \wedge FrontOf(c, b)) \vee Cube(d)$

8. $Larger(f, a) \wedge Larger(a, b)$

9. $\neg(Smaller(c, a) \vee Smaller(a, c))$

10. $Larger(d, b) \wedge Larger(f, d) \wedge$
    $\neg(Larger(e, d) \vee Larger(d, e))$

Sentenças de Sextus

1. $\neg(Home(carl) \wedge \neg Home(claire))$

2.
   ; Write the negation normal form of 1 here.
   ; Similarly for the other sentences that follow.

3. $\neg[Happy(max) \wedge$
   $(\neg Likes(carl, claire) \vee \neg Likes(claire, carl))]$

4.

5. $\neg\neg\neg[(Home(max) \vee Home(carl)) \wedge$
   $(Happy(max) \vee Happy(carl))]$

6.

Sentenças de Sheffer

1. $Tet(a) \wedge Small(a)$

2.

3. $Tet(a) \rightarrow Small(a)$

4.

5. $Tet(a) \leftrightarrow Small(a)$

6.

7. $(Cube(b) \wedge Cube(c)) \rightarrow (Small(b) \leftrightarrow Small(c))$
   ; This one is long. Tranlate the antecedant and
   ; consequent and then combine them using $\neg$ and $\vee$.

8.

Sentenças Sherlock

1. Tet(b) ↔ Tet(c)
2. Dodec(b) ↔ Dodec(c)
3. Cube(b) ↔ Cube(c)
4. Tet(a) ∧ ¬Tet(b)
5. FrontOf(a, b) → (FrontOf(b, c) ∨ FrontOf(c, b))
6. LeftOf(a, c) → ¬LeftOf(a, b)
7. BackOf(b, a) ↔ BackOf(c, b)

Sentenças de Skolem

1. Cube(c)
2. ∃x Cube(x)
3. Large(b)
4. ∃x Large(x)
5. Larger(e, c) ∧ Cube(e) ∧ Cube(c)
6. ∃x ∃y [Larger(y, x) ∧ Cube(y) ∧ Cube(x)]
7. ∀y [Dodec(y) ↔ y = d]
8. ∃x ∀y [Dodec(y) ↔ y = x]
9. ∀x (x = b ∨ x = c ∨ x = d ∨ x = e)
10. ∃u ∃v ∃w ∃z ∀x (x = u ∨ x = v ∨ x = w ∨ x = z)

Sentenças de Turing

1. ¬(Cube(a) ∧ Larger(a, b))
2. ; Write the negation normal form of 1 here.
   ; Similarly for the other sentences that follow.
3. ¬(Cube(a) ∨ ¬Larger(b, a))
4.
5. ¬(¬Cube(a) ∨ ¬Larger(a, b) ∨ a # b)
   ; The NNF of this sentence will have no negation
   ; signs in it.
6.
7. ¬(Tet(b) ∨ (Large(c) ∧ ¬Smaller(d, e)))
8.
9. Dodec(f) ∨ ¬(Tet(b) ∨ ¬Tet(f) ∨ ¬Dodec(f))
10.

Sentenças de Sócrates

1. ;========== Argument One ==========
2. Smaller(c, d) ;Premise
3. Smaller(d, f) ;Premise
4. Smaller(c, f) ;Conclusion
5. ;========== Argument Two ==========
6. FrontOf(c, d) ;Premise
7. d = e ;Premise
8. FrontOf(c, e) ;Conclusion
9. ;========== Argument Three ==========
10. FrontOf(c, b) ;Premise
11. SameRow(a, b) ;Premise
12. FrontOf(c, a) ;Conclusion
13. ;========== Argument Four ==========
14. Between(b, a, f) ;Premise
15. Between(b, f, a) ;Conclusion
16. ;========== Argument Five ==========
17. LeftOf(c, d) ;Premise
18. LeftOf(c, f) ;Premise
19. LeftOf(f, d) ;Conclusion
20. ;========== Argument Six ==========
21. SameCol(d, e) ;Premise
22. SameRow(e, d) ;Premise
23. e = d ;Conclusion
24. ;========== Argument Seven ==========
25. SameRow(a, f) ;Premise
26. Adjoins(a, b) ;Premise
27. SameRow(b, f) ;Conclusion
28. ;========== Argument Eight ==========
29. Adjoins(a, b) ;Premise
30. SameCol(b, c) ;Premise
31. FrontOf(c, b) ;Premise
32. FrontOf(c, a) ;Conclusion

Sentenças de Weiner

1. $(\text{Tet}(a) \wedge \text{Small}(a)) \vee (\text{Cube}(b) \wedge \neg\text{Cube}(b))$
2. $\text{Cube}(a) \wedge (\neg\text{Cube}(a) \vee \neg\text{Small}(a)) \wedge \text{Small}(a)$
3. $\text{Larger}(a, b) \wedge (\text{Larger}(a, a) \vee \neg\text{Larger}(a, b))$
4. $a \# b \wedge \neg\text{LeftOf}(a, b) \wedge \neg\text{RightOf}(a, b) \wedge \neg\text{FrontOf}(a, b) \wedge \neg\text{BackOf}(a, b)$
5. $\text{LeftOf}(a, b) \wedge \text{LeftOf}(b, c) \wedge \text{LeftOf}(c, d)$
6. $\text{Larger}(a, b) \wedge \text{Larger}(b, c) \wedge \text{Larger}(c, a)$
7. $\text{Between}(a, b, c) \wedge \text{LeftOf}(a, b) \wedge \text{LeftOf}(b, c)$
8. $\neg[(\text{Cube}(a) \vee \neg\text{Small}(a)) \wedge (\text{Cube}(a) \vee \neg\text{Medium}(a))] \wedge \text{Cube}(a)$
9. $\neg[\text{Dodec}(b) \wedge \text{Larger}(b, c)] \wedge [\text{Medium}(b) \vee \text{Tet}(b)]$
10. $\neg[\text{FrontOf}(a, b) \vee \neg\text{LeftOf}(b, c)] \wedge \text{Between}(b, c, a)$

Sentenças de Whitehead

1. $\exists x\, \exists y\, (x \# y)$
2. $\forall x\, \forall y\, \forall z\, (x = y \vee x = z \vee y = z)$
3. $\exists x\, \exists y\, (x \# y) \wedge \forall x\, \forall y\, \forall z\, (x = y \vee x = z \vee y = z)$
4. $\exists x\, \exists y\, (x \# y \wedge \forall z\, (z = x \vee z = y))$
5. $\exists x\, \exists y\, \exists z\, (x \# y \wedge y \# z)$
6. $\exists x\, \exists y\, \exists z\, (x \# y \wedge y \# z \wedge x \# z)$
7. $\exists x\, \exists y\, \exists z\, (x \# y \wedge y \# z \wedge x \# z \wedge \forall u\, (u = x \vee u = y \vee u = z))$
8. $\forall x\, (\text{Large}(x) \leftrightarrow x = a)$
9. $\exists y\, \forall x\, (\text{Large}(x) \leftrightarrow x = y)$
10. $\exists y\, \forall x\, (\text{Dodec}(x) \leftrightarrow x = y)$
11. $\exists y\, \forall x\, ((\text{Dodec}(x) \wedge \text{Medium}(x)) \leftrightarrow x = y)$
12. $\exists x\, \exists y\, (x \# y \wedge \text{Dodec}(x) \wedge \text{Dodec}(y))$
13. $\exists x\, (\text{Dodec}(x) \wedge \forall y\, (\text{Dodec}(y) \rightarrow x = y))$
14. $\exists x\, \exists y\, (x \# y \wedge \forall z\, (\text{Tet}(z) \leftrightarrow (z = x \vee z = y)))$

Sentenças de Zorn

1. $\forall x\, (x = a \vee x = b \vee x = c \vee x = d)$
2. $\exists x\, (x \# a \wedge x \# b \wedge x \# c \wedge x \# d \wedge x \# e)$
3. $\forall x\, (x = a \rightarrow x = d)$
4. $\exists x\, (\text{Between}(x, c, a) \wedge x \# b)$
5. $\forall x\, (\text{Between}(x, c, a) \rightarrow x = b)$
6. $\forall x\, ((\text{Tet}(x) \wedge \text{Medium}(x)) \rightarrow x = e)$
7. $\forall x\, (x = e \rightarrow (\text{Tet}(x) \wedge \text{Medium}(x)))$
8. $\forall x\, ((\text{Tet}(x) \wedge \text{Small}(x)) \leftrightarrow x = b)$
9. $\exists y\, (y \# e \wedge \text{SameRow}(y, e))$

Sentenças de Wittgenstein

1. Large(d)
   ; This sentence claims that the block named d is
   ; large.

2. Large(c)

3. Large(f)

4. Medium(c)

5. Medium(b)

6. Medium(f)

7. Larger(c, b)
   ; This claims that block c is larger than block b.

8. Larger(d, e)

9. Larger(d, f)

10. Larger(a, e)

11. BackOf(b, e)
    ; This claims that block b is back of block e.

12. BackOf(b, f)

13. BackOf(f, a)

14. LeftOf(b, a)

15. LeftOf(a, b)

16. LeftOf(a, a)

17. RightOf(a, b)

18. RightOf(b, d)

19. SameSize(d, f)

20. SameSize(b, c)

21. SameShape(b, c)

22. SameShape(a, b)

23. SameShape(d, d)

24. SameRow(a, d)

25. SameRow(d, e)

26. SameRow(a, a)

27. SameCol(a, d)

28. SameCol(d, e)

29. SameCol(a, c)

30. Adjoins(a, b)

31. Adjoins(b, c)

32. Adjoins(d, e)

33. Adjoins(b, b)

34. Between(d, e, f)
    ; Notice that this claims that block d is between
    ; blocks e and f, not that e is between d and f!
    ; Sentence 35 says that e is between d and f. Make
    ; sure you understand the difference.

35. Between(e, d, f)

36. Between(c, a, f)

37. Between(c, d, b)

38. Between(c, b, e)

39. Between(c, c, c)

40. Between(c, a, e)

Cabeçalhos de Prova
das Seções Experimente

Provas - Experimente

Ana Con 1

1.	FrontOf(b, d)	
2.	SameRow(b, c)	
3.	SameRow(a, c)	
4.	Cube(b)	
5.	SameRow(e, d)	
6.	Cube(c)	
7.	Medium(c)	
8.	Large(d)	
9.	SameSize(a, d)	
10.	SameShape(c, b)	**Ana Con**
11.	SameRow(b, a)	**Ana Con**
12.	BackOf(e, c)	**Ana Con**
13.	SameSize(d, a)	**Rule?**
14.	Larger(a, c)	**Rule?**
15.	SameCol(b, b)	**Rule?**

Goals

- SameShape(c, b)
- SameRow(b, a)
- BackOf(e, c)
- SameSize(d, a)
- Larger(a, c)
- SameCol(b, b)

Conditional 1

1. $(A \vee B) \rightarrow C$

Goals

- $A \rightarrow C$

Conditional 2

1. Tet(a) → Small(a)
2. Tet(a) → Larger(b, a)
3. (Small(a) ∧ Larger(b, a)) → LeftOf(b, a)
   4. Tet(a)
   5. → Elim 4,1
   6. → Elim 4,2
   7. ∧ Intro 5,6
   8. → Elim 7,3
9. → Intro 4-8
10. Rule?

Goals

Tet(a) → LeftOf(b, a)

Conditional 3

1.

Goals

(P → Q) ↔ (¬Q → ¬P)

Conjunction 1

1. Tet(a) ∧ Tet(b) ∧ Tet(c) ∧ (Small(a) ∨ Small(b))

Goals

Tet(a)
Small(a) ∨ Small(b)
Tet(c)

Conjunction 2

1. Dodec(c) ∧ ¬Large(c)
2. Tet(d) ∧ Medium(d)
3. Likes(c, d) ∨ Likes(d, c)

Goals

Medium(d) ∧ ¬Large(c)
Tet(d) ∧ Dodec(c) ∧ (Likes(c, d) ∨ Likes(d, c))

Conjunction 3

1. Tet(a) ∧ Tet(b) ∧ (Tet(c) ∧ (Small(a) ∨ Small(b)))

Goals

- Tet(c) ∧ Tet(a)
- (Small(a) ∨ Small(b)) ∧ Tet(b)

Conjunction 4

1. Dodec(a) ∧ Dodec(b)
2. Small(a) ∧ Small(b)
3. ∧ **Elim** 1
4. ∧ **Elim** 2
5. ∧ **Intro** 4,3
6. ∧ **Intro** 3,4

Disjunction 1

1. (Cube(c) ∧ Large(c)) ∨ Medium(c)

Goals

- Medium(c) ∨ Large(c)

Disjunction 2

1. Cube(b)
2. Small(b) ∨ Large(b)
3. Small(b)
4. ∧ **Intro** 1,3
5. ∨ **Intro** 4
6. Large(b)
7. ∧ **Intro** 1,6
8. (Cube(b) ∧ Small(b)) ∨ (Cube(b) ∧ Large(b)) ∨ **Intro** 7
9. ∨ **Elim** 2,3-5,6-8

Goals

- (Cube(b) ∧ Small(b)) ∨ (Cube(b) ∧ Large(b))

Elimination Exercise 2

1. $\forall x\ (Cube(x) \rightarrow Small(x))$
2. $\exists x\ Cube(x)$
3. $\exists x\ Cube(x) \rightarrow Cube(c)$
4. $Small(c) \rightarrow \exists x\ Small(x)$
5. $\neg(Cube(c) \rightarrow Small(c)) \rightarrow \exists x\ \neg(Cube(x) \rightarrow Small(x))$
6. $\neg\forall x\ (Cube(x) \rightarrow Small(x)) \leftrightarrow \exists x\ \neg(Cube(x) \rightarrow Small(x))$
7. $\exists x\ Small(x)$ **Taut Con** 6,5,4,3,2,1

~~Rule?~~

Goals

- $\exists x\ Small(x)$

Identity 1

1. $SameRow(a, a)$
2. $b = a$

Goals

- $SameRow(b, a)$

Negation 1

1. A

Goals

- $\neg\neg A$

Negation 2

1. $Cube(a)$
2. $\neg Cube(a)$
3. $a = b$
4. $\neg Cube(b)$
5. $Dodec(a)$
6. $SameSize(a, b)$
7. $Larger(a, b)$
8. $SameRow(b, c)$
9. $\neg SameRow(c, c)$
10. $b \neq b$
11. $\neg Cube(a) \vee \neg Larger(a, b)$
12. $\bot$ **Rule?** 9
13. $\bot$ **Rule?** 10
14. $\bot$ **Rule?** 6,7
15. $\bot$ **Rule?** 1,2
16. $\bot$ **Rule?** 1,3,4
17. $\bot$ **Rule?** 5,1
18. $\bot$ **Rule?** 1,7,11

Negation 3

1. $P \vee Q$
2. $\neg Q$

Goals

- P

Negation 4

1. $\neg(\neg Dodec(b) \vee \neg Dodec(c))$
   2. $\neg Dodec(b)$
   3. $\neg Dodec(b) \vee \neg Dodec(c)$ — $\vee$ Intro 2
   4. — $\perp$ Intro 3,1
5. — $\neg$ Intro 2-4
6. — $\neg$ Elim 5
   7. $\neg Dodec(c)$
   8. $\neg Dodec(b) \vee \neg Dodec(c)$ — $\vee$ Intro 7
   9. — $\perp$ Intro 8,1
10. — $\neg$ Intro 7-9
11. — $\neg$ Elim 10
12. — $\wedge$ Intro 6,11

Goals

- $Dodec(b) \wedge Dodec(c)$

Proof 2.19

1. Smaller(a, b)
2. Smaller(b, c)
3. Smaller(a, c) — ✔ Ana Con 1,2 ~~Rule?~~

Goals

- Smaller(a, c) — ✔ 3

Strategy 1

1. $\neg P \vee \neg Q$

Goals

- $\neg(P \wedge Q)$

Taut Con 1

1. Home(max) ∨ ¬Happy(carl)
2. Happy(carl) ∨ Hungry(carl)
3. Hungry(carl) ∨ Hungry(pris)
4. Home(max) ∨ ¬Hungry(pris)

Goals

- Home(max) ∨ Hungry(carl)
- Hungry(carl) ∨ (Home(max) ∧ Hungry(pris))

Taut Con 2

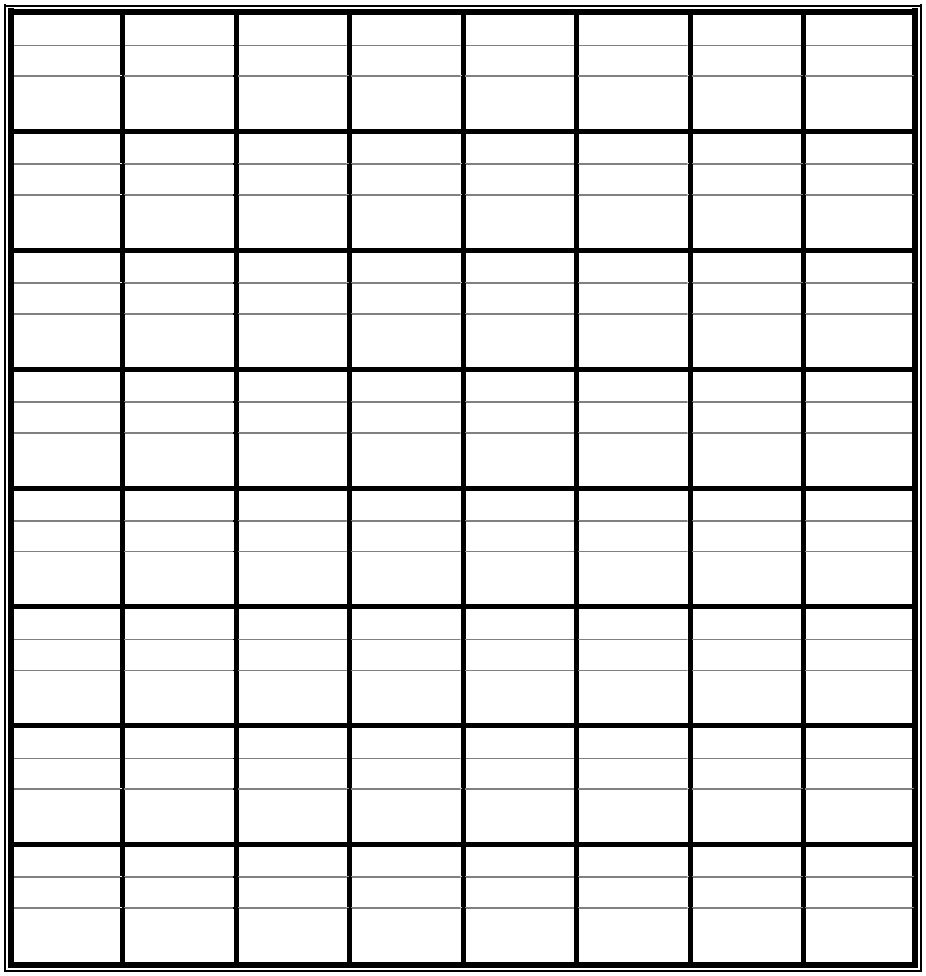
1. LeftOf(a, b) ∨ (Dodec(a) ∧ Small(a))
2. ¬LeftOf(a, b) ∨ (Cube(b) ∧ ¬Large(b))
3. ¬Small(a)
4. Small(c)
5. SameSize(a, b) ∨ SameCol(a, b)
6. b = c ∨ ¬(Cube(c) ∨ ¬Dodec(c))
7. LeftOf(a, b) **Rule?** 1,3
8. SameSize(a, b) **Rule?** 5,7
9. Cube(b) ∧ ¬Large(b) **Rule?** 7,2
10. ¬Large(b) **Rule?** 9
11. ¬Small(b) **Rule?** 3,8
12. Medium(b) **Rule?** 11,10
13. ¬(Cube(c) ∨ ¬Dodec(c)) **Rule?** 11,4,6
14. ¬Cube(c) ∧ ¬¬Dodec(c) **Rule?** 13
15. Dodec(c) **Rule?** 14
16. Medium(b) ∧ Dodec(c) **Rule?** 15,12

Rule?

Goals

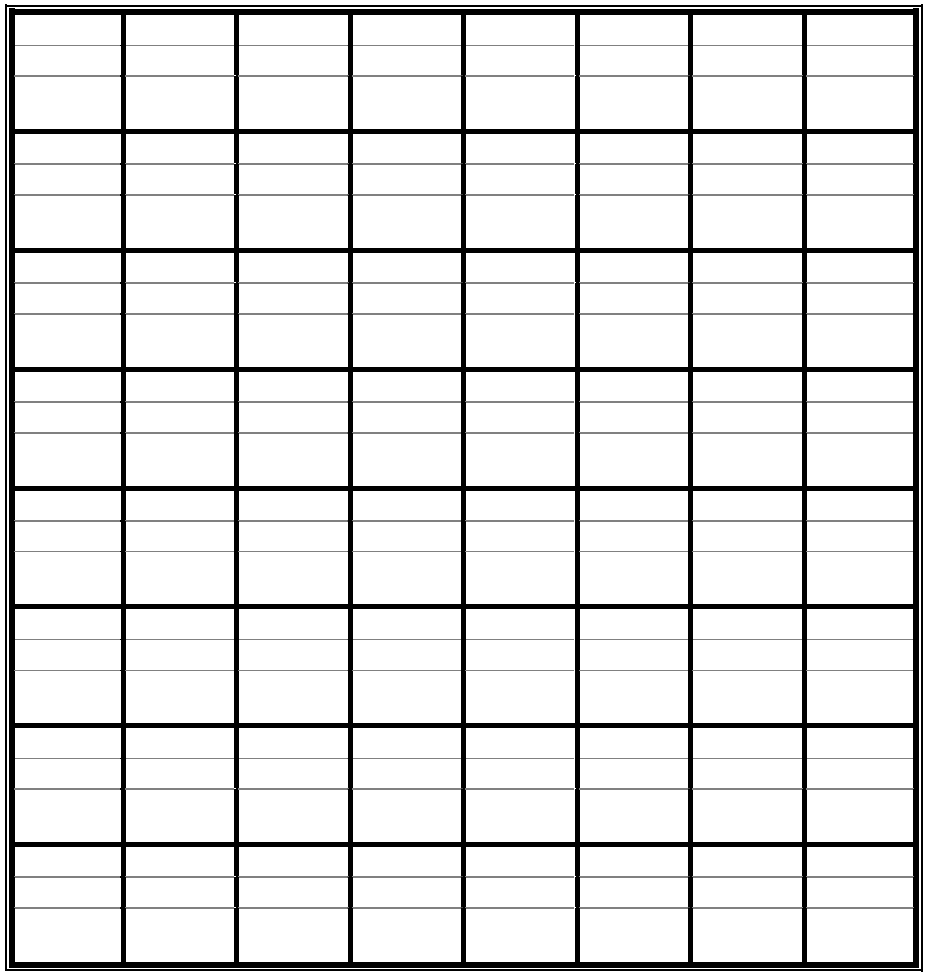
- Medium(b) ∧ Dodec(c)

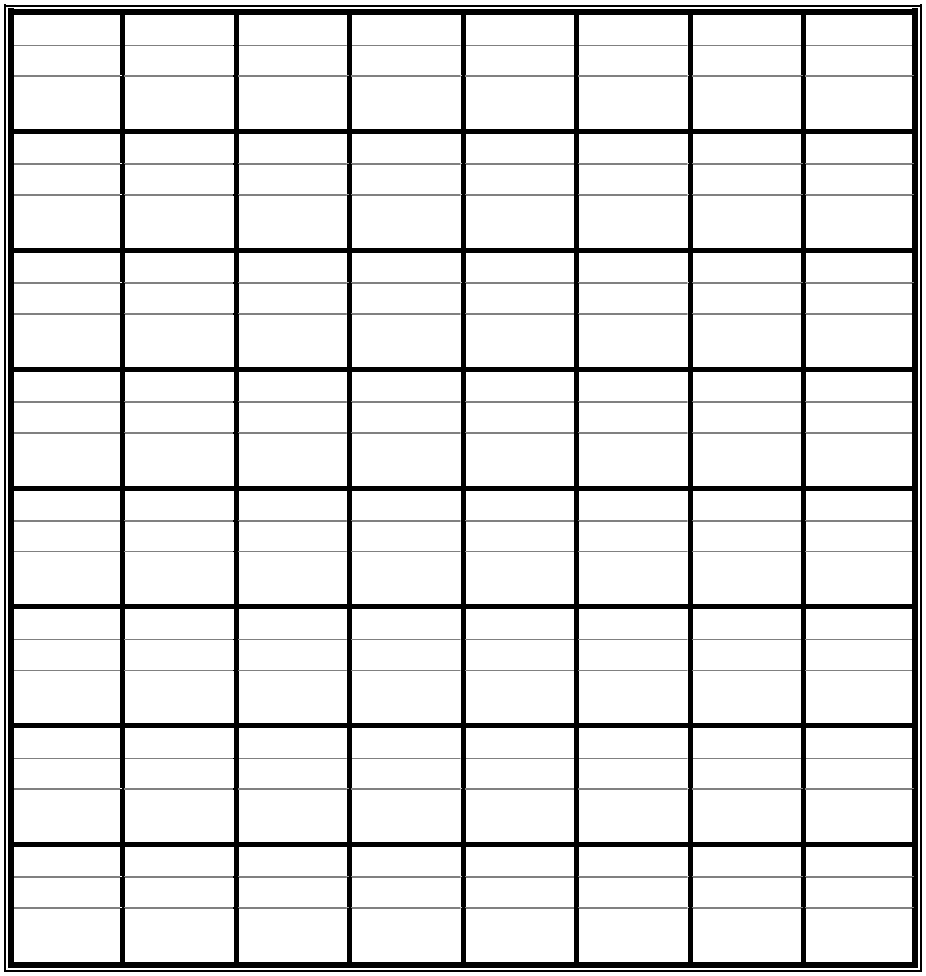
Diagramas para Desenho de Mundos



Exercício:_____.

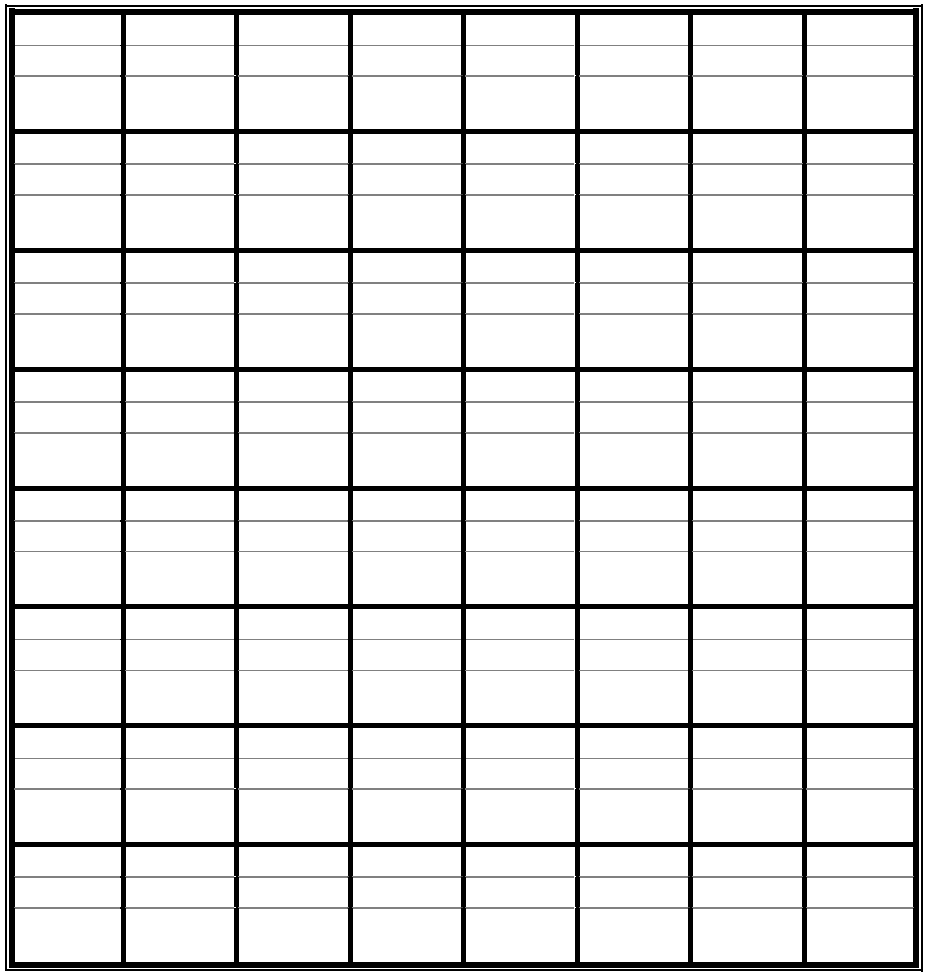
Exercício:_____.

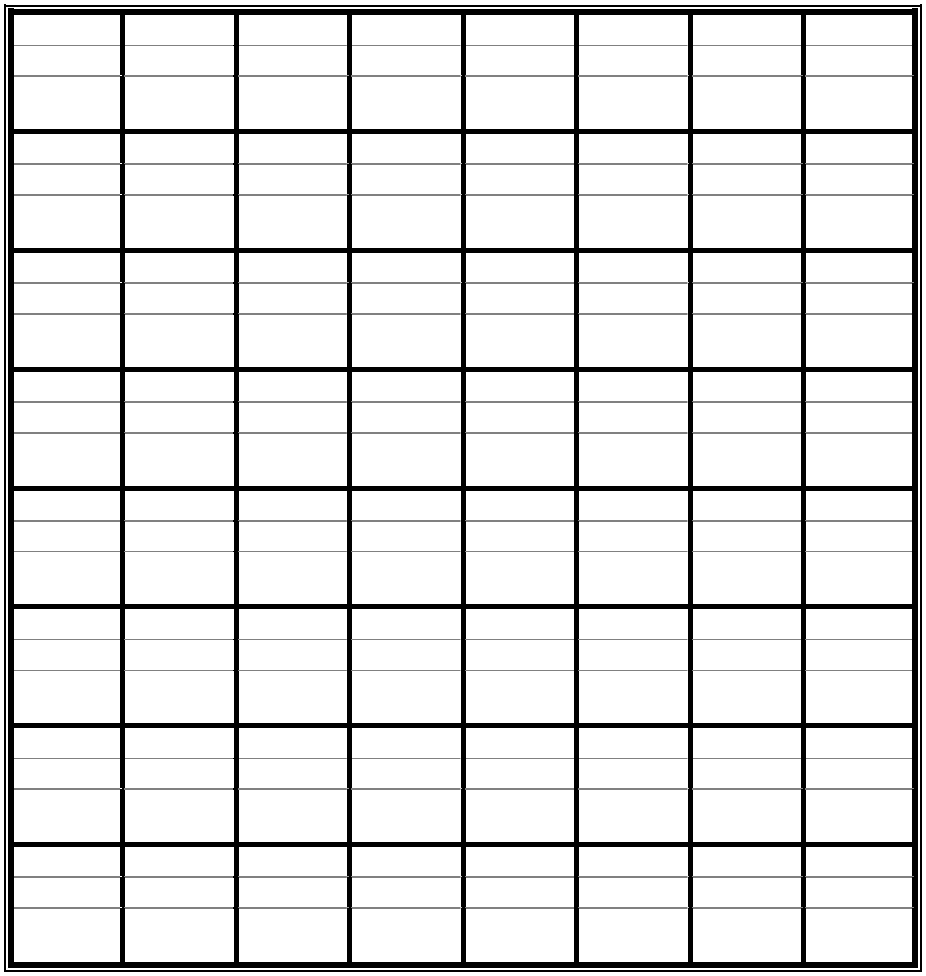




Exercício:_____.

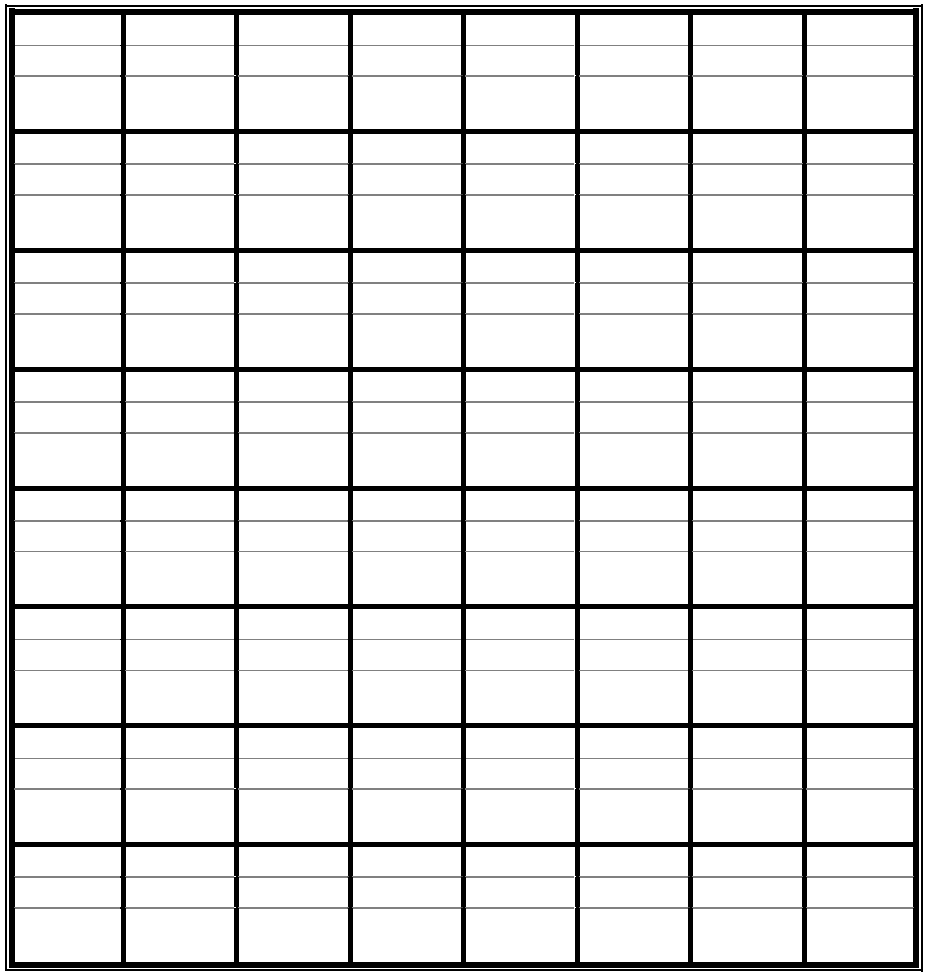
Exercício:_____.

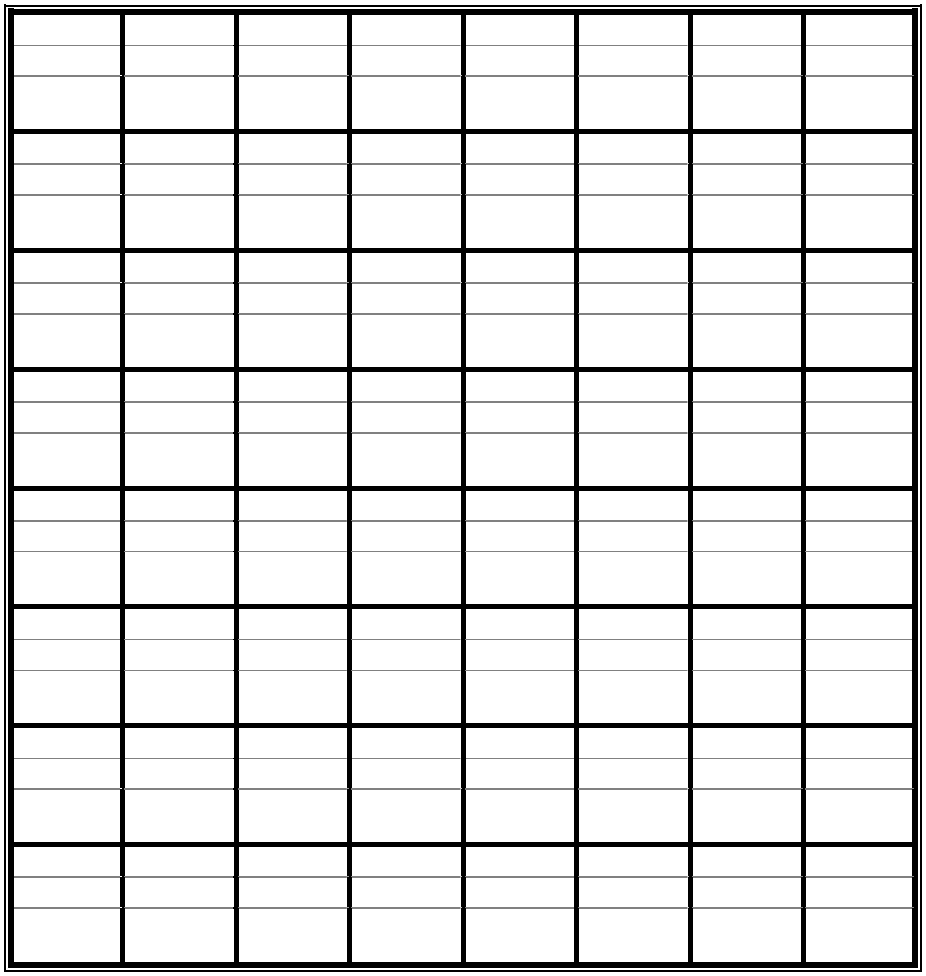




Exercício:_____.

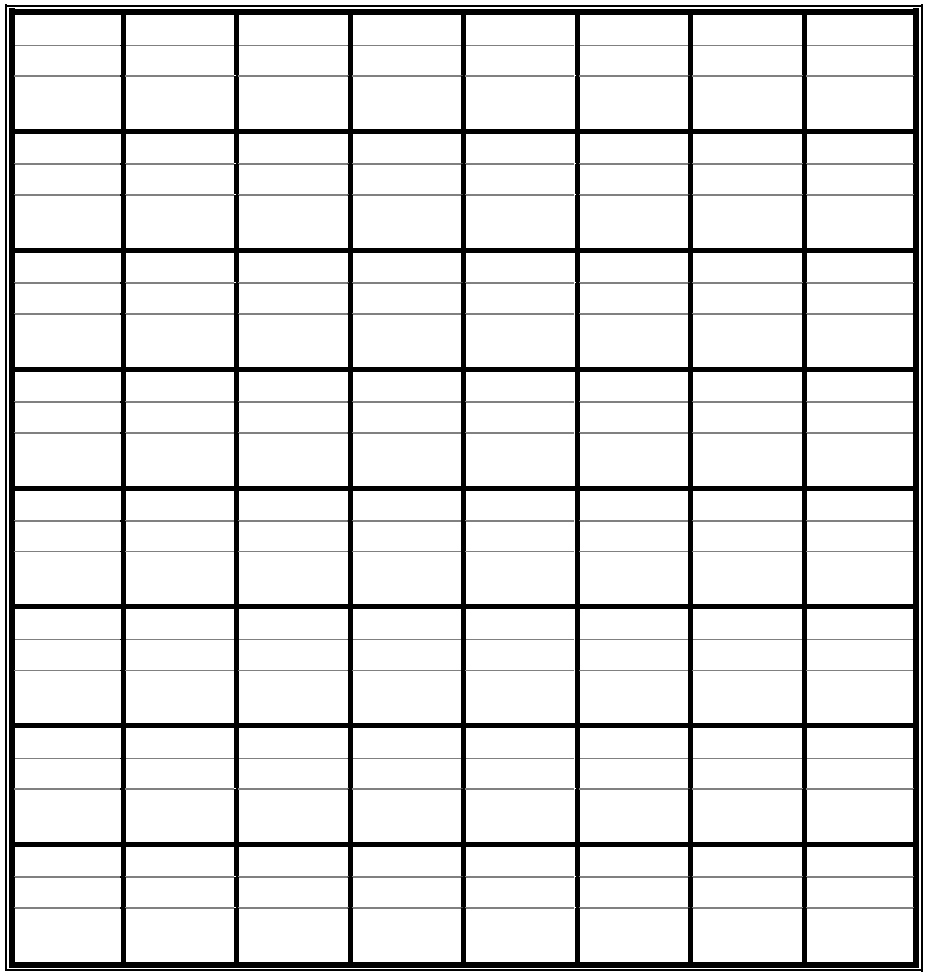
Exercício:_____.

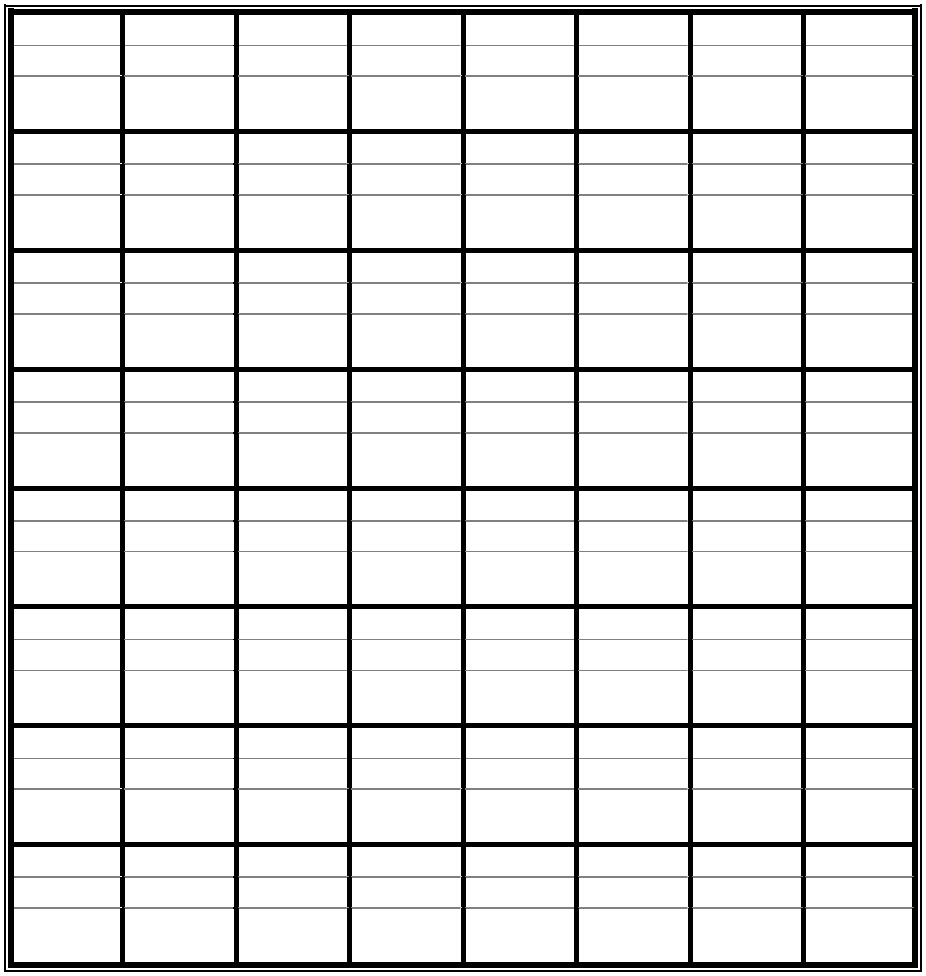




Exercício:_____.

Exercício:_____.





Exercício:_____.

Exercício:_____.