Introdução à Lógica Matemática para Alunos de Computação

Jefferson O. Andrade

Versão: 2022-08-03

Lista de Figuras

Lista de Tabelas

Lista de Códigos Fonte

Sumário

Prefácio

Propósito e Objetivos

Este livro é uma primeira introdução à lógica formal voltada principalmente à resolução de problemas. Ao longo dos anos ministrando a disciplina de Lógica em cursos da área de informática, tais como Sistemas de Informação, Análise e Desenvolvimento de Sistemas, e Administração com Ênfase em Processamento de Dados, tenho percebido que a forma tradicional de abordar a lógica que é usada em cursos mais técnicos tais como Ciência da Computação, e Engenharia da Computação tem pouca receptividade nos alunos dos cursos da área de informática, i.e., dos cursos que trabalham a computação mais como meio do que como fim. Para os alunos dos cursos em que a computação é a área fim, faz sentido tratar de questões tais como demonstração do completude da lógica proposicional, ou algoritmos de unificação, ou provas automáticas de teoremas. Entretanto, para alunos dos cursos em que a computação é meio, estas questões tem muito pouca relevância. Para estes, a lógica deve ser tratada como ferramenta para a resolução de problemas.

Assim, meu propósito ao iniciar a escrita deste livro foi preencher um vazio que, na minha opinião, existe na bibliografia de livros didáticos sobre lógica. Por um lado, embora exista uma boa quantidade de títulos sobre "raciocínio lógico" (principalmente para concursos), estes livros, de modo geral, não apresentam o rigor matemático que as aplicações de lógica em computação costumam exigir (sim, mesmo como ferramenta a lógica pede rigor), podendo ser classificados como livros sobre lógica informal. Por outro lado, os livros voltados para o público de nível superior, que apresentam um grau aceitável de rigor matemático, por via de regra, tratam o estudo da lógica como um fim em si mesmo, se atendo principalmente em provas de propriedades sobre os sistemas dedutivos apresentados, e sobre métodos e algoritmos de provas automáticas de teoremas. É evidente que estes temas são importantes no estudo da lógica (eu mesmo sou fascinado pro eles), mas não são muito relevantes ao estudante que deseja utilizar a lógica apenas como ferramenta.

Se, após ler este livro e praticar os exercícios indicados, o estudante tiver atingido um nível em que ele compreenda que a lógica não apenas é um dos fundamentos da computação, mas também que a lógica pode ser utilizada diretamente para resolver uma enorme quantidade de problemas, e este estudante tiver confiança em sua proficiência na resolução de alguns destes problemas utilizando os métodos indicados neste livro, então o objetivo da existência deste livro terá sido atingido e as muitas horas de trabalho gastas com escrita, preparação de exercícios, revisão, correção e formatação terão sido recompensadas.

Um objetivo secundário ao elaborar este livro, foi redigi-lo de tal modo que ele fosse o mais autocontido possível, para minimizar a necessidade de procurar material complementar em outras fontes. O objetivo não é, de forma alguma, sugerir que este livro esgota o tema ou desencorajar os estudantes a procurar outros materiais. Estudar o mesmo tema em mais de uma fonte, ter mais de um ponto de vista e mais de uma forma de explicar o mesmo assunto sempre é benéfico e certamente irá ajudar a entender melhor os tópicos estudados. A motivação para tornar o material autocontido é propiciar um fluxo de estudo com o mínimo de interrupções possível. De modo que o estudante possa "atacar" um determinado tópico do começo ao fim, e depois disso, caso queira, poderá procurar outras fontes sobre o mesmo tópico.

Abordagem

Embora o propósito e o objetivo deste livro estivessem claros para mim desde o início, a forma de atingir este objetivo não estava muito clara – e talvez ainda não esteja, ou talvez sempre vá estar meio nebulosa e precisando ser redefinida. Mas não dá pra ficar parado sem fazer nada esperando perfeição, então considerei algumas possibilidades. Uma era a de utilizar uma abordagem totalmente baseada em problemas, onde o problema fosse apresentado e em seguida a solução utilizando lógica formal fosse apresentada. O problema dessa abordagem, ao meu ver, é que uma vez que a lógica matemática exige formalismo o aluno seria apresentado a uma solução em uma linguagem que ele ainda não conhece e haveria pouca chance de que ele fosse conseguir entender a solução apresentada. Além disso, depois de pensar sobre o assunto por um tempo, não me pareceu muito humano jogar os alunos na vastidão da lógica formal, para que eles achem seus próprios caminhos, sem um primeiro curso de "sobrevivência na selva".

A outra possibilidade que considerei, e que foi aquela pela qual acabei me decidindo, foi a de dividir o tema da lógica matemática em "blocos aplicáveis" (teoria dos conjuntos, cálculo proposicional, lógica de predicados) e apresentar a fundamentação teórica de cada um desses temas, seguido imediatamente pelo estudo de uma das aplicações desses tópicos. Qualquer um dos tópicos da lógica matemática abordados neste livro têm múltiplas aplicações práticas, então a escolha de qual aplicação prática adotar para cada tópico também é um problema a se considerar. O critério que adotei para a escolha das aplicações a estudar foi o seguinte: uma vez que o livro é voltado para alunos de curso na área de informática, escolhi aplicações mais próximas da atividade de programação de computadores. Por exemplo, para o cálculo de predicados a aplicação escolhida poderia ter sido o estudo de circuitos lógicos, mas essa aplicação está mais próxima do contexto do engenheiro de computação dou do engenheiro eletrônico do que do bacharel em sistemas de informação ou do engenheiro de software, por exemplo. Assim, para o caso do cálculo proposicional, optei pelo estudo de prova de correção de algoritmos, que é uma atividade na qual qualquer profissional de informática que desenvolva, ou supervisione o desenvolvimento de sistemas deveria ter proficiência.

Público Alvo

Este livro é voltado principalmente para os estudantes dos períodos iniciais dos cursos de informática. O material não assume nenhuma experiência prévia com lógica formal, nem com programação. Entretanto, os alunos que já tem alguma experiência com programação provavelmente terão mais facilidade em trabalhar os capítulos de aplicações que lidam com as linguagens Python e Prolog.

A maioria dos estudantes de informática tem algum contato com linguagens de programação imperativas e orientadas a objetos em algum ponto dos seus cursos de graduação. A linguagem Python é uma linguagem orientada a objetos pura, i.e., todos os valores na linguagem são tratados como objetos, e provavelmente a maioria dos alunos dos cursos de informática já tiveram, ou terão, algum contato com esta linguagem. Já a linguagem Prolog, por outro lado, é uma linguagem que pertence ao paradigma de programação lógica, que não costuma ser visto na maioria dos cursos de informática, ou é visto de modo superficial. Assim, mesmo que o estudante já tenha tido algum contato com programação, é razoável assumir que ele, provavelmente, não terá visto Prolog.

Com relação ao conhecimento prévio de matemática, o que se espera é que o estudante tenha visto a matemática básica do ensino médio. Principalmente no que se relaciona à aritmética e resolução de equações moderadamente simples. Sem esse conhecimento, será difícil acompanhar alguns dos exemplos, sobretudo quando falarmos sobre conjuntos, relações e funções.

Visão Geral

O conteúdo deste livro está dividido em duas partes:

(I) Conceitos Preliminares

(II) Lógica Matemática

Na parte destinada aos conceitos preliminares temos um capítulo voltado par contar um pouco da história e da evolução da lógica enquanto disciplina sistemática de estudo, um capítulo para tratar dos conceitos de conjuntos, relações e funções, e um capítulo que apresenta a forma como a linguagem Python lida com conjuntos e alguns estudos de casos sobre resolução de problemas de programação usando conjuntos. Em algumas grades curriculares, há disciplinas de matemática discreta que já abrangem o tópico de conjuntos, muitas vezes no contexto da análise combinatória. Se este for o caso o professor da disciplina pode optar por omitir a parte de conceitos preliminares e iniciar o curso diretamente na parte de lógica matemática sem prejuízo para o curso.

A parte destinada à lógica matemática propriamente dita inicia com um capítulo apresentando os conceitos fundamentais da lógica formal. Conceitos como dedução e indução, verdade e validade, raciocínio lógico e a diferença entre lógica formal e lógica informal são discutidos neste capítulo inicial.

Em seguida temos o capítulo voltado para a lógica proposicional, que descreve os conceitos de proposição e conectivos lógicos, valoração lógica, tradução da linguagem natural para a forma simbólica e o sistema de prova por dedução natural, e abordamos brevemente a demonstração de argumentos formais.

O capítulo seguinte trata sobre demonstração de correção de algoritmos, utilizando Lógica de Hoare. A cobertura dada ao tópico de demonstração de correção não é completa, nem tão pouco aprofundada o suficiente para aplicação em sistemas de grande porte. Este tipo de aplicação exigiria o uso de assistentes de prova e/ou de provadores automáticos de teoremas, que estão fora do escopo de uma abordagem introdutória como a deste livro. Também não abordamos

lógicas mais sofisticadas como Lógica de Separação, por exemplo, pelo mesmo motivo. Ao invés disso, focamos em uma lógica consideravelmente similar à lógica proposicional e em demonstrações de algoritmos relativamente simples. O propósito deste capítulo é o de mostrar como é possível **provar** a correção de algoritmos através da lógica, em contraste com meramente *testar* os algoritmos.

Após o capítulo sobre demonstração de correção de algoritmos, temos um capítulo sobre lógica de predicados de primeira ordem, ou lógica de primeira ordem, de modo mais resumido. Assim como no capítulo sobre lógica de predicados, neste capítulo temos as noções mais teóricas, descrevendo o sistema lógico, os conceitos fundamentais de predicados, variáveis, funções lógicas, quantificadores, interpretações, etc., e ao final do capítulo falamos sobre o sistemas de provas por dedução natural para a lógica de primeira ordem, e sobre argumentos formais.

O último e mais "denso" capítulo do livro trata sobre modelagem de problemas usando lógica e da resolução destes problemas através de programação lógica. Se você tem alguma experiência na área de informática, já ouviu a expressão "lógica de programação". Não confunda "lógica de programação" com "programação lógica".

"lógica de programação" geralmente significa a solução de problemas através do desenvolvimento de algoritmos e a implementação destes algoritmos em alguma linguagem de programação.

"programação lógica" é um *paradigma de programação* baseado na lógica formal. Um programa escrito em uma linguagem de programação lógica é formado por um conjunto de sentenças em forma lógica, expressando fatos e regras sobre algum *domínio de problema*.

Se as expressões "paradigma de programação" e "domínio de problema" não estão muito claras para você, não se preocupe, iremos falar mais e melhor delas no último capítulo.

Agradecimentos

Primeiramente, preciso agradecer à minha família. Agradeço a Karin, a Anyssa, ao Jefferson e à Samiya, pela paciência que tiveram e continuam tendo comigo, por não reclamarem (muito) de todo o tempo que deveria ser livre, mas que eu gasto escrevendo, revisando e aperfeiçoando este livro.

Este livro tem estado em constante desenvolvimento, pelo menos, durante os últimos 3 anos. É claro que ele teve, tem e terá muitos erros. Ao longo do tempo em que eu tenho utilizado este livro como notas de aula, e depois como livro texto, vários alunos identificaram incorreções e, gentilmente, me indicaram essas incorreções para que eu pudesse corrigi-las. A estes alunos o meu muito obrigado.

Jefferson O. Andrade
Vitória, 31 de outubro de 2016.

Capítulo 1

Introdução

Sumário

1.1 O Que é Lógica?

A lógica, é uma ciência formal fortemente ligada à filosofia e à matemática. A lógica é o ramo da filosofia que estuda as regras do pensar, ou do pensar corretamente. A aprendizagem da lógica não constitui um fim em si. Ela é uma ferramenta para garantir que nosso pensamento proceda corretamente a fim de chegar a conhecimentos válidos. Podemos, então, dizer que a lógica trata dos argumentos, isto é, das conclusões a que chegamos através da apresentação de evidências que as sustentam. Um argumento lógico é estruturado de forma a dar razões para acreditar em uma certa conclusão. Aqui está um exemplo deste tipo de argumento:

1. Está chovendo muito.
2. Se estiver chovendo e você não tiver um guarda-chuva, você vai ficar encharcado.

∴ Se você não quer ficar encharcado, você deveria ter um guarda-chuva.

Os três pontos na terceira linha do argumento (∴) significam "portanto", e eles indicam que a frase final é a conclusão do argumento. As outras frases são *premissas* do argumento. Se você aceitar que as premissas são verdadeiras, o argumento deveria lhe fornecer uma razão para aceitar que a conclusão também é verdadeira.

Aristóteles é considerado o fundador e principal organizador da lógica clássica como ciência.[1] Ele divide a lógica em *formal* e *material*.

[1] Houve outros filósofos que trataram de lógica, mas Aristóteles foi o primeiro a tentar sistematizar a lógica como uma ciência e a estudar a dedução lógica à partir do formato dos argumentos.

Lógica Formal Também chamada de *Lógica Simbólica*, trata da estrutura do raciocínio. A lógica formal lida com as relações entre conceitos e fornece meios de compor provas de declarações. Na lógica formal os conceitos são rigorosamente definidos, e as orações são transformadas em notações simbólicas precisas, compactas e não ambíguas.

Lógica Material Trata da *aplicação* das operações do pensamento. Neste caso, a lógica é a própria metodologia de cada ciência. Logo, é somente na lógica material que se pode falar de *verdade*:[2] o argumento é válido quando as premissas são verdadeiras e se relacionam adequadamente à conclusão.

Há várias outras formas de se classificar e organizar o estudo de lógica e veremos mais sobre os tipos de lógica no Capítulo 4, mas por hora a classificação acima é suficiente pois neste curso estaremos voltados principalmente para o estudo da lógica formal e mais especificamente para o estudo da lógica matemática.

1.2 História da Lógica

A lógica formal se desenvolveu na antiguidade na China, na Índia e na Grécia. A lógica grega, principalmente a lógica aristotélica, encontrou grande aceitação e aplicação na filosofia, nas ciências e na matemática.

A lógica de Aristóteles teve seu desenvolvimento aprofundado posteriormente por filósofos cristãos e islâmicos na Idade Média, atingindo o seu ponto auto em meados do século XIV. O período compreendido entre o século XIV e o início do século XIX foi, em sua maior parte, de declínio e abandono da lógica.

A lógica formal foi revivida em meados do século XIX, o início de um período revolucionário quando este assunto se desenvolveu em uma disciplina rigorosa e produziu os métodos de prova utilizados na matemática. O desenvolvimento da lógica matemática simbólica foi o maior desenvolvimento nos mais de dois mil anos de história da lógica e é, possivelmente, um dos mais memoráveis eventos na história intelectual humana.

O progresso na lógica matemática na primeiras décadas do século XX, principalmente vindo dos trabalhos de Gödel e Tarski, teve um impacto significante na filosofia analítica e na lógica filosófica, principalmente a partir dos anos de 1950, em assuntos como lógicas modais, lógicas temporais, lógicas deonticas, lógicas de relevância e lógicas paraconsistentes.

1.2.1 Pré-história da Lógica

O raciocínio válido tem sido empregado em todos os períodos da história humana. Entretanto, a lógica estuda os *princípios*, ou regras, do raciocínio válido, das inferências e das demonstrações. É provavel que a necessidade de demonstrar uma conclusão tenha aparecido inicialmente ligada à geometria, que originalmente significava o mesmo que "medição de terra". Em particular os egipcios descobriram empiricamente alguns resultados geométricos como a fórmula do volume de uma piramide truncada.

Outra origem possível pode ser observada na Babilônia. O *Manual de Diagnóstico* médico de Esagil-kin-aplin, do século XI BCE, era baseado em um sistema lógico de axiomas e suposições. Além disso, astrônomos babilônicos dos séculos VIII e VII BCE empregaram uma lógica interna dentro de seus sistemas planetários preditivos, uma importante contribuição à filosofia da ciência.

1.2.2 Lógica na Ásia

Embora a lógica tenha sido estudada formalmente tanto na China quanto na Índia, essas tradições não chegaram até a cultura ocidental, ou tiveram pouca influência nesta parte do mundo. Em particular, o uso da lógica na matemática e na ciência da computação sofreu muito pouca influência dos desenvolvimentos da lógica na Ásia. Devido à relativa falta de informações históricas sobre a lógica nesses países, são apresentados apenas alguns comentários breves sobre eles.

Lógica na China

Mozi, um contemporâneo de Confúcio, é creditado como o fundador da escola Mohista, cujos ensinamentos lidavam com os problemas relacionados com a inferência e com as condições das conclusões corretas. Em particular, uma das escolas que nasceu do Mohismo, a "Escola dos Nomes", é creditada por alguns estudiosos como sendo uma das primeiras escolas a investigar a lógica formal. Infelizmente, em função da violenta imposição da severa regra do *legalismo*[3] durante a dinastia Qin, essa linha de investigação desapareceu da China até a introdução da filosofia indiana pelos Budistas.

[2] Na lógica formal, como veremos no Capítulo 4, falamos em *validade*.

[3] O legalismo foi uma filosofia que enfatizava a obediência rigoroza ao sistema legal. Foi uma filosofia política utilitarista que não buscava responder questões mais elevadas como o propósito ou a natureza da vida. Pode significar, de uma maneira geral, "filosofia política que sustenta o poder da lei".

Lógica na Índia

Os sutras Nyaya de Akasapada Gautama – cerca do século II ACE – são os textos centrais da escola da Nyaya, uma das seis escolas ortodoxas da filosofia Hindu. Esta escola criou um rígido esquema de cinco elementos de inferência envolvendo uma premissa inicial, uma razão, um exemplo, uma aplicação e uma conclusão. A filosofia budista do idealismo se tornou a principal escola concorrente dos Naiyayikas. Nagarjuna, o fundador da Madhyamika, o "caminho do meio", desenvolveu uma análise conhecida como "catuskoti" ou tetralema. Essa análise sistematicamente examinava a afirmação de uma proposição, sua negação, sua afirmação e negação conjunta, e finalmente, a negação de sua afirmação e negação conjuntas. De modo semelhante ao visto, posteriormente, na lógica multivalorada de Gödel.

Entretanto foi com Dignaga e o seu sucessor, Dharmakirti, que a lógica budista atingiu seu ápice. A base da analise deste pensadores é a definição da necessidade de uma dedução lógica, "vyapti", também conhecida como *concomitância invariável* ou *ubiquidade*. Para esse fim uma doutrina chamada "apoha" ou diferenciação foi desenvolvida. As dificuldades envolvidas neste sistema, em parte, estimularam a escola dos neo-escolásticos de Navya-Nyaya, que introduziu a análise formal da inferência no século XVI.

A escola Navya-Nyaya desenvolveu teorias que guardam semelhanças com a lógica moderna, tais como a "distinção entre sentido e referência dos nomes próprios" de Gottlob Frege e sua definição de número, bem como a teoria Navya-Nyaya de "condições restritivas para os universais" que antecipou alguns desenvolvimentos da teoria de conjuntos moderna.

1.2.3 Lógica na Grécia

Na Grécia, duas importantes tradições emergiram. A Lógica estóica com as suas raízes em Euclides de Megara, um pupilo de Sócrates, que é baseada na lógica proposicional que talvez foi a mais próxima da lógica moderna. Entretanto, a tradição que sobreviveu para mais tarde influenciar outras culturas foi a lógica aristotélica, o primeiro tratado grego sobre a sistematização da lógica. Na inspeção de Aristóteles sobre os silogismo há quem diga que existe uma interessante comparação com o esquema de inferência dos indianos e com a menos rígida discussão chinesa.

Difundida através de suas traduções para o latim, na Europa, e outras línguas mais ao oeste, como árabe e armênio, a tradição aristotélica era considerada uma codificação superior das leis do raciocínio. Somente no século XIX, com uma maior familiaridade com a cultura clássica indiana e um conhecimento mais profundo da China é que essa percepção mudou no ocidente.

1.2.4 Lógica Medieval

A "lógica medieval" (também conhecida como lógica escolástica) é a lógica aristotélica desenvolvida na era medieval no período de 1200–1600 ACE. Esta tradição foi fundamentada através de textos como o "Tractatus" de Pedro da Espanha (século XIII), cuja verdadeira identidade é desconhecida. Tomás de Aquino foi o filósofo que ousou mudar a antiga concepção tradicional, baseada em Platão e Agostinho, concebendo uma visão aristotélica, e desenvolvendo a escolástica tomista.

Essa antiga tradição também recebeu diversas considerações diferentes no século XIV com as obras de Guilherme de Ockham (1287-1347) e Jean Buridan.

As últimas obras dessa tradição são "Lógica" de John Poinsot (1589-1644, também conhecido como John de St Thomas), e o "Discussões Metafísicas" de Francisco Suarez (1548-1617).

1.2.5 Lógica Tradicional

Esta tradição começou com o livro *Lógica, ou a arte do pensamento* ou *Lógica de Port-Royal* de Antoine Arnauld e Pierre Nicole. Publicado em 1662, esse livro foi a mais influente introdução em lógica até o inicio do século XX. O Lógica de Port-Royal apresenta ao leitor uma doutrina cartesiana com uma estrutura que deriva da lógica aristotélica e medieval. O livro teve oito edições entre 1664 e 1700. Ele foi reimpresso em inglês ate o fim do século XIX.

A descrição das proposições que o filósofo John Locke faz na obra *Uma Tese a Respeito do Entendimento Humano* é a mesma do Port-Royal. "Proposições verbais, que são palavras, os sinais de nossa ideia reunidas ou separadas em sentenças afirmativas ou negativas. Pro tal meio de afirmar ou separar, esses sinais formados de sons são, como se fossem, reunidos ou separados entre si. De sorte que as proposições consistem em reunir ou separar sinais; e a verdade consiste em reunir ou separar estes sinais, de acordo com as coisas que eles significam para concordar ou discordar." [Loc99, Livro IV, Capítulo V, pág. 244]

Obras que se enquadram nessa tradição incluem "Lógica: Ou, o Correto Uso da Razão" de Isaac Watts (2025), "Lógica" de Richard Wately (1826), e uma das últimas grande obras dessa tradição "Sistema de Lógica Dedutiva e Indutiva" de

John Stuart Mill (1843), que além de formalizar os princípios do raciocínio indutivo, também teve grande importância na filosofia da ciência e influenciou diretamente vários cientístas, mais notadamente Paul Dirac, ganhador do Prêmio Nobel de Física por seus trabalhos em física quântica.

1.2.6 A Lógica Moderna

René Descartes foi, provavelmente, o primeiro filósofo a utilizar as técnicas algébricas como meio de exploração científica. A idéia de um "cálculo do raciocínio" também foi cultivada por Gottfried Wilhelm Leibniz.

Durante o século XIX a lógica simbólica começou a se tornar popular entre os filósofos e matemáticos porque eles estavam interessados no conceito do que constitui uma prova correta em matemática. Um marco no desenvolvimento da lógica formal ocorreu em meados do século XIX com a publicação de *"Uma Investigação sobre as Leis do Pensamento"* pelo matemático e filósofo inglês George Boole [Boo58].

Gottlob Frege no *Begriffschrift*, ou Ideografia, criou um sistema de representação simbólica para representar formalmente a estrutura dos enunciados lógicos e suas relações, e a invenção do cálculo dos predicados. Esta parte da decomposição funcional da estrutura interna das frases (substituindo a velha dicotomia analítica sujeito–predicado, herdada da tradição lógica aristotélica, pela oposição matemática função–argumento) e da articulação do conceito de quantificação (implícito na lógica clássica da generalidade), tornando assim possível a sua manipulação em regras de dedução formal. Os enunciados "para todo o x" e "existe um x" que denotam operações de quantificação sobre variáveis lógicas têm a sua origem no seu trabalho fundador, ex: "Todos os humanos são mortais" se torna "Todos os x são tais que, se x é um humano então x é mortal".

Ao contrário de Aristóteles, e mesmo de Boole, que procuravam identificar as formas válidas de argumento, a preocupação básica de Frege era a sistematização do raciocínio matemático, ou dito de outra maneira, encontrar uma caracterização precisa do que é uma "demonstração matemática". Frege havia notado que os matemáticos da época frequentemente cometiam erros em suas demonstrações, supondo assim que certos teoremas estavam demonstrados, quando na verdade não estavam. Para corrigir isso, Frege procurou formalizar as regras de demonstração, iniciando com regras elementares, bem simples, sobre cuja aplicação não houvesse dúvidas. O resultado que revolucionou a lógica, foi a criação do cálculo de predicados (ou lógica de predicados).

Em 1889 Giuseppe Peano publicou seus nove axiomas sobre números naturais. Mais tarde cinco destes vieram a ser conhecido com axiomas de Peano e, destes cinco, um veio a ser a formalização do princípio da indução matemática.

Os lógicos do século XIX e início do século XX esperavam estabelecer a lógica formal como o fundamento para a matemática. Embora os fundamentos da lógica matemática tenham se desenvolvido nesta época e trabalhos importantes como o *"Begriffsschrift"* de Frege [Fre79] e *"Principia Matematica"*, de Whitehead e Russell [WR25], tenham avançado muito o entendimento sobre lógica e a formalização da matemática, o objetivo de "logicização" de toda a matemática nunca se concretizou por completo[4]. Apesar disso, a lógica permitiu aos matemáticos apontar o motivo pelo qual uma suposta prova estava errada, ou em que parte da prova o raciocínio foi falho.

O trabalho de Frege nos fundamentos da lógica e a formulação da teoria dos conjuntos de Peano, abriram para o enorme desenvolvimento da lógica matemática que se seguiu no século XX. Nomes como o de Bertrand Russell, Alan Turing e Kurt Gödel, trilharam o caminho aberto por Frege e Peano, e mudaram para sempre o panorama da lógica, da matemática e da computação.

Uma grande parte do crédito por esse feito deve-se ao fato de que, escrevendo os argumentos na linguagem simbólica ao invés de escreve-los em linguagem natural (que é repleta de ambiguidade), verificar a corretude de uma prova se torna uma tarefa muito mais viável.

Apenas a título de exemplo, a Figura 1.1 mostra a formulação em *ideografia* para a Sentença de Geach-Kaplan. Esta sentença é um exemplo bem conhecido de uso de lógica de alta ordem e, em português, a sentença de Geach-Kaplan é:

> *Alguns críticos admiram somente uns aos outros.*

Também com exemplo, a Figura 1.2 mostra a mesma sentença representada na notação moderna da Lógica de Predicados de 2ª Ordem. Não se preocupe se você não entender as representações apresentadas nas Figuras 1.1 e 1.2. Elas estão postas aqui apenas para dar um vislumbre do que é trabalhar com lógica moderna. Tanto a *ideografia* quanto as lógicas de alta ordem (como a de 2ª ordem, por exemplo) estão fora do escopo de um curso introdutório de lógica. A *ideografia* faz parte apenas da história, e as lógicas de alta ordem pertencem a cursos mais avançados.

[4]Devido principalmente ao resultado do *Teorema da Incompletude*, de Göedel.

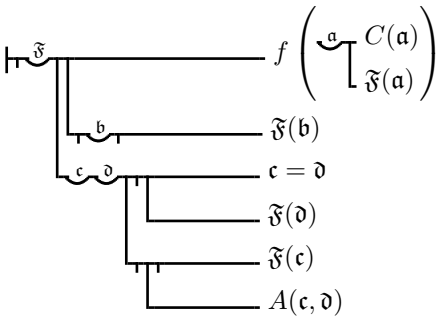


Figura 1.1: Notação do *Begriffschrift* para a Sentença de Geach-Kaplan.

$$\exists F : [\exists a : \exists b : (F(a) \wedge F(b) \wedge A(a,b)) \wedge \exists x : \neg F(x) \wedge \forall c : \forall d : (F(c) \wedge A(c,d) \rightarrow F(d))]$$

Figura 1.2: Notação da lógica de segunda ordem para a Sentença de Geach-Kaplan.

1.3 Lógica Informal

Como vimos na Seção 1.2, durante séculos o estudo da lógica inspirou a idéia de que seus métodos poderiam ser aproveitados nos esforços para entender e melhorar o pensamento, o raciocínio, e a argumentação na forma como eles ocorrem no dia a dia: em debates e discussões públicas; e no direito, na medicina, na filosofia, e em outras profissões. A *lógica informal* é um esforço de construir uma lógica apropriada para este propósito. Ela combina o estudo da argumentação, de evidências, de provas, e de demonstrações, com uma perspectiva que contribui para enfatizar sua utilidade na análise de argumentos da vida real.

A lógica informal, como campo de estudo moderno, emergiu na segunda metade do século XX, quando muitos filósofos e logicistas começaram a voltar sua atenção para a análise, avaliação, e aperfeiçoamento de argumentos da vida real. Entretanto, a lógica informal deve ser entendida como uma continuação de muitas tentativas antigas de filósofos que propuseram métodos para entender e avaliar argumentos reais.

Ao desenvolver uma explicação sobre argumentação, os logicistas estudaram a inferência e muitos outros tópicos relevantes, dentre os quais temos: explicações concorrentes sobre a natureza dos argumentos; critérios para avaliação de argumentos; esquemas de argumentos; falácias; modelos de inferência dedutivos, indutivos, condutores e abdutivos; abordagens retóricas e dialéticas à argumentação; ônus e responsabilidade da prova; o estudo empírico da argumentação; diagramação (ou "mapeamento") de argumentos; Predisposições cognitivas; a história da análise de argumentos; o papel da emoção no argumento; e as regras que regem a troca argumentativa em diferentes contextos de comunicação.

De acordo com Blair [Bla15], existem duas tarefas principais para a lógica informal: (a) desenvolver formas de identificar e extrair argumentos do discurso em linguagem natural; e (b) desenvolver métodos e diretrizes para avaliar a força de convicção (ou irrefutabilidade) dos argumentos. Ambas as tarefas assumem alguma definição do que conta como um argumento.

1.3.1 Argumentos Informais

A lógica informal entende o termo "argumento" como uma tentativa de oferecer evidência em favor de algum ponto de vista. Neste sentido, argumentar é um ato intencional, um ato de fala ou de comunicação, funcionando como uma tentativa de resolver algum conflito ou discordância.

No sentido comprovativo, a lógica informal entende argumentos como coleções de premissas e conclusões. As premissas fornecem as evidências que dão suporte às conclusões. Essa ideia é apresentada no exemplo abaixo.

Exemplo 1.1

"Uma superfície cinza parece vermelha se tivermos olhado para um azul-esverdeado; O papel plano parece liso se nós estivermos sentido uma lixa ou áspero se nós tivermos sentido uma placa de vidro; e água da torneira tem

gosto doce se tivermos comido alcachofras. Alguma parte do que chamamos de vermelho ou suave ou doce deve, portanto, estar nos olhos, ou nas pontas dos dedos, ou na língua do observador, do apalpador, ou do provador." [Ski71, p. 103]

Podemos analisar este argumento da seguinte forma:

Premissa:	Uma superfície cinza parece vermelha se tivermos olhado para um azul-esverdeado.
Premissa:	O papel plano parece liso se nós estivermos sentido uma lixa.
Premissa:	O papel plano parece áspero se nós tivermos sentido uma placa de vidro.
Premissa:	Água da torneira tem gosto doce se tivermos comido alcachofras.
Conclusão:	Alguma parte do que chamamos de vermelho ou suave ou doce deve, portanto, estar nos olhos, ou nas pontas dos dedos, ou na língua do observador, do apalpador, ou do provador.

Os próximos exemplos mostram que identificar e extrair argumentos de seus contextos nativos pode apresentar alguns desafios.

Exemplo 1.2

"O advogado do réu afirmou que a sentença proferida pelo juiz não era proporcional ao crime porque tinha uma duração sem precedentes."

O argumento pode ser resumido como:

Premissa:	A sentença (proferida pelo juiz) tinha uma duração sem precedentes.
Conclusão:	A sentença não era proporcional ao crime.

Note que neste caso, a premissa aparece no texto após a conclusão. A palavra "porque" funciona como um indicador de premissa, neste caso.

Exemplo 1.3

"Um coordenador da Sociedade Humanista apoiou uma sentença de prisão, alegando que as penas menores normalmente associadas a condenações por contravenção não são um elemento suficientemente desencorajante neste caso."

Este argumento pode ser resumido como:

Premissa:	as penas menores normalmente associadas a condenações por contravenção não são um elemento suficientemente desencorajante neste caso.
Conclusão:	Uma sentença de prisão é necessária.

Novamente temos a premissa ocorrendo após a conclusão no texto. O indicador de premissa, neste exemplo, foi a expressão "alegando que", que serviu para indicar uma explicação ou justificativa da conclusão.

Exemplo 1.4

"As pequenas empresas são as que mais contribuem para a economia. Elas representam mais de dois terços das oportunidades de emprego."

Resumidamente:

Premissa:	As pequenas empresa representam mais de dois terços das oportunidades de emprego.
Conclusão:	As pequenas empresas são as que mais contribuem para a economia.

Muitas vezes, os componentes de um argumento são intercalados com digressões e observações repetitivas que não são relevantes para o argumento ou sua avaliação. No processo de identificação e extração de um argumento de seu

contexto, isso pode significar que devemos reconhecer o que é implícito, mas relevante para o argumento, ao mesmo tempo que descartamos o que é explícito, mas redundante ou irrelevante.

Nos Exemplos 1.2 e 1.3, os argumentos dependem de alegações implícitas que devem ser consideradas na avaliação dos argumentos. No processo de identificação dos componentes de um argumento, tais alegações são identificadas como premissas (ou conclusões) *implícitas*. Em nossos dois exemplos, isso significa que podemos identificar os componentes dos argumentos como:

Exemplo 1.5 (Continuação do Exemplo 1.2)

Premissa:	A sentença (proferida pelo juiz) tinha uma duração sem precedentes.
Premissa Implícita:	As sentenças por crimes não devem ter uma duração sem precedentes.
Conclusão:	A sentença não era proporcional ao crime.

Exemplo 1.6 (Continuação do Exemplo 1.3)

Premissa:	As penas menores normalmente associadas a condenações por contravenção não são um elemento suficientemente desencorajador neste caso.
Premissa Implícita:	As sanções penais devem ter um efeito desencorajador.
Conclusão:	Uma sentença de prisão é necessária.

É importante identificar premissas e conclusões implícitas porque são elementos que precisam ser analisados e avaliados em qualquer tentativa de avaliar os argumentos que os contêm. No Exemplo 1.3, por exemplo, a questão de saber se as punições devem ser atribuídas considerando seus efeitos desencorajadores é uma questão fundamental que deve ser considerada para decidir se o argumento fornece evidência convincente para sua conclusão.

1.3.2 Preparação de Argumentos

Quando analisamos argumentos da vida real, nossa primeira tarefa é a identificação de suas partes componentes. Esta tarefa às vezes é chamado de "preparar" o argumento. Ou seja, tomar os argumentos como eles aparecem em seus contextos de vida real, e identificar e isolar seus componentes chaves de uma maneira que prepara o caminho para a avaliação dos argumentos.

Premissas e Conclusões

A tarefa mais elementar na preparação de um argumento é identificar suas premissas e conclusões. Tanto a lógica formal quanto a lógica informal entendem premissas e conclusões como os componentes centrais de um argumento. Em casos simples, eles são claramente e explicitamente indicados e facilmente identificados.

Preparar argumentos é um aspecto chave da análise de argumentos que ocorrem no discurso cotidiano porque tais argumentos são frequentemente obscuros e/ou abertos à interpretação. Ao isolar suas premissas e conclusões, isso significa que podemos ter que realizar uma ou mais das tarefas abaixo.

- Descartar digressões irrelevantes e comentários extemporâneos ("ruído");
- Eliminar questões retóricas e outros dispositivos estilísticos que obscurecem seu significado;
- Escolher entre formas alternativas de expressá-los;
- Resolver as questões levantadas por declarações e declarações incompletas, vagas ou ambíguas;
- Identificar elementos visuais, auditivos, olfativos e outros tipos de premissas que fornecem evidência para uma conclusão.

Premissas e Conclusões Implícitas

A possibilidade de premissas ou conclusões implícitas já era reconhecida desde a antiguidade. Alguns exemplos de premissas implícitas já foram apresentados anteriormente. As premissas implícitas identificam a ligação entre a premissa explícita e a conclusão. Ao preparar o argumento essas premissas precisam ser explicitadas porque precisam ser aferidas em qualquer tentativa de avaliar o argumento.

A tarefa de identificar premissas ou conclusões implícitas levanta questões teóricas porque há muitas circunstâncias em que diferentes premissas ou conclusões implícitas podem ser atribuídas a um argumento. Um princípio comum usado em tais circunstâncias é o "Princípio da Caridade", que nos leva a buscar uma premissa implícita que torna o argumento o mais forte possível. Uma abordagem alternativa atribui a um argumento o "mínimo lógico" – isto é, a premissa mais fraca necessária para ligar as premissas do argumento à sua conclusão.

Tipos de Diálogos

Um componente externo dos argumentos é o tipo de diálogo no qual eles estão inseridos. Em um diálogo de investigação, por exemplo, os argumentos são usados como ferramentas na tentativa de estabelecer o que é verdadeiro. Entendidos dentro deste contexto, os argumentos devem aderir a padrões rígidos que determinam o que conta como evidência ou como contra-evidência para algum ponto de vista. Em um diálogo de negociação, os argumentos funcionam de maneira diferente. O objetivo aqui é estabelecer um acordo entre duas partes que têm interesses conflitantes e pontos de vista diferentes, possivelmente inconciliáveis. Pode-se resumir essas diferenças dizendo que as expectativas, normas e procedimentos para argumentar dependem do tipo de diálogo no qual um argumento é proposto.

Podemos entender um diálogo como uma troca composta de uma fase de abertura, uma fase de argumentação e uma fase de encerramento. Na fase de abertura, os argumentadores no diálogo concordam em participar. As regras para o diálogo definem que tipos de movimentos são permitidos, que tipos de perguntas e respostas são permitidas, e que normas devem ser aderidas. Segundo Walton [Wal07], há sete tipos de diálogos que podem ser resumidos como mostra a Tabela 1.1.

Tipo	Situação	Objetivo do Argumentador	Objetivo do Dialogo
Persuasão	Conflito de opiniões	Persuadir a outra parte	Resolver problemas
Investigação	Precisa haver prova	Verificar evidências	Provar uma hipótese
Descoberta	Necessidade de explicação	Encontrar uma hipótese	Sustentar uma hipótese
Negociação	Conflito de interesses	Assegurar seus interesses	Resolver a questão
Informação	Necessidade de informação	Adquirir informação	Troca de informações
Deliberação	Escolha prática	Ajustar metas e ações	Decidir o que fazer
Erístico	Conflito pessoal	Atacar o oponente	Revelar conflito intenso

Tabela 1.1: Tipos de diálogos, segundo a classificação de Walton [Wal07].

Dentro destas categorias gerais, tipos mais específicos de diálogo podem ser governados por regras estritas. Uma das subespécies do diálogo negocial é, por exemplo, a negociação coletiva, que proíbe legalmente a "negociação de má-fé" - isto é, a negociação que ocorre fora da mesa de negociação. Neste tipo de negociação, o uso de ameaças (para atacar ou bloquear os funcionários) é uma parte fundamental do processo. Em contraste nítido, as ameaças não são aceitáveis na investigação crítica, onde são classificadas como exemplos da falácia[5] *ad baculum*.[6]

A questão de saber se há outros tipos de diálogo que ainda precisam ser reconhecidos permanece aberta, bem como a questão de saber se existem tipos de argumentação que não podem ser categorizados em termos de qualquer das formas padrão de diálogo (porque eles são um híbrido de diferentes tipos de diálogo, ou alguma nova forma de diálogo que não esteja claramente definida). Reconhecer um diálogo no qual um argumento está embutido é uma parte importante da

[5] Uma *falácia* é o uso de raciocínio inválido ou defeituoso, ou "movimentos errados" na construção de um argumento. Algumas falácias são cometidas intencionalmente para manipular ou persuadir por engano, enquanto outros são cometidos involuntariamente devido à negligência ou ignorância.

[6] *Argumentum ad baculum*, também conhecido como "apelo à força", é um argumento em que a força, a coerção ou a ameaça de força é dada como justificativa.

preparação do argumento quando isso impõe padrões de troca argumentativa aos quais os argumentadores podem não ter aderido adequadamente.

Da Preparação à Avaliação

Preparar argumentos é um precursor para a avaliação do argumento. A preparação pode ser um empreendimento complexo que requer o descarte de aspectos irrelevantes da comunicação, uma interpretação de uma situação em que as premissas e as conclusões não são claramente demarcadas, a identificação de premissas ou conclusões implícitas. Em muitos casos, requer o reconhecimento de que está embutido em um determinado tipo de diálogo, faz parte de uma troca com algum oponente argumentativo identificável ou é endereçado a um público específico que um argumentador está tentando convencer. Desta forma, uma preparação completa de um argumento requer uma descrição detalhada de suas características internas e externas. É uma atividade que requer treino e atenção. Não é, de modo algum, uma tarefa trivial.

1.3.3 Avaliação de Argumentos

O objetivo maior da lógica informal é normativo. Sua meta é desenvolver uma teoria de argumentação que possa ser usada para decidir quando os argumentos são fortes ou fracos; bons ou maus; e plausíveis e implausíveis. Os logicistas informais preparam argumentos para a avaliação. Em vista disso, muito da lógica informal é uma tentativa de desenvolver padrões, critérios e processos para julgar argumentos. Argumentos particulares podem ser avaliados em termos de critérios gerais de boa argumentação; ou como um exemplo de uma falácia ou de um esquema de argumentos.

Uma maneira de avaliar os argumentos no discurso cotidiano é traduzindo-os em uma linguagem formal e avaliando-os adequadamente. Este é um método que os logicistas informais usam algumas vezes, especialmente nas discussões sobre inteligência artificial, abordagens baseadas em teoria dos jogos para o diálogo e relatos formais de vários tipos de raciocínio. Métodos mais informais têm sido enfatizados no ensino de argumentação e raciocínio. No uso da lógica informal no discurso público (argumentar a favor ou contra, ou na análise de pontos de vista particulares) e no estudo de argumentos que não são facilmente acomodados pelos métodos formais.

Critérios AV e ARS

Na lógica clássica, um argumento é (dedutivamente) válido se sua conclusão deriva de suas premissas, i.e., se é impossível que suas premissas sejam verdadeiras e sua conclusão seja falsa. O objetivo final de argumentar é um argumento "sólido": um argumento válido com premissas verdadeiras. Muitas questões surgem quando se tenta aplicar esses critérios a argumentos informais sem adaptá-los de várias maneiras, mas eles apontam de forma útil que a força de um argumento é função de duas coisas: (i) a viabilidade de suas premissas, e (ii) a força da inferência dessas premissas para sua conclusão.

Dentro da lógica informal, os critérios mais simples para julgar os argumentos são um análogo informal da "solidez". Este critério exige que as premissas de um argumento sejam aceitáveis e que sua conclusão resulte dessas premissas. Podemos chamar esta última de validade "informal" e estes dois critérios são os critérios "AV" (aceitabilidade e validade) para avaliar argumentos.

Muitos logicistas informais entendem a validade informal em termos de relevância e suficiência, tornando os critérios para bons argumentos *aceitabilidade, relevância e suficiência* (os critérios "ARS"). As premissas de um argumento contam como relevantes para sua conclusão quando elas fornecem algum suporte para a conclusão e suficiente quando elas fornecem apoio suficiente para estabelecer a conclusão como plausível.

A lógica informal favorece a aceitabilidade sobre a verdade como critério para julgar premissas. Ambos os critérios AV e ARS avaliar premissas como aceitável ou inaceitável em vez de verdadeiro ou falso. Isso é feito por várias razões. Porque na vida real discussões tendem a ocorrer em contextos caracterizados pela incerteza, o que tornam difícil fazer julgamentos de verdade ou falsidade. Porque esses argumentos frequentemente giram em torno de julgamentos éticos e estéticos que não são facilmente categorizados como verdadeiros ou falsos. Porque há contextos (por exemplo, negociação) em que bons argumentos não dependem de premissas que podem claramente ser categorizadas como verdadeiras ou falsas. Porque existem outros contextos (em lidar com públicos específicos, por exemplo) nos quais a verdade não pode, por si só, tornar uma premissa aceitável para um argumento informal. E, mais geralmente, porque discussões filosóficas sobre a "verdade" que levantaram muitas perguntas sobre o que realmente conta como verdadeiro e falso.

Uma situação possível em que as premissas de um argumento são verdadeiras e sua conclusão não é verdadeira, é chamada de *contraexemplo* do argumento. Uma outra forma de definir seria simplesmente dizendo que um argumento válido é um que não possui contraexemplos.

Quando falamos de situações possíveis, o termo 'possível' deve ser entendido em um sentido bastante amplo. Para ser possível, uma situação não precisa ser algo que possamos realizar; nem sequer tem que obedecer às leis da física. Ela só tem de ser algo que podemos conceber coerentemente – ou seja, sem auto-contradição. Um contra-exemplo não precisa ser uma situação real, embora possa ser; basta que a situação seja conceitualmente possível.

Uma descrição bem definida de um contraexemplo deve conter três elementos:

1. Afirmações de todas as premissas do argumento.
2. Uma negação da conclusão do argumento.
3. Uma explicação de como isso pode acontecer, isto é, como a conclusão pode ser falsa enquanto as premissas são todas verdadeiras.

Pode haver muitos, ou até infinitos, contraexemplos para um determinado argumento. Nós não precisamos encontrar todos eles. Basta achar um contraexemplo para podermos afirmar que o argumento não é válido.

Exemplo 1.7

"Eles disseram no rádio que hoje será um dia lindo. Então, hoje será um dia lindo."

Contraexemplo: "Eles disseram no rádio que hoje será um dia lindo. Mas eles estão errados. Uma frente fria está se movendo inesperadamente e trará chuva em vez de um dia lindo."

Todos os três elementos estão presentes. A primeira frase afirma a premissa. A segunda nega a conclusão. A terceiro explica como a conclusão pode ser falsa mesmo que a premissa seja verdadeira.

Observe novamente que o contraexemplo não precisa ser uma situação real. É apenas uma história, um cenário, uma ficção.

Dedutivismo em Linguagem Natural

Na avaliação dos argumentos, uma das principais questões levantadas pela lógica informal é a questão de como devemos entender a validade informal, que é um componente chave de argumentos fortes. Embora os logicistas informais comumente distinguam entre a validade dedutiva e vários tipos alternativos de validade, alguns assumem uma visão alternativa, que é chamado de "Dedutivismo da Linguagem Natural" (DLN). O DLN sustenta que todos os argumentos informais podem ser melhor interpretados como tentativas de criar inferências dedutivamente válidas, e devem ser analisados e avaliados de acordo.

De modo geral, uma conclusão dedutiva é tão certa quanto as premissas em que se baseia. Este é um ponto importante em contextos informais, onde os argumentos se baseiam em premissas que são razoáveis ou plausíveis ao invés de certas. Nesses contextos, os argumentos dedutivos produzem conclusões que não são certas, mas razoáveis ou plausíveis. Considere o exemplo abaixo.

Exemplo 1.8

"A população do mundo crescerá de 7 para 10 bilhões nos próximos 30 anos, por isso, se quisermos fornecer comida suficiente para todos, precisamos de uma maneira de alimentar 3 bilhões de pessoas adicionais."

Este é um argumento obviamente dedutivo, mas depende de uma premissa que não é certa, mas razoável – porque é apoiada por outros raciocínios que extrapolam as tendências demográficas estabelecidas. O argumento é dedutivamente válido, mas sua conclusão é razoável e não certa.

Ao lidar com argumentos que não são explicitamente dedutivos, a abordagem DLN interpreta um argumento como um argumento dedutivo atribuindo premissas implícitas que o tornam dedutivo. Por exemplo:

Exemplo 1.9

"Paulo Freire disse, então deve ser verdade."

O Exemplo 1.9 é um padrão de raciocínio comum frequentemente criticado que apela para a autoridade de um personalidade famosa para justificar a conclusão. Se julgarmos apenas pela premissa explícita (*Paulo Freire disse*), este não é um argumento dedutivamente válido pois a premissa pode ser verdadeira e a conclusão falsa (para horror de muitos, *Paulo Freire pode estar errado*). A DLN contorna essa questão atribuindo ao argumento uma premissa implícita que o torna dedutivo. Assim o argumento pode ser resumido como:

Exemplo 1.10 (Continuação do Exemplo 1.9)

Premissa:	Paulo Freire disse isso.
Premissa Implícita:	Se Paulo Freire disse isso, isso deve ser verdade.
Conclusão:	Isso deve ser verdade.

O Exemplo 1.10 é uma reconstrução plausível do argumento apresentado no Exemplo 1.9. Pois não podemos extrair a conclusão a partir da premissa declarada sem aceitar a premissa implícita. Uma vez tornado explícito, o argumento é claramente dedutivo. Isso não o torna um bom argumento e, em vez disso, sugere que devemos avaliar sua força, não avaliando sua validade, mas avaliando a plausibilidade de suas premissas. Como a premissa implícita é implausível uma análise da DLN implica a conclusão (correta) de que o argumento é fraco.

A DLN é uma abordagem de avaliação de argumentos que torna explícitas algumas das alegações e dos pressupostos chaves de que os argumentos individuais dependem. Uma vez que a DLN analisa todos os argumentos como instâncias de uma forma de inferência bem compreendida, ela elimina a necessidade de distinguir entre argumentos que são dedutivos, indutivos, condutores, abdutivos, etc. – uma distinção que pode ser difícil de fazer ao lidar com argumentos informais. Aristóteles foi uma figura chave que adota uma abordagem dedutivista.

Teoria das Falácias

Na busca de maneiras diferenciadas de lidar com a lógica informal, a pesquisa nesta área inicialmente se voltou para a teoria das falácias. Em seu tratamento das falácias, reviveu uma tradição que pode ser atribuída a Aristóteles. Na história da lógica e da filosofia, seu significado se reflete nos escritos de filósofos como John Locke, Richard Whately e John Stuart Mill. Hoje, esta tradição aparece nos manuais e nos sites web que tentam ensinar raciocínio informal ensinando estudantes como detectar as falácias mais comuns.

As discussões teóricas sobre falácias nuca produziram uma taxonomia universalmente aceita, mas há um conjunto comum de falácias que são frequentemente usadas na análise de argumentos informais. Incluindo falácias formais como afirmar o consequente e negar o antecedente; e falacias informais como *ad hominem* (“contra a pessoa”), declive escorregadio, *ad bacculum* (“apelo à força”), *ad misericordiam*, “generalização precipitada” e “dois erros”. Nos livros de lógica informal, os autores podem elaborar sua própria nomenclatura para destacar as propriedades de casos particulares de argumentos falaciosos.

Embora alguns problemas com o estudo das falácias tradicionais tenham diminuído seu significado na teoria da argumentação, eles continuam a ser uma maneira popular de analisar o raciocínio e o argumento cotidianos. Uma boa introdução ao estudo das falácias é a obra clássica “*Como Vencer um Debate Sem Precisar Ter Razão*” [Sch03].

Esquemas de Argumentos

Os esquemas de argumento são padrões recorrentes de argumento. Uma vez identificados, eles podem ser usados para avaliar uma instância de argumentação, ou como receitas que nos dizem como construir argumentos fortes.

Uma abordagem padrão aos esquemas de argumentos combina um padrão particular de argumento com um conjunto de “questões críticas” que levanta. Considere o esquema “Apelo à Autoridade” (também chamado de “Apelo à Opinião de Perito”), que pode ser resumido como segue.

Exemplo 1.11 (Apelo à Autoridade)

A é uma autoridade no domínio D.

A diz que T é verdade.

T está dentro de D.

(Portanto) T é verdadeira.

Questões críticas:

1. Como é credível A?
2. A é uma autoridade no domínio D?
3. O que A afirma que implica T?
4. A é alguém que pode ser confiável?
5. T é consistente com o que outros especialistas afirmam?
6. A afirmação de A sobre T é baseada em evidências?

O comentário abaixo é um exemplo de uso do um apelo de autoridade.

Exemplo 1.12

"Não devemos armazenar armas nucleares. Precisamos dar ouvidos à advertência de Einstein, após o bombardeio de Hiroshima, de que a proliferação de bombas atômicas levaria inevitavelmente 'à destruição ainda mais terrível do que a atual destruição da vida'."

Podemos resumir o argumento do Exemplo 1.12 como uma instância do argumento do esquema de Apelo à Autoridade, onde A = "Einstein", D = "política" (de armas nucleares), T = "a proliferação de armas nucleares vai levar à destruição terrível". Podemos então avaliar o argumento perguntando se existem respostas satisfatórias às questões críticas do argumento. Neste caso, as questões de particular importância são a questão 2 ("Einstein é uma autoridade sobre a política de armas nucleares?") e a questão 5 ("A alegação de que a proliferação de armas nucleares vai levar a uma destruição terrível é consistente com o que outros especialistas em política de armas nucleares afirmam?"). Uma vez que não é claro que qualquer das duas questões tem uma resposta positiva, o Exemplo 1.12 é um argumento fraco.

Pode-se entender um esquema como uma espécie de regra de inferência, algumas questões críticas garantem a verdade das premissas, outras asseguram que o contexto da inferência é apropriado, etc. Os esquemas dedutivos são comumente codificados em regras padrão de inferência como *modus ponens*, negação dupla, modus tollens, etc.

Para uma visão mais profunda deste tema, Walton, Reed e Macagno [WRM08] fornecem um coletânea de 96 esquemas de argumento.

1.3.4 Outros Tópicos

Como qualquer campo de estudo, a lógica informal está em constante evolução. Sua tentativa de entender os argumentos e como eles ocorrem em uma ampla gama de situações da vida real continua a empurrá-la na direção de visões mais amplas dos argumentos que reconhecem novas maneiras em que a argumentação pode ocorrer e como e quando um argumento deve ser considerado bem sucedido. A seguir falamos brevemente de alguns poucos novos tópicos nessa área.

Mapas de Argumentos

Quando se entende um argumento como uma coleção de premissas e uma conclusão com uma estrutura de inferência pode-se esclarecer e analisar esta estrutura com diagramas de argumento ou "mapas". Este tipo de diagrama teve seus primeiros usos nos séculos XIX e XX.

À medida que o interesse na análise do raciocínio informal se intensificou, esse interesse foi acompanhado por um renovado interesse no mapeamento de argumentos. De certa forma isso reviveu a prática de mapeamento, mas de uma nova maneira que permite abordagens muito mais complexas e sofisticadas para o mapeamento. Mais significativamente, tem sido acompanhado pelo desenvolvimento de softwares tais como:[7]

- Argumentative;

[7] Os links apresentados foram acessados em 2016-11-15. Não existe qualquer garantia de que eles continuarão ativos no futuro.

- argunet;
- Online Visualisation of Argument (OVA);
- PIRIKA (PIlot for the RIght Knowledge and Argument);
- Rationale© da ReasoningLab;
- Compendium;
- Theseus.

Além disso, existem ferramentas on-line para construção colaboração colaborativa de mapas de argumentos.

- AGORA-net;
- bCisive;
- Carneades;
- DebateGraph;

Todas essas ferramentas são desenhadas para auxiliar no mapeamento de argumentos da vida real. Porém elas não substituem o trabalho de preparação e análise do logicista.

Emoções em Argumentos

Tradicionalmente, apelos emocionais têm sido vistos como elementos falaciosos em argumentos. Esta é uma visão muito simples do papel da emoção no discurso informal, onde os apelos à emoção desempenham um papel significativo que não pode ser rejeitado de imediato. Em uma discussão da política nuclear, a emoção inerente a uma descrição das consequências da guerra nuclear (digamos, a queda da bomba em Hiroshima) pode desempenhar um papel legítimo no argumento. Nos processos judiciais, há espaço explícito para o relato emocional da vítima sobre o impacto de um crime, o que geralmente é considerado na sentença.

Alguns filósofos e logicistas tem estudado os argumentos emocionais, investigando o papel que diferentes tipos de emoção e expressões de emoção desempenham em argumentos da vida real, onde eles frequentemente funcionam como uma maneira de convencer um público de um ponto de vista particular. Quando um aluno, por exemplo, chora no escritório de um professor enquanto pede uma nota mais alta,[8] sua efusão emocional é um componente chave de sua tentativa de convencer o professor de que sua nota deve ser mudada. No intercâmbio real, esses apelos emocionais são frequentemente bem-sucedidos. Ao distinguir entre apelos que devem ser aceitos e rejeitados, os logicistas sugerem uma reconcepção da argumentação que reconhece os argumentos emocionais como um gênero único de argumento que precisa ser avaliado de uma maneira diferente da que as abordagens tradicionais sugerem.

Inteligência Artificial

Como a lógica informal, o desenvolvimento da inteligência artificial requer modelos teóricos que podem explicar a argumentação informal em contextos amplamente diversos. Dessa forma, modelos de lógica informal influenciaram a tentativa de modelar a argumentação entre agentes em sistemas multiagentes que imitam ou auxiliam o raciocínio humano. Ferramentas computacionais têm sido aplicadas a redes de argumentos interconectados de larga escala, e a raciocínio sobre decisões médicas, questões legais, propriedades químicas e outros sistemas complexos. Assessoramento automatizado de argumentos funciona como um auxílio computacional que pode ajudar na construção de um argumento. O trabalho de Eemeren et al. [Eem+14] fornece uma boa visão geral destas questões.

Na medida em que a lógica informal busca desenvolver uma lógica acessível ao raciocínio cotidiano, ela e a modelagem computacional continuarão a ser empreendimentos distintos. Mas ambos assumem uma compreensão teórica da maneira como o raciocínio informal funciona e deve ser avaliado. A longo prazo, a modelagem formal que a inteligência artificial requer pode restabelecer laços mais fortes entre lógica formal e a informal. Os resultados podem fomentar o desenvolvimento da lógica informal dentro de uma lógica (ou teoria da argumentação) mais integrada que reconhece as diferenças entre a lógica formal e informal, mas reconhece um modelo de argumentação abrangente que explica ambos.

[8] Este é apenas um exemplo didático, sem qualquer conexão com o autor deste livro ou o professor da disciplina.

1.3.5 Lógica Informal e Filosofia

Até certo ponto, a relação entre lógica informal e filosofia flui em ambos os sentidos. A prática da filosofia assume inevitavelmente (e muitas vezes desenvolve) algum noção de argumento enquanto reúne evidências para diferentes perspectivas filosóficas. A lógica informal assume (e muitas vezes desenvolve) alguma visão da natureza da razão, da racionalidade e do que conta como evidência e conhecimento, à medida que desenvolve sua teoria de argumentação. As questões filosóficas em jogo estão ligadas a questões complexas e incertas sobre conhecimento e evidência.

Até certo ponto, a relevância da lógica informal para a filosofia não reside na sua influência sobre as principais disciplinas filosóficas, tanto quanto na sua influência sobre ela. A lógica informal é um campo no qual os filósofos aplicam suas teorias de argumentação (racionalidade, conhecimento, etc.) ao argumento cotidiano. Em consonância com isso, os filósofos continuam a ser os principais contribuintes para a lógica informal, e os departamentos de filosofia nas faculdades e universidades são o departamento central que ensinam os cursos que são o foco da lógica informal.

Fora da filosofia, a lógica informal tem sido cada vez mais influenciada pela retórica, estudos de comunicação, análise do discurso, semiótica, linguística e inteligência artificial. Dentro e fora da filosofia, sua tentativa de construir uma teoria abrangente de raciocínio e argumento tem importantes implicações para nossa visão da racionalidade, a natureza da mente e seus processos eo papel social, político e epistemológico do raciocínio e do argumento. Neste ponto, as explorações de suas implicações para a filosofia da mente, ética e epistemologia permanecem raras e seu foco continua a ser o desenvolvimento de uma ampla teoria da argumentação que pode ser uma base para a análise e avaliação de argumentos em um leque muito mais amplo de Contextos que aqueles que caracterizaram a lógica tradicional.

Para uma visão mais geral sobre a filosofia, que é um tema fascinante por si só, uma excelente fonte é o livro "História da Filosofia Ocidental" do filósofo e logicista Bertrand Russell [Rus15]. Uma obra excelente, que além de introduzir os principais conceitos da filosófica, também os coloca em seu contexto histórico facilitando o entendimento de como foram desenvolvidos.

1.4 Motivação

A lógica tem sido chamada de *"o cálculo da informática"*. A justificativa é que a lógica desempenha um papel fundamental em ciência da computação, semelhante ao desempenhado pelo cálculo diferencial e integral nas ciências físicas e engenharias tradicionais. Na verdade, a lógica desempenha um papel importante em áreas da ciência da computação tão diversas como arquitetura de computadores (portas lógicas), engenharia de software (especificação e verificação de programas), linguagens de programação (semântica, lógica de programação), bancos de dados (álgebra relacional e SQL), inteligência artificial (prova automática de teoremas e sistemas baseados em regras), algoritmos (complexidade e expressividade) e teoria da computação (noções gerais de computabilidade).

Ao longo dos séculos, vários matemáticos haviam anunciado provas matemáticas como corretas, que mais tarde foram refutadas por outros matemáticos. Todo o conceito de lógica está apoiado sobre o que é um argumento correto, em oposição a um errôneo ou falho, mas durante séculos (pelo menos desde Platão e Aristóteles) não foi tentada nenhuma formulação rigorosa para capturar a noção de um argumento correto que guiasse o desenvolvimento da matemática.

Desde a segunda metade do século XX, a lógica tem sido usada em informática para vários fins, desde a especificação e verificação de programas até a prova automática de teoremas. Inicialmente, seu uso era restrito apenas a especificação de programas e inferências sobre suas implementações. Este uso é bem exemplificado por duas pesquisas:

1. O *Cálculo de Pré-condição Mais Fraca (Weakest Precondition Calculus)*, de Dijkstra [Dij75; Dij76; Bak80]. Aplicado no desenvolvimento de programas corretos usando lógica de primeira ordem.

2. A *Lógica de Hoare*. Também conhecida como *Lógica Floyd–Hoare* [Hoa69; Ten02], é um método que combina expressões da lógica de primeira ordem e sentenças de programas em um mecanismo de especificação e inferência bastante útil no desenvolvimento de pequenos programas.

Outro uso de computadores em prova de teoremas ou verificação de modelos (*model checking*) é a verificação de projetos de grandes circuitos antes que o chip seja fabricado. Analisar circuitos com um bilhão de transistores é, na melhor das hipóteses, propenso a erros e, na pior, humanamente impossível. Este tipo de análise é melhor realizada por máquinas usando técnicas de prova de teoremas [CL73; Gal86] ou técnicas de verificação de modelos [Ber+01; HR04].

Um poderoso paradigma de programação declarativa, chamado de *programação lógica*, tem evoluído desde o final dos anos 70 e tem encontrado diversas aplicações em ciência da computação e inteligência artificial. A maioria dos programadores que usam este paradigma lógico utiliza uma linguagem chamada *Prolog* que é basicamente uma implementação da lógica de primeira ordem.[9] Mais recentemente, cientistas da computação estão trabalhando em uma forma

[9] Na verdade, um sub-conjunto da lógica de primeira ordem chamado *cláusulas de Horn*.

de lógica de programação chamada *lógica de restrições*.

1.5 Objetivos

O principal objetivo deste curso é introduzir os conceitos fundamentais da lógica matemática: as notações lógicas (sintaxe) e como atribuir-lhes significado (semântica). Tentaremos motivar o estudo de lógica exemplificando alguns usos para a lógica matemática na ciência da computação. Vamos então estudar sistemas formais para a construção de argumentos lógicos (teoria de provas), estudando em particular o sistema de dedução natural para a lógica proposicional e lógica de predicados. Naturalmente, estaremos preocupados com a correção e a integridade desses sistemas dedutivos.

Esperamos que ao final do curso todos os alunos tenham adquirido conhecimentos básicos sobre Lógica Proposicional e Lógica de Predicados, e de como a lógica pode ser utilizada na formalização de problemas e especificação de propriedades.

1.6 Exercícios de Revisão

Exercício 1.1 Faça uma pesquisa na *Wikipédia, a enciclopédia livre*, e identifique ao menos 7 tipos diferentes de lógica. Para cada tipo, apresente um resumo de um parágrafo descrevendo o tipo de lógica.[10]

Exercício 1.2 Construa uma linha do tempo indicando os principais eventos da história da lógica. Use algum software para construir a sua linha do tempo.

Exercício 1.3 (Argumentos Informais na Mídia) Identifique pelo menos 5 argumentos informais nas mídias a que você tem acesso (jornais, revistas, sites de notícias, blogs, etc.). Para cada argumento, transcreva o argumento e dê a referência bibliográfica completa no estilo Harvard.[11]

Exercício 1.4 Para cada um dos argumentos informais apresentados no Exercício 1.3, apresente o resumo em formato de tabela, contendo as *premissas*, as *premissas implícitas* e as *conclusões*.

Exercício 1.5 Para cada um dos argumentos abaixo, justifique por quê os argumentos abaixo são **inválidos**, e dê contraexemplos.

a) Todas as partículas com carga possuem massa. Nêutrons são partículas que possuem massa. Logo, nêutrons são partículas com carga.

b) O bilhete vencedor é o número 720. Rita tem o bilhete 719. Portanto, Rita não tem o bilhete vencedor.

c) Ninguém nesta sala é mais alto do que Ana. Bruno está nesta sala. Logo, Bruno é mais baixo do que Ana.

d) Algumas pessoas fumam cigarros. Algumas pessoas fumam cachimbos. Portanto, algumas pessoas fumam cigarros e cachimbos.

e) Algumas pessoas fumam cigarros. Logo, algumas pessoas não fumam cigarros.

Exercício 1.6 Classifique os argumento a seguir como válidos ou inválidos. Para aqueles que forem inválidos, descreva um contraexemplo certificando-se de que a sua descrição inclui todos os três elementos de um contraexemplo bem descrito. Considere todos os argumentos como estão, ou seja, não inclua premissas implícitas, ou outros elementos externos ao argumento.

a) Nenhuma planta é senciente. Todas as coisas moralmente consideráveis são sencientes. Logo, nenhuma planta é moralmente considerável.

b) Todas as verdades matemática são conhecíveis. Todas as verdades matemáticas são eternas. Logo, tudo o que é conhecível é eterno.

c) Muitos gênios estiveram perto da loucura. Newton foi um gênio. Logo, Newton esteve perto da loucura.

d) Um alto imposto na gasolina é a forma mais efetiva de reduzir o deficit comercial. Nós precisamos reduzir o deficit comercial. Então, nós precisamos de um alto importo na gasolina.

[10] Não faça "copy & paste" da Wikipédia. Leia o artigo sobre a lógica e escreva o resumo com suas próprias palavras.
[11] Harvard Citation Style. http://guides.is.uwa.edu.au/harvard

e) Alguns anjos caíram. Então, alguns anjos não caíram.

f) Conhecer alguma coisa é estar certo dela. Nós não podemos estar certos de nada. Então, nós não podemos conhecer nada.

g) A área da superfície da China é menor do que a área da superfície da Rússia. Então, a área da superfície da Rússia é maior do que a área da superfície da China.

h) Algumas pessoas são mortais. Logo, alguns mortais são pessoas.

i) A testemunha disse que um ou dois tiros foram disparados contra a vítima. Dois projéteis foram encontrados no corpo da vítima. Então, dois tiros foram disparados contra a vítima.

j) Algumas pessoas escalam o monte Everest sem tanques de oxigênio. Então, é possível escalar o monte Everest sem tanques de oxigênio.

k) Alguns tolos são gananciosos. Alguns tolos são lascivos. Existem alguns tolos que são gananciosos e lascivos.

l) Nenhuma pessoa viveu por mais de 200 anos. Então, ninguém nunca irá viver.

m) O DNA contém o código da vida. A vida é sagrada. Logo, é errado manipular o DNA.

n) Há menos de um bilhão de pessoas no Brasil. São Paulo é apenas uma parte do Brasil. Então, há menos de um bilhão de pessoas em São Paulo.

Exercício 1.7 (Diálogos na Mídia) Procure nas mídias a que você tem acesso (jornais, revistas, sites de notícias, blogs, vlogs, etc.) ao menos um exemplo de cada um dos sete tipos de diálogos indicados na Tabela 1.1. Transcreva os diálogos abaixo, indicando o tipo, e dê a referência bibliográfica completa no estilo Harvard.

Exercício 1.8 Procure nas mídias em que você tem acesso, quatro exemplos de argumentos que **não sejam** dedutivos, e aplique o método de DLN[12] para identificar premissas implícitas e tornar estes argumentos dedutivos. Apresente cada um dos argumentos em linguagem natural, e resumidos em formato de tabela.

Exercício 1.9 (Esquemas de Argumentos) Procure nas fontes de consulta (livros, periódicos, internet, etc.) cinco tipos de esquemas de argumentos diferentes do apresentado na Seção 1.3.3. Para cada esquema, transcreva a estrutura do esquema e as questões críticas relevantes a ele, como foi feito no Exemplo 1.11; e apresente um exemplo de aplicação do esquema.

Exercício 1.10 (Mapas de Argumentos) Usando alguma ferramenta computacional de sua escolha, construa quatro mapas de argumentos, para argumentos retirados de fontes das mídias a que você tenha acesso. Dê preferências a temas moderadamente complexos, como política ou economia.

[12] Dedutivismo em Linguagem Natural.

Capítulo 2

Fundamentos de Teoria dos Conjuntos

Sumário

2.1 Introdução

Esta seção introduz os conceitos básicos relativos à *Teoria do Conjuntos* bem como algumas notações que serão utilizadas no restante do curso. O objetivo não é dar um tratamento formal completo à teoria dos conjuntos, mas apenas definir os conceitos essenciais que serão necessários para acompanhar os próximos capítulos. A versão de teoria dos conjuntos

que veremos neste capítulo é muitas vezes chamada de *Teoria Simples dos Conjuntos*[1], em contraste com a versão mais formal (e mais avançada) que é chamada de *Teoria Axiomática dos Conjuntos*. O conceito de conjunto é fundamental para o nosso estudo, pois praticamente todos os conceitos que veremos mais adiante são baseados em conjuntos e construções sobre conjuntos.

Definição 2.1 (Conjunto)

Um *conjunto* é uma coleção de zero ou mais *elementos* distintos sem qualquer ordem específica associada e que não contém a si mesma.

Usaremos a notação $\{a, b, c\}$ para indicar o conjunto cujos elementos são a, b e c.
Vamos analisar as partes da Definição 2.1.

- Um conjunto é uma coleção de elementos distintos, ou seja, sem repetições. O termo "elemento" é usado aqui de modo bem amplo. Qualquer coisa pode ser um elemento de um conjunto, até mesmo outros conjuntos. Consideraremos "elemento" como um conceito fundamental, i.e., um conceito que não precisa (ou não pode) ser definido formalmente.

- Um conjunto não possui qualquer ordem específica associada aos seus elementos. Isso significa, entre outras coisas, que não faz sentido falar em primeiro elemento do conjunto, ou segundo elemento, etc. É claro que, ao escrever o conjunto precisaremos escolher alguma ordem para a escrita, mas esta ordem é apenas um "mal necessário" para que possamos escrever o conjunto e não se deve considerar que a ordem faça parte do conjunto. Por exemplo, se temos um conjunto com os elementos a, b e c, podemos escrever este conjunto como $\{a, b, c\}$, $\{a, c, b\}$, $\{b, a, c\}$, $\{b, c, a\}$, $\{c, a, b\}$ ou $\{c, b, a\}$; todas estas formas de escrever representam o mesmo conjunto.

- Um conjunto não contém a si mesmo. Conjuntos podem conter quaisquer tipos de elementos, inclusive outros conjuntos. Entretanto, se permitirmos que um conjunto contenha a si mesmo, corremos o risco de permitir *paradoxos* em nosso sistema.[2] Para evitar o surgimento de paradoxos, impomos a condição de que um conjunto não possa conter a si mesmo. Infelizmente, uma explicação completa de paradoxos e dos problemas relacionados à autorreferência está fora do escopo deste texto.

Exemplo 2.1

São conjuntos:

1. As vogais: a, e, i, o, u.
2. Os dígitos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
3. Os números pares: 0, 2, 4, ...
4. Todos os brasileiros.
5. Os jogadores da seleção brasileira de futebol.
6. $A = \{\{a, b, c\}, d\}$. O conjunto A contém dois elementos: $\{a, b, c\}$ e d.

Podemos descrever um conjunto qualquer enumerando todos os elementos do conjunto ou especificando as propriedades que caracterizam exclusivamente os elementos do conjunto. Assim, o conjunto de todos os inteiros pares não superiores a 10 pode ser descrita como $S = \{2, 4, 6, 8, 10\}$ ou, equivalentemente, como $S = \{x \mid x \text{ é um inteiro positivo par}, x \leq 10\}$.

[1] A tradução literal do termo em inglês *"naïve set theory"* seria algo como "teoria ingênua dos conjuntos".
[2] Um *paradoxo* é uma sentença ou conjunto de sentenças que leva à uma contradição lógica ou a uma situação de desafia a lógica e a razão.

A definição de um conjunto listando todos os seus elementos é chamada de *definição por extensão*, por exemplo:

$$\text{Vogais} = \{a, e, i, o, u\}$$

que deve ser lida como "Vogais é o conjunto cujos elementos são a, e, i, o, u".

A definição de um conjunto especificando propriedades é chamada de *definição por intensão* ou *definição por compreensão*, por exemplo:

$$\text{Pares} = \{n \mid n \text{ é inteiro par}\}$$

que deve ser lida como "Pares é o conjunto de todos os elementos n tais que n é um inteiro par".

Usaremos a notação $x \in S$ para indicar que x é um elemento do (ou pertence ao) conjunto S. Para indicar o inverso, ou seja, que x **não** pertence ao conjunto S, usaremos a notação $x \notin S$.

Um *conjunto vazio* é aquele que não contém elementos e vamos denotá-lo com o símbolo $\emptyset$. Por exemplo, seja S o conjunto de todos os alunos que reprovarem neste curso. S pode vir a ser vazio (se todos estudarem muito).

Exemplo 2.2

As seguintes sentença sobre pertinência de elementos a conjunto são todas verdadeiras.

1. $a \in \{a, b, c, d\}$
2. $a \in \text{Vogais}$
3. $\{a\} \notin \text{Vogais}$
4. $0 \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
5. $\{0\} \notin \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
6. $\emptyset \notin \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
7. $f \in \{\{a, b, c\}, \{d, e\}, f\}$
8. $a \notin \{\{a, b, c\}, \{d, e\}, f\}$
9. $e \notin \{\{a, b, c\}, \{d, e\}, f\}$
10. $\{d, e\} \in \{\{a, b, c\}, \{d, e\}, f\}$
11. $\{e, d\} \in \{\{a, b, c\}, \{d, e\}, f\}$
12. $\{a, b\} \notin \{\{a, b, c\}, \{d, e\}, f\}$
13. $\{c, a, b\} \in \{\{a, b, c\}, \{d, e\}, f\}$
14. $\{f\} \notin \{\{a, b, c\}, \{d, e\}, f\}$
15. $\{\{e, d\}\} \notin \{\{a, b, c\}, \{d, e\}, f\}$
16. $\emptyset \notin \emptyset$

2.1.1 Alguns Conjuntos Importantes

Usaremos a seguinte notação para denotar alguns conjuntos de padrão:

- O conjunto vazio: $\emptyset$
- O conjunto Universo: $\mathbb{U}$
- O conjunto dos números naturais[3]: $\mathbb{N} = \{0, 1, 2, \ldots\}$
- O conjunto dos números inteiros: $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ — também é comum vermos o uso de $\mathbb{Z}^*$ para representar os inteiros não-negativos, i.e. $\mathbb{Z}^* = \mathbb{N}$ e de $\mathbb{Z}^+$ para representar os inteiros positivos, i.e., $\mathbb{Z}^+ = \{1, 2, 3, 4, \ldots\}$.
- O conjunto dos números reais: $\mathbb{R}$
- O conjunto dos números racionais: $\mathbb{Q}$
- O conjunto dos números irracionais: $\mathbb{I}$ ou $\mathbb{R} \setminus \mathbb{Q}$
- O conjunto dos valores lógicos[4]: $\mathbb{B} = \{\textit{falso}, \textit{verdadeiro}\}$
- O conjunto de todos os pontos do *plano cartesiano*: $\mathbb{R}^2$

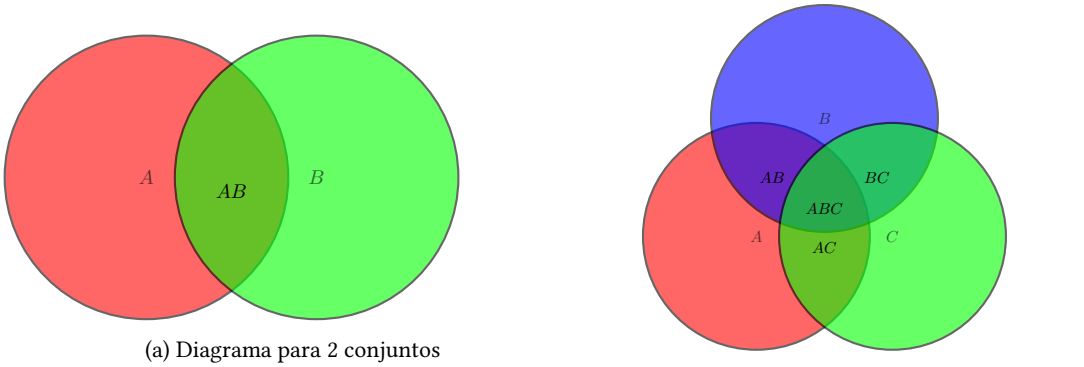
[3] Embora existam alguns autores que não considerem o zero como um número natural, nós iremos ficar do lado dos autores que consideram o zero um número natural, afinal é bastante natural tirar zero em uma prova, por exemplo.

[4] Também chamados de valores *Booleanos*.

2.1.2 Diagramas de Venn

Em muitos casos é util usar diagramas para representar relações e operações sobre conjuntos. No final do século XIX, o matemático inglês John Venn inventou uma forma de construir estes diagramas, que desde então são chamados *Diagramas de Venn* [Ven80]. Os diagramas de Venn são usados no ensino da teoria básica de conjuntos, bem como para ilustrar relações simples entre conjuntos no estudo de probabilidades, lógica, estatística, linguistica e ciência da computação.

Construir diagramas de Venn para dois ou três conjuntos é bastante fácil, entretanto, construir diagramas de Venn para mais do que três conjuntos passa a ser cada vez mais difícil à medida que o número de conjuntos aumenta, e é ainda mais difícil se tentarmos usar apenas formas geométricas "bem comportadas" como círculos ou elipses [RSW06]. A Figura 2.1 mostra exemplos de diagramas para 2 e 3 conjuntos.



(a) Diagrama para 2 conjuntos

(b) Diagrama para 3 conjuntos

Figura 2.1: Diagramas de Venn para 2 e 3 conjuntos.

O que torna difícil a construção de diagramas de Venn para mais de três conjuntos é o fato de que um diagrama de Venn deve necessariamente mostrar **todas** as possíveis interações (interferências) entre os conjuntos envolvidos. Por exemplo, para dois conjuntos — A e B — devemos ter no diagrama uma parte que mostre os elementos que são apenas de A, outra que mostre os elementos que são apenas de B e outra que mostre os elementos que pertencem ao mesmo tempo a A e a B. Assim temos que ter três regiões distintas. Para três conjuntos — A, B e C — precisaremos ter sete regiões: A, B, C, AB, AC, BC e ABC. De modo geral, um diagrama de Venn para n conjuntos precisará de $2^n - 1$ regiões distintas, o que significa que para cada novo conjunto o número de regiões do diagrama de Venn aproximadamente dobra. A Figura 2.2 mostra diagramas de Venn para 4 e 5 conjuntos.

2.1.3 Notação para Descrever e Definir Conjuntos

Como vimos, é possível representar graficamente os conjuntos através dos diagramas de Venn. Mas para poder trabalhar com eles, é necessário também representá-los em linguagem matemática.

Usamos as chaves, $\{\}$, para representar e definir os conjuntos, e no seu interior estão os elementos que o formam separados por vírgula.

Por exemplo, se quisermos escrever o conjunto $\mathcal{F}$ formado pelos elementos 1, p, z, e 3 podemos usar a seguinte forma:

$$\mathcal{F} = \{1, p, z, 3\}$$

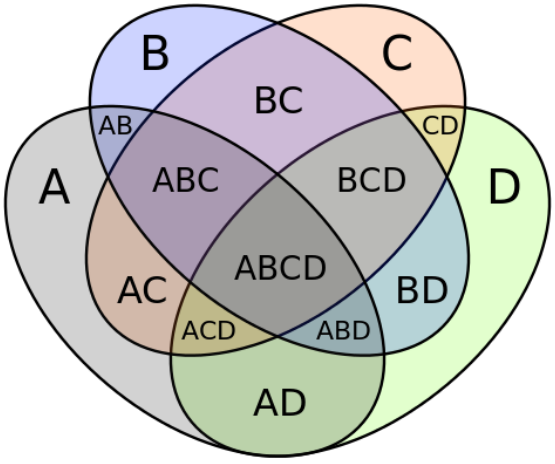
Descrição dos Conjuntos por Extensão

Para descrever os elementos de um determinado conjunto podemos mencioná-los um a um, o que é chamado de **descrição por extensão**. Definimos como o $\mathcal{Q}$ conjunto formado pelas cores do arco íris, desta forma vamos descrever o conjunto $\mathcal{Q}$ por extensão dessa forma:

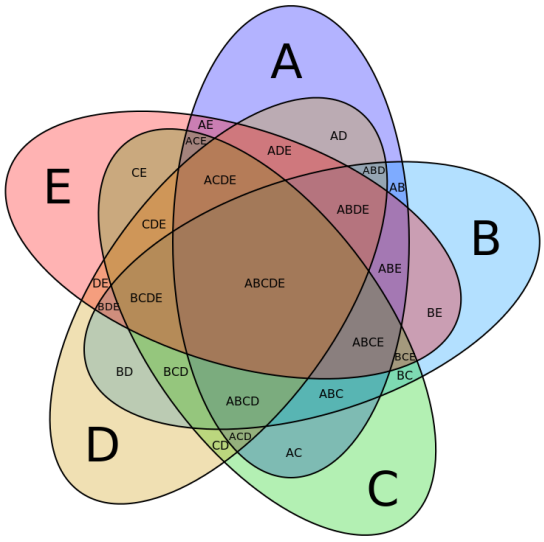
$$\mathcal{Q} = \{\text{vermelho, laranja, amarelo, verde, azul, índigo, violeta}\}$$

Se um conjunto tem muitos elementos, podemos usar **reticências**, i.e., três pontos, para descrevê-lo. Por exemplo, o conjunto $\mathcal{W}$ é formado pelos cem primeiros números inteiros positivos e podemos representá-lo da seguinte forma:

$$\mathcal{W} = \{1, 2, 3, \ldots, 98, 99, 100\}$$



(a) Diagrama para 4 conjuntos



(b) Diagrama para 5 conjuntos

Figura 2.2: Diagramas de Venn para 4 e 5 conjuntos. As regiões estão identificadas com os nomes dos conjuntos que estão interagindo na região.

Neste caso não mostramos os cem elementos que formam o conjunto, contudo, as reticências representam todos os elementos que não escrevemos.

Conjuntos por Compreensão

Em alguns casos, os conjuntos podem ter uma grande variedade de elementos e a descrição por extensão fica muito difícil. O que podemos fazer é descrever os conjuntos mencionando as características comuns dos elementos que o forma. Por exemplo, se $\mathcal{C}$ é o conjunto formado por todos os países do mundo, podemos escrevê-lo assim:

$$\mathcal{C} = \{x \mid x \text{ é um país}\}$$

A barra ($|$) é lida como "tal que". Assim a expressão acima pode ser lida da forma: "$\mathcal{C}$ é o conjunto dos elementos x, tais que x é um país". Neste caso o símbolo x é usado simplesmente para representar os elementos do conjunto $\mathcal{C}$.

Conectivos

Algumas vezes os elementos que formam um conjunto devem satisfazer mais de uma condição, ou pelo menos uma de várias. Nestes casos usamos os conectivos disjunção e conjunção para criar condições compostas.

A Disjunção

Observe o seguinte exemplo:

$$\mathcal{A} = \{a \mid a \text{ é um animal mamífero ou voador}\}$$

Neste caso há duas condições para os animais que formam o conjunto: Ser mamífero *ou* ser voar. A disjunção é a palavra **"ou"** que os conecta e significa que os elementos que formam o conjunto **devem atender alguma destas condicionais ou as** .

Neste caso por exemplo, a abelha atende à condição de voar, e por isso deve pertencer ao conjunto. O gato por sua vez, atende à condição de ser mamífero e por isso também pertence ao conjunto $\mathcal{A}$. O morcego atende às duas condições, já que é um mamífero que voa e por isto também pertence a $\mathcal{A}$.

A Conjunção

Considere agora uma variação do conjunto definido acima:

$$\mathcal{B} = \{b \mid b \text{ é um animal mamífero e voador}\}$$

Neste caso também existem duas condições mas estão unidas pela conjunção **"e"**. Isto significa que os elementos que pertencem ao conjunto **devem atender às duas condições ao mesmo tempo**.

No caso acima, nem o gato, nem a abelha fariam parte do conjunto $\mathcal{B}$ pois cada um deles só atende a uma das condições; entretanto, o morcego faria parte do conjunto $\mathcal{B}$ pois atende às duas condições simultaneamente, i.e., é mamífero e voa.

Consideremos outro exemplo. Definimos o conjunto $\mathcal{P}$ assim:

$$\mathcal{P} = \{p \mid p \text{ é um número maior que zero e menor que zero}\}$$

Como não existem números que atendem as duas condições ao mesmo tempo, concluimos que o conjunto $\mathcal{P}$ não tem elementos, ou seja, **é vazio**.

Também é possível combinar os conectivos anteriores para estabelecer as condições que devem cumprir os elementos de um determinado conjunto. Por exemplo:

$$\mathcal{K} = \{k \mid k \text{ é um número maior ou igual a } 4 \text{ e menor que } 8\}$$

Na definição dos elementos do conjunto há duas condições: "ser maior ou igual a 4" e "ser menor que 8", como estas condições estão unidas por uma conjunção, i.e., "e", ambas devem ser cumpridas. Entretanto, a condição "ser maior ou igual a 4" está composta por duas condições menores unidas por uma disjunção, i.e., "ou", e isso significa que a atendem os números que são maiores que 4 ou iguais a 4.

2.2 Relações entre Conjuntos

Assim como temos operações de comparação entre números, tais como igualdade, maior, menor, etc., também temos operações que nos permitem fazer comparações entre conjuntos.

A mais elementar comparação que podemos fazer entre dois conjuntos é perguntar se esses conjuntos são iguais ou não. Dizemos que dois conjuntos são iguais se eles contém exatamente os mesmos elementos. A Definição 2.2 formaliza este conceito.

Definição 2.2 (Igualdade de Conjuntos)

Dados dois conjuntos A e B são ditos conjuntos iguais, escrevemos $A = B$, se e somente se, A e B contém exatamente os mesmos elementos. Ou seja, se $x \in A$, então $x \in B$ e vice-versa.

Utilizando a definição de igualdade de conjuntos podemos ver que $\{1, 3, 2\}$ e $\{2, 1, 3\}$ são iguais (lembre-se que a ordem dos elementos é irrelevante nos conjuntos), mas que $\{1, 2, 4\}$ e $\{1, 2, 5\}$ não são.

Uma outra comparação que podemos fazer entre conjuntos é a comparação de *subconjunto*, ou de *estar contindo*. Um conjunto A é um subconjunto de outro conjunto B, denotada por $A \subseteq B$, se $x \in B$, sempre que $x \in A$. Alternativamente, $A \subseteq B$ se e somente se todos os elementos de A também forem elementos de B. Também podemos ler $A \subseteq B$ como "A está contido em B" ou "B contém A". A Definição 2.3 formaliza este conceito.

Definição 2.3 (Subconjunto)

Dizemos que um conjunto A é *subconjunto* de um conjunto B, denotado por $A \subseteq B$, se e somente se para todo elemento $x \in A$, temos que $x \in B$. Para denotar a noção inversa, ou seja, que A não é subconjunto de B, escreveremos $A \nsubseteq B$.

Por definição, o conjunto vazio $\emptyset$ é subconjunto de todos os conjuntos.

Iremos assumir a existência de um *universo de discurso*, o conjunto $\mathbb{U}$. Todos os conjuntos que iremos considerar são subconjuntos de $\mathbb{U}$. Assim, por definição, temos:

1. $\emptyset \subseteq A$, para qualquer conjunto A.
2. $A \subseteq \mathbb{U}$, para qualquer conjunto A.

Além da noção de subconjunto, existe uma outra noção mais restrita, a de *subconjunto próprio*. Se $A \subseteq B$, mas existe algum elemento $x \in B$ tal que $x \notin A$, então dizemos que A é um subconjunto próprio de B, escrito $A \subset B$. Outra forma de definir subconjunto próprio é dizer que $A \subset B$ sse $A \subseteq B$ e $A \neq B$.

Definição 2.4 (Subconjunto Próprio)

Dizemos que um conjunto A é *subconjunto próprio* de um conjunto B, denotado por $A \subset B$, se para todo elemento $x \in A$, temos que $x \in B$, mas existe ao menos um elemento $x \in B$ tal que $x \notin A$. Para denotar a noção inversa, ou seja, que A não é subconjunto próprio de B, escreveremos $A \not\subset B$.

A noção de *continência* (relação subconjunto) nos permite dar uma outra definição para a igualdade de conjuntos.

Definição 2.5 (Igualdade de Conjuntos)

Dois conjuntos A e B são ditos *conjuntos iguais*, denotado por $A = B$, se e somente se possuem exatamente os mesmos elementos. De modo mais formal, dizemos que $A = B$ se e somente se $A \subseteq B$ e $B \subseteq A$.

Exemplo 2.3

As seguintes proposições são verdadeiras para os conjuntos indicados:[a]

1. $\{a, b\} \subseteq \{b, a\}$
2. $\{a, b\} \not\subset \{b, a\}$
3. $\{a, b\} \subseteq \{a, b, c\}$
4. $\{a, b\} \subset \{a, b, c\}$
5. $\{1, 2, 3\} \subseteq \mathbb{N}$
6. $\{1, 2, 3\} \subset \mathbb{N}$
7. $\mathbb{N} \subseteq \mathbb{Z}$
8. $\mathbb{N} \subset \mathbb{Z}$
9. $\mathbb{Z} \subseteq \mathbb{Z}$
10. $\mathbb{Z} \not\subset \mathbb{Z}$
11. $\emptyset \subseteq \{a, b, c\}$
12. $\emptyset \subset \{a, b, c\}$
13. $\{1, 2, 3\} = \{x \mid x \in \mathbb{N} \text{ e } 0 < x < 4\}$
14. $\mathbb{N} = \{x \mid x \in \mathbb{Z} \text{ e } x \geq 0\}$
15. $\emptyset = \{\}$
16. $\emptyset \neq \{\emptyset\}$

[a] Os símbolos $\mathbb{N}$ e $\mathbb{Z}$ denotam os conjuntos dos naturais e inteiros, respectivamente.

2.3 Operações sobre Conjuntos

Veremos nesta seção as seguintes operações sobres conjuntos:

1. União
2. Interseção
3. Diferença (Complemento Relativo)
4. Diferença Simétrica
5. Complemento Absoluto
6. Conjunto das Partes (*Powerset*)
7. Produto Cartesiano

Todas as operações sobre conjuntos retornam novos conjuntos.

2.3.1 União

A *união* de dois conjuntos A e B, denotada por $A \cup B$, é o conjunto cujos elementos são os elementos que pertencem a A ou que pertencem a B (ou a ambos). Na Figura 2.3 as regiões preenchidas representam as partes dos conjuntos A e B que pertencem a $A \cup B$.

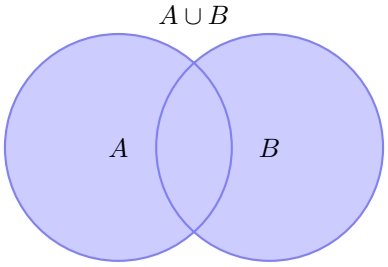


Figura 2.3: Diagrama representando a união dos conjuntos A e B.

Definição 2.6 (União de Conjuntos)

A *união* de dois conjuntos A e B, denotada por $A \cup B$, é definida como:

$$A \cup B = \{x \mid x \in A \text{ ou } x \in B \text{ ou ambos}\}$$

Exemplo 2.4

Os itens abaixo são exemplos de união de conjuntos.

1. $\{a, b, c, d\} \cup \{e, f, g\} = \{a, b, c, d, e, f, g\}$
2. $\{a, b, c, d\} \cup \{d, e, f\} = \{a, b, c, d, e, f\}$
3. $\{0, 1, 2, 3\} \cup \{1, 3, 5, 7\} = \{0, 1, 2, 3, 5, 7\}$
4. $\{0, 2, 4, \ldots\} \cup \{1, 3, 5, \ldots\} = \{0, 1, 2, 3, 4, 5, \ldots\}$
5. $\{0, 1, 2, 3, 4\} \cup \{1, 2\} = \{0, 1, 2, 3, 4\}$
6. $\emptyset \cup \{a, b, c, d\} = \{a, b, c, d\}$
7. $\{a, \{a, b, c\}, b\} \cup \{a, b, c\} = \{a, b, c, \{a, b, c\}\}$
8. $\{\{a, b\}, c, d\} \cup \{a, b, \{c, d\}\} = \{a, b, \{a, b\}, c, d, \{c, d\}\}$

2.3.2 Interseção

A *interseção* de dois conjuntos A e B, denotada por $A \cap B$, é o conjunto cujos elementos são os elementos que pertencem a A e pertencem a B simultaneamente. Na Figura 2.4 a região preenchida representa as partes dos conjuntos A e B que pertencem a $A \cap B$.

Definição 2.7 (Interseção de Conjuntos)

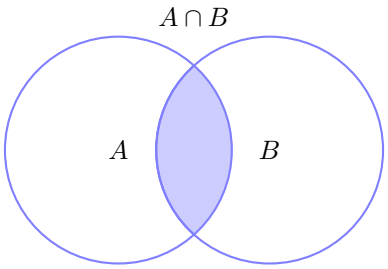


Figura 2.4: Diagrama representando a interseção dos conjuntos A e B.

A *interseção* de dois conjuntos A e B, denotada por $A \cap B$, é definida como:

$$A \cap B = \{x \mid x \in A \text{ e } x \in B\}$$

Exemplo 2.5

Os item abaixo são exemplos de interseção de conjuntos.

1. $\{a, b, c, d\} \cap \{e, f, g\} = \emptyset$
2. $\{a, b, c, d\} \cap \{d, e, f\} = \{d\}$
3. $\{0, 1, 2, 3\} \cap \{1, 3, 5, 7\} = \{1, 3\}$
4. $\{0, 2, 4, \ldots\} \cap \{1, 3, 5, \ldots\} = \emptyset$
5. $\{0, 1, 2, 3, 4\} \cap \{1, 2\} = \{1, 2\}$
6. $\emptyset \cap \{a, b, c, d\} = \emptyset$
7. $\{a, \{a, b, c\}, b\} \cap \{a, b, c\} = \{a, b\}$
8. $\{\{a, b\}, c, d\} \cap \{a, b, \{c, d\}\} = \emptyset$

2.3.3 Diferença Simétrica

A *diferença simétrica* entre os conjuntos A e B, escrita $A \triangle B$, é o conjunto formado pelos elementos que pertencem ao conjunto A ou ao conjunto B, mas não a ambos. A Figura 2.5 mostra o diagrama representando a diferença simétrica. As regiões preenchidas representam as partes dos conjuntos A e B que pertencem a $A \triangle B$.

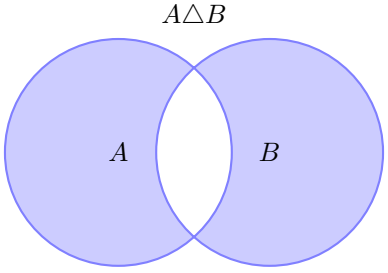


Figura 2.5: Diagrama representando a diferença simétrica dos conjuntos A e B.

Definição 2.8 (Diferença Simétrica)

A *diferença simétrica* entre os conjuntos A e B, escrita como $A \triangle B$, é definida como:

$$A \triangle B = \{x \mid x \in (A \cup B) \text{ e } x \notin (A \cap B)\}$$

2.3.4 Complemento Relativo

O *complemento de B em relação a A*, ou diferença entre A e B, denotada $A \setminus B$, é definido como o conjunto de todos os elementos de A que não pertencem a B. A Figura 2.6 representa a diferença entre A e B, i.e. $A \setminus B$. A região preenchida representa a parte de A que pertencem a $A \setminus B$.

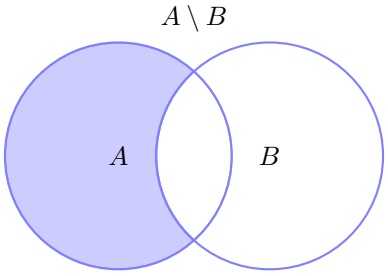


Figura 2.6: Diagrama representando a diferença entre os conjuntos A e B.

Definição 2.9 (Complemento Relativo)

O *complemento de B em relação a A*, ou diferença entre A e B, denotada por $A \setminus B$, é definida como:

$$A \setminus B = \{x \mid x \in A \text{ e } x \notin B\}$$

A expressão $A \setminus B$ também pode ser lida "A menos B". Além disso, o complemento de B em relação a A também pode ser denotado por $\complement_A B$, ou por $A - B$.

2.3.5 Complemento Absoluto

O complemento absoluto de um conjunto A, denotado $A^{\complement}$ é o complemento de A em relação a um conjunto especial que chamamos conjunto universo — normalmente o conjunto universo é representado por $\mathbb{U}$. Alternativamente, podemos pensar em $A^{\complement}$ como sendo o conjunto de todos os elementos do universo, i.e. de $\mathbb{U}$, que não pertencem a A.

O conjunto universo pode variar de uma situação para outra. Por exemplo, em uma situação o conjunto universo pode ser igual ao conjunto dos naturais, e em outra pode ser igual ao conjunto dos estados do Brasil. Entretanto, o conjunto universo sempre deve respeitar a propriedade de que, qualquer conjunto que apareça no nosso "problema" estará contido no conjunto universo.[5] Desse modo, qualquer que seja o conjunto A podemos sempre afirmar que $A \subseteq \mathbb{U}$.

A Figura 2.7 representa o complemento absoluto de A. A região preenchida representa a parte do conjunto universo que pertencem a $A^{\complement}$.

Definição 2.10 (Complemento Absoluto)

[5] Ao invés de dizer "nosso problema" também podemos dizer "domínio" ou "domínio do problema".

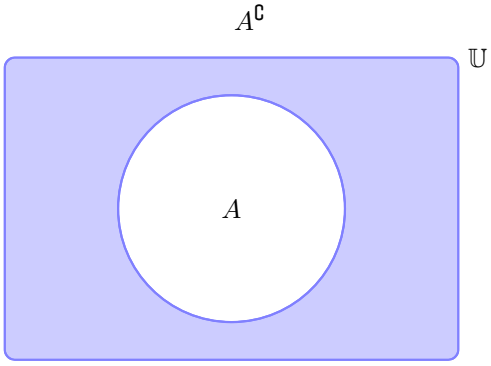


Figura 2.7: Diagrama representando complemento de A.

O *complemento absoluto* de A, ou simplesmente complemento de A, denotado por $A^{\complement}$, é definido como:

$$A^{\complement} = \mathbb{U} \setminus A$$

Outras notações comumente usadas para denotar o complemento absoluto do conjunto S são: S', $\sim S$, $\overline{S}$ e $\complement S$.

2.3.6 Conjunto das Partes ou *Powerset*

Uma outra operação que aparece com alguma frequência em computação o a operação *conjunto das partes* ou *powerset*. Essa operação, quando aplicada a um conjunto S, gera o conjunto contendo todos os subconjuntos de S, incluindo o conjunto vazio (que é subconjunto de todos os conjuntos) e o próprio conjunto S (já que todo conjunto é subconjunto de si mesmo).

Definição 2.11 (Conjunto das Partes)

O *conjunto das partes* de S, ou *powerset* de S, escrito como 2^S ou $\wp(S)$, é o conjunto formado por todos os subconjuntos do conjunto S.

Exemplo 2.6

1. $2^{\emptyset} = \{\emptyset\}$
2. Seja $A = \{1\}$, então $2^A = \{\emptyset, \{1\}\}$
3. Seja $B = \{a, b\}$, então $2^B = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$
4. Seja $C = \{1, 2, 3\}$, então $2^C = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$

2.3.7 Produto Cartesiano

O *par ordenado* formado pelos elementos a e b, denotado por (a, b), é uma coleção de exatamente dois elementos na qual é possível distinguir entre um *primeiro elemento* e um *segundo elemento*. Mais especificamente, no par ordenado (a, b), a é o primeiro elemento, enquanto b é o segundo elemento.

Podemos estender a ideia de pares ordenados para lidar com três ou mais componentes. O valor formado por três componentes, por exemplo (a, b, c), é chamado de *trio* ordenado. Para quatro componentes falamos em *quádrupla* ordenada, para cinco componentes falamos em *quíntupla* ordenada, e assim por diante. De modo geral, para n componentes, $(x_1, x_2, \ldots, x_n)$ falamos em *n-upla* ordenada.[6]

Definição 2.12 (Produto Cartesiano)

O produto cartesiano de dois conjuntos A e B, denotada por $A \times B$, é o conjunto de todos os pares ordenados (a, b) tais que $a \in A$ e $b \in B$. Assim,

$$A \times B = \{(a, b) \mid a \in A \text{ e } b \in B\}$$

Exemplo 2.7

Sejam $A = \{a, b, c\}$ e $B = \{1, 2, 3, 4\}$, o produto cartesiano de A e B é dado por:

$$A \times B = \{(a,1), (a,2), (a,3), (a,4), (b,1), (b,2), (b,3), (b,4), (c,1), (c,2), (c,3), (c,4)\}$$

Podemos estender o conceito de par ordenado para 3 ou mais conjuntos. Por exemplo, dados os conjuntos A, B e C, podemos definir o produto cartesiano destes 3 conjuntos como:

$$A \times B \times C = \{(a, b, c) \mid a \in A \text{ e } b \in B \text{ e } c \in C\}$$

De modo geral, para n conjuntos, podemos Dados os conjuntos $A_1, A_2, \ldots, A_n$, os elementos do conjunto $A_1 \times A_2 \times \cdots \times A_n$ são chamados de *tuplas n-árias* ordenadas ou *n-tuplas* oredenadas. Também podemos falar, alternativamente, em *n-uplas* ordenadas.

Exemplo 2.8

Sejam $A = \{a, b, c\}$, $B = \{1, 2\}$, $C = \{i, j, k\}$ e $D = \{\alpha, \beta\}$ o produto cartesiano de A, B, C e D é dado por:

$$\begin{aligned} A \times B \times C \times D \quad = \quad & \{(a,1,i,\alpha), (a,1,i,\beta), (a,1,j,\alpha), (a,1,j,\beta), (a,1,k,\alpha), (a,1,k,\beta), \\ & (a,2,i,\alpha), (a,2,i,\beta), (a,2,j,\alpha), (a,2,j,\beta), (a,2,k,\alpha), (a,2,k,\beta), \\ & (b,1,i,\alpha), (b,1,i,\beta), (b,1,j,\alpha), (b,1,j,\beta), (b,1,k,\alpha), (b,1,k,\beta), \\ & (b,2,i,\alpha), (b,2,i,\beta), (b,2,j,\alpha), (b,2,j,\beta), (b,2,k,\alpha), (b,2,k,\beta), \\ & (c,1,i,\alpha), (c,1,i,\beta), (c,1,j,\alpha), (c,1,j,\beta), (c,1,k,\alpha), (c,1,k,\beta), \\ & (c,2,i,\alpha), (c,2,i,\beta), (c,2,j,\alpha), (c,2,j,\beta), (c,2,k,\alpha), (c,2,k,\beta)\} \end{aligned}$$

Exemplo 2.9

Sejam $A = \{a, b, c\}$, $B = \{1, 2\}$, $C = \{i, j, k\}$ e $D = \{\}$ o produto cartesiano de A, B, C e D é dado por:

$$A \times B \times C \times D = \emptyset$$

A notação A^n denota o conjunto formado pelo o produto cartesiano do conjunto A com ele mesmo n vezes.

[6] Também é comum vermos a nomenclatura *t-upla* ordenada — para t componentes, ao invés de n.

$$A^n = \underbrace{A \times \ldots \times A}_{n \text{ vezes}}$$

Assim, podemos também dizer que o conjunto A^n é o conjunto de todas as n-tuplas ordenadas $(a_1, a_2, \ldots, a_n)$ tais que $a_1 \in A, a_2 \in A, \ldots,$ e $a_n \in A$.

Exemplo 2.10

Seja $A = \{a, b, c\}$. Então, temos:

$$A^2 = \{(a,a), (a,b), (a,c), (b,a), (b,b), (b,c), (c,a), (c,b), (c,c)\}$$

2.3.8 Identidades sobre Conjuntos

Temos também as seguintes identidades para todos os conjuntos A, B e C.

1. Associatividade
   a) $(A \cap B) \cap C = A \cap (B \cap C)$
   b) $(A \cup B) \cup C = A \cup (B \cup C)$
2. Distributividade
   a) $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
   b) $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
3. Comutatividade
   a) $A \cap B = B \cap A$
   b) $A \cup B = A \cup B$
4. Elemento Neutro
   a) $A \cap \mathbb{U} = A$
   b) $A \cup \emptyset = A$
5. Elemento Absorvente
   a) $A \cap \emptyset = \emptyset$
   b) $A \cup \mathbb{U} = \mathbb{U}$
6. Idempotência
   a) $A \cap A = A$
   b) $A \cup A = A$
7. De Morgan
   a) $(A \cup B)^{\complement} = A^{\complement} \cap B^{\complement}$
   b) $(A \cap B)^{\complement} = A^{\complement} \cup B^{\complement}$
8. Absorção
   a) $A \cap (A^{\complement} \cup B) = A \cap B$
   b) $A \cup (A^{\complement} \cap B) = A \cup B$

2.4 Conjuntos Finitos e Infinitos

Um conjunto pode ter um número finito ou infinito de elementos. A definição formal de conjuntos finitos e infinitos está fora do escopo desta introdução, para os nossos propósitos uma definição informal será suficiente. Informalmente, temos:

1. *Conjunto finito* – **pode** ser denotado por extensão, ou seja, listando-se exaustivamente todos os elementos do conjunto.
2. *Conjunto infinito* – **não** pode ser denotado por extensão.

Exemplo 2.11

Conjuntos Finitos e Infinitos.

1. São conjuntos finitos:

a) $\emptyset$

b) $\{a, e, i, o, u\}$

c) $\{\mathbb{Z}\}$

d) $A = \{x \mid x \in \mathbb{N} \text{ e } x > 0 \text{ e } x < 4\}$

e) $B = \{x \mid x \text{ é brasileiro}\}$

2. São conjuntos infinitos:

a) $\mathbb{Z}$

b) $\mathbb{R}$

c) $\{x \mid x \in \mathbb{Z} \text{ e } x \geq 7\}$

d) Pares $= \{y \mid y = 2x \text{ e } x \in \mathbb{N}\}$

e) Ímpares $= \{y \mid y = 2x + 1 \text{ e } x \in \mathbb{N}\}$

2.5 Relações

> "Quando dois objetos, qualidades, classes ou atributos, vistos juntos pela mente, são vistos sob alguma conexão, essa conexão é chamada de relação."
>
> – Augustus De Morgan

Intuitivamente, uma relação expressa a ideia de que elementos de dois ou mais conjuntos estão associados entre si. Como exemplos de relações podemos citar, entre outros: (a) cidadãos brasileiros e seus CPF[7]; (b) pessoas de uma mesma família; (c) números de chapas de automóveis e de seus números de chassi; (d) ordenação entre números inteiros; (e) contratante, contratado, e objeto do contrato.

2.5.1 Relações Binárias

No caso mais simples, uma relação envolve elementos de dois conjuntos apenas. Podemos dizer, então, que uma relação associa elementos de um conjunto de origem com elementos de um conjunto de destino. Se uma relação ρ associa elementos do conjunto A como elementos conjunto B, escrevemos $\rho : A \to B$ e chamamos o conjunto A de conjunto de origem (ou partida) e o conjunto B de conjunto de destino (ou chegada).

Numa abordagem um pouco mais formal, dizemos que uma relação ρ de A em B é um subconjunto de $A \times B$. Ou seja, escrever $\rho : A \to B$ é o mesmo que escrever $\rho \subseteq A \times B$. Se, na relação ρ, um elemento a do conjunto de origem está associado a um elemento b do conjunto de destino, então escrevemos que $(a, b) \in \rho$.

Definição 2.13 (Relação Binária)

Dizemos que ρ é uma *relação binária*, ou simplesmente *relação*, do conjunto A, chamado conjunto de origem, para o conjunto B, chamado conjunto de destino, se e somente se ρ é um subconjunto do produto cartesiano $A \times B$ e escrevemos

$$\rho : A \to B$$

É importante notar que, para o nosso estudo, **relações são conjuntos**. Isso significa que tudo o que aprendemos sobre conjuntos também se aplica às relações. É possível, portanto, comparar se duas relações são iguais, se uma relação é um subconjunto de outra, fazer a união ou a interseção de duas relações, e assim por diante. Tudo o que se aplica a conjuntos também se aplica a relações.

Definição 2.14 (Domínio e Contradomínio)

[7] Cadastro de Pessoa Física.

Seja $R : A \to B$ uma relação binária. O *domínio de definição*, ou simplemente *domínio*, da relação R, denotado por $dom(R)$, é definido por:

$$dom(R) = \{a \mid a \in A \text{ e } (a, b) \in R, \text{ para algum } b \in B\}$$

O *contradomínio*, ou *imagem*, da relação R, denotado por $im(R)$, é definido por:

$$im(R) = \{b \mid b \in B \text{ e } (a, b) \in R, \text{ para algum } a \in A\}$$

Também é comum usar-se a notação $a\rho b$ para representar $(a, b) \in R$. Neste texto, por questões de legibilidade, buscaremos usar a notação *infixa* apenas quando a relação for identificada por um símbolo, por exemplo: $a = b, a \leq b, a \odot b, a \rhd b, a\rho b$, etc. E usaremos a notação de conjuntos para os demais casos, por exemplo: $(a, b) \in R, (a, b) \in \text{Ord}, (a, b) \in \Xi$, etc.

Exemplo 2.12 (Relação Binária)

Sejam $A = \{1, 2, 3, 4, 5, 6\}$ e $B = \{2, 4, 6, 8, 10\}$, e seja a relação binária $R : A \to B$ definida como:

$$R = \{(1, 2), (1, 4), (1, 8), (2, 4), (2, 8), (3, 8), (5, 10)\}$$

Então temos:

1. O domínio de R é $dom(R) = \{1, 2, 3, 5\}$.
2. A imagem de R é $im(R) = \{2, 4, 8, 10\}$.

Algumas vezes, para relações binárias simples, é conveniente representar a relação graficamente. A forma mais comum de representação gráfica de relações binárias consistem em usar círculos ou elipses para representar os conjuntos de origem e de destino, com os elementos dos conjuntos representados como pontos com rótulos. As associações entre os elementos do conjunto de origem com o conjunto de destino são representadas por setas indo do elemento de origem para o elemento de destino. Um exemplo dessa forma de representação gráfica é mostrada na Figura 2.8

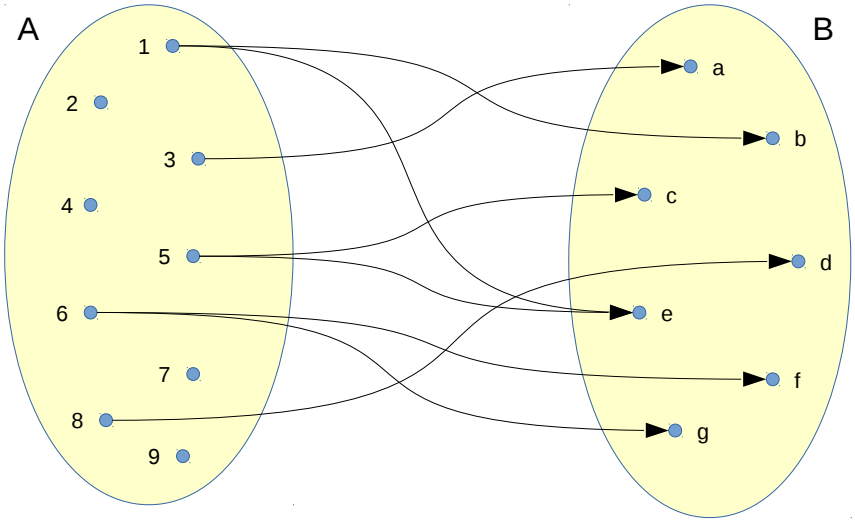


Figura 2.8: Representação gráfica de uma relação binária. O conjunto A é o conjunto de origem e o conjunto B é o conjunto de destino; as setas representam os pares ordenados que compõem a relação.

2.5.2 Propriedades de Relações Binárias

No estudo de relações binárias, muitas vezes é útil classificar as relações de acordo com algumas propriedades. Para relações binárias em geral são classificadas com relação quatro propriedades:

1. Total
2. Funcional
3. Injetora
4. Sobrejetora

Abaixo definimos as principais propriedades de relações binárias.

Definição 2.15 (Relação Total)

Seja a relação binária $R : A \to B$. Dizemos que R é uma relação *total* em A se e somente se R relaciona todos os elementos de A com *ao menos um* elemento de B, ou seja, se $dom(R) = A$.

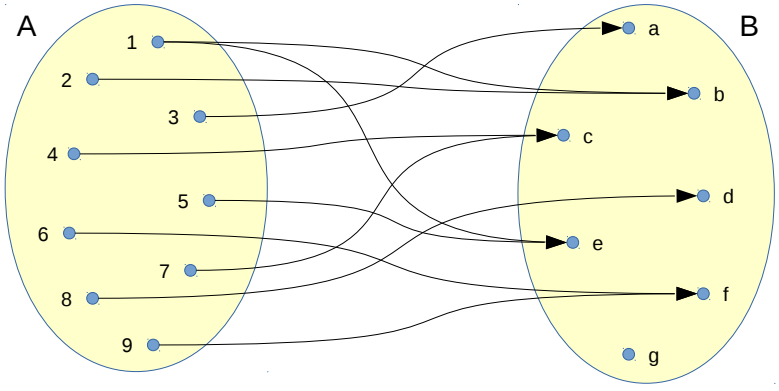


Figura 2.9: Relação total do conjunto A para o conjunto B.

Definição 2.16 (Relação Funcional)

Seja a relação binária $R : A \to B$. Dizemos que R é uma relação *funcional* em A se e somente se R relaciona cada elemento de A com *no máximo um* elemento de B.

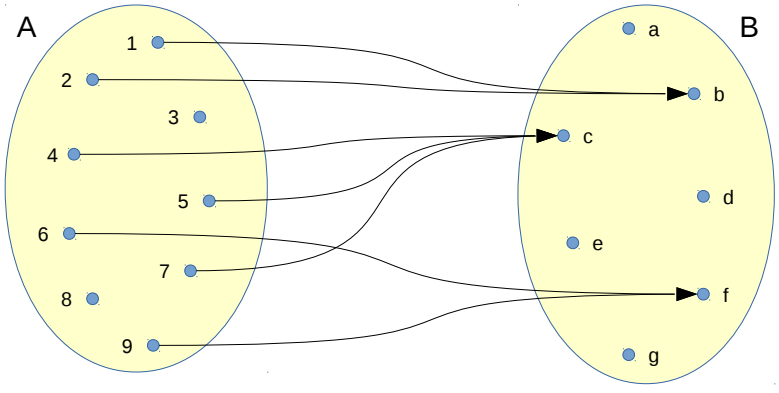


Figura 2.10: Relação funcional do conjunto A para o conjunto B.

Definição 2.17 (Relação Injetora)

Seja a relação binária $R : A \to B$. Dizemos que R é uma relação *injetora* em B se e somente se R relaciona cada elemento de B com *no máximo um* elemento de A.

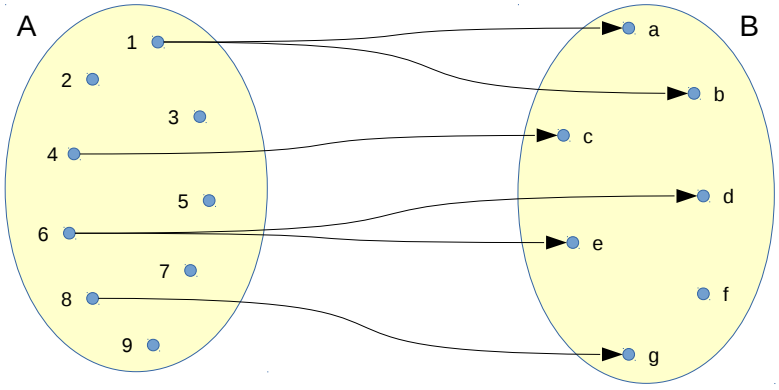


Figura 2.11: Relação injetora do conjunto A para o conjunto B.

Definição 2.18 (Relação Sobrejetora)

Seja a relação binária $R : A \to B$. Dizemos que R é uma relação *sobrejetora* em B se e somente se R relaciona todos os elementos de B com *ao menos um* elemento de A, ou seja, se $im(R) = B$.

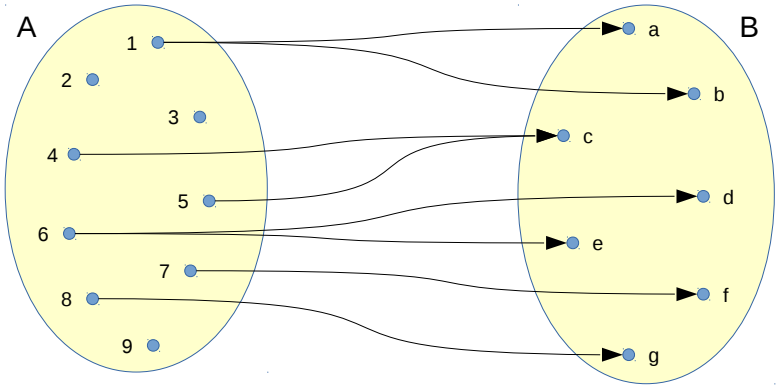


Figura 2.12: Relação sobrejetora do conjunto A para o conjunto B.

Quando uma relação é simultaneamente injetora e sobrejetora, dizemos que esta função é *bijetora.*

Definição 2.19 (Relação Bijetora)

Seja a relação binária $R : A \to B$. Dizemos que R é uma relação *bijetora* se e somente se R é tanto injetora quanto sobrejetora. Ou seja, se R relaciona cada um e todos os elementos de B com *exatamente um* elemento de A.

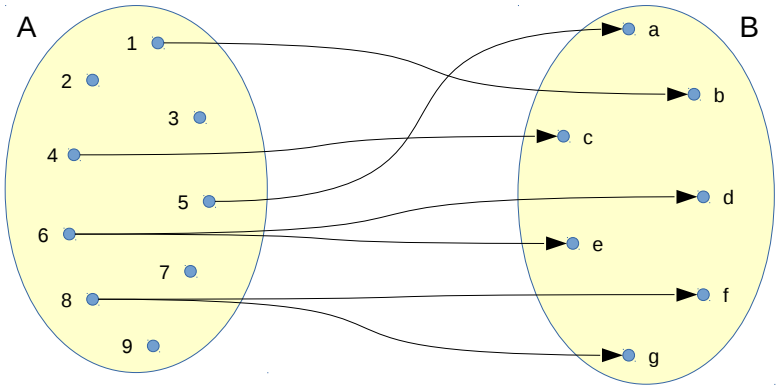


Figura 2.13: Relação bijetora do conjunto A para o conjunto B.

Exemplo 2.13

A seguir estão alguns exemplos de relações binárias familiares juntamente com os suas propriedades.

1. A relação $\leq$ é uma relação de $\mathbb{N}$ para $\mathbb{N}$, que é ao mesmo tempo total e sobrejetora em $\mathbb{N}$. Ou seja, tanto a imagem quanto o domínio da relação $\leq$ em $\mathbb{N}$ são o próprio $\mathbb{N}$.
2. A relação binária que associa seqüências de teclas de um teclado de computador com os respectivos códigos ASCII de 8-bit é um exemplo de uma relação total.
3. A relação binária que associa um código de caracter ISO-8859-1 com o código correspondente de caracter Unicode é um exemplo de uma relação funcional e injetora.

2.5.3 Endorrelações

Em uma relação binária R, quando o conjunto de origem e o conjunto de destino são o mesmo conjunto A, dizemos que R é uma *endorrelação* em A, ou ainda, que R é uma *auto-relação* em A, e escrevemos $R \subseteq A^2$. Várias relações comumente usadas no dia-a-dia são endorrelações. Algumas endorrelações comuns em $\mathbb{N}$, por exemplo, são $=, \neq, <, \leq, >, \geq$. Assim, os elementos do conjunto $\{(0,0), (0,1), (0,2), \ldots (1,1), (1,2), \ldots\}$ são todos membros da relação $\leq$ que é um subconjunto de $\mathbb{N} \times \mathbb{N}$.

De modo geral, uma relação n-ária entre os conjuntos $A_1, A_2, \ldots, A_n$ é um subconjunto do conjunto $A_1 \times A_2 \times \ldots \times A_n$.

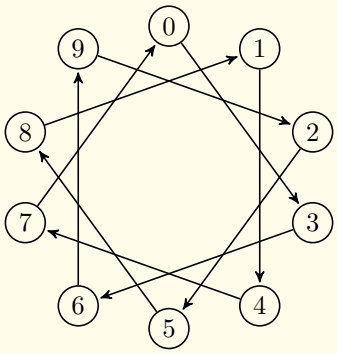
Como vimos na subsection 2.5.1, relações binárias entre dois conjuntos podem ser representadas mostrando-se os dois conjuntos como círculos ou elipses, indicando-se os elementos dentro dos conjuntos e representando os pares ordenados da relação por setas com origem no primeiro elemento do par ordenado e destino no segundo elemento do par. Como mostrado na Figura 2.8.

Essa técnica pode ser utilizada também para endorrelações. Entretanto, para endorrelações, o conjunto de origem e de destino são o mesmo, o que torna a representação de dois conjuntos redundante. Por esta razão, frequentemente as endorrelações são representadas por uma técnica diferente. Desenham-se apenas os elementos do conjunto sobre o qual a endorrelação está definida, sem representar as fronteiras do conjunto. Os pares ordenados que compõem a relação continuam sendo representados por setas com origem no primeiro elemento do par e destino no segundo elemento do par. Para se conseguir uma apresentação gráfica mais clara, costuma-se organizar os elementos em círculo, mas isso não é obrigatório.

Exemplo 2.14

Considere a relação $R \subseteq A$, onde $A = \{0,1,2,3,4,5,6,7,8,9\}$ e $R = \{(x,y) \mid y = (x+3) \mod 10\}$. Com estas

definições temos que a representação gráfica de R é a seguinte:



No exemplo acima, a operação $a \mod b$ representa o resto da divisão inteira de a por b. Por exemplo, $13 \mod 10 = 3$.

2.5.4 Propriedades de Endorrelações

Uma endorelação em um conjunto A pode apresentar algumas propriedades de interesse para o nosso estudo. Considere por exemplo a relação de *igualdade* no conjunto $\mathbb{N}$. Esta relação apresenta três propriedades importantes:

1. Para qualquer número $x \in \mathbb{N}$, $x = x$. Chamamos esta proprieade de *reflexividade*.
2. Para quaisquer números $x, y \in \mathbb{N}$, $x = y$ implica que $y = x$. Chamamos esta propriedade de *simetria*.
3. Para quaisquer números $x, y, z \in \mathbb{N}$, $x = y$ e $y = z$ implica que $x = z$. Chamamos esta propriedade de *transitividade*.

Considere agora a relação *menor-ou-igual* no conjunto $\mathbb{N}$. Essa relação é reflexiva, pois para qualquer $x \in \mathbb{N}$, $x \leq x$. Ela também é transitiva, pois para quaisquer $x, y, z \in \mathbb{N}$, se $x \leq y$ e $y \leq z$, então $x \leq z$. Entretanto, não podemos dizer que essa relação é simétrica, pois se tomarmos dois número quaisquer $x, y \in \mathbb{N}$, **não** podemos afirmar que $x \leq y$ implica em $y \leq x$. De fato, só teremos $x \leq y$ e $y \leq x$ se $x = y$. Esta característica descreve a propriedade que chamamos de *antissimetria*.

Assim, de modo geral, endorrelações podem ser classificadas como *reflexivas*, *simétricas*, *antissimétricas* ou *transitivas*.

Definição 2.20 (Relação Reflexiva)

Uma endorrelação $R \subseteq A^2$ é *reflexiva* se, e somente se, todo elemento de A está relacionado a si mesmo. Ou seja, quando $(a, a) \in R$ para todo elemento $a \in A$.

A Figura 2.14 mostra a representação gráfica da endorrelação $R_1 \subseteq A_1$, onde $A_1 = \{2, 3, 4, 5, 6, 7\}$ e $R_1 = \{(x, y) \mid x \text{ divide } y\}$. Como todo número divide a si mesmo, todos os elementos de A_1 aparecem com setas em laço, voltando para si mesmos. Além disso, 2 aparece com setas para 4 e 6, visto que 2 divide 4 e 2 divide 6; e 3 aparece com uma seta para 6, visto que 3 divide 6. Uma vez que todo elemento do conjunto de origem, A, está relacionado a si mesmo, podemos dizer que a endorrelação R_1 é reflexiva.

Definição 2.21 (Relação Simétrica)

Uma endorrelação $R \subseteq A^2$ é *simétrica* se, e somente se, sempre que $(a, b) \in R$, então $(b, a) \in R$, para quaisquer elementos $a, b \in A$.

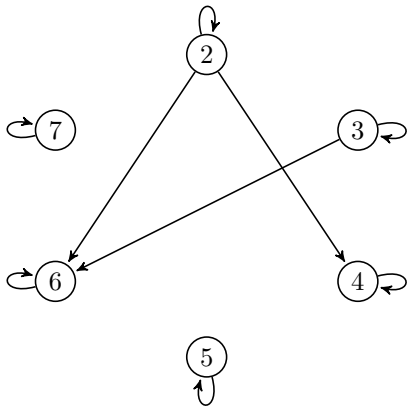


Figura 2.14: Diagrama da endorrelação reflexiva "x divide y" sobre o conjunto $A = \{2, 3, 4, 5, 6, 7\}$.

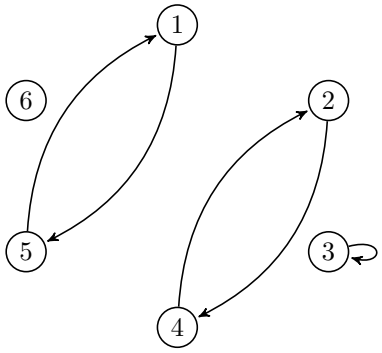


Figura 2.15: Diagrama da endorrelação simétrica "$x + y = 6$" sobre o conjunto $A = \{1, 2, 3, 4, 5, 6\}$.

A Figura 2.15 mostra a representação gráfica da endorrelação $R_2 \subseteq A_2$, onde $A_2 = \{1, 2, 3, 4, 5, 6\}$ e $R_2 = \{(x, y) \mid x + y = 6\}$. Podemos ver que estão representados os pares ordenados $(1, 5)$, $(2, 4)$, $(3, 3)$, $(4, 2)$ e $(5, 1)$. Também podemos notar que para todo par ordenado do tipo (a, b) pertencente à relação, existe o par simétrico, i.e., o par (b, a). Por exemplo, temos o par $(2, 4)$ e temos o simétrico $(4, 2)$. Sendo assim, podemos dizer que a endorelação R_2 é simétrica.

Note também, que apesar da relação R_2 possuir um par ordenado onde um elemento se relaciona a ele mesmo, **não** podemos dizer que R_2 é reflexiva, pois para ser reflexiva, todos os elementos deveriam estar relacionados a si mesmos, e não é o caso.

Definição 2.22 (Relação Antissimétrica)

Uma endorelação $R \subset A^2$ é *antissimétrica* se, e somente se, nunca existirem ambos $(a, b) \in R$ e $(b, a) \in R$, para quaisquer elementos $a, b \in A$. Exceto se $a = b$.

A Figura 2.16 mostra a representação gráfica da endorrelação $R_3 \subseteq A_3$, onde $A_3 = \{1, 2, 3, 4, 5, 6, 7\}$, e $R_3 = \{(x, y) \mid x - y = 3\}$. Podemos observar nesse gráfico que nunca há setas "voltando", ou seja, nunca vemos um par ordenado (a, b) e seu simétrico (b, a). Nessa situação, podemos dizer que a endorrelação R_3 é antissimétrica.

Definição 2.23 (Relação Transitiva)

Uma endorrelação $R \subseteq A^2$ é *transitiva* se, e somente se, sempre que $(a, b) \in R$ e $(b, c) \in R$ então $(a, c) \in R$, para quaisquer elementos $a, b, c \in A$.

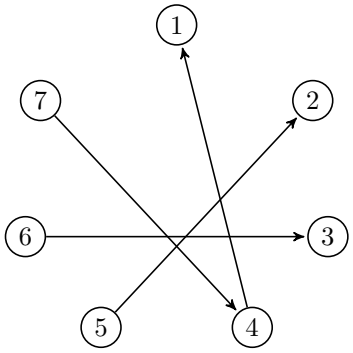


Figura 2.16: Diagrama da endorrelação antissimétrica "$x - y = 3$" sobre o conjunto $A = \{1, 2, 3, 4, 5, 6\}$.

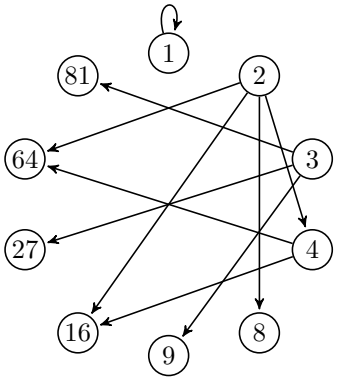


Figura 2.17: Diagrama da endorrelação transitiva "y é potência de x" sobre o conjunto $A_4 = \{1, 2, 3, 4, 9, 16, 27, 64, 81\}$.

A Figura 2.17 mostra a representação gráfica da endorrelação $R_4 \subseteq A_4$, onde $A_4 = \{1, 2, 3, 4, 8, 9, 16, 27, 64, 81\}$ e $R_4 = \{(x, y) \mid y = x^n, n \in \mathbb{N}, n > 1\}$. Podemos observar que sempre que temos dois pares ordenados do tipo (a, b) e (b, c) em R_4, também ocorre o par ordenado (a, c). Por exemplo, tempo $(2, 4)$ e $(4, 16)$ e também temos $(2, 16)$; temos $(2, 8)$ e $(8, 64)$ e também temos $(2, 64)$; temos $(3, 9)$ e $(9, 81)$ e também temos $(3, 81)$; e assim por diante. Nesta situação, podemos dizer que R_4 é uma endorrelação transitiva.

2.5.5 Relações n-árias

Há situações em que as relações envolvem mais de dois conjuntos. Considere, por exemplo, a relação envolvendo três papéis que as pessoas podem desempenhar, expressa em uma declaração da forma "X acha que Y gosta de Z". Os fatos de uma situação concreta poderiam ser organizados em no conjunto de trios ordenado S:

$$\begin{aligned} S = \{&(\text{Alice}, \text{Beto}, \text{Denise}), (\text{Carlos}, \text{Alice}, \text{Beto}), (\text{Carlos}, \text{Carlos}, \text{Alice}), \\ &(\text{Denise}, \text{Denise}, \text{Denise})\} \end{aligned}$$

O conjunto S define uma relação ternária sobre o conjunto P de pessoas em discussão:

$$P = \{\text{Alice}, \text{Beto}, \text{Carlos}, \text{Denise}\}$$

Cada $n$upla de S registra um fato, i.e., faz uma afirmação do formulário "X acha que Y gosta de Z". Por exemplo, o trio $(\text{Alice}, \text{Beto}, \text{Denise})$ diz, na verdade, "Alice acha que Beto gosta de Denise".

A relação S pode ser vista como um exemplo extremamente simples de um banco de dados relacional.

Definição 2.24 (Relações n-árias)

Uma relação L sobre os conjuntos $X_1, \ldots, X_n$ é um subconjunto do seu produto cartesiano, escrito $L \subseteq X_1 \times$

$\cdots \times X_n$.

As relações são classificadas de acordo com o número de conjuntos no produto cartesiano que as define. Por isso:

- $u \in L$ ou $L(u)$ denota uma relação ou propriedade **unária**;
- $(u, v) \in L$, ou $L(u, v)$ ou $u \, L \, v$ denotam uma relação binária; L é uma *relação homogênea* quando $X_1 = X_2$ e uma *relação heterogênea* quando $X_1 \neq X_2$.
- $(u, v, w) \in L$ ou $L(u, v, w)$ denota uma relação **ternária**;
- $(u, v, w, x) \in L$ ou $L(u, v, w, x)$ denota uma relação **quaternária**.

Relações com mais de quatro termos são geralmente referidas como n-árias.

As seguintes considerações se aplicam:

- Os conjuntos X_j para j de 1 a n são chamados de domínios da relação.
- Se todos os domínios X_j são o mesmo conjunto X, então é mais simples referir-se a L como uma relação n-ária sobre X.
- Se algum dos domínios X_j estiver vazio, então o produto cartesiano de definição é vazio, e a única relação sobre essa sequência de domínios é a relação vazia $L = \varnothing$. Por isso é comummente estipulado que todos os domínios não são vazios.
- Se $P(x_1, x_2, \ldots, x_n)$ é um predicado, então

$$\{(a1, a2, \ldots, a_n) \mid P(a_1, a_2, \ldots, a_n) \text{ é verdadeiro}\}$$

é uma relação n-ária.

- Se $f(x_1, x_2, \ldots, x_{n-1}) = x_n$ é uma função, então

$$\{(a_1, a_2, \ldots, a_{n-1}, a_n) \mid f(a_1, a_2, \ldots, a_{n-1}) = a_n\}$$

é uma relação n-ária.

Veremos mais sobre relações n-árias e bancos de dados relacionais na subsection 2.7.2.

2.6 Funções

Uma relação funcional também pode ser chamada de função parcial. Se além de funcional, uma relação f for também uma relação total, dizemos que f é uma *função total* de A para B, usualmente representada por $f : A \to B$.

Definição 2.25 (Função Total)

Uma *função total*, ou simplesmente uma *função*, f de A para B é uma relação binária $f \subseteq A \times B$ tal que para cada elemento $a \in A$ existe um único elemento $b \in B$ de modo que $(a, b) \in f$ (geralmente denotada $f(a) = b$, e por vezes $f : a \mapsto b$).

1. Usaremos a notação $f : A \to B$ para denotar uma função f, de A em B.
2. O conjunto A é chamado de *domínio* da função f.
3. O conjunto B é chamado de *co-domínio* (ou *contra-domínio*) da função f.
4. O *intervalo* de uma função $f : A \to B$ é o conjunto $\{b \mid f(a) = b\}$ para algum $a \in A$.

Alternativamente, podemos definir uma *função total* f de A em B, como uma relação binária funcional total de A em B.

Uma *função parcial* g de A para B, denotada $g : A \nrightarrow B$ é uma função total de algum subconjunto de A para o conjunto B. Evidentemente, toda função total é também uma função parcial.

A Figura 2.18 ilustra o diagrama de uma relação funcional total, i.e., de uma função.

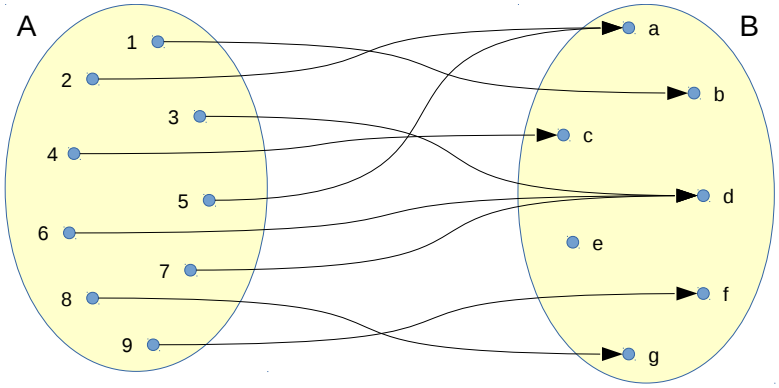


Figura 2.18: Relação funcional e total do conjunto A para o conjunto B. Este tipo de relação é chamado de função.

A palavra "função", salvo disposição em contrário, será usada com o sentido de "função total" – quando quisermos nos referir a "função parcial" usaremos explicitamente este termo.

Exemplo 2.15

Alguns exemplos familiares de funções parciais e totais são:

1. $+$ e $\cdot$ (adição e multiplicação) são funções totais do tipo $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$.
2. $-$ (subtração) é uma função parcial do tipo $f : \mathbb{N} \times \mathbb{N} \nrightarrow \mathbb{N}$.
3. *div* e *mod* são funções totais do tipo $f : \mathbb{N} \times \mathbb{P} \to \mathbb{N}$. Sejam, $a, b, q, r \in \mathbb{N}$, se $a = q \cdot b + r$ tal que $0 \leq r < b$, então as funções *div* e *mod* são definidas como $div(a, b) = q$ e $mod(a, b) = r$. Nós poderemos escrever estas funções como a *div* b e a *mod* b.

   Note que div e mod também são funções parciais do tipo $f : \mathbb{N} \times \mathbb{N} \nrightarrow \mathbb{N}$.
4. As relações binárias $=, \neq, <, \leq, >, \geq$ também pode ser imaginadas como funções totais do tipo $f : \mathbb{N} \times \mathbb{N} \to \mathbb{B}$.

2.6.1 Operações em Relações e Funções

Nesta seção vamos considerar algumas operações sobre relações binárias em geral, incluindo funções.

Em algumas situação é útil ter uma relação que relaciona cada elemento de um conjunto exclusivamente a si mesmo. Essas relações são chamadas de *função identidade*, e sua formalização é dada na Definição 2.26.

Definição 2.26 (Função Identidade)

Dado um conjunto A, a função identidade sobre A, denotada por $id_A : A \to A$, é o conjunto $\{(a, a) \mid a \in A\}$.

O exemplo mais comum de função identidade é a igualdade.

Exemplo 2.16

Seja o conjunto $A = \{2, 3, 5, 7, 11\}$. A função identidade do conjunto A é dada por $id_A = \{(2, 2), (3, 3), (5, 5), (7, 7), (11, 11)\}$.

Outra situação bastante comum, é quando temos uma relação de um conjunto A para um conjunto B, por exemplo, do conjunto de pessoas para o conjunto de números de CPF, e queremos a relação que faz o inverso, que relacione os elementos de B com seus respectivos pares em A. Continuando nosso exemplo, essa relação inversa levaria de números de CPF para pessoas. O conceito de *relação inversa* é formalizado na Definição 2.27.

Definição 2.27 (Relação/Função Inversa)

Dada uma relação binária R de A em B, a relação inversa de R, denotada por R^{-1} é a relação de B em A definido como $R^{-1} = \{(b, a) \mid (a, b) \in R\}$.

Note que, dada uma relação R, sempre existirá a relação inversa, R^{-1}. Entretanto, dada uma função f, nem sempre existirá a **função** inversa, f^{-1}. Observe os exemplos abaixo.

Exemplo 2.17

Seja $A = \{2, 3, 5, 7, 11, 13\}$, e seja a relação $R : A \to A$ definida como $R = \{(x, y) \mid x < y\}$. Então temos que: $R = \{(2, 3), (2, 5), (2, 7), (2, 11), (2, 13), (3, 5), (3, 7), (3, 11), (3, 13), (5, 7), (5, 11), (5, 13), (7, 11), (7, 13), (11, 13)\}$.

Sendo assim, a inversa de R é dada por: $R^{-1} = \{(3, 2), (5, 2), (7, 2), (11, 2), (13, 2), (5, 3), (7, 3), (11, 3), (13, 3), (7, 5), (11, 5), (13, 5), (11, 7), (13, 7), (13, 11)\}$.

Exemplo 2.18

Seja $B = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, e seja $f : B \to B$ definida como $f(x) = (x + 3) \mod 10$. Então temos que: $f = \{(0, 3), (1, 4), (2, 5), (3, 6), (4, 7), (5, 8), (6, 9), (7, 0), (8, 1), (9, 2)\}$.

Sendo assim, a inversa de f é dada por: $f^{-1} = \{(3, 0), (4, 1), (5, 2), (6, 3), (7, 4), (8, 5), (9, 6), (0, 7), (1, 8), (2, 9)\}$.

Exemplo 2.19

Seja $C = \{-1, 0, 1\}$, e seja $g : C \to C$ definida como $g(x) = x^2$. então temos que $g = \{(-1, 1), (0, 0), (1, 1)\}$.

Sendo assim, a inversa de g é dada por: $g^{-1} = \{(1, -1), (0, 0), (1, 1)\}$.

Note que nos três exemplos acima, foi possível determinar a relação inversa. Entretanto, devemos observar a diferença entre os dois últimos exemplos. No caso do Exemplo 2.18, tanto a relação original, f, quanto sua inversa, f^{-1}, são funções. Entretanto, no caso do Exemplo 2.19, embora a relação original, g, seja uma função, a sua inversa, g^{-1}, não é função.

Definição 2.28 (Composição de Relações)

Dadas duas relações binárias R de A em B e S de B em C, a composição de R com S, denotada por $S \circ R$, é definida como:

$$S \circ R = \{(a, c) \mid (a, b) \in R \text{ e } (b, c) \in S, \text{ para algum } b \in B\}$$

A notação $S \circ R$ deve ser lida como "composta de R com S" ou "S-composta-R". Se R é uma relação de A em B, e S é uma relação de B em C, então $S \circ R$ será uma relação de A em C, ou seja, do conjunto de partida da primeira relação para o conjunto de chegada da segunda relação. Para, para que seja possível compor duas relações é necessário que o conjunto de chegada da primeira relação seja o mesmo que o conjunto de partida da segunda relação.

Definição 2.29 (Composição de Funções)

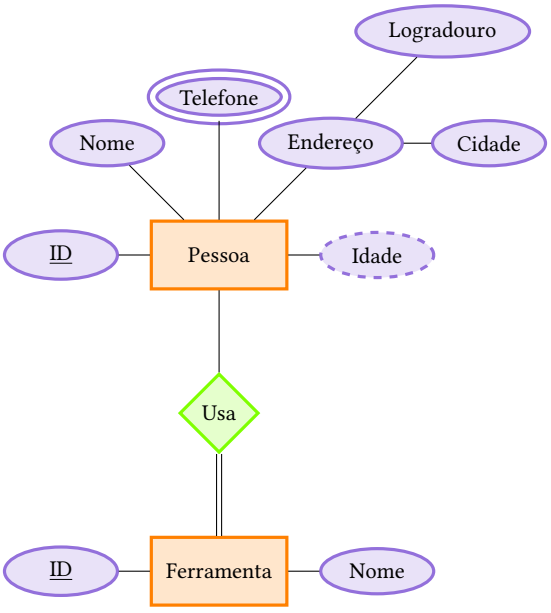


Figura 2.19: Exemplo de diagrama ER

Dadas duas funções (parciais ou totais) $f : A \nrightarrow B$ e $g : B \nrightarrow C$, sua composição é a função $g \circ f : A \to C$ definida simplesmente como a composição relacional das duas funções.

$$(g \circ f) = \{(a, c) \mid f(a) = b \text{ e } g(b) = c, \text{ para algum } b \in B\}$$

Assim, $(g \circ f)(a) = g(f(a))$.

Importante: Note que, embora digamos "função composta de f e g", a notação usada, $g \circ f$, coloca mais à esquerda a função que é aplicada por último (ou mais externa), i.e., em $g(f(x))$ g é aplicada por último, "sobre" o resultado de f, logo, g deve aparecer à equerda.

2.7 Relações e Bancos de Dados

Um banco de dados é um depósito de informações associadas. O usuário de um banco de dados pode recuperar qualquer fato específico armazenado no banco de dados. Mas um banco de dados bem projetado é mais do que simplesmente uma lista de fatos. O usuário pode executar consultas no banco de dados para recuperar informações não contidas em um único fato. O todo se torna mais do que a soma de suas partes.

Para projetar um banco de dados informatizado útil e eficiente, é necessário modelar o sistema do qual o banco de dados trata. Um **modelo conceitual** tenta capturar os recursos e trabalhos importantes do sistema. Pode ser necessária considerável interação com aqueles que estão familiarizados com o sistema real para obter todas as informações necessárias para formular o modelo.

2.7.1 Modelo Entidades-Relacionamentos

Uma representação em alto nível de sistemas é o **modelo entidade-relacionamento** (modelo ER). Neste modelo, objetos importantes no sistemas, ou entidades, são identificados juntamente com suas propriedades importantes, ou atributos. Os relacionamentos entre estas várias entidades também é identificado. Uma outra definição possível é dizer que um modelo ER descreve elementos de interesse inter-relacionados em um domínio de conhecimento específico. Estas informações são representadas graficamente por um **diagrama entidade-relacionamento**, ou diagrama ER. Em um diagrama ER, retângulos representam conjuntos de *entidades*, elipses representam *atributos* e losangos representam *relacionamentos*.

A Figura 2.19 mostra um exemplo de diagrama ER para um domínio bastante simples. Temos um conjunto de entidades Pessoal que se relaciona com um conjunto de entidades Ferramenta através de um relacionamento Usa. Podemos interpretar esta modelagem como significando "pessoas usam ferramentas". Também são mostrados os atributos da entidade Pessoa, i.e., ID, Nome, Telefone, Endereço, e Idade; e os atributos da entidade Ferramenta, i.e., ID, e Nome. Alguns elementos do diagrama merecem atenção especial:

- Os atributos que aparecem sublinhados (ID nas duas entidades) são **chaves**, i.e., servem para identificar unicamente instâncias de entidades no conjunto de entidades.
- A borda dupla no atributo Telefone, na entidade Pessoa, indica que este é um **atributo múltiplo**, ou seja, que uma pessoa pode ter vários telefones.
- O atributo Endereço, na entidade Pessoa, é um **atributo composto**. Ele se divide em dois outros subatributos: Logradouro e Cidade.
- A linha tracejada no atributo Idade, na entidade Pessoa, indica que este é um **atributo derivado**, i.e., que seu valor não é armazenado mas calculado de alguma forma, possivelmente à partir dos valores de outros atributos.
- A linha dupla entre o atributo Usa e a entidade Ferramenta indica que a participação do conjunto entidade Ferramenta no relacionamento é **total**, ou seja, toda instância de Ferramenta estará associada a ao menos uma instância de Pessoa.

Exemplo 2.20

A *Associação Brasileira dos Amigos do Bichinhos de Estimação* (ABABE) que desenvolver uma base de dados. A ABABE adquiriu listas de e-mail de fontes comerciais, e está interessada nas pessoas que possuem animais de estimação e em algumas informações básicas sobre estes animais, tais como nome, tipo do animal (cachorro, gato, etc.), e raça (caso seja aplicável).

A Figura 2.20 mostra o diagrama ER para o domínio da ABABE. Este diagrama diz que pessoas e animais são entidades. Pessoas tem os atributos *Nome*, *Endereço* (rua, nº), *Cidade*, e *UF* (estado). Animais tem os atributos *NomePet*, *Tipo*, e *Raça*. O diagrama também mostra que pessoas possuem animais. Pensando em termos de conjuntos de entidades, i.e., o conjuto Pessoas e o conjunto Animais, o relacionamento "Possui" é uma relação binária de Pessoa para Animal – a relação de propriedade é capturada pelo para ordenado (pessoa, animal). O "1" e o "N" na linhas do relacionamento indicam que esta relação binária é do tipo "um-para-muitos" ou "um-para-N"; ou seja, neste domínio em particular, uma pessoa pode possuir vários animais, mas nenhum animal pode ter múltiplos donos. (Animais com múltiplos donos resultariam em um relacionamento "muitos-para-muitos".) Além disso, neste exemplo, uma pessoa pode não possuir animais, um animal pode não ter dono.

O fato de que nenhum animal pode ter múltiplos donos é uma das "regras de negócio" deste domínio. Identificar as regras de negócio é muito importante, pois elas irão determinar várias características da nossa base de dados, como veremos adiante.

2.7.2 Modelo Relacional

Um outra forma de representar as informações de um domínio de conhecimento específico é através do **modelo relacional**, que pode ser desenvolvido à partir do modelo ER. No modelo relacional tanto as entidades quanto os relacionamentos do modelo ER se tornam *relações* (no sentido matemático do termo). As relações são representadas como tabelas. Um **banco de dados relacional** consiste em uma coleção desta tabelas.

Uma tabela é criada para cada conjunto entidade. Cada linha na tabela contém os valores dos n atributos de cada instância específica do conjunto entidade. Deste modo a tabela pode ser considerada como um conjunto de n-uplas (linhas). Assim como em um conjunto, assume-se que não existem tuplas duplicadas nas tabelas do modelo relacional, e não se assume nenhuma ordenação específica das tuplas na tabela. A ordem dos atributos não é importante, exceto pelo fato de que deve-se manter consistência, ou seja, cada coluna em uma tabela contém valores para um atributo específico em todas as tuplas. O número de atributos (colunas) da tabela é chamado de **grau** da relação. O número de tuplas (linhas) é chamado de **cardinalidade** da relação.

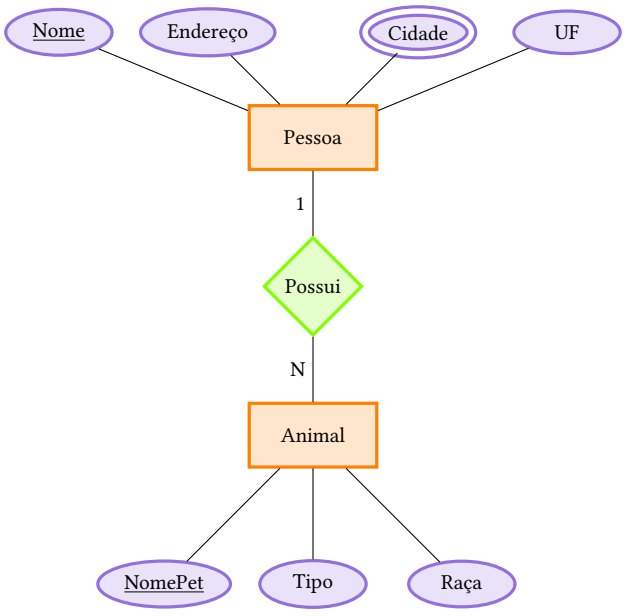


Figura 2.20: Exemplo de diagrama ER

Mais formalmente, uma relação de banco de dados é um subconjunto de $D_1 \times D_2 \times \cdots \times D_n$, onde D_i é o domínio do qual o atributo A_i toma seus valores. Isso significa que o modelo relacional usa o termo *relação* de forma consistem com a definição de relação n-aria dada em subsection 2.5.5.

Exemplo 2.21

A relação *Pessoa* no banco de dados ABABE pode conter dados como os mostrados abaixo.

Pessoa				
Nome	**Sexo**	**Endereço**	**Cidade**	**UF**
Ana Souza	F	Rua Quintino, 123	Vitória	ES
Bruno Trevis	M	Av. dos Desolados, 1021	Linhares	ES
Carlos Umbre	M	Rua Don Quixote, 357	Niterói	RJ
Diana Vicentino	F	Rua Brooklyn, 99	Santos	SP
Elvira Walker	F	Trav. do Viradouro, 56	Itabira	MG
Fausto Xylander	M	Rua João da Cruz, 101	Serra	ES
Gilda Yankov	F	Rua da Revolução, 1917	Vila Velha	ES
Heitor Zoz	M	Av. Antônio Carlos, 67	Rio de Janeiro	RJ

A relação *Animal* poderia ser como a seguir.

Animal		
NomePet	**Tipo**	**Raça**
Mancha	cão	cão de caça
Princesa	gato	siamês
Garotão	cão	collie
Lassie	cão	collie
Moicano	peixe	zanclidae
Piu-piu	pássaro	canário
Tigre	gato	pêlo curto

Como não há duplicação de tuplas em uma relação, indicar o valor de todos os n atributos de uma tupla claramente a distingue de todas as outras. Entretanto, podem existir muitos subconjuntos mínimos de atributos que possam ser usados para distinguir cada tupla. Esses subconjuntos de atributos, caso existam, são chamados de **chave primária**. Se o subconjunto for formado por um único atributo, ele é dito uma *chave primária simples*; se o subconjunto consistir de mais de um atributo, ele é uma *chave primária composta*. Tipicamente, na tabela que descreve a relação, os nomes dos atributos que compõem a chave primária aparecem sublinhados.

Uma outra regra de negócios do domínio da ABABE é que pessoas tem nomes únicos, portanto o atributo *Nome* é suficiente para identificar unicamente cada tupla e foi escolhido como chave primária da relação *Pessoa*. De modo semelhante, o atributo *NomePet* foi escolhido como chave primária na relação *Animal* o que indica deve haver uma regra de negócios que garante que não haverá repetição nos nomes de animais de estimação.

Estas regras de negócio que determinam que não haverá duplicação de nomes são, certamente, simplistas e usadas apenas para propósitos didáticos. Em domínios de conhecimento reais, tipicamente utiliza-se atributos numéricos tais como CPF, número de passaporte, ou algum outro para identificar pessoas pois há garantia de que não haverá repetição. No caso dos animais de estimação não há um identificador como estes, assim, em um cenário mais realista, seria necessário criar um atributo "artificial" único para cada animal de estimação para ser usado como chave primária. Esta chave primária não teria nenhum correspondente no domínio de conhecimento real associado ao sistema, assim o usuário do banco de dados nunca precisaria tomar conhecimento dela; este tipo de chave é chamada de *chave substituta*.[8]

Tipicamente, as implementações de bancos de dados relacionais assumem que cada domínio D_i de atributo em uma base de dados relacional contém – além de seus valores "reais" – um valor especial *nulo*, de modo que uma dada tupla pode ter um ou mais de seus atributos com valor nulo (vazio). Entretanto, nenhum atributo da chave primária pode ser nulo. Essa restrição de **integridade de entidade** meramente confirma que cada tupla deve ter um valor de chave primária que a distingua de todas as outras tuplas, e que todos os atributos da chave primária são necessários para identificar a tupla unicamente.

Um atributo em uma relação (chamada relação "filha") pode ter o mesmo domínio que o atributo de chave primária em outra relação (chamada relação "pai"). Este atributo é chamado de **chave estrangeira** (da relação pai) na relação filha. A relação (no modelo relacional) que representa um relacionamento entre duas entidades (no modelo entidade-relacionamento) usa chaves estrangeiras para estabelecer conexões entre estas entidades. Haverá uma chave estrangeira na relação de relacionamento pra cada entidade que participe do relacionamento.

Exemplo 2.22

A base de dados da ABABE identificou as seguintes instâncias para o relacionamento *Possui*. O atributo *Nome* em *Possui* é uma chave estrangeira da relação *Pessoa*; o atributo *NomePet* em *Possui* é uma chave estrangeira de *Animal*.

Possui	
NomePet	**NomePet**
Fausto Xylander	Mancha
Gilda Yankov	Princesa
Ana Souza	Garotão
Heitor Zoz	Lassie
Bruno Trevis	Piu-Piu
Carlos Umbre	Tigre

A primeira tupla estabelece o relacionamento *Possui* entre Fausto Xylander e Mancha; ou seja, ela indica que Fausto Xylander é dono de Mancha.

Pessoas que não possuem animais de estimação não são representadas em *Possui*, nem são representados os animais de estimação que não tem dono. A chave primária da relação *Possui* foi definida como sendo o atributo *NomePet*. Recorde a regra de negócio que estabelece que um animal de estimação não pode ter múltiplos donos. Se fosse possível um animal de estimação ter múltiplos donos precisaríamos usar uma chave primária composta com *Nome* e *NomePet* para identificar as tuplas da relação *Possui*.

[8] https://en.wikipedia.org/wiki/Surrogate_key

Algumas vezes não é necessário definir uma tabela separada para representar um relacionamento, como fizemos no exemplo anterior. A tabela separada para o relacionamento nunca é necessária se o relacionamento for do tipo um-para-um, e ela pode frequentemente ser removida em relacionamentos do tipo um-para-muitos como o do nosso exemplo.

Exemplo 2.23

Como cada animal de estimação só pode ter um dono (ou nenhum), podemos guardar o nome do dono diretamente na tupla que já armazena as outras informações do animal. Desta forma não seria necessário criar uma tabela separada para representar o relacionamento, ele já estaria representado na relação que representa o animal. Vamos chamar a relação que contém as informação do animal e o nome do dono de *AnimalDono*.

AnimalDono			
Nome	**NomePet**	**Tipo**	**Raça**
Fausto Xylander	Mancha	cão	cão de caça
Gilda Yankov	Princesa	gato	siamês
Ana Souza	Garotão	cão	collie
Heitor Zoz	Lassie	cão	collie
(null)	Moicano	peixe	zanclidae
Bruno Trevis	Piu-piu	pássaro	canário
Carlos Umbre	Tigre	gato	pêlo curto

A relação *AnimalDono* substitui tanto a relação *Animal* quanto a relação *Possui* sem perda de informação. Note que a relação *AnimalDono* possui uma tupla com o valor (null) no atributo *Nome*. Isto permite que animais sem dono sejam representados na relação e não viola a integridade da entidade pois o atributo *Nome* não faz parte da chave primaria.

2.7.3 Álgebra Relacional

A **álgebra relacional** é uma álgebra derivada da álgebra de conjuntos e da lógica de primeira ordem. Ela é uma álgebra com uma semântica (significado) bem fundamentada usada para modelar dados armazenados em bases de dados relacionais, e para definir consultas nestas bases de dados. A principal aplicação da álgebra relaciona é fornecer a fundamentação teórica para as bases de dados relacionais, particularmente para as linguagens de consulta, dentre as quais a principal é a linguagem SQL.

A álgebra relacional foi descrita pela primeira vez por E. F. Codd em 1970, enquanto ele trabalhava para a IBM. As cinco operações primitivas da álgebra relacional descrita por Codd são a *seleção*, a *projeção*, o *produto cartesiano*, a *união*, e *diferença* de conjuntos. Mais tarde também foi acrescentada a operação de *renomeação*. Há outras operações possíveis, e bastante úteis, tais como *junção natural*, *junção-θ*, *equijunção*, *semijunção* e *divisão*. Mas estas podem ser derivadas à partir das operações primitivas, então nós não iremos discutí-las (exceto pela junção-θ, como falaremos adiante).

Para uma abordagem mais detalhada sobre a álgebra relacional e suas operações, veja a página Álgebra relacional da Wikipédia.

A seguir teremos uma breve visão da álgebra relacional, de suas operações primitivas. Também falaremos da operação de junção-θ.

Operadores de Conjuntos

A álgebra relacional usa as operações de união, diferença e produto cartesiano da teoria dos conjuntos, mas adiciona algumas restrições a essas operações.

Para união e a diferença de conjuntos, só é possível realizar as operações se as duas relações envolvidas forem **compatíveis para união** – isto é, as duas relações devem ter os mesmos atributos, na mesma ordem. Uma vez que, na álgebra relacional, a interseção de conjuntos é definida em termos da união e da diferença; na interseção também é necessário que as duas relações envolvidas sejam compatíveis.

Para definir o produto cartesiano, as duas relações envolvidas devem ter cabeçalhos disjuntos – isto é, as duas relações não podem ter atributos com os mesmos nomes.

Na álgebra relacional o produto cartesiano é definido de maneira ligeiramente diferente da forma como ele é definido em teoria dos conjuntos.

- Na teoria dos conjuntos, os elementos do dois conjuntos envolvidos no produto formam um par ordenado.
- Na álgebra relacional, os dois conjuntos envolvidos no produto são relações e seus elementos são tuplas. Ao efetuar o produto, ao invés de parear as tuplas elas são **concatenadas**.

Isto é, o produto cartesiano de um conjunto de n-uplas com um conjunto de m-uplas resulta um conjunto $(n+m)$-uplas. Mais formalmente, $R \times S$ é definido como segue:

$$R \times S = \{(r_1, r_2, \ldots, r_n, s_1, s_2, \ldots, s_m) \mid (r_1, r_2, \ldots, r_n) \in R, (s_1, s_2, \ldots, s_m) \in S\}$$

A cardinalidade do produto Cartesiano é o produto das cardinalidades dos seus fatores, i.e., $|R \times S| = |R| \times |S|$.
O produto cartesiano é, tipicamente, utilizando em conjunto com as operações de seleção e projeção.

Projeção (π)

A projeção é uma operação unária escrita como $\pi_{[a_1,\ldots,a_n]}(R)$ onde $a_1, \ldots, a_n$ é um conjunto de nomes de atributos. O resultado de tal projeção é definida como o conjunto que é obtido quando todas as tuplas em R são restritas ao conjunto de atributos $a_1, \ldots, a_n$.

Isto especifica o subconjunto de colunas (atributos de cada tupla) a ser selecionado. Por exemplo, para obter os nomes e cidades da relação *Pessoa*, a projeção poderia ser escrita $\pi_{[\text{Nome,Cidade}]}(\text{Pessoa})$. O resultado de tal projeção seria uma relação com apenas os atributos *Nome* e *Cidade* para cada entrada única em *Pessoa*.

Note que ao fazer a projeção, duas ou mais tuplas que originalmente eram distintas podem se tornar iguais (pode ser que os atributos que continham valores diferentes sejam removidos na projeção). Caso isso acontece, como o resultado das operações é sempre um conjunto, as duplicatas serão removidas e apenas uma instância permanecerá no resultado.

Seleção (σ)

Uma seleção é uma operação unária escrita como $\sigma_{\varphi}(R)$ onde φ é uma fórmula proposicional que consiste de átomos como é permitido na seleção normal e operadores os lógicos $\wedge$ (e), $\vee$ (ou) e $\neg$ (não). Esta seleção seleciona todas as tuplas em R para as quais o valor de φ é verdadeiro.

Por exemplo, para obter a lista de todas as pessoas do sexo feminino que moram no Espírito Santo, a seleção poderia ser escrita como:

$$\sigma_{\text{Sexo=F} \wedge \text{UF=ES}}(\text{Pessoa})$$

O resultado seria uma relação que tinha todos atributos de todos os registo únicos onde *Sexo* tem valor `F` ou onde o estado é igual a *ES*.

No artigo acadêmico de Codd de 1970, a seleção é chamada de *restrição*.

Renomeação (ρ)

Renomear é uma operação unária escrita como $\rho_{a/b}(R)$ onde o resultado é um conjunto idêntico ao conjunto R exceto pelo fato de que o atributo b em todas as tuplas é renomeado para um atributo a. Isto é usado para mudar o nome do atributo de uma relação ou da própria relação.

Para renomear o atributo Tipo para Espécie na relação *Animal*, podemos escrever a expressão:

$$\rho_{\text{Espécie/Tipo}}(\text{Animal})$$

Se mais do que uma renomeação for necessária, elas podem ser separadas por vírgulas:

$$\rho_{\text{Espécie/Tipo,Variedade/Raça}}(\text{Animal})$$

O operador de renomeação também pode ser usado para mudar o nome da relação. Isso frequentemente é feito quando se precisa usar duas "cópias" da mesma relação em uma expressão. A forma como o operador ρ é usado para mudar o nome das relações é a seguinte:

$$\rho_{\text{Pet}}(\text{Animal})$$

Junção Natural (⋈)

A junção natural (⋈) é uma operação binária escrito como $R \bowtie S$, onde R e S são relações. O resultado da junção natural é o conjunto de todas as combinações de tuplas em R e S que são iguais em seus atributos comuns, i.e., atributos de mesmo nome. Para um exemplo, considere as tabelas *Funcionário* e *Departamento* e sua junção natural:

Funcionário

Nome	FuncId	NomeDepto
Horácio	3415	Finanças
Sônia	2241	Vendas
Jorge	3401	Finanças
Helena	2202	Vendas

Departamento

NomeDepto	Gerente
Finanças	Jorge
Vendas	Helena
Produção	Carlos

Funcionário ⋈ Departamento

Nome	FuncId	NomeDepto	Gerente
Horácio	3415	Finanças	Jorge
Sônia	2241	Vendas	Helena
Jorge	3401	Finanças	Jorge
Helena	2202	Vendas	Helena

A junção natural é, provavelmente, um dos operadores mais importantes, pois é a contraparte relacional da conjunção ("e") lógico. Observe cuidadosamente que se a mesma variável aparecer em cada um dos dois predicados conectados por um "e", então essa variável representa a mesma coisa e ambas as ocorrências devem sempre ser substituídas pelo mesmo valor. Em particular, a junção natural permite a combinação de relações associadas por uma *chave estrangeira*. Por exemplo, no exemplo acima, provavelmente há uma uma chave estrangeira entre os campos *Funcionário.NomeDepto* e *Departamento.NomeDepto*. A junção natural de *Funcionário* e *Departamento* combina todos os funcionários com seus departamentos. Isso funciona porque a chave estrangeira é válida entre os atributos com o mesmo nome. Se esse não for o caso, como na chave estrangeira de *Departamento.Gerente* para *Funcionário.Nome*, temos que renomear essas colunas antes de tomarmos a junção natural. Tal junção é às vezes também referida como uma equjunção (veja junção-θ).

Junção-θ ($\bowtie_\theta$)

A junção-θ entre duas relações R e S, que é escrita $R \bowtie_\theta S$, pode ser vista como uma operação que combina o produto cartesiano e a seleção, da seguinte forma:

$$R \bowtie_\theta S \equiv \delta_\theta(R \times S)$$

Onde θ representa um critério de seleção do tipo $a \odot b$ ou $a \odot v$ em que $\odot$ é um dos operadores relacionais ($<, \leq, =, \neq, \geq, >$); a e b são atributos do produto cartesiano $R \times S$, e v é um valor constante.

Como exemplo, considere a expressão abaixo:[9]

$$\rho_P(\text{Pessoa}) \bowtie_{\text{P.nome=A.nome}} \rho_A(\text{AnimalDono})$$

Essa expressão renomeia a relação *Pessoa* para P e a relação *AnimalDono* para A (apenas para "encurtar" a expressão). Em seguida realiza uma junção-θ entre as relações P e A com critério de seleção P.nome = A.nome. Ou seja, será feito um produto cartesiano entre *Pessoa* e *AnimalDono* e, sobre o resultado do produto cartesiano, será aplicado um filtro para selecionar apenas as tuplas onde o *Nome* que "veio" de *AnimalDono* seja igual ao *Nome* que "veio" de *Pessoa*.

O mesmo resultado poderia ser obtido pela expressão abaixo:

$$\delta_{\text{P.nome=A.nome}}(\rho_P(\text{Pessoa}) \times \rho_A(\text{AnimalDono}))$$

Onde primeiramente é feito, explicitamente, o produto cartesiano e em seguida é aplicada, também explicitamente, a seleção.

[9] Note que estamos igualando atributos de mesmo nome, logo, poderíamos ter usado uma junção natural ao invés da junção-θ.

2.7.4 Integridade de Bancos de Dados

2.7.5 Exercícios

2.8 Exercícios de Revisão

Exercício 2.1 Para $A = \{1\}$, $B = \{1, 2\}$ e $C = \{\{1\}, 1\}$, marque as afirmações corretas:

1. ( ) $A \subseteq B$
2. ( ) $A \subset B$
3. ( ) $A \in B$
4. ( ) $A = B$
5. ( ) $A \subseteq C$
6. ( ) $A \subset C$
7. ( ) $A \in C$
8. ( ) $A = C$
9. ( ) $1 \in A$
10. ( ) $1 \in C$
11. ( ) $\{1\} \in A$
12. ( ) $\{1\} \in C$
13. ( ) $\emptyset \notin C$
14. ( ) $\emptyset \subseteq C$

Exercício 2.2 Sejam os conjuntos A, B, C, D, E, F e G conforme definido abaixo:

$A = \{1, 2, 3, 4\}$
$B = \{a, b, c, d, e, f\}$
$C = \{1, 2\}$
$D = \{1, 3, 4, a, b\}$
$E = \{\{1\}, 2, \{a, 1\}\}$
$F = \{1, c, d\}$
$G = \{d, e, 2, 3\}$

Considere que o universo de discurso é $\mathbb{U} = A \cup B \cup C \cup D \cup E \cup F \cup G$. Liste os elementos dos seguintes conjuntos.

1. $C \setminus D$
2. $A \cap F$
3. $A \cap B$
4. $C^{\complement} \cap F^{\complement}$
5. $E \cap C$
6. $(C \cup D) \setminus (C \cap D)$
7. $F \cup C$
8. $G^{\complement} \cap C$
9. $A \cap E$
10. $(E \cup B) \cup D$

Exercício 2.3 Determine quais são todos os subconjuntos dos seguintes conjuntos:

1. $A = \{a, b, c\}$
2. $B = \{a, \{b, c\}\}$
3. $C = \{a, \{b, c\}, \{d\}\}$

Exercício 2.4 Sejam $A = \{0, 1, 2, 3, 4, 5\}$, $B = \{3, 4, 5, 6, 7, 8\}$, $C = \{1, 3, 7, 8\}$, $D = \{3, 4\}$, $E = \{1, 3\}$, $F = \{1\}$ e X um conjuntos desconhecido. Em cada um dos itens abaixo, determine quais dos conjunto A, B, C, D, E ou F podem ser iguais a X, para que o que é afirmado no item seja verdadeiro.

1. $X \subseteq A$ e $X \subseteq B$
2. $X \not\subset B$ e $X \subseteq C$
3. $X \not\subset A$ e $X \not\subset C$
4. $X \subseteq B$ e $X \not\subset C$

5. $E \subseteq X$ e $D \subseteq X$

Exercício 2.5 Sejam A, B e C conjuntos tais que $A \subseteq B$ e $B \subseteq C$. Suponha que $a \in A$, $b \in B$, $c \in C$, $d \notin A$, $e \notin B$, $f \notin C$. Quais das seguintes afirmações podemos afirmar como verdadeiras?

1. ( ) $a \in C$
2. ( ) $b \in A$
3. ( ) $c \notin A$
4. ( ) $d \in B$
5. ( ) $e \notin A$
6. ( ) $a \in B$
7. ( ) $f \notin A$
8. ( ) $e \notin C$

Exercício 2.6 Sejam,

$$A = \{x \mid x \in \mathbb{N} \text{ e } x \geq 5\} \qquad B = \{10, 12, 16, 20\} \qquad C = \{x \mid y \in \mathbb{N} \text{ e } x = 2y\}$$

Quais das seguintes afirmações são verdadeiras?

1. ( ) $B \subseteq C$
2. ( ) $A \subseteq C$
3. ( ) $\{11, 12, 13\} \subseteq A$
4. ( ) $\{12\} \in B$
5. ( ) $\{x \mid x \in \mathbb{N} \text{ e } x < 20\} \not\subseteq B$
6. ( ) $\{\emptyset\} \subseteq B$
7. ( ) $B \subset A$
8. ( ) $26 \in C$
9. ( ) $\{12\} \subseteq B$
10. ( ) $5 \subseteq A$
11. ( ) $\emptyset \notin A$
12. ( ) $\emptyset \not\subset A$

Exercício 2.7 Suponha o conjunto universo $\mathbb{U} = \{p, q, r, s, t, u, v, w\}$, bem como os seguintes conjuntos:

$$A = \{p, q, r, s\} \qquad B = \{r, t, v\} \qquad C = \{p, s, t, u\}$$

Determine os seguintes conjuntos:

1. $B \cap C$
2. $A \cup C$
3. $C^{\complement}$
4. $A \cap B \cap C$
5. $B \setminus C$
6. $C \setminus B$
7. $(A \cup B)^{\complement}$
8. $A \times B$
9. $(A \cup B) \cap C^{\complement}$
10. $(A \setminus B)^{\complement} \setminus C^{\complement}$

Exercício 2.8 Determine se os conjuntos abaixo são finitos (F) ou infinitos (I):

1. ( ) Os números inteiros maiores que 10.
2. ( ) $\{2, 3, 5, 7, 11, 13, 17, 19, \ldots\}$
3. ( ) Todos os paises da Terra.
4. ( ) Os programas que podem ser escritos em Pascal.
5. ( ) $\{x \mid x \in \mathbb{N},\ x < 2010\}$
6. ( ) $\{x \mid x \in \mathbb{N},\ x \in \text{Pares}\}$
7. ( ) $\{(x, y) \mid x, y \in \mathbb{R},\ 3x + 5y = 0\}$

8. ( ) $\{(x, y) \mid x, y \in \mathbb{Z},\ 10x^2 + 10y^2 = 0\}$

Exercício 2.9 Marque os conjuntos que são alfabetos.

1. ( ) Conjunto dos números naturais
2. ( ) Conjunto dos números primos
3. ( ) Conjunto das letras latinas: $\{a, b, c, d, \ldots, x, y, z\}$
4. ( ) Conjunto dos algarismos arábicos
5. ( ) Conjunto dos algarismos romanos
6. ( ) Conjunto $\{a, b, c, d, e\}$
7. ( ) Conjunto $\{a, i, u, e, o\}$
8. ( ) Conjunto das letras gregas

Exercício 2.10 Sejam $L = \{a, b, c, d, \ldots, x, y, z\}$ e $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ dois alfabetos. Então:

1. Descreva o conjunto de todas as palavras para cada um dos alfabetos abaixo.
   a) L
   b) D
2. Indique se as afirmativas abaixo são verdadeiras ou falsas e tente justivicar sua resposta.
   a) Português é uma linguagem sobre L, ou seja, é um subconjunto de L^*.
   b) $\mathbb{N}$ é uma linguagem sobre D, ou seja, é um subconjunto de D^*.
   c) $\mathbb{N} = D^*$. *Dica*: Em $\mathbb{N}$ e em D^* as palavras 001 e 1 tem significados iguais ou diferentes?

Exercício 2.11 Sejam $A = \{2, 3, 4, 5\}$ e $B = \{3, 4, 5, 6, 10\}$. Para cada uma das relações $R_i \subseteq A \times B$ indicadas abaixo determine:

- Os elementos (pares ordenados) da relação;
- O domínio da relação;
- A imagem da relação;

1. $R_1 = \{(x, y) \mid x \text{ é divisível por } y\}$
2. $R_2 = \{(x, y) \mid x \cdot y = 12\}$
3. $R_3 = \{(x, y) \mid x = y + 1\}$
4. $R_4 = \{(x, y) \mid x \leq y\}$

Exercício 2.12 Para cada uma das relações $\rho_i \subseteq \mathbb{N}^2$ indicadas abaixo determine os elementos (pares ordenados) da relação. *Nota*: Caso o número de pares seja muito grande (ou infinito) represente ao menos os primeiros 5 e os 2 últimos pares (caso haja). Por exemplo $\{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5), \ldots, (x_{n-1}, y_{n-1}), (x_n, y_n)\}$.

1. $\rho_1 = \{(x, y) \mid x + y < 9\}$
2. $\rho_2 = \{(x, y) \mid x = y + 3\}$
3. $\rho_3 = \{(x, y) \mid 2x + 3y = 12\}$
4. $\rho_3 = \{(x, y) \mid y \text{ é um cubo perfeito}\}$[10]

[10] Um cubo perfeito é um número cuja raíz cúbica é um número inteiro.

Exercício 2.13 Para cada uma das relações ρ_i definidas abaixo determine se a relação é reflexiva, simétrica, transitiva ou anti-simétrica. *Nota*: Uma relação pode possuir mais de uma destas propriedades simultaneamente. Indique todas as propriedades apropriadas.

1. $\rho_1 \subseteq \mathbb{N}^2$;
   $\rho_1 = \{(x, y) \mid x + y \text{ é par}\}$
2. $\rho_2 \subseteq \mathbb{P}^2$;
   $\rho_2 = \{(x, y) \mid x \text{ divide } y\}$
3. $\rho_3 \subseteq \{l \mid l \text{ é uma reta no plano cartesiano}\}^2$;
   $\rho_3 = \{(x, y) \mid x \text{ é paralela a } y \text{ ou } x \text{ é coincidente a } y\}$
4. $\rho_4 \subseteq \mathbb{N}^2$;
   $\rho_4 = \{(x, y) \mid x = y^2\}$
5. $\rho_5 \subseteq \{0, 1\}^2$;
   $\rho_5 = \{(x, y) \mid x = y^2\}$
6. $\rho_6 \subseteq \{1, 2, 3\}^2$;
   $\rho_6 = \{(1, 1), (2, 2), (3, 3), (1, 2), (2, 1)\}$
7. $\rho_7 \subseteq \{p \mid p \text{ é uma pessoa}\}^2$;
   $\rho_7 = \{(x, y) \mid x \text{ é mais velho que } y\}$
8. $\rho_8 \subseteq \{p \mid p \text{ é uma pessoa}\}^2$;
   $\rho_8 = \{(x, y) \mid x \text{ é ao menos tão velho quanto } y\}$
9. $\rho_9 \subseteq \{a \mid a \text{ é um aluno desta turma}\}^2$;
   $\rho_9 = \{(x, y) \mid x \text{ senta na mesma fileira que } y\}$
10. $\rho_{10} \subseteq \{u \mid u \text{ é usuário do sistema}\}^2$;
    $\rho_{10} = \{(x, y) \mid x \text{ tem o mesmo direito de aceso que } y\}$

Exercício 2.14 Sejam $A = \{4, 5, 6, 7, 8\}$ e $B = \{8, 9, 10, 11\}$. E seja $f : A \to B$ definida como $f = \{(4, 8), (5, 8), (6, 9), (7, 9), (8, 10)$

1. Desenhe um diagrama representando o conjunto A e seus elementos, o conjunto B e seus elementos e a função f.
2. Qual o domínio da função f? Qual a imagem de f?
3. Qual a imagem de 5? E de 7?
4. Quais são as imagens inversas (pré-imagens) de 9? E de 10?
5. A função f é injetora? É sobrejetora?

Exercício 2.15 Seja a função $f : \mathbb{R} \to \mathbb{R}$ definida como $f(x) = 2x - 1$, determine $f(A)$ para cada um dos casos abaixo.

1. $A = \{0, 1, 2\}$
2. $A = \{1, 2, 4, 5\}$
3. $A = \{\sqrt{7}, 1, 5\}$
4. $A = \{x \mid x \in \mathbb{Z} \text{ e } x = 2y \text{ para algum } y \in \mathbb{Z}\}$

Exercício 2.16 Nas definições a seguir determine se a fução indicada está definida para os conjuntos de origem e destino indicados. Quais são funções injetoras? Quais são funções sobrejetoras? Descreva a função inversa para as funções bijetoras.

1. $f : \mathbb{Z} \to \mathbb{N}$, onde $f(x) = x^2 + 1$.

2. $g : \mathbb{N} \to \mathbb{Q}$, onde $g(x) = \frac{1}{x}$.
3. $h : \mathbb{Z} \times \mathbb{N} \to \mathbb{Q}$, onde $h(z, n) = \frac{z}{n+1}$.
4. $f : \{0, 1, 2, 3\} \to \{p, q, r, s\}$, onde $f = \{(0, s), (1, q), (2, r), (3, p)\}$.
5. $g : \mathbb{N} \to \mathbb{N}$, onde $g(x) = 2^x$.
6. $h : \mathbb{N}^2 \to \mathbb{R}^2$, onde $h(x, y) = (y + 1, x + 1)$.
7. $f : \mathbb{Z}^2 \to \mathbb{N}$, onde $f(x, y) = x^2 + 2y^2$.
8. $g : \mathbb{R} \to \mathbb{R}$, onde $g(x) = \frac{1}{\sqrt{x+1}}$.
9. $h : \mathbb{N}^3 \to N$, onde $h(x, y, z) = x + y - z$.
10. $d : \mathbb{R}^4 \to \mathbb{R}$, onde $d(x_1, y_1, x_2, y_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

Capítulo 3

Conjuntos em Python

Sumário

A linguagem de programação Python foi criada no final da década de 1980 e vem ganhando bastante popularidade desde sua criação. Python é considerada uma linguagem de programação orientada a objetos pura, assim como Smalltalk, Eiffel, Ruby e Scala, entre outras. Isso significa que todos os valores em Python são considerados "objetos".[1] Também é possível identificar em Python alguns elementos de linguagens funcionais, tais como compreensão de listas (do inglês *list comprehension*), uma forma limitada de clausuras (do inglês *closure*), e funções de alta ordem.

Em outubro de 2000 foi anunciada a versão 2.0 de Python que incluiu várias características novas, tais como gerenciamento automático de memória completo e suporte a texto em codificação Unicode.

Python 3.0, uma grande reformulação da linguagem, foi lançado em Dezembro de 2008. A versão 3.0 de Python não é totalmente compatível com as versões anteriores da linguagem, o que significa que um programa escrito em Python 2.x, por exemplo, pode não executar em Python 3.x e vice-versa. Do ponto de vista de projeto e teoria de linguagens

[1] Por mais frustrante que seja para os alunos iniciantes, neste texto não vamos explicar o que são objetos ou entrar em detalhes sobre os vários paradigmas de programação. Vamos apenas usar o termo "objeto" como um conceito fundamental e deixar as explicações mais aprofundadas para cursos mais avançados de programação.

de programação, Python 3 é considerado uma linguagem superior às suas versões antecessoras, dando um tratamento mais uniforme às estruturas sintaticas da linguagem.

A resolução de problemas não-triviais requer que se tenha as estruturas de dados adequadas para o uso com seus algoritmos. Uma das razões para a popularidade da linguagem Python é, certamente, a diversidade de tipos e estruturas de dados disponíveis na linguagem. Python possui listas, tuplas, conjuntos, dicionários, filas, e muitos outros tipos de dados nativos da linguagem. Além disso, através de bibliotecas como NumPy, SciPy e Pandas, por exemplo, é possível incorporar novos tipos de dados à linguagem.

No restante desta seção trataremos de como a linguagem Python lida com conjuntos e quais a limitações deste tratamento. Evidentemente, uma limitação que é inerente a **todas** as linguagens de programação, é o fato de que não se pode representar conjuntos infinitos em um computador de memória finita. Assim, nossa abordagem tratará apenas de conjuntos finitos.

3.1 Entrando Código em Python

Antes de começarmos a falar sobre conjuntos, relações e funções em Python, é necessário falarmos um pouco sobre como escrever código fonte Python e como executar esse código fonte. Vamos assumir neste texto, que você tem alguma experiência com o uso de computadores pessoais, mas que talvez não tenham experiência prévia com Python.

Nosso objetivo aqui é apresentar uma introdução à lógica, e talvez aos fundamentos da matemática discreta, e neste capítulo especificamente, procuramos mostrar como usar a linguagem Python para praticar os conceitos sobre conjuntos, relações e funções que vimos no Capítulo 2.

É claro que uma introdução completa à linguagem Python, ou a programação de modo geral, está além do escopo deste livro. Caso você queira conhecer um pouco mais sobre a linguagem Python do que é mostrado neste capítulo, o Apêndice D apresenta um mini tutorial de Python, com descrições breves da sintaxe dos principais elementos da linguagem. Para aqueles que desejem textos mais completos e mais aprofundados sobe a linguagem Python, há inúmeros recursos disponíveis, tanto pagos quanto gratuitos.

Existem várias formas de executar código em Python, a Seção D.2 apresenta algumas destas formas. Neste capítulo, vamos assumir que todas as expressões e comando em Python serão entradas através do interpretador de comandos *REPL* do Python. A sigla REPL vem de *Read Eval Print Loop*, ou "laço de leitura avaliação e impressão", em português. O *REPL* do Python pode ser executado através do comando `python` no prompt de comando do seu sistema operacional. Após a execução do comando `python` será exibida a apresentação do *REPL*, semelhante à mostrada abaixo:

```
Python 3.5.2 (default, Sep 10 2016, 08:21:44)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

O texto "`>>>`" mostrado na última linha é chamado de *prompt*. Ele indica que o *REPL* está pronto para ler uma nova expressão. Após ler a expressão, o *REPL* irá avaliar a expressão e imprimir o seu resultado.

```
>>> 1 + 1
2
>>> 5 > 4
True
>>> print("Ola, mundo!")
Ola, mundo!
```

Nos excertos de código que iremos apresentar, como o mostrado acima, não vamos incluir a apresentação do *REPL*, vamos começar sempre com o prompt.

Normalmente, uma expressão em Python, termina no final da linha, mas há exceções:

- Se uma expressão for muito grande e você quiser continuá-la na próxima linha, pode incluir uma contrabarra, "\" no final da linha.

```
>>> 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 \
... + 11 + 12 + 13 + 14 + 15 + 16 + 17 + 18 + 19 + 20 \
... + 21 + 22 + 23 + 24 + 25 + 26 + 27 + 28 + 29 + 30
465
```

- Se a expressão em questão for uma tupla, uma lista, um dicionário ou qualquer outro tipo de coleção (veja a Seção D.6), a expressão só irá terminar quando terminar a coleção.

```
>>> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
... 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
... 20, 21, 22, 23, 24, 25, 26, 27, 28, 29
... ]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]
```

- Expressões que envolvem blocos, como os condicionais e os lações (veja a Seção D.4), tem os blocos delimitados pela endentação, i.e., pela quantidade de espaços em branco que precedem as expressões. A expressão só terminará quando o bloco for terminado. No *REPL*, marca-se o fim do bloco por uma linha em branco.

```
>>> for i in [1,2,3,4,5]:
...     j = 2*i
...     print(i, j)
...
1 2
2 4
3 6
4 8
5 10
```

Note que quando uma expressão ocupa mais de uma linha, o prompt do *REPL*, na segunda linha em diante, muda para "`...`".

3.2 Definição de Conjuntos

Desde a versão 2.6, Python suporta conjuntos como tipos de dados pré-definidos, através dos tipos `set` e `frozenset`. Um conjunto em Python é uma coleção não-ordenada de objetos. Isso significa que, diferentemente de listas e tuplas, um conjunto não pode ser indexado, ou seja, não é aceito pela linguagem Python que tente acessar os elementos de um conjunto por um índice. Conjuntos não podem ter elementos duplicados, ou seja, um objeto qualquer aparece exatamente 0 ou 1 vez em um conjunto. Além disso, todos os elementos de um conjunto precisam ser imutáveis[2] Números inteiros, números em ponto flutuante, tuplas, e strings, por exemplo, são objetos imutaveis. Por outro lado, listas, dicionários, e conjuntos (`sets`) não são imutáveis e portanto não podem ser elementos de um conjunto em Python.

A restrição de que um conjunto não pode ser elemento de outro conjunto constitui um problema sério, pois há vários algoritmos que utilizam conjuntos de conjuntos em seu processamento. Para contornar essa dificuldade a linguagem Python suporta um tipo de dados chamado `frozenset`, que representam conjutnos imutáveis, ou seja, uma vez que o conjunto tenha sido criado, não é possível alterar que objetos compõem o conjunto. Assim, um `frozenset` pode ser elementos de um conjunto, e pode também ter outros `frozensets` como seus elementos.

Uma forma de construir conjuntos em Python é passar um objeto qualquer que represente uma sequência para o construtor `set`. Por exemplo:

[2]Na verdade, a definição oficial diz que o objeto precsa ser *hashable*, ou seja, tem que ser possível calcular um valor para a função de *hashing* do objeto e este valor não pode mudar durante todo o tempo de vida do objeto. Na prática, um objeto *hashable* é quase sempre um objeto imutável.

```
>>> set()
set()
>>> set("estudantes")
{'d', 't', 'e', 'n', 'u', 'a', 's'}
>>> set([0,1,2,3])
{0, 1, 2, 3}
>>> set((2,0,1,4))
{0, 1, 2, 4}
>>> set([('a',1), ('b',2), ('c',3), ('d',4)])
{('a', 1), ('c', 3), ('b', 2), ('d', 4)}
>>> set([0, set()])
Traceback (most recent call last):
  File "<pyshell#12>", line 1, in <module>
    set([0, set()])
TypeError: unhashable type: 'set'
```

É importante notar alguns pontos no exemplo acima:

1. A forma de criar um conjuntos vazio ($\emptyset$) é usando o comando `set()`.

2. Caso haja repetições de elementos na sequência passada ao construtor (como as letras repetidas "e" e "s" da palavra "estudantes") estas repetições serão eliminadas.

3. A forma padrão com Python representa conjuntos mutáveis é através do uso de chaves ('{','}').

4. A tentativa de criar um conjunto, em que um dos elementos era um conjunto vazio, gerou um erro.

Para construir conjuntos imutaveis pode-se fazer o mesmo, porém trocando-se o contrutor `set` pelo construtor `frozenset`:

```
>>> frozenset()
frozenset()
>>> frozenset("estudantes")
frozenset({'d', 't', 'e', 'n', 'u', 'a', 's'})
>>> frozenset([0,1,2,3])
frozenset({0, 1, 2, 3})
>>> frozenset((2,0,1,4))
frozenset({0, 1, 2, 4})
>>> frozenset([('a',1), ('b',2), ('c',3), ('d',4)])
frozenset({('a', 1), ('c', 3), ('b', 2), ('d', 4)})
>>> frozenset([0, frozenset()])
frozenset({0, frozenset()})
```

Note que ao tentar criar um conjunto contendo como elementos o zero e o conjunto imutável vazio, não houve erro e o conjunto foi criado corretamente.

No caso de conjuntos mutáveis, ou seja, `sets`, em alguns casos é possível usar uma sintaxe abreviada para a criação de conjuntos, substituindo a chamada ao construtor `set(…)` pelo uso de chaves ('{', '}'). Por exemplo:

```
>>> {1, 2, 3, 4}
{1, 2, 3, 4}
>>> {('a', 1), ('c', 3), ('b', 2), ('d', 4)}
{('c', 3), ('a', 1), ('b', 2), ('d', 4)}
>>> {'d', 't', 'e', 'n', 'u', 'a', 's'}
{'d', 't', 'e', 'n', 'u', 'a', 's'}
```

Entretanto, esta sintaxe abreviada não pode ser utilizada em todos os casos. A expressão `{}`, por exemplo, não retorna um conjunto vazio, ao invés disso ela retorna um objeto do tipo `dict`, i.e., um dicionário. Também não é possível criar um conjunto com as letras de uma palavra utilizando-se a sintaxe abreviada. Ao invés de obter um conjunto com a letras, o que obteremos é um conjunto com a palavra inteira como 1 elemento.

```
>>> {}
{}
>>> type({})
<class 'dict'>
>>> {"estudantes"}
{'estudantes'}
```

3.3 Compreensão de Conjuntos

Na subseção anterior vimos como definir conjuntos por extensão, ou seja, como definir conjuntos litando todos os seus elementos. Entretanto, como dito anteriormente neste capítulo, frequentemente é mais interessante definir os elementos de um conjunto por intensão. Python possuir um mecanismo muito conveniente para suprir essa necessidade, trata-se da *compreensão de conjuntos*.

Talvez a melhor forma de explicar compreensão de conjuntos seja começar dando um exemplo de como ela é utilizada:

Exemplo 3.1

Seja $A = \{x \mid x \in \mathbb{N} \quad \text{e} \quad 1 \leq x \leq 100\}$, desejamos definir o conjunto B que contenha os elementos de A que são múltiplos de 3. A definição matemática de B é dada por

$$B = \{x \mid x \in A \quad \text{e} \quad x \bmod 3 = 0\}$$

Em Python, as definições equivalentes de A e B seriam:

```
>>> A = frozenset([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
... 15, 16, 17, 18, 19])
>>> B = frozenset({x for x in A if x % 3 == 0})
>>> B
frozenset({0, 18, 3, 6, 9, 12, 15})
```

A segunda linha do Exemplo 3.1 define o conjunto B utilizando compreensão de conjuntos. A compreensão de conjuntos é uma expressão formada por 3 partes, que gera um conjunto. As três partes de uma compreensão de conjuntos são mostradas no esquema abaixo:

$$\{\textit{expr fonte filtro}\}$$

expr Parte obrigatória. Descreve a forma geral dos termos que irão compor o conjunto. Tipicamente envolve ao menos uma variável que será descrita na parte *fonte*, mas o uso dessa variável não é obrigatório. No Exemplo 3.1 a expressão é x, o que significa que cada elemento do conjunto B será formado pelo valor de x.

fonte Parte obrigatória. Uma ou mais ocorrências de uma expressão do tipo "`for` *var* `in` *coleção*". Onde *var* é o nome de uma variável e *coleção* é uma coleção (conjunto, lista, tupla, dicionário, etc.) de dados, de onde será obtidos os dados para a variável. No Exemplo 3.1 a fonte é "**for** x **in** A". O que significa que o valor de x virá do conjunto A, ou seja, que a expressão será avaliada para cada um dos elementos do conjunto A.

filtro Parte opcional. Uma expressão do tipo "**if** *condição*". Para cada grupo de valores gerados pela *fonte* a condição é avaliada. Apenas se a condição for verdadeira é que o elementos correspondente do conjunto é gerado. No Exemplo 3.1 o filtro é "**if** $x\%3 == 0$", o que significa que o item só será gerado se o resto da divisão de x por 3 for igual a zero, ou seja, se x for divisível por 3.

É muito comum, se utilizar compreensões em Python conjuntamente com o contrutor `range`. Este construtor gera uma sequência de números inteiros e pode ser usado de três formas diferentes:

1. **range(n)** — gera a sequência de números de 0 até $n-1$, i.e., $0, 1, 2, \ldots, n-1$.
2. **range(i, j)** — gera a sequência de números de i até $j-1$, i.e., $i, i+1, \ldots, j-1$.
3. **range(i, j, d)** — gera a sequência de números de i até $j-1$, com incrementos de d, i.e., $i, i+d, i+2d, \ldots, j-1$. Se o número $j-1$ não fizer parte da sequência, para no maior número da sequência que seja anterior a $j-1$.

Exemplo 3.2

Crie o conjunto com os números naturais menores que 10.

```
>>> frozenset(range(10))
frozenset({0, 1, 2, 3, 4, 5, 6, 7, 8, 9})
```

Exemplo 3.3

Crie o conjunto com os números naturais menores ou iguais a 10.

```
>>> frozenset(range(11))
frozenset({0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10})
```

Exemplo 3.4

Crie o conjunto com os números inteiro entre -20 e 10, inclusive.

```
>>> frozenset(range(-20,11))
frozenset({0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, -20, -19, -18,
-17, -16, -15, -14, -13, -12, -11, -10, -9, -8, -7, -6, -5,
-4, -3, -2, -1})
```

Exemplo 3.5

Crie o conjunto com os primeiros 10 múltiplos de 7 (use compreensão).

Solução 1:

```
>>> frozenset({x*7 for x in range(10)})
frozenset({0, 35, 7, 42, 14, 49, 21, 56, 28, 63})
```

Solução 2:

```
>>> frozenset({x for x in range(10*7) if x % 7 == 0})
frozenset({0, 35, 7, 42, 14, 49, 21, 56, 28, 63})
```

Exemplo 3.6

Crie o conjunto contendo os quadrados dos números inteiros de 1 a 10.

```
>>> frozenset({x*x for x in range(1,11)})
frozenset({64, 1, 4, 36, 100, 9, 16, 49, 81, 25})
```

Exemplo 3.7

Sejam os conjuntos $A = \{10, 30, 50\}$ e $B = \{20, 40, 60\}$, crie o conjunto C contendo elementos formados à partir da soma dos elementos de A com os elementos de B, ou seja, $C = \{x + y \mid x \in A \quad \text{e} \quad y \in B\}$.

```
>>> A = frozenset({10, 30, 50})
>>> B = frozenset({20, 40, 50})
>>> C = frozenset({x+y for x in A for y in B})
>>> C
frozenset({80, 50, 100, 70, 90, 60, 30})
```

Exemplo 3.8

Dados os conjuntos $A = \{1, 2, 3\}$ e $B = \{a, b, c\}$, gere o conjunto $A \times B$.

```
>>> A = frozenset({1, 2, 3})
>>> B = frozenset("abc")
>>> frozenset({(x,y) for x in A for y in B})
frozenset({(3, 'a'), (1, 'a'), (1, 'c'), (3, 'c'), (2, 'b'),
(3, 'b'), (2, 'c'), (2, 'a'), (1, 'b')})
```

3.4 Relações entre Conjuntos

As relações básicas de conjuntos (pertinência, igualdade, subconjunto) são todas definidas na linguagem Python através de operadores. Suas sintaxes são explicadas abaixo.

3.4.1 Pertinência

Definição 3.1 (Pertinência)

A relação de pertinêcia em Python é implementada pelos operadores `in` e `not in`. Deste modo, para testar se o valor de x pertence a um conjunto S, i.e., $x \in S$, em Python escrevemos `x in S`. De modo similar, para testar se $x \notin S$, escrevemos `x not in S`.

Exemplo 3.9

Escreva os comandos em Python que testem se o elemento 3 pertence ao conjunto $A = \{1, 3, 5, 7\}$.

```
>>> A = frozenset({1, 3, 5, 7})
>>> 3 in A
True
```

Exemplo 3.10

Escreva os comandos em Python que testem se o elemento 3 pertence ao conjunto $B = \{x \mid x \in \mathbb{Z} \quad \text{e} \quad -100 \leq x \leq 100\}$.

```
>>> B = frozenset(range(-100,101))
>>> 3 in B
True
```

Exemplo 3.11

Escreva os comandos em Python que testem se o elemento 7 não pertence ao conjunto dos múltiplos de 3 entre 0 e 100.

```
>>> multi3 = frozenset({x for x in range(101) if x % 3 == 0})
>>> 7 not in multi3
True
```

Exemplo 3.12

Escreva os comandos em Python para verificar que o conjunto das soluções inteiras, entre 0 e 100, da equação $x^3 - 6x^2 + 3x + 10 = 0$ é igual a $\{2, 5\}$.

```
>>> X = frozenset({x for x in range(101) if x**3 - 6*x**2 + 3*x + 10 == 0})
>>> X == frozenset({2, 5})
True
```

3.4.2 Igualdade

Definição 3.2 (Igualdade)

Para testar se dois conjuntos A e B são iguais em Python, escrevemos `A == B`. De modo semelhante, para testar se $A \neq B$ escrevemos `A != B`.

Exemplo 3.13

Escreva os comandos em Python que testem se os conjuntos $S = \{1, 3, 5, 7, 9\}$ e $T = \{x \mid x \in \mathbb{N} \quad \text{e} \quad 0 < x < 10 \quad \text{e} \quad x \bmod 2 = 0\}$ são iguais.

```
>>> S = frozenset({1,3,5,7,9})
>>> T = frozenset({x for x in range(1,10) if x % 2 == 0})
>>> S == T
False
>>> S
frozenset({9, 1, 3, 5, 7})
>>> T
frozenset({8, 2, 4, 6})
```

Exemplo 3.14

Escreva os comandos em Python que testem se os conjuntos $S = \{1, 3, 5, 7, 9\}$ e $T = \{x \mid x \in \mathbb{N} \quad \text{e} \quad 0 < x < 10 \quad \text{e} \quad x \bmod 2 \neq 0\}$ são iguais.

```
>>> S = frozenset({1,3,5,7,9})
>>> T = frozenset({x for x in range(1,10) if x % 2 != 0})
>>> S == T
True
>>> S
frozenset({9, 1, 3, 5, 7})
>>> T
frozenset({1, 9, 3, 5, 7})
```

3.4.3 Subconjunto

Definição 3.3 (Subconjunto)

Para testar se um conjunto A é subconjunto de B, i.e., $A \subseteq B$, em Python escrevemos `A <= B`. Também é possível escrever `A.issubset(B)`, com o mesmo significado.

Exemplo 3.15

Sejam os conjuntos $F = \{1, 2, 3, 5, 8, 13, 21\}$, $P = \{2, 3, 5, 7, 11, 13, 17\}$, $I = F \cap P$ e $J = F \cup P$. Escreva os comandos em Python que testem se $F \subseteq P$, $P \subseteq F$, $I \subseteq F$, $I \subseteq P$, $F \subseteq J$, $P \subseteq J$, $I \subseteq J$, $I \subseteq I$ e $J \subseteq J$.

```
>>> F = frozenset({1, 2, 3, 5, 8, 13, 21})
>>> P = frozenset({2, 3, 5, 7, 11, 13, 17})
>>> I = F & P
>>> J = F | P
>>> F <= P
False
>>> P <= F
False
>>> I <= F
True
>>> I <= P
True
>>> F <= J
True
>>> P <= J
True
>>> I <= J
True
>>> I <= I
True
>>> J <= J
True
```

3.4.4 Subconjunto Próprio

Definição 3.4 (Subconjunto Próprio)

A expressão `A < B`, em Python, significa que A é subconjunto próprio de B, ou seja, `A < B`, significa que `A <= B` e `A != B`.

Exemplo 3.16

Sejam os conjuntos $F = \{1, 2, 3, 5, 8, 13, 21\}$, $P = \{2, 3, 5, 7, 11, 13, 17\}$, $I = F \cap P$ e $J = F \cup P$. Escreva os comandos em Python que testem se $F \subset P$, $P \subset F$, $I \subset F$, $I \subset P$, $F \subset J$, $P \subset J$, $I \subset J$, $I \subset I$ e $J \subset J$.

```
>>> F = frozenset({1, 2, 3, 5, 8, 13, 21})
>>> P = frozenset({2, 3, 5, 7, 11, 13, 17})
>>> I = F & P
>>> J = F | P
>>> F < P
False
>>> P < F
False
>>> I < F
True
>>> I < P
True
>>> F < J
True
>>> P < J
True
>>> I < I
False
>>> J < J
False
```

3.4.5 Superconjunto e Superconjunto Próprio

Os conceitos de *superconjunto* e *superconjunto próprio* são simétricos aos conceitos de subconjunto e subconjunto próprio.

Definição 3.5 (Superconjunto e Superconjunto Próprio)

Dados dois conjuntos A e B, dizemos que A é superconjunto de B, escrito $A \supseteq B$, se todo elemento que pertence a B também pertence a A. De modo análogo, dizemos que A é superconjunto próprio de B, escrito $A \supset B$, se todo elemento de B pertence a A, e existe ao menos um elementos de A que não pertence a B, ou seja, $A \supset B \equiv (A \supseteq B$ e $A \neq B)$.

Estritamente falando, os operadores $\supset$ e $\supseteq$ são desnecessários pois sempre é o caso que $A \supseteq B = B \subseteq A$ e $A \supset B = B \subset A$. Apesar desses operadores não serem necessários, a linguagem Python define os operadores de superconjunto e superconjunto próprio.

Em Python a expressão `A >= B` significa que A é superconjunto de B. e `A > B` significa que A é superconjunto próprio de B.

Exemplo 3.17

```
>>> F = frozenset({1, 2, 3, 5, 8, 13})
>>> E = frozenset({1, 2, 3, 5, 8})
>>> F >= E
True
>>> F > E
True
>>> F >= F
True
>>> F > F
False
>>> E >= E
True
>>> E > E
False
```

3.5 Operações sobre Conjuntos

Python define praticamente todas as operações sobre conjuntos que foram vistas anteriormente neste capítulo, para a maioria das operações de conjunto a linguagem Python define duas sintaxes diferentes, uma utilizando operadores e uma utilizando *métodos*. Apresentaremos as duas sintaxes, mas daremos preferência ao uso da sintaxe com operadores.

3.5.1 Cardinalidade

Definição 3.6 (Cardinalidade)

A quantidade de um conjunto S, ou seja, a quantidade de elementos de S, escrita como $|S|$, é implementada em Python pela a função `len(S)`.

Exemplo 3.18

Sejam os conjuntos $X = \{1, 3, 5, 7, 9\}$, $Y = \{y \mid y = x^2 \quad \text{e} \quad x \in X\}$ e $Z = \{\{x, y\} \mid x \in X \quad \text{e} \quad y \in Y\}$, use a operação `len` do Python para determinar a cardinalidade de cada um dos conjuntos.

```
>>> X = frozenset({1, 3, 5, 7, 9})
>>> len(X)
5
>>> Y = frozenset({x**2 for x in X})
>>> len(Y)
5
>>> Z = frozenset({frozenset({x,y}) for x in X for y in Y})
>>> len(Z)
24
```

Como exercício, tente entender porque a cardinalidade do conjunto Z foi 24 e não 25. Liste os elementos do conjunto Z.

3.5.2 União

Definição 3.7 (União)

A união de dois conjuntos A e B, $A \cup B$, é implementada em Python pelo operador "`|`". Assim a união dos conjuntos A e B é escrita como `A | B` em Python. Alternativamente, também é possível escrever `A.union(B)` para a união.

Exemplo 3.19

Sejam os conjuntos $F = \{1, 2, 3, 5, 8, 13\}$ e $P = \{2, 3, 5, 7, 11, 13\}$. Escreva os comandos em Python que calculam a união de F e P, i.e., $F \cup P$.

```
>>> F = frozenset({1, 2, 3, 5, 8, 13})
>>> P = frozenset({2, 3, 5, 7, 11, 13})
>>> F | P
frozenset({1, 2, 3, 5, 7, 8, 11, 13})
```

3.5.3 Interseção

Definição 3.8 (Interseção)

A interseção de dois conjuntos A e B, $A \cap B$, é implementada em Python pelo operador "&". Assim a união dos conjuntos A e B é escrita como `A & B` em Python. Alternativamente, também é possível escrever `A.intersection(B)` para a interseção.

Exemplo 3.20

Sejam os conjuntos $F = \{1, 2, 3, 5, 8, 13\}$ e $P = \{2, 3, 5, 7, 11, 13\}$. Escreva os comandos em Python que calculam a interseção de F e P, i.e., $F \cap P$.

```
>>> F = frozenset({1, 2, 3, 5, 8, 13})
>>> P = frozenset({2, 3, 5, 7, 11, 13})
>>> F & P
frozenset({2, 3, 5, 13})
```

3.5.4 Diferença

Definição 3.9 (Diferença)

A diferença do conjunto A em relação ao conjunto B, ou simplesmente, diferença entre A e B, $A \setminus B$, é implementada em Python pelo operador "-". Assim a diferenção entre A e B, é escrita em Python como `A - B`. Alternativamente, também é possível escrever `A.difference(B)` para a differença.

Exemplo 3.21

Sejam os conjuntos $F = \{1, 2, 3, 5, 8, 13\}$ e $P = \{2, 3, 5, 7, 11, 13\}$. Escreva os comandos em Python que calculem a diferença entre F e P, i.e., $F \setminus P$, e a diferença entre P e F, i.e., $P \setminus F$.

```
>>> F = frozenset({1, 2, 3, 5, 8, 13})
>>> P = frozenset({2, 3, 5, 7, 11, 13})
>>> F - P
frozenset({8, 1})
>>> P - F
frozenset({11, 7})
```

3.5.5 Diferença Simétrica

Definição 3.10 (Diferença Simétrica)

A diferença simétrica entre os conjuntos A e B, $A\Delta B$, é implementada em Python pelo operador "^". Assim a diferenção simétrica entre A e B, é escrita em Python como `A ^ B`. Alternativamente, também é possível escrever `A.symmetric_difference(B)` para a differença simétrica.

Exemplo 3.22

Sejam os conjuntos $F = \{1, 2, 3, 5, 8, 13\}$ e $P = \{2, 3, 5, 7, 11, 13\}$. Escreva os comandos em Python que calculem a diferença simétrica entre F e P, i.e., $F\Delta P$.

```
>>> F = frozenset({1, 2, 3, 5, 8, 13})
>>> P = frozenset({2, 3, 5, 7, 11, 13})
>>> F ^ P
frozenset({1, 7, 8, 11})
```

Estas operações funcionam tanto para conjuntos mutáveis (`sets`) quanto para conjuntos imutáveis (`frozensets`). Para o caso dos conjuntos mutáveis, há ainda algumas outras operações que alteram o conjunto, mas nós não discutiremos essas funções neste texto.

3.6 Funções Pré-definidas sobre Conjuntos

Além das operações tradicionais sobre conjuntos vistas acima, a linguagem Python também possui diversas outras funções pré-definitas para manipulação de conjuntos. Abaixo são descritas algumas destas funções.

max(S) — retorna o maior valor no conjunto S.

```
>>> S = frozenset({19, 2, 18, 3, 17, 4, 16, 5, 15, 6, 14})
>>> max(S)
19
```

min(S) — retorna o menor valor no conjunto S.

```
>>> S = frozenset({19, 2, 18, 3, 17, 4, 16, 5, 15, 6, 14})
>>> min(S)
2
```

enumerate(S) — enumera os elementos de um conjunto. Essa operação produz uma sequência de tuplas baseada no conjunto S. Cada tupla tem dois elementos: um número sequêncial, indicando a ordem do elemento e um item do conjunto original.

Note que uma vez conjuntos não tem uma ordem definida para os elementos, esta operação pode, em princípio, produzir ordens diferentes para os elementos do conjunto. Ou seja, chamar esta função duas vezes para o mesmo conjunto, não necessariamente irá produzir os mesmos índices para os mesmos elementos.

```
>>> S = frozenset({'azul', 'verde', 'vermelho', 'amarelo', 'laranja'})
>>> list(enumerate(S))
[(0, 'verde'), (1, 'amarelo'), (2, 'vermelho'), (3, 'laranja'),
(4, 'azul')]
```

sorted(S,*key*,*reverse*) — retorna uma sequência (um iterator em Python) com os elementos do conjunto ordenados. Apenas o argumento S é obrigatório. O argumento *key* é uma função que recebe um elementos do conjunto e retorna um inteiro representando a chave de ordenação deste elementos. O argumento *reversed* é do tipo booleano, seu valor padrão é `False`, se for definido como `True`, os elementos serão ordendos na ordem reversa.

```
>>> S = frozenset({19, 2, 18, 3, 17, 4, 16, 5, 15, 6, 14})
>>> sorted(S)
[2, 3, 4, 5, 6, 14, 15, 16, 17, 18, 19]
>>> sorted(S, reverse=True)
[19, 18, 17, 16, 15, 14, 6, 5, 4, 3, 2]
```

sum(S**)** — retrona a soma de todos os elementos do conjunto.

```
>>> A = frozenset({x for x in range(1,101)})
>>> sum(A)
5050
```

all(*teste* **for** *var* **in** S**)** — testa se todos os valores de S fazem a condição *teste* ser verdadeira.

```
>>> N = frozenset({2, 3, 5, 7, 11, 13, 17, 19})
>>> all(x > 1 for x in N)
True
>>> all(x > 10 for x in N)
False
>>> all(x % 2 == 0 for x in N)
False
```

any(*teste* **for** *var* **in** S**)** — testa se algun dos valores de S faz a condição *teste* ser verdadeira.

```
>>> N = frozenset({2, 3, 5, 7, 11, 13, 17, 19})
>>> any(x > 1 for x in N)
True
>>> any(x > 10 for x in N)
True
>>> any(x > 20 for x in N)
False
>>> any(x % 2 == 0 for x in N)
True
```

3.7 Relações em Python

A linguagem Python não possui mecanismos específicos para a definição de relações, mas há formas de conseguir criar essas definições. Uma dessas formas é a utilização de conjuntos, que já vimos, outra é a utilização de *dicionários*, que serão apresentados mais adiante.

3.7.1 Definição de Relações em Python

Como foi dito, relações podem ser definidas em Python como conjuntos de pares ordenados (tuplas em Python), exatamente da forma como vimos a definição de relações na Seção 2.5.

Exemplo 3.23 (Relação como Conjunto de Pares Ordenados)

Sejam $A = \{1, 2, 3, 4, 5\}$ e $B = \{\text{one}, \text{two}, \text{three}, \text{four}, \text{five}\}$. Desejamos definir em Python a relação R que associa cada número em A com a palavra correspondente em B.

```
>>> R = frozenset({(1,'one'), (2,'two'), (3,'three'), (4,'four'),
... (5,'five')})
>>> R
frozenset({(5, 'five'), (3, 'three'), (1, 'one'), (2, 'two'),
(4, 'four')})
```

Note que no Exemplo 3.23, assim como nos outros exemplos em que definimos conjuntos, ao consultarmos o valor da variável `R`, a ordem dos elementos é diferente da ordem em que eles foram especificados na definição de `R`. É importante relembrar: a ordem interna em que os elementos de um conjunto são armazenados em Python não é determinada pela especificação da linguagem e pode variar de uma execução do código para outra.

Uma vez que definimos nossa relação como conjunto, podemos executar qualquer operação válida para conjuntos em Python. Como, por exemplo, as operações de pertinência, e de subconjunto.

Exemplo 3.24

Continuando com a relação definida no Exemplo 3.23.

```
>>> (2,'two') in R
True
>>> ('one',1) in R
False
>>> frozenset({(3,'three'), (5,'five')}) < R
True
```

Uma desvantagem de definir relações como conjuntos em Python, é o fato de que não é trivial consultar se um elemento do conjunto de origem ou do conjunto de destino fazem parte da relação. Não é possível simplesmente perguntar, por exemplo, se `1 in R`, pois o resultado será falso, embora o 1 apareça na relação. Para realizar o teste desejado precisaremos usar a operação `any`, como mostrado no exemplo abaixo.

Exemplo 3.25

Teste para determinar se um elemento da origem, ou do destino, faz parte da relação.

```
>>> any(a == 1 for (a,b) in R)
True
>>> any(b == 'four' for (a,b) in R)
True
```

Uma outra forma de definir relações é Python é utilizando **dicionários**, também chamados de tabelas associativas. Entretanto os dicionários são menos gerais do que os conjunto. Um dicionário em Python associa uma *chave* a um *valor*, e posteriormente permite que se inquira o valor à partir da chave.

Exemplo 3.26

Exemplo de uso de dicionários em Python.

```
>>> D = {1:'one', 2:'two', 3:'three', 4:'four', 5:'five'}
>>> D
{1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five'}
```

É possível testar se uma determinada chave pertence ao dicionário com o operador `in`, mas não é possível perguntar diretamente se um par ordenado pertence ao dicionário. Entretanto, esta restrição desaparece se usarmos o método `items()`, que retorna os pares ordenados do dicionário. Também existem os métodos `keys()` e `values()`) que retornam as listas de chaves e valores respectivamente.

Exemplo 3.27

Testes para determinar se uma determinada chave ou par ordenado faz parte da relação.

```
>>> 4 in D
True
>>> (4,'four') in D
False
>>> (4,'four') in D.items()
True
>>> D.items()
dict_items([(1, 'one'), (2, 'two'), (3, 'three'), (4, 'four'),
(5, 'five')])
>>> D.keys()
dict_keys([1, 2, 3, 4, 5])
>>> D.values()
dict_values(['one', 'two', 'three', 'four', 'five'])
```

Outro ponto positivo dos dicionário é que, com eles, é possível ter acesso ao valor de destino de modo bastante direto e fácil, à partir do valor de origem (chave). Entretanto, a tentativa de acessar o valor para ua chave que não consta do dicionário causará um erro. Como mostrado no exemplo abaixo.

Exemplo 3.28

Acesso direto aos valores do dicionário.

```
>>> D[4]
'four'
>>> D[7]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 7
```

Embora os dicionários tenham alguns recursos que facilitam o trabalho com relações em alguns casos, eles têm duas limitações quando comparados com o uso de conjuntos para definir relações:

1. Dicionários não aceitam dicionários como chaves. E não existe algo como um *frozen*-dicionário em Python. Esta limitação tem consequências menores pois não é comum querermos ou precisarmos usar dicionários como conjunto de partida nas relações.

2. Dicionários só aceitam que uma determinada chave apareça uma vez. Ou seja, dicionários só podem ser usados para representar relações funcionais. Embora boa parte das situações em computação lide com relações funcionais, é relativamente comum aparecerem conceitos que são melhor modelados por relação não-funcionais. Nestes casos o uso de dicionários não é o mais indicado para representar estas relações.

Uma vez que uma chave no dicionário pode estar associada a no máximo um valor, os dicionários são mais próximos de relações funcionais parciais do que de relações gerais como vimos na Seção 2.5.

Considere agora, que ao invés de uma relação funcional, com aquela definida no Exemplo 3.23, tenhamos uma relação em que cada número aparece relacionado a mais de uma palavra. Como podemos manipular uma relação como esta? O Exemplo 3.29 mostra como essa relação pode ser definida, e como podemos inquirir sobre elementos desta relação.

Exemplo 3.29

Definição de uma relação não-funcional em Python:

```
>>> S = frozenset({
...     (1,'one'), (2,'two'), (3,'three'), (4,'four'), (5,'five'),
...     (1,'un'), (2,'deux'), (3,'trois'), (4,'quatre'), (5,'cinq')
... })
>>> S
frozenset({(1, 'one'), (3, 'trois'), (2, 'two'), (1, 'un'),
(5, 'five'), (3, 'three'), (2, 'deux'), (4, 'four'), (4, 'quatre'),
(5, 'cinq')})
```

Teste se um determinado elemento de origem, ou de destino, pertence à relação.

```
>>> any(a == 3 for (a,b) in S)
True
>>> any(a == 7 for (a,b) in S)
False
>>> any(b == 'cinq' for (a,b) in S)
True
>>> any(b == 'neuf' for (a,b) in S)
False
```

Operação pra obter os elementos de destino associados com um determinado elementos de origem, e vice-versa.

```
>>> {y for (x,y) in S if x == 1}
{'one', 'un'}
>>> {y for (x,y) in S if x == 3}
{'three', 'trois'}
>>> {x for (x,y) in S if y == 'cinq'}
{5}
>>> {x for (x,y) in S if y == 'five'}
{5}
```

3.7.2 Operações sobre Relações em Python

Na Seção 2.6.1 vimos três operações que podem ser aplicadas a relações e funções:

1. Relação identidade.
2. Relação inversa.
3. Composição de relações.

Infelizmente, a linguagem Python, não possui formas específicas, pré-definidas, de executar estas operações. Mas não é difícil implementá-las com os recursos existentes na linguagem. Para facilitar a explicação, vamos considerar que as relações foram definidas como conjuntos.

Relação Identidade

Dado um conjunto `A`, é bastante simples definir uma relação identidade sobre este conjunto através de uma expressão de compreensão de conjuntos, como demonstrado no exemplo abaixo:

Exemplo 3.30

O código abaixo define um conjunto `A` e, em seguida, define uma relação identidade sobre `A`, chamada `idA`.

```
>>> A = frozenset({1, 2, 4, 8, 16, 32, 64})
>>> idA = {(x,x) for x in A}
>>> idA
{(16, 16), (64, 64), (32, 32), (4, 4), (8, 8), (2, 2), (1, 1)}
```

Relação Inversa

Uma vez que estamos usando conjuntos de pares ordenados para representar relações em Python, é bastante simples construir a relação inversa de qualquer relação dada.

Exemplo 3.31

O código abaixo define uma relação `R` e, em seguida, define a relação `invR` que é a relação inversa de `R`.

```
>>> R = frozenset({(1,'a'), (2,'b'), (3,'c'), (4,'d'), (5,'e')})
>>> invR = {(y,x) for (x,y) in R}
>>> invR
{('a', 1), ('d', 4), ('b', 2), ('e', 5), ('c', 3)}
```

Composição de Relações

Novamente, a representação de relações através de conjuntos em Python torna bastante simples a construção de relações compostas através de compreensão de conjuntos.

Exemplo 3.32

O código abaixo define duas relações, `R`, de inteiro para caracter, e `S`, de caracter para string, e em seguida cria a relação composta $S \circ R$, de inteiro para string.

```
>>> R = frozenset({(1, 'a'), (1,'b'), (2,'b'), (2,'c'), (3,'d'),
... (4,'e'), (4,'f'), (5,'g'), (6,'h'), (6,'i'), (7,'j')})
>>> S = frozenset({('a','alpha'), ('b','beta'), ('d', 'delta'),
... ('e', 'epsilon'), ('g', 'gamma'), ('i', 'iota')})
>>> SoR = {(x,v) for (x,y) in R for (u,v) in S if y == u}
>>> SoR
{(5, 'gamma'), (1, 'alpha'), (1, 'beta'), (3, 'delta'), (6, 'iota'),
(4, 'epsilon'), (2, 'beta')}
```

3.8 Funções em Python

Evidentemente, como função são relações, nós poderíamos definir funções usando os mesmos mecanismos que usamos para definir funções. Entretanto, Python, assim como praticamente todas as outras linguagens de programação, possui uma forma específica de definir funções.

A sintaxe para definição de funções em Python é, de certo modo, parecida com a sintaxe familiar de definição de funções na matemática. Considere as seguintes definições em matemática:

$$f(x) = x + 3 \qquad\qquad g(x) = x^2 - 1$$

A definição das mesmas funções em Python seria a seguinte:

```
>>> def f(x):
...     return x + 3
...
>>> def g(x):
...     return x**2 + 1
...
>>> f(4)
7
>>> g(4)
17
```

Como pode ser visto na definição acima, a palavra reservada **def** é usada para iniciar a definição da função; em seguida temos o nome da função, seguido de parênteses. Entre os parênteses segue-se a lista de parâmetros da função, que pode conter zero ou mais parâmetros. Em seguida temos os comandos que realizam o cálculo do valor de retorno da função. O valor de retorno em si é indicado pela palavra reservada **return**. Para usar a função para realizar cálculos, ou seja, para se obter um resultado da função, basta escrever o seu nome e passar valores para os parâmetros.

Exemplo 3.33

A função abaixo recebe como parâmetros os comprimentos dos dois catetos de um triângulo retângulo e retorna o comprimento da hipotenusa deste triângulo.

```
>>> from math import sqrt
>>> def h(a,b):
...     return sqrt(a**2 + b**2)
...
>>> h(3,4)
5.0
```

3.8.1 Função Identidade

A construção de uma função identidade em Python é trivial, e essa função funciona para qualquer conjunto de origem/destino. O código da função identidade em Python é mostrado abaixo:

```
>>> def id(x):
...     return x
...
```

Veja que a função identidade, `id`, simplesmente retorna o argumento x que foi passado.

3.8.2 Composição de Funções

A composição de funções em Python também pode ser obtida facilmente. Um exemplo de código que gera uma função composta é mostrado abaixo.

Exemplo 3.34

Definição das funções $f(x) = x + 3$, $g(x) = x^2 + 1$, e de sua composta $f \circ g$.

```
>>> def f(x):
...     return x + 3
...
>>> def g(x):
...     return x**2 + 1
...
>>> def fog(x):
...     return f(g(x))
...
>>> fog(4)
20
```

3.8.3 Função Inversa

Infelizmente, diferentemente do que ocorrer com relações definidas como conjuntos, não há um modo geral de criar funções inversas em Python. Como vimos na Seção 2.6.1, nem toda função possui uma função inversa (a inversa sempre existirá, mas nem sempre será função), por essa razão, se a função inversa de uma função f qualquer for necessária, ela terá que ser explicitamente definida pelo programador. O exemplo abaixo ilustra como essa definição pode ser feita.

Exemplo 3.35

Definição da função $g(x) = x^2 + 1$, e de sua inversa $g^{-1}(x) = \sqrt{x - 1}$.

```
>>> from math import sqrt
>>> def g(x):
...     return x**2 + 1
...
>>> def g_inv(x):
...     return sqrt(x - 1)
...
>>> g(4)
17
>>> g_inv(17)
4.0
```

3.9 Exercícios de Revisão

Exercício 3.1 Escreva expressões em Python que realizem o seguinte:

1. Definam os conjuntos $A = \{1\}$, $B = \{1, 2\}$ e $C = \{\{1\}, 1\}$.
2. Com os conjuntos definidos no item anterior realize os seguintes testes:

a) $A \subseteq B$
b) $A \subset B$
c) $A \in B$
d) $A = B$
e) $A \subseteq C$
f) $A \subset C$
g) $A \in C$
h) $A = C$
i) $1 \in A$
j) $1 \in C$
k) $\{1\} \in A$
l) $\{1\} \in C$
m) $\emptyset \notin C$
n) $\emptyset \subseteq C$
o) $B \subset C$

Exercício 3.2 Escreva expressões em Python que realizem o seguinte:

1. Defina os seguintes conjuntos:

   a) $A = \{1, 2, 3, 4\}$
   b) $B = \{a, b, c, d, e, f\}$
   c) $C = \{1, 2\}$
   d) $D = \{1, 3, 4, a, b\}$
   e) $E = \{\{1\}, 2, \{a, 1\}\}$
   f) $F = \{1, c, d\}$
   g) $G = \{d, e, 2, 3\}$
   h) $\mathbb{U} = A \cup B \cup C \cup D \cup E \cup F \cup G$

2. Calcule os elementos dos seguintes conjuntos:

   a) $C \setminus D$
   b) $A \cap F$
   c) $A \cap B$
   d) $C^{\complement} \cap F^{\complement}$
   e) $E \cap C$
   f) $(C \cup D) \setminus (C \cap D)$
   g) $F \cup C$
   h) $G^{\complement} \cap C$
   i) $A \cap E$
   j) $(E \cup B) \cup D$
   k) $A \triangle D$
   l) $A \triangle B \triangle C$

Exercício 3.3 Escreva expressões em Python que realizem o seguinte:

1. Defina os seguintes conjuntos:

   a) $\mathbb{U} = \{p, q, r, s, t, u, v, w\}$
   b) $A = \{p, q, r, s\}$
   c) $B = \{r, t, v\}$
   d) $C = \{p, s, t, u\}$

2. Determine os seguintes conjuntos:

   a) $B \cap C$
   b) $A \cup C$
   c) $C^{\complement}$
   d) $A \cap B \cap C$
   e) $B \setminus C$
   f) $C \setminus B$
   g) $(A \cup B)^{\complement}$
   h) $A \times B$
   i) $(A \cup B) \cap C^{\complement}$
   j) $(A \setminus B)^{\complement} \setminus C^{\complement}$

Exercício 3.4 Sejam $A = \{2, 3, 4, 5\}$ e $B = \{3, 4, 5, 6, 10\}$. Para cada uma das relações $R_i \subseteq A \times B$ indicadas abaixo, use compreensão de conjuntos em Python para calcular:

i. Os elementos (pares ordenados) da relação;

ii. O domínio da relação;

iii. A imagem da relação;

1. $R_1 = \{(x, y) \mid x \text{ é divisível por } y\}$
2. $R_2 = \{(x, y) \mid x \cdot y = 12\}$
3. $R_3 = \{(x, y) \mid x = y + 1\}$
4. $R_4 = \{(x, y) \mid x \leq y\}$

Exercício 3.5 Seja $N_1 = \{x | x \in \mathbb{N}, -1000 \leq x \leq 1000\}$. Para cada uma das relações $\rho_i \subseteq {N_1}^2$ indicadas abaixo use compreensão de conjuntos em Python par determinar os elementos (pares ordenados) das relação.

1. $\rho_1 = \{(x, y) \mid x + y < 9\}$
2. $\rho_2 = \{(x, y) \mid x = y + 3\}$

3. $\rho_3 = \{(x, y) \mid 2x + 3y = 12\}$
4. $\rho_3 = \{(x, y) \mid y \text{ é um cubo perfeito}\}$[3]

Exercício 3.6 Sejam $N_1 = \{x \mid x \in \mathbb{N}, -1000 \leq x \leq 1000\}$, $E_1 = \{x \mid x \in N_1, x \text{ é par}\}$, e $O_1 = \{x \mid x \in N_1, x \text{ é ímpar}\}$.Para cada uma das relações ρ_i definidas abaixo, use Python para defini as relações e escrever expressões para determinar quais propriedades (reflexiva, simétrica, antissimétrica e/ou transitiva) a relação possui.[4]

1. $\rho_1 \subseteq {N_1}^2$;
   $\rho_1 = \{(x, y) \mid x + y \text{ é par}\}$
2. $\rho_2 \subseteq {E_1}^2$;
   $\rho_2 = \{(x, y) \mid x \text{ divide } y\}$
3. $\rho_3 \subseteq {O_1}^2$;
   $\rho_3 = \{(x, y) \mid x \text{ e } y \text{ são coprimos}\}$
4. $\rho_4 \subseteq {N_1}^2$;
   $\rho_4 = \{(x, y) \mid x = y^2\}$
5. $\rho_5 \subseteq \{0, 1\}^2$;
   $\rho_5 = \{(x, y) \mid x = y^2\}$
6. $\rho_6 \subseteq \{1, 2, 3\}^2$;
   $\rho_6 = \{(1, 1), (2, 2), (3, 3), (1, 2), (2, 1)\}$

Exercício 3.7 Para cada relação ρ_i do Exercício 3.6, use Python para definir a relação inversa, ${\rho_i}^{-1}$, e para escrever expressões para determinar quais propriedades (reflexiva, simétrica, antissimétrica e/ou transitiva) a relação inversa possui.

Exercício 3.8 Use Python para definir as seguintes funções:

1. $f(x) = 2x + 1$
2. $g(x) = x^2 + 3$
3. $c(x) = f(x) \cdot g(x)$
4. $(f \circ g)(x)$
5. $(g \circ f)(x)$
6. $k(x, y) = h(f(x), g(y))$

[3] Um cubo perfeito é um número cuja raíz cúbica é um número inteiro.

[4] Este exercício usa o conceito de números **coprimos**, ou "primos entre sí". Dois números a e b são coprimos se eles não possuem fatores primos comuns, em outras palavras, dois números a e b são coprimos se o máximo divisor comum entre eles é 1.

Capítulo 4

Lógica Formal

Sumário

4.1 Introdução

Embora existam muitas definições para o campo de estudo da lógica, essas definições não diferem essencialmente umas das outras; há um certo consenso entre os autores de que a Lógica tem, por objeto de estudo, as leis gerais do pensamento, e as formas de aplicar essas leis corretamente.

Em sua obra chamada Organum Aristóteles estabeleceu princípios tão gerais e tão sólidos que dominou o pensamento ocidental durante dois mil anos, e até hoje são considerados válidos.

Aristóteles tinha como objetivo a busca da verdade, e para isso procurava caracterizar os instrumentos de que se servia a razão. Em outras palavras, Aristóteles se preocupava com as formas de raciocínio que, a partir de conhecimentos considerados verdadeiros, permitiam obter novos conhecimentos. Caberia, pois, à Lógica a formulação de leis gerais de encadeamentos de conceitos e juízos que levariam à descoberta de novas verdades.

Essa forma de encadeamento é chamado, em Lógica, de argumento, enquanto as afirmações envolvidas são chamadas proposições; um argumento é, pois, um conjunto de proposições tal que se afirme que uma delas é derivada das demais; usualmente, a proposição derivada é chamada conclusão, e as demais, premissas. Em um argumento válido, as premissas são consideradas provas evidentes da verdade da conclusão.

Eis um exemplo de argumento:

Se eu ganhar na loteria, serei rico.
Eu ganhei na loteria.

Logo, sou rico.

Como a conclusão "sou rico" é uma decorrência lógica das duas premissas, esse argumento é considerado válido.

É preciso deixar claro que a Lógica se preocupa com o relacionamento entre as premissas e a conclusão, com a estrutura e a forma do raciocínio, e não com seu conteúdo, isto é, com as proposições tomadas individualmente. Em outras palavras, não é objeto da Lógica saber se quem ganha na Loteria fica rico ou não, ou se eu ganhei ou não na Loteria. O objeto da Lógica é determinar se a conclusão é ou não uma conseqüência lógica das premissas. Por esse motivo, por que o objeto da Lógica é a forma pela qual o raciocínio está estruturado, a Lógica costuma receber o nome de Lógica Formal.

A validade do argumento está diretamente ligada à forma pela qual ele se apresenta, como pode ser mostrado pelo enunciado abaixo:

Se eu ganhar na loteria, serei rico.
Não ganhei na loteria.

Logo, não sou rico.

Que embora seja semelhante ao anterior tem outra forma, e, nessa forma, a conclusão não se segue logicamente das premissas, e, portanto, não é um argumento válido.

4.2 Dedução e Indução

A Lógica dispõe de duas ferramentas principais que podem ser utilizadas pelo pensamento na busca de novos conhecimentos: a dedução e a indução, que dão origem a dois tipos de argumentos, dedutivos e indutivos. Os argumentos dedutivos pretendem que suas premissas forneçam uma prova conclusiva da veracidade da conclusão. Um argumento dedutivo é válido quando suas premissas, se verdadeiras, fornecem provas convincentes para sua conclusão, isto é, quando for impossível que as premissas sejam verdadeiras e a conclusão falsa; caso contrário, o argumento dedutivo é dito inválido. Os dois argumentos citados anteriormente são do tipo dedutivo, o primeiro válido e o segundo inválido.

Os argumentos indutivos, por outro lado, não pretendem que suas premissas forneçam provas cabais da veracidade da conclusão, mas apenas que forneçam indicações dessa veracidade. Veja um exemplo de argumento indutivo:

Joguei uma pedra no lago, e a pedra afundou; Joguei outra pedra no lago e ela também afundou; Joguei mais uma pedra no lago, e também esta afundou; Logo, se eu jogar uma outra pedra no lago, ela vai afundar.

Os termos "válidos" e "inválidos" não se aplicam aos argumentos indutivos; eles costumam ser avaliados de acordo com a maior ou menor possibilidade com que suas conclusões sejam estabelecidas.

Costuma-se dizer que os argumentos indutivos partem do particular para o geral, isto é, a partir de observações particulares, procura estabelecer regras gerais, que, no caso das ciências naturais, devem ser provadas por outros meios; os argumentos dedutivos, por seu lado, partem de regras gerais para estabelecer a veracidade de acontecimentos particulares. O desenvolvimento da ciência tem dependido, em grande parte, da habilidade em combinar os dois tipos de raciocínio.

4.3 Lógica Clássica e Lógica Simbólica

Os argumentos formulados em uma linguagem natural, como o inglês ou português, são, muitas vezes, de difícil avaliação, principalmente por causa da ambiguidade inerente às linguagens naturais, e das construções às vezes vagas ou confusas dos termos. Em virtude desses fatos, a partir dos trabalhos de George Boole, em meados do século XIX, foram sendo utilizados cada vez mais símbolos de origem matemática para expressar os enunciados e raciocínios da Lógica. A Lógica apresentada dessa forma é chamada Lógica Matemática ou Lógica Simbólica, enquanto a Lógica baseada em linguagem natural é chamada Lógica Clássica.

À medida que a Lógica Simbólica desenvolve sua própria linguagem técnica, ela vem se tornando um instrumento cada vez mais poderoso para a análise e a dedução dos argumentos. A utilização de uma simbologia matemática ajuda a expor, com maior clareza, as estruturas lógicas das proposições e dos argumentos, que podem não ficar suficientemente claras se expressas em linguagem natural.

4.4 Proposições e Predicados

Muitas das idéias envolvidas nos argumentos podem ser apresentadas através de *proposições* (também chamados *enunciados* ou *sentenças*) que sereferem a um único objeto; por exemplo, "Eu ganhei na loteria", "José atirou uma pedra no lago", "Sócrates é um homem". Tais proposições são chamadas singulares.

No entanto, existem outras proposições que fazem referência a conjuntos de objetos; por exemplo, "todos os homens são mortais", "alguns astronautas foram à Lua", "nem todos os gatos caçam ratos".

Os termos "homens", "astronautas" e "gatos" são conceitos; não se referem a nenhum homem, astronauta ou gato em particular, mas sim ao conjunto de propriedades que faz com que um objeto esteja em uma categoria ou em outra. Tais propriedades são chamadas predicados.

Como a lógica que trata apenas das proposições singulares é mais simples que a que trata também com conjuntos de objetos, os logicistas e autores preferiram separar o estudo da Lógica Matemática em duas partes:

- O *Cálculo Proposicional*, ou *Lógica Sentencial*, que se ocupa das proposições singulares; e
- o *Cálculo de Predicados*, ou *Lógica dos Predicados*, que trata dos conjuntos de objetos e suas propriedades.

Para tratar com objetos e suas propriedades, o Cálculo de Predicados apresenta dois conceitos matemáticos, a *variável*, para se referir a um objeto genérico de uma categoria, e os *quantificadores*, expressões como "para todo" e "existe algum" para se referirem à quantidade de objetos que partilham o mesmo predicado; assim a proposição "todos os homens são mortais" assume a forma "para todo x, se x é um homem, então x é mortal" e as proposições "alguns astronautas foram à Lua" e "nem todos os gatos caçam ratos" assumem respectivamente as formas "existe um x tal que x é um astronauta e x foi à Lua" e "existe um x tal que x é um gato e x não caça ratos".

Quando as variáveis e quantificadores se referem apenas aos objetos, o Cálculo de Predicados também é chamado *Lógica de Primeira Ordem*; mas podemos pensar em uma situação na qual as variáveis e quantificadores se refiram também aos predicados; por exemplo, considere o enunciado "existe um predicado que todas as pessoas possuem", que pode ser expressa por "existe um p tal que p é um predicado e tal que para todo x, se x é uma pessoa, então x possui p".

Quando as variáveis e quantificadores se referem também aos predicados, como na expressão acima, temos o que chamamos *Lógica de Segunda Ordem.* Um exemplo importante da Lógica de Segunda Ordem é o Principio de Indução Matemática:

> "Se o numero 1 tiver um predicado, e o fato de n possuir esse predicado implica em que $n + 1$ também o possua, então o predicado se aplica a todos os números naturais."

Os predicados de primeira ordem são, portanto, aqueles que se aplicam a indivíduos; os de segunda ordem são aqueles que se aplicam a indivíduos e aos predicados de primeira ordem. Essa generalização pode prosseguir, considerando-se predicados de terceira ordem, de quarta ordem, e assim por diante, cada um deles aplicando-se aos indivíduos e aos predicados das ordens anteriores.

Em nosso curso vamos nos limitar ao Cálculo Proposicional e ao Cálculo de Predicados de Primeira Ordem.

4.5 Verdade e Validade

A *verdade* e a *validade* são diferentes. A verdade (tal como a falsidade) é uma característica das proposições. A validade (tal como a invalidade) é uma característica dos argumentos. Por isso, é incorrecto dizer que uma proposição é válida ou inválida, tal como é incorrecto dizer que um argumento é verdadeiro ou falso.

A validade diz respeito à relação entre o valor lógico (verdadeiro ou falso) das premissas e o valor lógico da conclusão. Um argumento válido é um argumento em que as premissas justificam a conclusão, pois esta é uma consequência lógica daquelas. Isso significa que a verdade das premissas assegura a verdade da conclusão.

Deste modo, podemos dizer que a verdade e a falsidade são características possíveis das diferentes partes de um argumento: premissas e conclusão. A validade e a invalidade são características da ligação dessas partes – ou seja, do próprio argumento.

A lógica formal distingue raciocínios válidos de raciocínios inválidos (ou formalmente incorrectos).

A validade é independente do conteúdo das proposições; a validade é função unicamente da forma, ou da estrutura das relações que são estabelecidas entre as proposições (que são tomados como ponto de partida – premissas) e a conclusão. Vejamos um exemplo:

"Se todos os A são B" e "se todos os B são C", daí resulta necessariamente que "todos os A são C". Esta inferência (ou raciocínio dedutivo) é válida, seja qual for a verdade ou a falsidade das premissas; trata-se de um raciocínio formal, pois é válido seja qual for o conteúdo dos termos A, B ou C.

Outro exemplo: "Se todos os homens são mortais" e "Pedro é homem", então "Pedro é mortal". Se substituirmos Pedro por João ou por Maria, ou se se substituir os termos por símbolos ("Se todos os X são Y" e "Z é X", então "Z é Y"), mantendo a mesma forma, ainda assim o raciocínio continua válido. A validade lógica do raciocínio depende deste encadeamento formal do raciocínio, independentemente dos conteúdos atribuídos aos símbolos utilizados.

Consederemos agora:

Todos os homens têm asas
Pedro é homem

Logo, Pedro tem asas

Podemos verificar que:

(a) O raciocínio obedece à mesma estrutura formal anterior; por isso, do ponto de vista da validade formal, é válido.

(b) A conclusão é *materialmente* falsa, ou seja, no confronto com a realidade, verificamos que a proposição "Pedro tem asas" é falsa. Isto aconteceu porque, apesar da conclusão decorrer necessariamente das premissas, parte da premissa era falsa: "Todos os homens têm asas".

Portanto, é importante distinguir entre *validade formal* e *verdade material* de um raciocínio. Esta última avalia-se no confronto entre o que é afirmado ou negado e a realidade. A conformidade com as regras lógicas é uma condição necessária, mas não suficiente, para garantir que se atinja conclusões verdadeiros. É preciso ainda que se parta de premissas verdadeiras.

4.6 Raciocínio Lógico

Antes de iniciarmos o estudo sistemático da Lógica, exercitemos desde já nosso raciocínio, e apelemos ao velho e útil bom senso para resolver os seguintes problemas:

1. Se eu não tenho carro, a afirmação "meu carro não é azul" é verdadeira ou falsa ?

2. Existe um ditado popular que afirma que "toda regra tem exceção". Considerando que essa frase é, por sua vez, também uma regra, podemos garantir que é verdadeira? Ou que é falsa?

3. Tenho 9 pérolas idênticas, mas sei que uma delas é falsa, e é mais leve que as outras; como posso identificar a pérola falsa, com apenas duas pesagens em uma balança de dois pratos?

4. Tenho 12 pérolas idênticas, mas uma delas é falsa e tem peso um pouco diferente das demais, não sei se mais leve ou mais pesada; como posso identificar a pérola falsa, e se ela é mais leve ou mais pesada, com apenas três pesagens em uma balança de dois pratos? Se for o caso, como?

5. Tenho 10 grupos com 10 moedas cada um; todas as moedas pesam 10 gramas cada uma, exceto as de um grupo, no qual as moedas pesam 9 gramas cada uma; como posso identificar o grupo de moedas mais leves, com apenas uma pesagem em uma balança de um prato?

6. Durante uma expedição, um explorador encontra uma caverna com três deuses: o deus da sinceridade, que sempre fala a verdade; o deus da diplomacia, que às vezes diz a verdade, às vezes, não; e o deus da falsidade, cujas declarações são sempre mentirosas. O deus A diz: "B é o deus da sinceridade", mas o deus B retruca: "Não, eu sou o deus da diplomacia", e o deus C completa: "Nada disso, B é o deus da mentira". Afinal, quem é quem?

7. Há muitos anos atrás, vivia em uma pequena cidade, um barbeiro, que ganhava a vida fazendo a barba dos habitantes da região. Um dia, ele ficou muito doente, e, na iminência de morrer, fez uma promessa ao santo de sua devoção: se ficasse bom, faria gratuitamente, uma vez por ano, a barba de todos os homens, e unicamente desses homens, que não fizessem sua própria barba. O barbeiro foi melhorando, melhorando, até que ficou bom; dispôs-se então a cumprir a promessa: na data aprazada, passou todo o dia barbeando os homens que não faziam sua própria barba. À noite, antes de dormir, foi se barbear, e verificou que estava diante de um impasse: se fizesse sua própria barba, estava barbeando um homem que fazia sua própria barba, o que quebrava a promessa; por outro lado, se não fizesse, estaria deixando de fazer a barba de um homem que não fazia sua própria barba, o que tambem quebrava a promessa. Você tem idéia de como sair desse impasse?

8. Um rei resolveu dar a um prisioneiro a oportunidade de obter a liberdade. Levou-o até uma sala, com duas portas de saída, chamadas A e B, cada uma com um guarda. Disse: "Uma das portas leva à liberdade, enquanto a outra leva à fôrca; alem disso, um dos guardas fala sempre a verdade, enquanto o outro só fala mentiras. Voce pode fazer uma única pergunta a um dos guardas e escolher uma porta para sair." O prisioneiro pensou durante alguns segundos; depois, dirigiu-se a um dos guardas e disse: "Se eu perguntasse a seu companheiro qual a porta que leva à liberdade, o que ele me diria ?". Depois de alguns segundos, o guarda respondeu: "A". "Obrigado", disse o prisioneiro, e passou pela porta B. O prisioneiro obteve a liberdade ou foi para a fôrca? Como saber?

9. Um outro rei resolveu dar a três prisioneiros uma oportunidade de obter a liberdade. Mandou vir três chapéus brancos e dois vermelhos, e escolheu um chapéu para cada prisioneiro; ganharia a liberdade aquele que fosse capaz de dizer a cor de seu próprio chapéu, observando unicamente a cor dos chapéus de seus companheiros. O primeiro prisioneiro observou o chapéu dos outros dois prisioneiros, mas não foi capaz de dizer a cor de seu próprio chapéu e voltou para a prisão; o segundo, à sua vez, após observar os chapéus dos outros prisioneiros também não soube dizer que cor tinha seu chapéu, e também voltou para a prisão. O rei, ao perceber que o terceiro prisioneiro era

cego, nem ia se dar ao trabalho de perguntar, mas este insistiu que deveria ter a mesma oportunidade. Inquirido, declarou corretamente a cor de seu chapéu e ganhou a liberdade. Qual a cor do chapéu do prisioneiro cego, e como ele chegou à conclusão correta?

Capítulo 5

Lógica Proposicional

Sumário

5.1 Introdução

Este capítulo trata da *lógica proposicional*. Descreveremos a sintaxe e a semântica da lógica proposicional e introduziremos noções chaves sobre os conceitos de satisfatividade, validade e equivalência lógica. Também apresentaremos outras noções tais como a de fórmulas bem formadas, conectivos lógicos, valores lógicos, consequência lógica, prova formal e poder de expressividade de linguagens.

Assim como outras lógicas, a lógica proposicional pode expressar propriedades. Essas propriedades são expressas usando uma linguagem própria. Expressões nesta linguagem são chamadas fórmulas proposicionais. Estas fórmulas realizam o concento matemático de proposição (asserção, afirmativa, sentença). Intuitivamente falando, uma proposição é qualquer coisa que possa ser *verdadeira* ou *falsa*. Quando lidamos como fórmulas proposicionais, nós abstraímos os seus significados concretos. Ou seja, nós consideramos estas fórmulas apenas à partir de um ponto de vista abstrato, de tal modo que o significado de fórmulas complexas seja definido em termos de fórmulas mais simples. A análise das fórmulas é feita com base no formato das fórmulas e não em seu significado concreto.

As inferências lógica mais básicas dizem respeito a combinações de sentenças (proposições) através de expressões que as conectam, tais como 'não', 'e', 'ou', 'se...então', ... Estas combinações nos permitem descrever situação e propriedades que estas situações possuem. Ao mesmo tempo, estas combinações de sentenças lógicas são fundamentais também em outro sentido: elas estruturam a forma como nos comunicamos e construímos argumentações.

A lógica proposicional foi formal e sistematicamente desenvolvida pela primeira vez no famoso livro de 1854 do matemático inglês George Boole [Boo58]. Boole foi o primeiro a dar um tratamento algébrico à lógica de proposições.

Esse tipo de álgebra hoje é conhecido como Álgebra Booleana. Um tratamento mais completo do que o visto aqui – e mais moderno do que o livro original de Boole – pode ser encontrado no livro [GH09]. Se você estiver interessado em entretenimento e em história da lógica, deveria ler o livro "Logic: A Graphic Guide" [CSM08], que apresenta de modo bastante descontraído a história de vários dos principais desenvolvimentos da lógica moderna. Outra opção de entretenimento ligada à lógica são os excelentes livros de quebra-cabeças lógicos de Raymond Smullyan [Smu82; Smu90].

5.2 Cálculo Proposicional

A lógica pode ser vista em ação em vários contextos. Por exemplo:

> Em um restaurante, o seu Pai pede peixe, a sua Mãe pede uma salada, e você pede carne. Algum tempo depois o garçom vem da cozinha trazendo três prato. O que irá acontecer?

O que nós esperamos que aconteça é que o garçom irá perguntar algo como "Quem pediu carne?", e servirá este prato; então ele irá perguntar "Quem pediu peixe?", e servirá este prato. Então, sem perguntar mais nada, ele servirá o último prato para a pessoa correta. O que aconteceu aqui?

Iniciando pelo passo final, quando o garçom serve o terceiro prato sem perguntar à quem deve servir, nós podemos ver uma aplicação direta da lógica "ao vivo": o garçom formula um conclusão! A informação nas respostas das duas primeiras perguntas permite ao garçom inferir automaticamente a quem ele deve servir o terceiro prato. Nós podemos representar este fato em um esquema de inferência como mostrados abaixo:

$$P \text{ ou } S \text{ ou } C, \text{não } C, \text{não } P \Longrightarrow S$$

Que podemos interpretar como: "A última pessoa deveria receber peixe, salada ou carne (são os três pratos que o garçom trazia); ela não receberá peixe (já foi servido); ela não receberá carne (já foi servido); **portanto**, receberá salada."

5.2.1 Sentenças e Proposições

Quando raciocinamos, nós usamos sentenças. A sentença "A garo que está apaixonada pelo Carlos ofendeu a maioria de seus colegas." implica, *inter alia*[1], a sentença "Alguma garota está apaixonada pelo Carlos." Uma pessoa que aceite a primeira sentença como verdadeira deve também aceitar a segunda sentença. Entretanto, nem todas as sentenças são próprias para fazer parte de um relacionamento inferencial com outras sentenças. Nada pode ser concluido de uma sentença do como "Woohoo!" (a não ser, talvez que a pessoa que disse a sentença está excitada com alguma coisa). Uma sentença como esta não contém informação suficiente para servir de base para formular conclusões.

Esta é a razão pela qual é muito importante fazer uma distinção clara entre sentenças e *proposições*. Sentenças são identificada com as frases gramaticamente corretas em alguma linguagem — no nosso caso, o Português. Uma proposição, por outro lado, é aquilo que uma *sentença declarativa* afirma inequivocamente. A mesma proposição pode ser afirmada usando sentenças diferentes. Por exemplo:

- Lógica é a disciplina mais chata que Clara já cursou.
- A disciplina mais chata que Clara já cursou é Lógica.

As duas sentenças acima dizem a mesma coisa, expressam a mesma proposição. Além disso, também temos uma mesma proposição sendo expressa nas seguintes sentenças:

- Lógica é chata. [Porgutês]
- Logic is boring. [Inglês]
- Logique est ennuyeux. [Francês]
- Lógica es aburrida. [Espanhol]
- Logik ist langweilig. [Alemão]
- Logika jest nudna. [Polonês]

[1] Latin: "entre outras coisas".

Logicistas[2] se importam com a proposições porque as proposições são adequadas para "carregar" os chamados valores-lógicos. Proposições podem ser verdadeiras ou falsas, e na lógica clássica há dois valores-lógicos: verdadeiro e falso.

Dessa forma, vimos duas características fundamentais das proposições:

- Uma proposição é aquilo que é afirmado inequivocamente por uma sentença declarativa.
- Uma proposição precisa ser verdadeira ou falsa (mas não ambos).

Vejamos como estas duas características de proposições estão relacionadas uma com a outra analisando se sentenças não declarativa (como exclamações e perguntas) e sentenças declarativas ambíguas podem ser verdadeiras ou falsas.

(a) *Exclamações não são proposições.* Exclamações como "Woohoo!" ou "Venha cá!" não são proposições, uma vez que elas não são nem verdadeiras nem falsas. Na verdade, exclamações não podem ser usada para afirmar coisa alguma.

(b) *Perguntas não são proposições.* Considere a seguinte pergunta:

> Você já está aborrecido?

Esta pergunta é verdadeira ou falsa? Se você estiver confuso para achar a respostas, isso é normal uma vez que **a pergunta** não pode ser verdadeira nem falsa. O que pode ser verdadeira ou falsa é a respostas à esta pergunta, mas não a pergunta em si. Novamente, não é possível afirmar coisa alguma com uma pergunta.

(c) *Sentenças declarativas ambíguas não são proposições.* Vamos considerar uma outra sentença, uma sentença declarativa desta vez:

> Os diretores não deram permissão aos alunos para protestar, uma vez que eles eram *skinheads*.

Aparentemente a sentença acima não tem problemas, e está afirmando algo que pode, de fato, ser verdadeiro ou falso. Entretanto, essa impressão está equivocada porque esta sentença é ambígua. Não está claro se "eles" se refere aos alunos ou aos diretores. As duas sentenças abaixo expressam proposições corretamente:

(i) Os diretores não deram permissão aos alunos para protestar, uma vez que os alunos eram *skinheads*.

(ii) Os diretores não deram permissão aos alunos para protestar, uma vez que os diretores eram *skinheads*.

A seguir há outro exemplo de sentença ambígua:

> Susana comeu o atum de biquíni.

Esta sentença também é ambígua porque não fica claro quem estava de biquíni, Susana ou o atum.

(i) Susana estava usando um biquíni, ela comeu um atum.

(ii) Susana comeu um atum, que foi servido sobre um biquíni.

(d) *Sentenças declarativas com expressões de indexação não são proposições.* Há uma classe de sentenças contendo as chamadas expressões "de indexação" (como "eu", "ele", "ela", "lá", "agora", etc.) que são notoriamente ambíguas porque as expressões de indexação mudam seu referente dependendo do contexto em que elas são proferidas. Vamos considera a sentença:

> Eu estou em Vitória hoje.

Nós podemos pensar que a sentença é verdadeira, mas o que realmente é afirmado por esta sentenças. Quando eu (o autor) uso esta sentença, o que ela significa é:

> O Prof. Jefferson está em Vitória em 13 de Maio de 2015.

[2]Optamos por usar o termo "logicista" para designar a pessoa que estuda a lógica, ao invés do termo "lógico". O termo lógico designa tudo o que está relacionado à lógica e por isso, pode causar mal-entendidos.

O que é verdadeiro. Mas você (o leitor) pode usar esta sentença também, e neste caso ela significaria:

«o nome do leitor» está em Vitória em «data real de hoje».

E neste caso a proposição será, provavelmente, falsa. O ponto principal é que, dependendo da pessoa que profere a sentença (e onde, e quando), ela representará proposições diferentes.

Os pontos explicados acima devem ser suficientes para evidenciar uma diferença marcante entre sentenças e proposições. Em uma proposição, o conteúdo é explicitamente definido ao passo que sentenças frequentemente deixam parte do conteúdo dependente do contexto no qual a sentença é proferida. Portanto, sentenças dependem do contexto, e proposições são independentes de contexto.

Eventualmente, algumas pessoas podem utilizar as palavras "sentenças" e "proposições" como sinônimos. Neste texto, nós procuraremos evitar isso para não causar confusão.

Problema 5.1

Quais das sentenças a seguir expressam proposições?

- ☐ Fernando cozinho demais a couve-flor.
- ☐ Não existe força que possa parar você.
- ☐ Estava muito escuro lá.
- ☐ Meus amigos foram à mata colher cogumelos.
- ☐ Renata se esgueirou para dentro da cozinha.
- ☐ Paula se grudou na parede recém pintada.
- ☐ Se Fred Astaire não fosse um dançarino, Greta Garbo não seria uma atriz.
- ☐ Se ao menos as crianças soubessem mais do que seus pais!
- ☐ Será que o Henrique algum dia vai gostar de garotas?
- ☐ Floriano Peixoto é uma mulher.

5.2.2 Conectivos Lógicos

Um dos objetivos da lógica, enquanto ciência, é entender que argumentos são válidos e quais não são (e por quê). Esta tarefa se mostrou bastante difícil e os logicistas a abordaram passo a passo, primeiramente propondo teorias relativamente simples, e mais tarde propondo outras teorias que foram gradualmente se tornando mais e mais complexas, e também mais eficientes em capturar aquilo que nós consideramos, intuitivamente, como argumentos válidos.

A primeira, e a mais simples, destas teorias é a Lógica Proposicional. Uma das suposições básicas desta teoria diz respeito à questão de como é a estrutura lógica das proposições. De acordo com a lógica proposicional, proposições são construídas de outras proposições mais simples por meio dos **conectivos proposicionais** – ou apenas *conectivos* para encurtar o nome. Os conectivos são expressões como 'não', 'e', 'ou', 'se...então' e 'se e somente se'.

Argumentos são válidos ou inválidos em função de suas formas. Toda teoria lógica deve definir quais são as formas lógica aceitáveis. É neste ponto que a distinção entre proposições simples e complexas aparece.

Em lógica proposicional, proposições complexas são aquelas que são compostas por outras proposições através de algum dos cindo conectivos: 'não', 'e', 'ou', 'se...então', ou 'se e somente se'. Proposições que não são compostas desta maneira – e que, portanto, não podem ser divididas em proposições menores – são chamadas de proposições simples. Vamos considerar primeiro algumas proposições compostas (os conectivos estão em negrito):

- Carlos **não** perguntou que horas são.
- Suzana pegou um livro emprestado com Ana **e** ela **não** devolveu.
- **Se** Alan mantiver seu quarto arrumado por mais de uma semana, **então** ele vai ganhar um gato **ou** um cachorro.

- Cláudia vai sair com Tiago **se e apenas se** ele se desculpar com ela primeiro.

Já as proposições abaixo, são proposições simples:

- A Lua é um satélite da Terra.
- Sócrates é um homem.
- Jefferson estuda lógica.
- Maria vai alimentar o hamster.
- Todos os homens são ciumentos.
- Quem vive na esquina da Rua 1 com a Rua 2 tem uma vista horrível.
- O Presidente da Câmara confirmou o rumor de que a votação do projeto de lei sobre a saúde será adiada para depois do Natal.

Note que as proposições simples não precisam ser expressa por sentenças simples. O que constitui a "simplicidade" em lógica proposicional é a falta de conectivos na proposição.

Definição 5.1 (Proposição Simples)

Em lógica proposicional, uma *proposição simples* é uma proposição que não é construída à partir de outras pelo uso de conectivos.

Definição 5.2 (Proposição Composta)

Em lógica proposicional, uma *proposição composta* é uma proposição construída à partir de outras proposições pelo uso de conectivos.

Problema 5.2 (Proposição Simples vs. Composta)

Quais das proposições abaixo são simples e quais são compostas? No caso de proposições compostas sublinhe os conectivos usados para construir a proposição.

(a) Tiago é um homem extraordinariamente agradável que odeia todas as mulheres que usam chapéus grandes.

(b) Tiago convidou Suzana para sair e ela aceitou.

(c) Se Suzana não chegar na hora, Tiago ficará preocupado.

(d) Suzana foi bastante pontual.

(e) Tiago não acreditou no que viu.

(f) Suzana estava usando o maior chapéu que Tiago já viu na vida.

(g) Tiago começou a pedir para Suzana tirar aquela coisa horrível que deve ter sido uma das coisas mais feias já produzidas por um ser humano.

(h) Suzana concordou em tirar o chapéu se e somente se Tiago tirar a gravata borboleta e as botas de cowboy.

5.2.3 Forma Simbólica

Como foi dito antes, a lógica está interessada na forma de um argumento e não no seu significado concreto. Para facilitar a análise da forma dos argumentos a lógica matemática não trabalha com as sentenças que expressam as proposições, ao invés disso, a lógica matemática trabalha com *fórmulas* que representam as proposições.

Estas formulas são chamadas de *forma simbólica* das proposições e argumentos, e uma das coisas mais importantes que nós faremos é a "simbolização", ou conversão para a forma simbólica das proposições e dos argumentos. Nós representaremos proposições expressa por sentenças em Português usando a notação simbólica da lógica proposicional. A primeira coisa que se deve saber é que proposições simples são representadas pelas chamadas constantes proposicionais, que serão sempre letras maiúsculas do alfabeto latino (A, B, C, etc.)

Que letra representará qual proposição é uma escolha completamente arbitrária, embora seja útil selecionar uma letra que nos ajude a lembrar da proposição que a letra representa. Como exemplo, se formos escolher letras para representar as proposições abaixo:

- Tiago é elegante.
- Tiago é rico.
- Suzana é bonita.
- Suzana é pobre.

Seria mais útil usar a primeira letra do nome das características que estão sendo atribuídas a Tiago e Suzana. Uma atribuição possível seria a seguinte:

E – Tiago é elegante.

R – Tiago é rico.

B – Suzana é bonita.

P – Suzana é pobre.

Mas outras atribuições de letras são possíveis. Essa lista de atribuições de constantes proposicionais (ou letras) para proposições simples é chamado de *chave de simbolização*.

Embora a atribuição de letras às proposições simples seja arbitrária, três regras devem ser obedecidas:

1. *Uma proposição não pode ser representada por duas ou mais letras diferentes.* Essa regra é mais fácil de quebrar do que parece porque uma mesma proposição pode ser expressa por sentenças diferentes em Português. Considere, por exemplo, as seguintes sentenças:

   (i) Tiago é rico.
   (ii) Tiago é um homem rico.
   (iii) Tiago é uma pessoa rica.

   Foram usadas sentenças diferentes, mas elas expressam a mesma proposição e devem ser "simbolizadas" pela mesma letra.

2. *Uma letra não pode representar mais do que uma proposição.* Seria um erro, por exemplo, criar a seguinte chave de simbolização:

   S – Suzana é bonita.
   S – Suzana é pobre.

   O erro ocorre porque a mesma letra, S, está sendo usada para duas proposições diferentes.

3. *Uma letra não pode representar uma proposição composta.* A chave de simbolização não pode incluir qualquer proposição composta. Nós representaremos proposições compostas utilizando símbolos lógicos para os conectivos.

Um outro ponto importante a lembrar é que em lógica proposicional, nós geralmente desconsideramos os indicadores de tempo nas sentenças. Nós vamos considerar que sentenças como "Tiago convida Suzana para sair", "Tiago convidou Suzana para sair" e "Tiago irá convidar Suzana para sair" representam a mesma proposição.

Problema 5.3 (Chave de Simbolização)

Construa a chave de simbolização para as proposições contidas nas sentenças abaixo. Lembre-se de obedecer às três regras de construção de chaves de simbolização e de que, às vezes, sentenças diferentes expressam a mesma proposição.

(a) Tiago convidou Suzana para sair e ela aceitou sair com ele.

(b) Se Suzana aceitar sair com Tiago, então ela irá usar um chapéu grande.

(c) Suzana vai usar um chapéu grande se e somente se ela quiser ensinar uma lição ao Tiago.

(d) Se Suzana quiser ensinar uma lição ao Tiago, então ele não irá convidá-la para sair.

Como dito acima, as proposições compostas são construídas à partir das proposições simples através do uso de conectivos. A Lógica Proposicional utiliza cinco conectivos: "não", "e", "ou", podemos construir as seguintes proposições compostas: "se... então", e "se e somente se".

Em Lógica Simbólica, a ação de combinar proposições é chamada "operação", e os conectivos são chamados "operadores", e são representados por símbolos específicos. Apresentamos na Tabela 5.1 as cinco operações lógicas, com seus respectivos conectivos e símbolos.

Tabela 5.1: Operações, conectivos e símbolos usados na lógica proposicional.

Operação	**Conectivo**	**Símbolo**
Negação	"não"	$\neg$
Conjunção	"e"	$\wedge$
Disjunção	"ou", "ou então"	$\vee$, $\veebar$
Condicional	"se... então"	$\rightarrow$
Bicondicional	"se e somente se"	$\leftrightarrow$

O significado intuitivo das operações lógicas é explicado a seguir. Mais adiante, na Seção 5.2.5, será dada uma definição mais operacional destes conectivos, i.e., eles serão definidos em termos de manipulação de valores, de modo semelhante aos operadores aritméticos que nós já conhecemos.

Negação Se α é uma proposição, então a expressão $\neg\alpha$ é chamada negação de α. A negação inverte o valor lógico de uma expressão; se α for verdadeira, $\neg\alpha$ será falsa, e vice-versa.

Então, se a expressão "Maria foi ao cinema" é representada por P, expressões como:

- "Maria não foi ao cinema."
- "É falso que Maria tenha ido ao cinema."

devem ser representadas por $\neg P$.

Conjunção Se α e β são proposições, a expressão $\alpha \wedge \beta$ é chamada de conjunção de α e β, e as proposições α e β são chamadas **fatores**.

A proposição composta $\alpha \wedge \beta$ será verdadeira se os dois fatores α e β forem verdadeiros, e será falsa se qualquer um dos dois fatores, ou os dois, forem falsos.

Se a sentença "João é magro" e "José é alto" forem representadas por M e A, respectivamente, então as sentenças:

- "João é magro e José é alto."
- "João é magro. José é alto."
- "João é magro, mas José é alto."
- "João é matro, porém José é alto."

devem ser representadas por $M \wedge A$.

Disjunção A língua portuguesa, de modo geral, é ambígua no uso do conectivo "ou" — a utilização de "ou" entre duas parcelas indica que uma delas é verdadeira e a outra não, mas há situações em que ambas as parcelas podem ser verdadeiras; normalmente, na linguagem natural, procura-se resolver essa ambiguidade utilizando-se o contexto. Por exemplo:

> Todo aluno que tiver nota inferior a 60 pontos, ou menos de 75% de presença ficará reprovado.

Nesta frase o aluno ficará reprovado se tiver menos de 60 pontos, ou se tiver menos de 75% de presença ou ambos.

Outro exemplo:

> Mário foi ao cinema ou Marcelo ficou em casa.

Este exemplo é ainda pior, pois não há nenhuma indicação de se apenas um ou os dois fatos ocorreram. Como na Lógica não são permitidas ambiguidades, foi necessário definir dois operadores para o conectivo "ou": o "ou inclusivo", onde se permite que um dos fatos ou ambos ocorram, e o "ou exclusivo" onde um e apenas um dos fatos pode ocorrem.

Se α e β são proposições, a expressão $\alpha \vee \beta$ é chamada de *disjunção inclusiva* de α e β, e as proposições α e β são chamadas **parcelas**. A *disjunção exclusiva* de α e β é representada por $\alpha \veebar \beta$.

Em que condições uma sentença com "ou" é verdadeira?

Se interpretarmos o "ou" como uma *disjunção inclusiva* basta que Maria tenha ido a pelo menos um dos lugares, ou seja, para que uma disjunção inclusiva seja verdadeira, basta que uma das parcelas (ou ambas) seja verdadeira; apenas se ambas as parcelas forem falsas, a disjunção inclusiva será falsa.

Por outro lado, se interpretarmos o "ou" como *disjunção exclusiva*, a expressão só será verdadeira se Maria tiver ido a um dos lugares, mas não ao outro. A disjunção exclusiva será verdadeira se uma das parcelas for verdadeira e a outra falsa; se ambas as parcelas tiverem o mesmo valor lógico, a disjunção exclusiva será falsa.

Neste texto, o termo "disjunção" se referirá à disjunção inclusiva; quando se tratar da disjunção exclusiva, isso será expressamente citado.

Recentemente vem sendo cada vez mais comum se encontrar, em textos, a forma "e/ou" para indicar explicitamente que se trata de um ou inclusivo. Também é relativamente comum encontrar a expressão "ou então" sendo usada para indicar explicitamente que se trata de um ou exclusivo. A menos que fique claro pelo contexto, ou que a expressão "ou então" seja usuada, deve-se dar preferência ao uso do ou inclusivo.

Se a sentença "João é magro" e "José é alto" forem representadas por M e A, respectivamente, então as sentenças:

- "João é magro ou José é alto."
- "Ou João é magro, ou José é alto."

devem ser representadas por $M \vee A$.

Já as sentenças:

- "João é magro ou então José é alto."
- "João é magro ou José é alto, mas não ambos."

devem ser representadas por $M \veebar A$.

Condicional Considere a proposição abaixo:

> Se a chuva continuar a cair, então o rio vai transbordar.

Esta é uma proposição composta pelas duas proposições "a chuva continua a cair" e "o rio vai transbordar", ligadas pelo conectivo "se...então". Em Lógica Simbólica este conectivo é chamado "condicional" e representado pelo símbolo $\rightarrow$.

Então, se α e β são proposições, a expressão $\alpha \rightarrow \beta$ é chamada condicional de α e β; a proposição α é chamada de antecedente, e a proposição β de consequente da condicional. A operação condicional indica que o acontecimento de α é uma condição para que β aconteça. Ou ainda, que β acontecerá sempre que α acontecer.

Considere novamente a expressão do exemplo. Suponha que ambas as coisas aconteçam, isto é, que a chuva tenha continuado a cair, e o rio tenha transbordado; neste caso, a condicional é verdadeira. Por outro lado, suponha que a chuva tenha continuado a cair, mas que o rio não tenha transbordado; nesse caso, α não foi uma condição para β, por isso condicional é falsa. Em outras palavras, se o antecedente for verdadeiro e o consequente também, a condicional é verdadeira; mas se o antecedente for verdadeiro e o consequente for falso, a condicional é falsa.

Agora, o que acontece se o antecedente for falso? Considere que a chuva não tenha continuado a cair; nesse caso, independentemente do que tenha acontecido com o rio, a condicional é considerada verdadeira.

Por que isso ocorre? Por que motivo, a Lógica considera que se o antecedente for falso, a condicional é verdadeira, qualquer que seja o valor lógico do consequente?

Existem vários motivos para isso, e o mais intuitivo é o de que se o antecedente for falso, não é possível estabelecer se o antecedente foi ou não condição para o consequente. Não é possível **demonstrar** que o condicional estabelece uma relação falsa. Como a Lógica Formal Clássica só aceita verdadeiro ou falso e não podemos mostrar que o condicional é falso, somos "forçados" a considerá-lo verdadeiro. Uma razão mais operacional será dada na Seção 5.2.5.

Se representarmos por α a frase "A chuva continua a cair", e por β a frase "O rio vai transbordar", então as sentenças abaixo:

- "Se a chuva continuar a cair, então o rio vai transbordar."
- "Se a chuva continuar a cair, o rio vai transbordar."
- "O rio vai transbordar, se a chuva continuar a cair."
- "O fato de a chuva continuar a cair implica em o rio transbordar."
- "A chuva continuar a cair é condição suficiente para o rio transbordar."

podem ser representadas por $\alpha \to \beta$. Entretanto, a sentença "A chuva continuar a cair é condição necessária para o rio transbordar", deveria ser representada por $\beta \to \alpha$.

Bicondicional Agora, considere a proposição:

> João será aprovado se e somente se ele estudar.

Nesse caso, temos duas proposições "João será aprovado" e "João irá estudar", ligadas pelo conectivo "se e somente se". Em Lógica Simbólica, essa operação é chamada de "bicondiciol", e é representada pelo símbolo $\leftrightarrow$.

Se α e β são proposições, a expressão $\alpha \leftrightarrow \beta$ é chamada bicondicional de α e β. Dizemos que a bicondicional é verdadeira quando ambos os termos são verdadeiros ou quando ambos são falsos; quando um é falso e outro é verdadeiro, a bicondicional é falsa.

No exemplo acima, o conectivo "se e somente se" indica que se João estudar ele será aprovado, e que essa é a única possibilidade de João ser aprovado, isto é, se João não estudar ele não será aprovado. Os dois acontecimentos serão ambos verdadeiros ou ambos falsos, não há possibilidade de uma terceira opção.

Além de "se e somente se", a operação bicondicional é indicada por termos como "unicamente se", "exceto se", "condição necessária e suficiente" e outras análogas. Por exemplo, as sentenças:

- "João será aprovado se e somente se estudar."
- "João será aprovado unicamente se estudar."
- "João não será aprovado, exceto se estudar."
- "João estudar é condição necessária e suficiente para ser aprovado."

podem todas ser representadas por $\alpha \leftrightarrow \beta$, onde α representa "João será aprovado" e β representa "João irá estudar".

Problema 5.4

Utilizando uma chave de simbolização adequada e os conectivos lógicos apropriados, traduza as sentenças abaixo para a forma simbólica da lógica proposicional.

(a) João é magro e José é alto.

(b) Mário foi ao cinema, João foi ao teatro e Marcelo ficou em casa.

(c) Maria foi à praia ou ao mercado.

(d) Mário foi ao cinema ou Marcelo ficou em casa.

(e) A Lua não é o satélite da Terra.

(f) Se a chuva continuar a cair, então o rio vai transbordar.

(g) Se João estudar, será aprovado.

(h) João será aprovado se e somente se estudar.

5.2.4 Fórmulas Bem Formadas

Vimos que uma sentença, ou conjunto de sentenças, em linguagem natural pode ser traduzida para a forma simbólica, formando uma fórmula – ou um conjunto de fórmulas. Entretanto, nem toda sequência de símbolos lógicos e letras de proposição geram fórmulas válidas. Por exemplo, nenhuma das sequências de símbolos e letras a seguir é uma fórmula válida: $\vee AB\neg \wedge A$, $X_1 \wedge \vee X_2$, $P\neg \rightarrow Q \veebar$.

Uma vez que nem toda sequência de símbolos lógicos e letras gera uma fórmula válida, surge a questão de como definir e reconhecer uma fórmula válida, ou de como criar uma *fórmula bem formada (fbf)*.

Por definição, são considerada uma fórmula bem formada apenas aquelas sequências de símbolos lógicos e letras de proposição que podem ser obtidas através da aplicação de um conjunto bem definido de regras de geração. Essas regras são apresentadas na definição abaixo.

Definição 5.3 (Fórmula Bem Formada)

Uma *fórmula bem formada* (FBF) é uma expressão obtida pelas regras abaixo:

1. Se x é uma letra de proposição, então x é uma FBF;
2. Se α é uma FBF, então $\neg\alpha$ também é uma FBF;
3. Se α e β são FBF's, então:
   a) $(\alpha \wedge \beta)$ é uma FBF;
   b) $(\alpha \vee \beta)$ é uma FBF;
   c) $(\alpha \veebar \beta)$ é uma FBF;
   d) $(\alpha \rightarrow \beta)$ é uma FBF;
   e) $(\alpha \leftrightarrow \beta)$ é uma FBF;
4. Nada mais é FBF.

Assim, para se determinar se uma dada sequência de símbolos e letras é uma *fbf*, basta demonstrar como essa sequência pode ser construída através das regras da Definição 5.3.

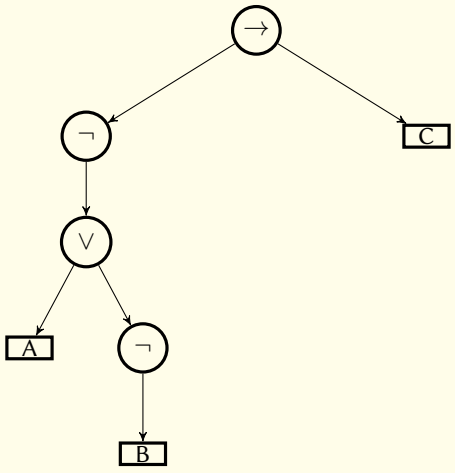
Exemplo 5.1

Para demonstrar que a fórmula $(\neg(A \vee \neg B) \rightarrow C)$ é uma *fbf*, procedemos da seguinte forma: A, B e C são *fbf*'s pela regra 1; $\neg B$ é *fbf* pela regra 2; $(A \vee \neg B)$ é *fbf* pela regra 3b; $\neg(A \vee \neg B)$ é *fbf* pela regra 2; finalmente, $(\neg(A \vee \neg B) \rightarrow C)$ é *fbf* pela regra 3d.

Uma outra forma de demonstrar que uma sequência de símbolos e letras é uma *fbf* é construindo a sua *árvore geradora*. Uma árvore geradora é um gráfico que representa como a fórmula é construída. Ela é formada por círculos contendo os operadores lógicos – os "nós" da árvore; retângulos contendo as variáveis proposicionais – as "folhas" da árvore; e linhas ou setas ligando os nós a outros nós ou a folhas – essas linhas representam os "ramos" da árvore.

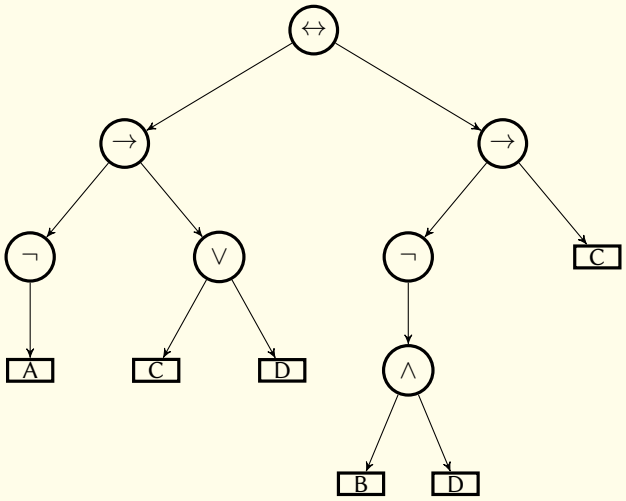
Exemplo 5.2

A árvore geradora para a fórmula $(\neg(A \vee \neg B) \rightarrow C)$ é mostrada abaixo:



Exemplo 5.3

A árvore geradora associada à fórmula $((\neg A \rightarrow (C \vee D)) \leftrightarrow (\neg(B \wedge D) \rightarrow C))$ é mostrada abaixo:



Como podemos observar pelos exemplos acima, no topo da árvore – o nó que é chamado de *raiz* da árvore – temos o operador que será avaliado por último na fórmula. Além disso, os nós que contém o operador de negação possuem exatamente um filho (uma linha ligando este nós a outro mais abaixo), enquanto todos os nós dos demais operadores possuem exatamente dois filhos.

As fórmulas bem formadas, como vistas acima, são construídas da forma que chamamos de *completamente parentetizadas*, ou seja, com todos os parênteses necessários para que sejam lidas sem ambiguidade. Ambiguidades podem surgir quando escrevemos fórmulas como $A \vee B \rightarrow C$; esta fórmula, em princípio, pode significar tanto $(A \vee B) \rightarrow C$, quanto $A \vee (B \rightarrow C)$.

Tabela 5.2: Ordem de precedência de avaliação dos operadores lógicos.

Operação	Símbolo	
Negação	$\neg$	precedência mais alta
Conjunção	$\wedge$	
Disjunção	$\vee$, $\veebar$	
Condicional	$\rightarrow$	
Bicondicional	$\leftrightarrow$	precedência mais baixa

Embora as fórmulas completamente parentetizadas evitem esse tipo de ambiguidade, o uso de todos esses parênteses acaba sendo tedioso e cansativo ao se escrever as fórmulas. Seria útil definir uma forma de escrever e ler fórmulas que evitasse a ambiguidade sem a necessidade do uso de todos esses parênteses. Para conseguir esta forma de escrever fórmulas usando menos parênteses, atribuiremos precedências ao operadores lógicos, da mesma forma que se faz na aritmética. Quando vemos a expressão $x + y \times z$ sabemos que a multiplicação deve ser avaliada antes da soma, porque a multiplicação tem precedência maior do que a soma; esta precedência dos operadores evita que sejamos obrigados a escrever $x + (y \times z)$.

A Tabela 5.2 apresenta os operadores lógicos ordenados por precedência, da mais alta para a mais baixa. Como podemos ver naquela tabela, a negação possui a maior precedência, ao passo que o bicondicional apresenta a menor.

Evidentemente, assim como na aritmética, a ordem de precedência pode ser alterada pelo uso de parênteses. Por exemplo, seguindo-se a ordem de precedência estabelecida na Tabela 5.2, a fórmula $A \wedge B \vee C$ deve ser lida como $(A \wedge B) \vee C$, entretanto, se quisermos que a disjunção seja avaliada primeiro, podemos escrever $A \wedge (B \vee C)$, usando explicitamente os parênteses.

5.2.5 Valores Lógicos e Tabelas-Verdade

Como foi dito na definição de proposição, toda proposição deve ser verdadeira ou falsa.[3] Dizemos que as proposições possuem *valor lógico* – também chamado de *valor verdade* – verdadeiro ($\top$ ou 1) ou falso ($\bot$ ou 0).

Valoração

Para as **proposições simples**, o valor da proposição deve ser determinado *ad hoc*.[4] Esses valores podem ser determinados por nosso próprio conhecimento prévio sobre o assunto do qual tratam as proposições ou pode ser dado como parte do contexto do problema.

O mapeamento entre proposições e valores lógicos – ou seja, quais proposições são verdadeiras e quais são falsas – é chamado de *valoração*. Representaremos a função que mapeia proposições simples em valores lógicos de $\mathcal{V}$; assim, para dizer que a proposição P possui valor verdadeiro, escreveremos $\mathcal{V}(P) = \top$, ao passo que $\mathcal{V}(Q) = \bot$ significa que a proposição Q tem valor falso.

Para um determinado conjunto de proposições existem várias valorações possíveis. Mais especificamente, como cada proposição pode assumir dois valores, se tivermos n proposições diferentes, teremos 2^n valorações diferentes.

Exemplo 5.4 (Valorações)

Considere o conjunto de proposições simples formado por A, B, e C. Como temos 3 proposições simples diferentes podemos ter $2^3 = 8$ valorações diferentes. Essas valorações são apresentadas na tabela abaixo:

[3] Na lógica proposicional clássica só existem estes dois valores lógico: verdadeiro e falso. Entretanto, há outros tipos de lógicas com mais do que dois valores lógicos. Essas lógicas são chamadas de lógicas multivaloradas.

[4] *Ad hoc* é uma expressão latina cuja tradução literal é "para isto" ou "para esta finalidade", mas também pode ser usado como "caso a caso".

#	$\mathcal{V}(A)$	$\mathcal{V}(B)$	$\mathcal{V}(C)$
$\mathcal{V}_0$	$\top$	$\top$	$\top$
$\mathcal{V}_1$	$\top$	$\top$	$\bot$
$\mathcal{V}_2$	$\top$	$\bot$	$\top$
$\mathcal{V}_3$	$\top$	$\bot$	$\bot$
$\mathcal{V}_4$	$\bot$	$\top$	$\top$
$\mathcal{V}_5$	$\bot$	$\top$	$\bot$
$\mathcal{V}_6$	$\bot$	$\bot$	$\top$
$\mathcal{V}_7$	$\bot$	$\bot$	$\bot$

ou

#	$\mathcal{V}(A)$	$\mathcal{V}(B)$	$\mathcal{V}(C)$
$\mathcal{V}_0$	1	1	1
$\mathcal{V}_1$	1	1	0
$\mathcal{V}_2$	1	0	1
$\mathcal{V}_3$	1	0	0
$\mathcal{V}_4$	0	1	1
$\mathcal{V}_5$	0	1	0
$\mathcal{V}_6$	0	0	1
$\mathcal{V}_7$	0	0	0

Já no caso das **proposições compostas**, o valor da proposição é determinado à partir dos valores lógicos das proposições simples utilizando-se as regras de avaliação dos operadores lógicos.

Definição 5.4 (Semântica dos Operadores Lógicos)

Sejam α e β *fórmulas bem formadas.* Os valores lógicos de proposições compostas, de acordo com os operadores que as compõem e com os valores lógicos de suas sub-fórmulas são definidos de acordo com as tabelas abaixo:

α	$\neg\alpha$
$\bot$	$\top$
$\top$	$\bot$

α	β	$\alpha \wedge \beta$
$\bot$	$\bot$	$\bot$
$\bot$	$\top$	$\bot$
$\top$	$\bot$	$\bot$
$\top$	$\top$	$\top$

α	β	$\alpha \vee \beta$
$\bot$	$\bot$	$\bot$
$\bot$	$\top$	$\top$
$\top$	$\bot$	$\top$
$\top$	$\top$	$\top$

α	β	$\alpha \rightarrow \beta$
$\bot$	$\bot$	$\top$
$\bot$	$\top$	$\top$
$\top$	$\bot$	$\bot$
$\top$	$\top$	$\top$

α	β	$\alpha \veebar \beta$
$\bot$	$\bot$	$\bot$
$\bot$	$\top$	$\top$
$\top$	$\bot$	$\top$
$\top$	$\top$	$\bot$

α	β	$\alpha \leftrightarrow \beta$
$\bot$	$\bot$	$\top$
$\bot$	$\top$	$\bot$
$\top$	$\bot$	$\bot$
$\top$	$\top$	$\top$

— ou —

α	$\neg\alpha$
0	1
1	0

α	β	$\alpha \wedge \beta$
0	0	0
0	1	0
1	0	0
1	1	1

α	β	$\alpha \vee \beta$
0	0	0
0	1	1
1	0	1
1	1	1

α	β	$\alpha \rightarrow \beta$
0	0	1
0	1	1
1	0	0
1	1	1

α	β	$\alpha \veebar \beta$
0	0	0
0	1	1
1	0	1
1	1	0

α	β	$\alpha \leftrightarrow \beta$
0	0	1
0	1	0
1	0	0
1	1	1

Nota 5.1: Deste ponto do texto em diante, por simplicidade, passaremos a utilizar apenas **1** para significar "verdadeiro" e **0** para significar "falso".

Deste modo, dada uma valoração para um conjunto de proposições simples, para se determinar o valor de uma proposição composta construída à partir destas proposições simples, substituímos para cada proposição simples o seu valor lógico e em seguida aplicamos as regras dadas na Definição 5.4, de acordo com a ordem de precedência definida na Tabela 5.2.

Exemplo 5.5

Dada a valoração $\mathcal{V}(A) = 1, \mathcal{V}(B) = 0, \mathcal{V}(C) = 1$, o valor lógico da fórmula

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

pode ser calculado por:

$$\begin{gathered}(\neg(A \vee \neg B) \rightarrow C) \\ (\neg(1 \vee \neg 0) \rightarrow 1) \\ (\neg(1 \vee 1) \rightarrow 1) \\ (\neg 1 \rightarrow 1) \\ (0 \rightarrow 1) \\ 1\end{gathered}$$

Exemplo 5.6

Dada a valoração $\mathcal{V}(A) = 0, \mathcal{V}(B) = 1, \mathcal{V}(C) = 0, \mathcal{V}(D) = 0$, o valor lógico da fómrula

$$((\neg A \rightarrow (C \vee D)) \leftrightarrow (\neg(B \wedge D) \rightarrow C))$$

é calculado por:

$$\begin{gathered}((\neg A \rightarrow (C \vee D)) \leftrightarrow (\neg(B \wedge D) \rightarrow C)) \\ ((\neg 0 \rightarrow (0 \vee 0)) \leftrightarrow (\neg(1 \wedge 0) \rightarrow 0)) \\ ((1 \rightarrow (0 \vee 0)) \leftrightarrow (\neg(1 \wedge 0) \rightarrow 0)) \\ ((1 \rightarrow 0) \leftrightarrow (\neg 0 \rightarrow 0)) \\ ((1 \rightarrow 0) \leftrightarrow (1 \rightarrow 0)) \\ (0 \leftrightarrow 0) \\ 1\end{gathered}$$

Tabelas-Verdade

Muitas vezes estamos interessados no comportamento geral de uma fórmula, para quaisquer valores de suas proposições simples, e não apenas no valor lógico para uma valoração em particular. Para analisar esse comportamento geral das fórmulas, uma ferramenta útil é construir a **tabela-verdade**.

Definição 5.5 (Tabela-Verdade)

Uma **tabela-verdade** para uma fórmula ϕ é uma tabela que contém uma linha para cada valoração possível das variáveis proposicionais de ϕ, e pelo menos uma coluna para cada variável proposicional de ϕ e mais uma coluna para a própria fórmula ϕ. Se ϕ contem n variáveis proposicionais, então a tabela verdade deve ter 2^n linhas.

Em cada linha da tabela verdade serão listadas, nas colunas de cada variável proposicional, o valor lógico da variável que rotula a coluna e na última coluna, rotulada por ϕ, será listado o valor de ϕ para valoração daquela linha.

Por exemplo, para a fórmula $A \vee B \rightarrow \neg(A \vee B)$, a tabela verdade deve conter, a menos, a informações mostradas abaixo:

A	B	$A \vee B \rightarrow \neg(A \vee B)$
0	0	1
0	1	0
1	0	0
1	1	0

Frequentemente, a tabela verdade inclui colunas auxiliares, com as sub-fórmulas da fórmula principal para ajudar a determinar o valor lógico da última coluna (correspondente à fórmula principal).

Exemplo 5.7

A tabela verdade da fórmula $A \vee B \rightarrow \neg(A \vee B)$ pode ser construída com o uso de colunas auxiliares da seguinte forma:

A	B	$A \vee B$	$\neg(A \vee B)$	$A \vee B \rightarrow \neg(A \vee B)$
0	0	0	1	1
0	1	1	0	0
1	0	1	0	0
1	1	1	0	0

Exemplo 5.8

A tabela-verdade da fórmula $P \vee (Q \wedge R) \rightarrow (P \wedge R) \vee Q$ pode ser construída da seguinte forma:

P	Q	R	$Q \wedge R$	$P \vee (Q \wedge R)$	$P \wedge R$	$(P \wedge R) \vee Q$	$P \vee (Q \wedge R) \rightarrow (P \wedge R) \vee Q$
0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	1
0	1	0	0	0	0	1	1
0	1	1	1	1	0	1	1
1	0	0	0	1	0	0	0
1	0	1	0	1	1	1	1
1	1	0	0	1	0	1	1
1	1	1	1	1	1	1	1

5.2.6 Tautologias, Contradições e Equivalências Lógicas

Considera a fórmula $A \rightarrow (B \rightarrow A)$, cuja tabela verdade é dada abaixo:

A	B	$B \rightarrow A$	$A \rightarrow (B \rightarrow A)$
0	0	1	1
0	1	0	1
1	0	1	1
1	1	1	1

Ao observar o valor da fórmula (i.e., a última coluna da tabela verdade) notamos que ela é verdadeira em todas as linhas da tabela. Em outras palavras, em todas as valorações possíveis das variáveis proposicionais a fórmula é verdadeira. As fórmula que tem esta característica são chamadas de *tautologias*.

Definição 5.6 (Tautologia)

Uma **tautologia** na lógica proposicional é uma *fórmula bem formada* que tem valor lógico verdadeiro (1) para todas as valorações possíveis de suas variáveis proposicionais. Ou, dito de outra forma, é uma fórmula que tem valor 1 em todas as linhas de sua tabela-verdade.

De modo análogo, considere agora a fórmula $(P \leftrightarrow Q) \wedge (\neg P \wedge Q)$, cuja tabela verdade é dada a seguir:

P	Q	$P \leftrightarrow Q$	$\neg P$	$\neg P \wedge Q$	$(P \leftrightarrow Q) \wedge (\neg P \wedge Q)$
0	0	1	1	0	0
0	1	0	1	1	0
1	0	0	0	1	0
1	1	1	0	0	0

Podemos notar que na tabela acima ocorre o oposto do que verificamos em uma tautologia, ou seja, a fórmula tem valor lógico falso (0) em todas as linha da tabela verdade. Fórmulas com esta característica são chamadas de *contradições*.

Definição 5.7 (Contradição)

Uma **contradição** na lógica proposicional é uma *fórmula bem formada* que tem valor lógico falso (0) para todas as valorações possíveis de suas variáveis proposicionais. Ou seja, é uma fórmula que tem valor 0 em todas as linhas de sua tabela-verdade.

Quando uma fórmula não é nem uma tautologia, nem uma contradição, dizemos que esta fórmula é uma **contingência**. Ou seja, em uma *contingência* as linhas da tabela verdade assumem tanto valor lógico 0 quanto valor lógico 1 dependendo da linha.

Problema 5.5

Para cada fórmula abaixo, construa sua tabela-verdade e determine se a fórmula é uma tautologia, uma contradição ou uma contingência.

a) $A \wedge B \rightarrow A$

b) $\neg P \wedge (P \wedge Q)$

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

d) $A \rightarrow (B \rightarrow A \wedge B)$

e) $P \rightarrow (0 \rightarrow \neg P)$

Se uma fórmula tem valor lógico verdadeiro para alguma interpretação, ela é dita **satisfatível**, ou consistente; evidentemente as tautologias e as contingências são exemplos de fórmulas satisfatíveis. Por outro lado, se uma fórmula assume valor lógico falso para alguma interpretação, ela é chamada de **inválida**; evidentemente, as contradições e as contingência são exemplos de fórmulas inválidas. Assim, uma outra forma de definir fórmulas contingentes é dizer que elas são fórmulas simultaneamente satisfatíveis e inválidas.

Uma fórmula proposicional na forma bicondicional $\alpha \leftrightarrow \beta$ que é também uma tautologia é chamada de uma *equivalência lógica*, ou simplesmente *equivalência*, entre α e β. Neste caso também dizemos que α é equivalente a β, que é escrito como $\alpha \equiv \beta$.

Definição 5.8 (Equivalência Lógica)

Dadas duas fórmulas proposicionais α e β, dizemos que α é **equivalente** a β (escrito $\alpha \equiv \beta$) se, e somente se, a fórmula $\alpha \leftrightarrow \beta$ for uma **equivalência lógica**, ou seja, se este bicondicional for uma tautologia.

Exemplo 5.9

A fórmula $P \rightarrow Q \leftrightarrow \neg Q \rightarrow \neg P$ é uma equivalência lógica. Isso pode ser verificado pela tabela-verdade abaixo:

P	Q	$P \rightarrow Q$	$\neg Q$	$\neg P$	$\neg Q \rightarrow \neg P$	$P \rightarrow Q \leftrightarrow \neg Q \rightarrow \neg P$
0	0	1	1	1	1	1
0	1	1	0	1	1	1
1	0	0	1	0	0	1
1	1	1	0	0	1	1

Escrevemos então, $P \rightarrow Q \equiv \neg Q \rightarrow \neg P$.

Decorre imediatamente da Definição 5.8 que, se duas fórmulas são equivalentes, então elas possuem a mesma tabela-verdade, e, reciprocamente, se duas fórmulas têm a mesma tabela-verdade, então são equivalentes. Uma bicondicional é

verdadeira se e somente se seus componentes têm os mesmos valores lógicos; como a fórmula deve ser uma tautologia, o seja, verdadeira em todas as valorações; isto significa que seus componentes têm o mesmo valor lógico em todas as valorações.

Decorre ainda da Definição 5.8 que todas as tautologias são equivalentes entre si, e também que todas as contradições são equivalentes entre si.

Listamos abaixo algumas das equivalência mais importantes (e úteis) da Lógica; cada uma delas pode ser provada simplesmente mostrando que a bicondicional correspondente é uma tautologia.

1. Bicondicional (bicond)
   $\alpha \leftrightarrow \beta \equiv (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$
2. Ou-exclusivo (ouex)
   a) $\alpha \veebar \beta \equiv (\alpha \vee \neg\beta) \wedge (\neg\alpha \wedge \beta)$
   b) $\alpha \veebar \beta \equiv (\alpha \vee \beta) \wedge \neg(\alpha \wedge \beta)$
3. Dupla negação (dn)
   $\neg\neg\alpha \equiv \alpha$
4. Comutatividade (com)
   a) $\alpha \wedge \beta \equiv \beta \wedge \alpha$
   b) $\alpha \vee \beta \equiv \beta \vee \alpha$
5. Associatividade (ass)
   a) $\alpha \wedge (\beta \wedge \gamma) \equiv (\alpha \wedge \beta) \wedge \gamma$
   b) $\alpha \vee (\beta \vee \gamma) \equiv (\alpha \vee \beta) \vee \gamma$
6. Distributividade (dist)
   a) $\alpha \wedge (\beta \vee \gamma) \equiv (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$
   b) $\alpha \vee (\beta \wedge \gamma) \equiv (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$
7. Condicional (cond)
   $\alpha \rightarrow \beta \equiv \neg\alpha \vee \beta$
8. De Morgan (dm)
   a) $\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$
   b) $\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$
9. Idempotência (id)
   a) $\alpha \wedge \alpha \equiv \alpha$
   b) $\alpha \vee \alpha \equiv \alpha$
10. Complemento (comp)
    a) $\alpha \wedge \neg\alpha \equiv 0$
    b) $\alpha \vee \neg\alpha \equiv 1$
11. Elemento neutro (en)
    a) $\alpha \wedge 1 \equiv \alpha$
    b) $\alpha \vee 0 \equiv \alpha$
12. Elemento absorvente (ea)
    a) $\alpha \wedge 0 \equiv 0$
    b) $\alpha \vee 1 \equiv 1$

5.2.7 Exercícios

Exercício 5.1 Quais das frases a seguir são proposições?

a. A Lua é feita de queijo verde.

b. Ele é, certamente, um homem alto.

c. Dois é um número primo.

d. O jogo vai acabar logo?

e. Os juros vão subir ano que vem.

f. Os juros vão descer ano que vem.

g. $x^2 - 4 = 0$

Exercício 5.2 Dada a seguinte valoração $\mathcal{V}(A) = 1, \mathcal{V}(B) = 0, \mathcal{V}(C) = 1$, qual o valor lógico de cada uma das fórmulas a seguir?

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

b. $(A \wedge B) \vee C$

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

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

Exercício 5.3 Qual o valor lógico de cada uma das proposições a seguir?

a. 8 é par ou 6 é ímpar.

b. 8 é par e 6 é ímpar.

c. 8 é ímpar ou 6 é ímpar.

d. 8 é ímpar e 6 é ímpar.

e. Se 8 é ímpar, então 6 é ímpar.

f. Se 8 é par, então 6 é ímpar.

g. Se 8 é ímpar, então 6 é par.

h. Se 8 for ímpar e 6 é par, então $8 < 6$.

Exercício 5.4 Encontre o antecedente e o consequente de cada uma das proposições a seguir.

a. O crescimento sadio de plantas é consequência de quantidade suficiente de água.

b. O aumento da disponibilidade de informação é uma condição necessária para um maior desenvolvimento tecnológico.

c. Serão introduzidos erros apenas se forem feitas modificações no programa.

d. A economia de energia para aquecimento implica boa insulação ou vedação de todas as janelas.

Exercício 5.5 São dadas diversas formas de negação para cada uma das proposições a seguir. Quais estão corretas?

a. A resposta é 2 ou 3.

   i. A resposta é nem 2 nem 3.
   ii. A resposta não é 2 ou não é 3.
   iii. A resposta não é 2 e não é 3.

b. Pepinos são verdes e têm sementes.

   i. Pepinos não são verdes e não têm sementes.
   ii. Pepinos não são verdes ou não tem sementes.
   iii. Pepinos são verdes e não tem sementes.

c. $2 < 7$ e 3 é ímpar.

   i. $2 > 7$ e 3 é par.
   ii. $2 \geq 7$ e 3 é par.
   iii. $2 \geq 7$ ou 3 é ímpar.
   iv. $2 \geq 7$ ou 3 é par.

Exercício 5.6 Escreva a negação de cada afirmação a seguir.

a. Se a comida é boa, então o serviço é excelente.

b. Ou a comida é boa, ou o serviço é excelente.

c. Ou a comida é boa e o serviço é excelente, ou então está caro.

d. Nem a comida é boa, nem o serviço é excelente.

Exercício 5.7 Escreva a negação de cada uma das afirmações a seguir.

a. O processador é rápido, mas a impressora é lenta.

b. O processador é rápido ou a impressora é lenta.

c. Se o processador é rápido, então a impressora é lenta.

d. Ou o processador é rápido e a impressora é lenta, ou então o arquivo está danificado.

e. Se o arquivo não está danificado e o processador é rápido, então a impressora é lenta.

f. A impressora só é lenta se o arquivo estiver danificado.

Exercício 5.8 Sejam A, B, e C as seguintes proposições:

A – Rosas são vermelhas.

B – Violetas são azuis.

C – Açúcar é doce.

Escreva as sentenças à seguir em notação simbólica.

a. Rosas são vermelhas e violetas são azuis.

b. rosas são vermelhas e, violetas são azuis ou então açúcar é doce.

c. Sempre que violetas são azuis, rosas são vermelhas e açúcar é doce.

d. Rosas são vermelhas apenas se violetas não forem azuis ou se açúcar não for doce.

e. Rosas são vermelhas e, se açúcar não for doce, então ou violetas são azuis ou então açúcar é doce.

Exercício 5.9 Converta as sentenças abaixo para a forma simbólica na Lógica Proposicional. Indique o significado das letras de proposição utilizadas, i.e., defina a **chave de simbolização** das fórmulas geradas.

1. Se não houver ruído, o papagaio vai morder.
2. Embora houvesse ruído, o papagaio mordeu.
3. O papagaio não vai morder, se houver ruído.
4. O papagaio morderá ou haverá ruído.
5. É falso que o papagaio não morda.
6. Ou há ruído ou não há ruído.
7. Houve ruído, mas o papagaio não mordeu.
8. O papagaio não mordeu; ainda que houvesse ruído.
9. Houve ruído e o papagaio mordeu.
10. Não houve ruído; o papagaio não mordeu.
11. O papagaio não morde, se não houver ruído.
12. Que o papagaio não mordeu é definitivamente verdade.
13. Que o papagaio tenha mordido é definitivamente falso.
14. Não só houve ruído, o papagaio também mordeu.
15. O papagaio mordeu, embora não o tenha feito.
16. Ele morde se e somente se houver ruído.

17. Se o papagaio não mordesse, não haveria ruído.
18. Ele não morde se e somente se não houver ruído.
19. As escolhas são ruído e um papagaio mordaz.
20. Sim, embora tenha havido ruído, não te mordeu.

Exercício 5.10 Usando a chave de simbolização indicada, traduza as fórmulas da Lógica Proposicional dada abaixo para linguagem natural sempre que for possível. Quando não for possível, ou seja, quando a sequência de símbolos não for uma *fórmula bem formada* escreva "NÃO É FBF!".

A – Maçãs são vermelhas C – Cenuras são crocantes M – Macacos são divertidos
B – Bananas são amarelas K – Cangurus pulam muito P – Papagaios são barulhentos

1. $A \wedge \neg M$
2. $\neg A \rightarrow \neg C$
3. $K(\wedge \neg M)$
4. $\rightarrow A, B$
5. $\neg P \wedge M$
6. $\neg A \wedge B$
7. $A \rightarrow BC$
8. $K \wedge (\neg M)$
9. $A \wedge B \vee C$
10. $(\neg A) \wedge B$
11. $A, B, \wedge C$
12. $\neg(A \wedge B)$
13. $(\neg A)(\wedge B)$
14. $(K \vee M) \vee \neg P$
15. $K \vee (M \vee \neg P)$
16. $M \wedge (\neg P \wedge \neg M) \vee P$
17. $(M \wedge \neg P) \wedge (\neg M \vee P)$
18. $\neg(\neg A) \wedge ((\neg B) \wedge (\neg C))$
19. $\neg(\neg(\neg M)) \wedge \neg\neg\neg P$
20. $\neg(B \vee \neg C) \rightarrow (\neg B \wedge C)$

Exercício 5.11 Converta as sentenças abaixo para a forma simbólica na Lógica Proposicional. Indique o significado das letras de proposição utilizadas, i.e., defina a chave de simbolização das fórmulas geradas.

1. Se este é um ornitorrinco, então põe ovos.
2. Isto não põe ovos, se é um ornitorrinco.
3. Se isto não é um ornitorrinco, não põe ovos.
4. Este é um ornitorrinco apenas se puser ovos.
5. Não é um ornitorrinco apenas se não puser ovos.

6. Coloca ovos se e somente se não for um ornitorrinco.
7. Isto não é um ornitorrinco, a menos que ponha ovos.
8. Isto põe ovos a menos que não seja um ornitorrinco.
9. A menos que ponha ovos, isto não é um ornitorrinco.
10. Somente se este é um ornitorrinco, põe ovos.
11. Não põe ovos sem ser um ornitorrinco.
12. Isto põe ovos sem que seja um ornitorrinco.
13. Sem colocar ovos, não é um ornitorrinco.
14. Ser um ornitorrinco requer que ele coloque ovos.
15. Este é um ornitorrinco, mas só se põe ovos.
16. Desde que coloque ovos, não é um ornitorrinco.

5.3 Dedução no Cálculo Proposicional

Na seção anterior vimos o conceito de proposição e como traduzir sentenças em linguagem natural para a forma simbólica, construindo chaves de simbolização e usando os operadores lógicos. O objetivo por trás deste processo de tradução é permitir que utilizemos a lógica proposicional para modelar e raciocinar sobre informações e sobre "discursos", i.e., sobre o que as pessoas falam e escrevem.

Na lógica e na filosofia, um *argumento* é uma série de declarações tipicamente usada para persuadir alguém de alguma coisa, ou para apresentar razões para aceitar uma conclusão. A forma geral de um argumento em uma linguagem natural consistem em apresentar *premissas* para fundamentar uma *conclusão*. Em um argumento dedutivo típico, as premissas se destinam fornecer uma garantia da validade da conclusão, enquanto em argumentos indutivos se considera que elas fornecem razões para apoiar a provável validade da conclusão.

Os padrões e critérios usados para avaliar argumentos e suas formas de raciocínio são estudados na lógica. Os caminhos para formular argumentos efetivos são estudados pela retórica e a teoria da argumentação. Uma argumento em uma linguagem formal mostra a forma lógica de um argumento em linguagem natural traduzido para a forma simbólica.

O que queremos fazer agora é usar ferramentas da lógica formal para permitir que deduzamos conclusões "corretas", i.e., válidas, à partir de proposições dadas.

5.3.1 Argumentos Válidos

Antes de darmos uma definição para *argumento* na lógica proposicional, precisamos estabelecer a noção de *consequência lógica*.

Definição 5.9 (Consequência Lógica)

Dizemos que uma *fórmula bem formada* β é uma **consequência lógica** de um conjunto de *fbf's* $\alpha_1, \alpha_2, \ldots, \alpha_n$, quando esse conjunto de premissas **acarreta** a conclusão β. Existem dois tipos de consequência lógica:

Consequência Semântica Dizemos que β é uma *consequência semântica* de $\alpha_1, \alpha_2, \ldots, \alpha_n$, escrito

$$\alpha_1, \alpha_2, \ldots, \alpha_n \vDash \beta$$

se e somente se não existir nenhuma valoração para as variáveis proposicionais que faça todas as fórmulas $\alpha_1, \alpha_2, \ldots, \alpha_n$ verdadeiras mas que faça β falsa.

Consequência Sintática Dizemos que β é uma *consequência sintática* de $\alpha_1, \alpha_2, \ldots, \alpha_n$, escrito

$$\alpha_1, \alpha_2, \ldots, \alpha_n \vdash \beta$$

se e somente se não existir existir uma "prova" (em algum sistema de prova) de que dadas as premissas $\alpha_1, \alpha_2, \ldots, \alpha_n$ é possível derivar a fórmula β.

Um resultado direto da definição de *consequência semântica* é que β será uma consequência lógica de $\alpha_1, \alpha_2, \ldots, \alpha_n$, se e somente se, a fórmula $\alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_n \rightarrow \beta$ for uma tautologia.

Exemplo 5.10

Como exemplo, afirmamos que Q é consequência lógica de $P \rightarrow Q$ e $\neg Q \rightarrow P$. Para testar a validade desta afirmação, construímos a tabela-verdade para a fórmula $(P \rightarrow Q) \wedge (\neg Q \rightarrow P) \rightarrow Q$ que, como podemos ver abaixo, é uma tautologia.

P	Q	$P \rightarrow Q$	$\neg Q$	$\neg Q \rightarrow P$	$(P \rightarrow Q) \wedge (\neg Q \rightarrow P)$	$(P \rightarrow Q) \wedge (\neg Q \rightarrow P) \rightarrow Q$
0	0	1	1	0	0	1
0	1	1	0	1	1	1
1	0	0	1	1	0	1
1	1	1	0	1	1	1

Assim, a lógica proposicional define um *argumento* como sendo uma sequência de proposições. Um *argumento válido* é uma sequência de proposições, onde a última proposição "segue" das demais, i.e., é consequência lógica das demais. Todos os outros argumentos são inválidos.

Definição 5.10 (Argumento)

Em lógica proposicional, um **argumento** é uma sequência de fórmulas escritas como $\alpha_1, \alpha_2, \ldots, \alpha_n \vdash \beta$, onde $\alpha_1, \alpha_2, \ldots, \alpha_n$ são as *premissas* do argumento e β é a *conclusão*.

Um argumento é um **argumento válido** se e somente se sua conclusão é consequência lógica de suas premissas.

Um dos argumentos mais simples é o *modus ponens*, que tem um exemplo mostrado abaixo:

$$\begin{array}{ll} 1. & P \rightarrow Q \\ 2. & P \\ \hline \therefore & Q \end{array}$$

Para verificar que o argumento acima é, de fato, válido, podemos construir a tabela-verdade para a fórmula $(P \rightarrow Q) \wedge P \rightarrow Q$:

P	Q	$P \rightarrow Q$	$(P \rightarrow Q) \wedge P$	$(P \rightarrow Q) \wedge P \rightarrow Q$
0	0	1	0	1
0	1	1	0	1
1	0	0	0	1
1	1	1	1	1

A definição de *consequência sintática* diz que $\alpha_1, \alpha_2, \ldots, \alpha_n \vdash \beta$ é válido sse houver uma prova por algum **sistema de prova** de que é possível derivar a fórmula β à partir das fórmulas $\alpha_1, \alpha_2, \ldots, \alpha_n$, mas não foi dada até agora uma definição do que é um "sistema de prova".

Definição 5.11 (Sistema de Prova)

Um **sistema de prova** para uma lógica é um conjunto de regras que para manipulação dos símbolos e das fórmulas desta lógica.

- Se, para do Γ e todo β, $\Gamma \vdash \beta$ implica $\Gamma \vDash \beta$, dizemos que o sistema de prova é **correto**.

- Se, para do Γ e todo β, $\Gamma \vDash \beta$ implica $\Gamma \vdash \beta$, dizemos que o sistema de prova é **completo**.

Existem diversos sistemas de provas diferentes. Na próxima seção iremos estudar um sistema que busca "imitar" a forma como logicistas constroem provas mentalmente.

5.3.2 Dedução Natural

Como vimos, em princípio, tabelas-verdade podem ser usadas para resolver qualquer questão na lógica proposicional clássica. Entretanto, esse método tem suas desvantagens. O tamanho da tabela cresce exponencialmente com o número de variáveis proposicionais distintas envolvidas nas fórmulas do argumento. Além disso, tabelas-verdade não são familiares aos nossos padrões de pensamento normais. Não se parecem com a forma como nós raciocinamos sobre argumentos. Existe uma outra forma de estabelecer a validade de um argumento que não apresenta estas desvantagens: **o método de dedução natural**. No método de dedução natural se faz uma tentativa de reduzir o raciocínio por trás de um argumento válido a uma série de passos, cada um deles intuitivamente justificado pelas premissas do argumento, ou por outros passos anteriores da série.

Considere o seguinte argumento declarado em linguagem natural:

> Ou pelo de gato ou então pelo de cachorro foi achado na cena do crime. Se foi encontrado pelo de cachorro na cena do crime, o policial Fagundez teve um ataque alérgico. Se pelo de gato foi encontrado na cena do crime, então Meireles é o responsável pelo crime. Mas o policial Fagundes não teve um ataque alérgico. Portanto, Meireles deve ser o responsável pelo crime.

A validade deste argumento pode ser apresentada de modo mais óbvio se representarmos a sequência de raciocínio que leva das premissas à conclusão:

1. Ou pelo de gato ou então pelo de cachorro foi achado na cena do crime. (Premissa)
2. Se foi encontrado pelo de cachorro na cena do crime, o policial Fagundez teve um ataque alérgico. (Premissa)
3. Se pelo de gato foi encontrado na cena do crime, então Meireles é o responsável pelo crime. (Premissa)
4. O policial Fagundes não teve um ataque alérgico. (Premissa)
5. Pelo de cachorro não foi encontrado na cena do crime. (Segue de 2 e 4.)
6. Pelo de gato foi encontrado na cena do crime. (Segue de 1 e 5.)
7. Meireles é responsável pelo crime. (Conclusão. Segue de 3 e 6.)

No desenvolvimento acima, nós não saltamos diretamente das premissas para a conclusão. Ao invés disto, nós mostramos como as inferências intermediarias são utilizadas para justificar a conclusão em uma sequência passo-a-passo. Cada passo na sequência representa uma forma de raciocínio simples e obviamente válida. Neste exemplo, a forma de raciocínio utilizada no item 5 é chamado de *modus tollens*, que envolve deduzir a negação do antecedente de um condicional à partir deste condicional e da negação de seu consequente. A forma de raciocínio utilizada no item 6 é chamada de *silogismo disjuntivo* e permite concluir um dos termos de uma disjunção à partir desta disjunção e da negação do outro termo. Finalmente, no item 7, a conclusão, foi usada uma forma de raciocínio chamada *modus ponens* que permite concluir o consequente de um condicional à partir deste condicional e de seu antecedente.

Definição 5.12 (Sequência de Demonstração)

Uma **sequência de demonstração** para um argumento formal válido é uma sequência de fórmulas na qual cada fórmula é uma premissa do argumento ou o resultado da aplicação de uma regra de dedução do sistema formal de dedução natural às fórmulas anteriores da sequência. A sequência de demonstração termina quando é gerada a conclusão do argumento.

Um sistema de dedução natural consiste em especificar um conjunto de regras de inferência intuitivamente verdadeiras para a construção de sequências de derivações.

As regras de dedução para um sistema de dedução natural devem ser escolhidas com cuidado. Se forem poderosas demais (i.e., permitirem deduzir fórmula com muita facilidade), não preservarão a validade das novas fórmulas e seremos capazes de deduzir qualquer coisa à partir de um dados conjunto de premissas. Se forem fracas demais (i.e., muito restritas no que é possível deduzir), existirão conclusões lógicas que não seremos capazes de provar a partir de um conjunto de premissas. Queremos um sistema lógico formal que seja **correto** (permita demonstrar apenas argumentos válidos) e **completo** (permita demonstrar todos os argumentos válidos). Além disso, a quantidade de regras não deve ser muito grande, de modo a tornar o sistema formal "tratável". Desejamos que o sistema tenha o menor número possível de regras de dedução que o torne completo.

As regras de dedução para lógica proposicional são de dois tipos, de substituição ou equivalência, e de inferência. As regras de substituição permitem que *fbf's* sejam reescritas mantendo o mesmo valor lógico, enquanto as regras de inferência permitem a dedução de novas *fbf's* a partir de *fbf's* anteriores na sequência de demonstração.

Exemplo 5.11

Demonstramos o argumento formal $P \to Q, P \wedge R \vdash Q$:

1	$P \to Q$	hip
2	$P \wedge R$	hip
3	P	simp, 2
4	Q	mp, 1, 3

□

Note que as fórmulas que compõem a sequência de demonstração do Exemplo 5.11 estão todas numeradas para futura referência, e apresentam todas uma justificativa para a sua introdução à direita da fórmula (nas justificativas são usadas as abreviações das regras de inferência). Note ainda que as premissas do argumento formal são justificadas por "hip", de "hipótese".

Exemplo 5.12

Demonstramos o argumento formal $P \vee Q \to R, \neg Q \wedge \neg R \vdash \neg(P \vee Q)$.

1	$P \vee Q \to R$	hip
2	$\neg Q \wedge \neg R$	hip
3	$\neg R$	simp, 2
4	$\neg(P \vee Q)$	mt, 1, 3

□

Não se preocupe em entender as justificativas das fórmulas das sequências de demonstração do Exemplos 5.11 e 5.12 agora. As regras de inferência serão explicada à seguir.

Regras de Inferência

De modo geral, uma regra de inferência diz que, se determinadas fórmulas constarem da sequência de demonstração, então podemos adicionar uma determinada nova fórmula na sequência, de acordo com o padrão estabelecido pela regra de inferência.

Quando se cria um sistema de provas para uma lógica qualquer, sempre é necessário provar que este sistema de provas é "correto", ou seja, que ele é capaz de provar apenas fórmulas válidas. Para isso, cada regra de inferência é analisada individualmente e em conjunto com as outras regras de inferência, tipicamente utilizando uma técnica chamada *indução estrutural* (Apêndice C). Como se pode imaginar, quanto mais regras de inferência, maior é a complexidade de se provar que o sistema de prova é correto. Por essa razão, sempre se busca manter o número de regras do sistema de provas

o menor possível, mas sem comprometer a capacidade do sistema de provar argumentos válidos. Para o sistema de dedução natural para a lógica proposicional, iremos trabalhar com as regras de inferência apresentadas abaixo:

1. Modus Ponens
2. Modus Tolens
3. Conjunção
4. Simplificação
5. Adição
6. Reapresentação

Iremos apresentar e explicar cada uma destas regras de inferência, mas antes disso precisamos apresentar uma definição mais concreta do que é uma regra de inferência.

Definição 5.13 (Regra de Inferência)

Uma regra de inferência é um mecanismo que nos permite transformar uma sequência de demonstração, acrescentando novas fórmulas, desde que certas condições sejam atendidas. As regras de inferência são definidas pela seguinte notação:

$$\frac{\gamma_1, \ldots, \gamma_n}{\alpha}$$

As fórmulas $\gamma_1, \ldots, \gamma_n$, acima do traço, indicam quais fórmulas devem existir na sequência de demonstração original para que a regra de inferência possa ser utilizada. A fórmula α, abaixo do traço, indicam que fórmula sera adicionada à sequência de demonstração caso a regra seja usada.

Tipicamente, ao apresentar uma regra, além de seu nome e de seu esquema, também é apresentada uma abreviação do nome da regra para ser usada na justificativa das sequências de demonstração. Também é usual usar letras do alfabeto grego para representar fórmulas ou partes de fórmulas na definição do esquema das regras de inferência. Veremos agora as regras de inferência básicas da dedução natural para a lógica proposicional.

Modus Ponens

Modus ponens ou *modus ponendo ponens* significa "o modo de afirmar afirmando". Esta regra é apresentada abaixo:

$$\frac{\alpha, \alpha \to \beta}{\beta} \qquad \text{(mp)}$$

Esta regra diz que, se tivermos na sequência de demonstração uma fórmula condicional ($\alpha \to \beta$) e outra fórmula da sequência que seja igual ao antecedente deste condicional (α), podemos acrescentar à sequência de demonstração uma fórmula que seja igual ao consequente deste condicional (β).

Exemplo 5.13

Demonstração do argumento $P, R, P \to (R \to Q) \vdash Q$.

1	P	hip
2	R	hip
3	$P \to (R \to Q)$	hip
4	$R \to Q$	mp, 1, 3
5	Q	mp, 2, 4

□

Modus Tolens

Modus Tollens o *modus tollendo tollens* significa "o modo de negar negando". Esta regra é definida pelo esquema abaixo:

$$\frac{\neg\beta, \alpha \to \beta}{\neg\alpha} \quad \text{(mt)}$$

Esta regra diz que, se tivermos na sequência de demonstração uma fórmula condicional ($\alpha \to \beta$) e outra fórmula da sequência se seja igual à negação do consequente deste condicional ($\neg\beta$), então podemos acrescentar à sequência de demonstração uma fórmula igual à negação do consequente deste condicional ($\neg\alpha$).

Exemplo 5.14

Demonstração do argumento $\neg Q, P \to Q, R \to P \vdash \neg R$.

1	$\neg Q$	hip
2	$P \to Q$	hip
3	$R \to P$	hip
4	$\neg P$	mt, 1, 2
5	$\neg R$	mt, 4, 3

□

Conjunção

A regra de conjunção faz exatamente o que o seu nome indica, i.e., a conjunção de outras fórmulas. O esquema desta regra é dado abaixo:

$$\frac{\alpha, \beta}{\alpha \wedge \beta} \quad \text{(conj)}$$

Assim, a regra de conjunção diz que, se tivermos na sequência de demonstração uma fórmula qualquer (α) e uma outra fórmula qualquer (β), então podemos acrescentar à sequência de demonstração a fórmula formada pela conjunção destas duas ($\alpha \wedge \beta$).

Exemplo 5.15

Demonstração do argumento $P, Q, R \vdash P \wedge Q \wedge R$.

1	P	hip
2	Q	hip
3	R	hip
4	$P \wedge Q$	conj, 1, 2
5	$P \wedge Q \wedge R$	conj, 4, 3

□

Simplificação

O esquema da regra de simplificação é o seguinte:

$$\frac{\alpha \wedge \beta}{\alpha} \text{ ou } \frac{\alpha \wedge \beta}{\beta} \quad \text{(simp)}$$

De certo modo, a regra de simplificação realiza a operação inversa da regra de conjunção, ou seja, dado que temos na sequência de demonstração uma fórmula que já é uma conjunção ($\alpha \wedge \beta$), a regra de simplificação nos permite adicionar à sequência qualquer um dos dois termos da conjunção (α ou β).

Exemplo 5.16

Demonstração do argumento $P \wedge Q \wedge R \vdash R$

1	$P \wedge Q \wedge R$	hip
2	$Q \wedge R$	simp, 1
3	R	simp, 2

□

Adição

A regra de adição costuma precisar de um pouco mais de explicação do que as demais. O esquema da regra de adição é o seguinte:

$$\frac{\alpha}{\alpha \vee \gamma} \qquad \text{(ad)}$$

O que esse esquema nos diz é que, se tivermos uma fórmula qualquer (α) na sequência de demonstração, então podemos acrescentar à sequência a fórmula formada pela disjunção desta fórmula com **qualquer outra fórmula** (γ) que desejarmos ($\alpha \vee \gamma$).

Isso parece pouco intuitivo, mas a justificativa é a seguinte: se α já está na sequência de demonstração, então, por hipótese, α é verdadeiro; se α já é verdadeiro, "fazer o ou" de α com qualquer outra coisa também irá resultar em uma fórmula verdadeira.

Exemplo 5.17

Demonstração do argumento $P \wedge Q \vdash P \vee Q$.

1	$P \wedge Q$	hip
2	P	simp, 1
3	$P \vee Q$	ad $[Q]$, 2

□

Reapresentação

Em algumas situações, ao desenvolver uma sequência de demonstração, é necessário reescrever fórmulas que já foram escritas antes na sequência. Isso tipicamente ocorre ao se desenvolver sub-provas (serão vistas adiante), mas também pode acontecer me outras situações. A regra de *reapresentação* permite que se faça exatamente isso, como mostrado no esquema abaixo:

$$\frac{\alpha}{\alpha} \qquad \text{(R)}$$

É um esquema extremamente simples. O que ele diz é que se temos uma fórmula qualquer (α) na sequência de demonstração, então podemos adicionar essa mesma fórmula (α), novamente, na sequência de demonstração.

Regras de Substituição

No sistema de dedução natural que estamos definindo, é permitido realizar certas substituições de fórmula de certo formato por outras, desde que as fórmulas em questão sejam logicamente equivalentes, como visto na Definição 5.8. As regras que permitem esta reescrita são chamada de *regras de substituição* ou *regras de equivalência*. As regras de substituição diferem das regras de inferência que vimos acima, porque estritamente falando, quando uma regras de substituição é usada, não estamos inferindo nada novo, apenas declarando a mesma coisa mas usando uma combinação diferente de símbolos.

Regras de substituição também diferem das regras de inferência de outras formas. Regras de inferência só podem ser aplicadas quando os operadores principais das fórmulas correspondem exatamente aos padrões dados e só podem ser aplicadas a fórmula completas. Regras de inferência também são estritamente unidirecionais, i.e., deve se inferior o que

está abaixo da linha horizontal da regra, à partir daquilo que está acima da linha e nunca o contrário. Em contrapartida, regras de substituição podem ser aplicadas a partes de fórmulas (sub-fórmulas); além disso, elas podem ser utilizada em qualquer direção.

As regras de substituição disponíveis em nosso sistema de dedução natural são aquelas listada na Seção 5.2.6.

Exemplo 5.18

Consideremos agora outra prova possível para o argumento $P \to Q, P \vdash Q$, que usa exclusivamente regras de substituição.

1	$P \to Q$	hip
2	P	hip
3	$\neg P \vee Q$	cond, 1
4	$P \wedge (\neg P \vee Q)$	conj, 2, 3
5	$(P \wedge \neg P) \vee (P \wedge Q)$	dist, 4
6	$0 \vee (P \wedge Q)$	comp, 5
7	$P \wedge Q$	en, 6
8	Q	simp, 7

□

Sub-provas, e Método Dedutivo

Em alguns casos, embora uma determinada conclusão seja consequência lógica das premissas, as premissas dadas não são suficientes para se construir uma demonstração pelo sistemas de dedução natural. Nestes casos é aceitável fazer suposições, desde que estas suposições sejam corretamente utilizadas.

Durante uma sequência de demonstração, podemos usar uma suposição para criar uma **sub-prova**. Dentro da sub-prova a suposição pode ser utilizada como se fosse uma premissa, porém a sub-prova deve eventualmente ser fechada e devemos sempre voltar para a prova principal. Ao se fechar a sub-prova, podemos acrescentar à prova principal um condicional onde o antecedente é a suposição que abriu a sub-prova, e o consequente é a última fórmula escrita na sub-prova.

Exemplo 5.19

O argumento $P \to Q, Q \to R, R \to S \vdash P \to S$ pode ser demonstrado pela sequência de demonstração abaixo:

1	$P \to Q$	hip
2	$Q \to R$	hip
3	$R \to S$	hip
4	$\quad P$	sup
5	$\quad Q$	mp, 1, 4
6	$\quad R$	mp, 2, 5
7	$\quad S$	mp, 3, 6
8	$P \to S$	fech, 4––7

□

Note que na linha 4 do Exemplo 5.19 é aberta uma sub-prova com a suposição P. Essa sub-prova continua até a linha 7. A linha 8 introduz na prova principal o condicional $P \to S$, que tem como antecedente a suposição que abriu a sub-prova e como consequente a última fórmula que foi escrita na sub-prova.

É possível criar sub-provas dentro de sub-provas se isso for necessário ou útil. Dentro de uma sub-prova é possível utilizar todas as fórmulas que aparecem nas prova e sub-provas mais externas, mas uma vez que uma sub-prova é fechada **nenhuma** das fórmula de dentro da sub-fórmula pode ser usada nos níveis mais externos.

Resumo das Regras de Inferência

As regras de inferências vistas acima, estão sumarizadas na Tabela 5.3.

Nome	Regra	Indicação	Obs.
Modus Ponens	$\dfrac{\alpha,\ \alpha \to \beta}{\beta}$	mp l_i l_j	l_n: número da linha n
Modus Tolens	$\dfrac{\neg\beta,\ \alpha \to \beta}{\neg\alpha}$	mt l_i l_j	
Conjunção	$\dfrac{\alpha,\ \beta}{\alpha \wedge \beta}$	conj l_i l_j	
Simplificação	$\dfrac{\alpha \wedge \beta}{\alpha}$ ou $\dfrac{\alpha \wedge \beta}{\beta}$	simp l_i	
Adição	$\dfrac{\alpha}{\alpha \vee \gamma}$	ad $[\gamma]$ l_i	para qualquer γ
Reapresentação	$\dfrac{\alpha}{\alpha}$	R l_i	

Tabela 5.3: Regras de inferências básicas para dedução natural.

Regras Derivadas

Como foi dito acima, o número de regras de inferência em um sistema de prova para qualquer lógica, é tipicamente mantido o menor possível pois isso facilita a tarefa de demonstrar a correção do sistema. Entretanto, ter poucas regras de inferência significa que as sequências de demonstração irão ficar longas, em alguns casos muito longas. Para evitar que isso aconteça, podemos lançar mão de regras derivadas.

Regras derivadas, são regras que não são estritamente necessárias para que o sistema de dedução natural funcione, mas que ajudam a abreviar sequências de demonstração. Essas regras não aumentam o poder de prova do sistema, e podem ser demonstradas utilizando as regras de inferência básicas – daí o nome "regras derivadas".

Nome	Regra	Uso	Demonstração
Silogismo Hipotético	$\dfrac{\alpha \to \beta, \beta \to \gamma}{\alpha \to \gamma}$	sh l_i l_j	Exercício 5.12
Silogismo Disjuntivo	$\dfrac{\alpha \vee \beta, \neg\alpha}{\beta}$	sd l_i l_j	Exercício 5.13
Contraposição	$\dfrac{\alpha \to \beta}{\neg\beta \to \neg\alpha}$	cont l_i	Exercício 5.14
Contraposição	$\dfrac{\neg\beta \to \neg\alpha}{\alpha \to \beta}$	cont l_i	Exercício 5.15
Autorreferência	$\dfrac{\alpha}{\alpha \diamondsuit \alpha}$	auto l_i	Exercício 5.16
Autorreferência	$\dfrac{\alpha \diamondsuit \alpha}{\alpha}$	auto l_i	Exercício 5.17
Exportação	$\dfrac{\alpha \wedge \beta \to \gamma}{\alpha \to (\beta \to \gamma)}$	exp l_i	Exercício 5.18
Inconsistência	$\dfrac{\alpha, \neg\alpha}{\gamma}$	inc l_i l_j	Exercício 5.19
Distributiva	$\dfrac{\alpha \wedge (\beta \vee \gamma)}{(\alpha \wedge \beta) \vee (\alpha \wedge \gamma)}$	dist l_i	Exercício 5.20
Distributiva	$\dfrac{\alpha \vee (\beta \wedge \gamma)}{(\alpha \vee \beta) \wedge (\alpha \vee \gamma)}$	dist l_i	Exercício 5.21

Tabela 5.4: Regras de inferência derivadas para dedução natural.

A Tabela 5.4 apresenta as regras derivadas mais comuns. Espera-se que a esta altura você já seja capaz de entender o funcionamento das regras apenas através do esquema apresentado na coluna "Regra". A demonstração de que cada uma das regras derivada apresentadas pode, de fato, ser obtida apenas através da aplicação das regras de inferência básicas é deixada como exercício. Na Tabela 5.4, a coluna "Demonstração" indica o número do exercício que demonstra a regra em questão.

5.3.3 Exercícios

Nos exercícios de 5.12 a 5.21 abaixo, demonstre o argumentos formais indicados usando apenas as regras de inferências básicas e a equivalência de dupla negação.

Exercício 5.12 $P \to Q, Q \to R \vdash P \to R$

Exercício 5.13 $P \land Q, \neg P \vdash Q$

Exercício 5.14 $P \to Q \vdash \neg Q \to \neg P$

Exercício 5.15 $\neg P \to \neg Q \vdash Q \to P$

Exercício 5.16 $P \vdash P \land P$

Exercício 5.17 $P \lor P \vdash P$

Exercício 5.18 $P \land Q \to R \vdash P \to (Q \to R)$

Exercício 5.19 $P, \neg P \vdash Q$

Exercício 5.20 $P \land (Q \lor R) \vdash (P \land Q) \lor (P \land R)$

Exercício 5.21 $P \lor (Q \land R) \vdash (P \lor Q) \land (P \lor R)$

5.4 Argumentos Verbais

Na Seção 1.3 falamos um pouco sobre argumentos informais e dissemos que uma das formas de se trabalhar com argumentos é traduzi-los para uma linguagem formal. Sendo assim, podemos verificar a validade de um argumento em um processo de duas etapas:

1. Traduzir o argumento para a forma simbólica utilizando lógica proposicional, como visto na Seção 5.2.
2. Provar que o argumento é válido construindo uma sequência de demonstração para ele usando o sistema de dedução natural.

Veja que, como foi dito na Seção 1.3, o processo de preparação de um argumento informal, para que ele possa ser transformado em um argumento formal pode exigir a introdução de *premissas implícitas*, assim como o descarte de "ruído" no argumento original. É uma tarefa que exige prática e atenção.

Muitas vezes, ao tentar preparar um argumento, assumimos premissas implícitas que não são adequadas. Essa é uma prática perigosa, pois pode nos levar a concluir erroneamente que um argumento é válido, quando na verdade ele é inválido. Veremos abaixo alguns exemplos de argumentos verbais inválidos.

Exemplo 5.20

Considere o argumento:

> "Sandy não é um homem. Portanto, Sandy é uma mulher."

À primeira vista, pode parecer que o argumento é válido, pois costuma-se assumir a premissa implícita de que "Sandy é humano/a". Entretanto, nada nos garante que esta premissa implícita seja válida.

Assim, o argumento é **inválido**. Um contraexemplo para justificar a invalidade do argumento seria: "Sandy é um hamster."

Exemplo 5.21

> "Se a TV está desconectada (da tomada), ela não funciona. A TV não está funcionando. Logo, a TV está desconectada."

O erro deste argumento é interpretar o condicional (se...então...) como se fosse um bicondicional (se e somente se...).

Contraexemplo: A TV está conectada, mas não funciona porque está com defeito.

5.4.1 Indicadores de Argumentos

Definimos um argumento como uma sequencia de frases declarativas, i.e., proposições. Uma das quais deve ser uma conclusão; e as demais, as premissas, se propõem a sustentar a conclusão. Consideramos agora as pistas gramaticais pelas quais as pessoas comunicam tais intenções. As pistas mais importantes são os indicadores de argumento, palavras ou frases que sinalizam a presença e indicam a estrutura dos argumentos. Os indicadores se dividem em duas classes: indicadores de premissa e indicadores de conclusão. Um indicador de premissa é uma expressão como "para", "desde" e "porque", que conecta duas afirmações, significando que uma é a premissa da qual a outra é inferida como uma conclusão.Assim, por exemplo, na frase:

> "A alma é indestrutível porque é indivisível."

O indicador de premissa "porque" indica que a afirmação "é indivisível" é uma premissa que sustenta a conclusão de que "a alma é indestrutível". Indicadores de premissa também podem ocorrer no início das frases, mas a regra ainda se mantém: a declaração à qual o indicador de premissa está anexado é a premissa; a outra é a conclusão. Assim, por exemplo, na frase:

> "Como os números são não físicos, existem objetos não físicos."

A palavra "como" mostra que a afirmação "números não são físicos" é uma premissa que leva à conclusão de que "existem objetos não físicos".

Os indicadores de conclusão são palavras ou frases que significam que a afirmação a que estão ligadas é uma conclusão que se segue de premissas previamente estabelecidas. Inglês é rico em indicadores de conclusão. Alguns dos mais comuns são "portanto", "assim", "assim", "por isso", "então", "segue-se que", "em conclusão", "em conformidade" e "consequentemente". No seguinte argumento, por exemplo, "consequentemente" indica que a terceira afirmação "a consciência existe" é uma conclusão a partir dos dois primeiros:

> "Sem consciência, não pode haver moralidade. Contudo a moralidade existe. Consequentemente a consciência existe."

Argumentos também podem ser declarados sem indicadores, nesses casos devemos confiar em pistas mais sutis de contexto, entonação ou ordem para discernir sua estrutura. Na maioria das vezes quando faltam indicadores de argumento, a conclusão é dada em primeiro lugar, seguida pelas premissas. Aqui está um exemplo:

> "Não há verdade sem pensamento. A verdade é uma correspondência entre pensamento e realidade. E uma correspondência entre duas coisas não pode existir, a menos que as próprias coisas existam."

Aqui a primeira afirmação é uma conclusão das duas restantes.

5.4.2 Tradução e Prova

Ao traduzir um argumento verbal para a forma simbólica, o primeiro passo é preparar o argumento e identificando as premissas, premissas implícitas e conclusões. Em seguida fazemos a simbolização de cada um destes elementos, criando uma chave de simbolização e à partir daí, escrevemos o argumento formal. Em seguida, cria-se a sequência de demonstração para provar a validade do argumentos.

Exemplo 5.22

Considere o seguinte argumento:

> "Se as taxas de juros caírem, o mercado imobiliário vai melhorar. Ou a taxa federal de descontos vai cair, ou então o mercado imobiliário não vai melhorar. As taxas de juros vão cair. Portanto, a taxa federal de descontos vai cair."

Vamos considerar o argumento tal como foi dados, ou seja, sem premissas implícitas. Construímos a seguinte chave de simbolização:

J: As taxas de juros vão cair.

I: O mercado imobiliário vai melhorar.

F: A taxa federal de descontos vai cair.

O argumento formal é:

$$J \to I, F \vee \neg I, J \vdash F$$

A sequência de demonstração para estabelecer a validade do argumento é:

1	$J \to I$	hip
2	$F \vee \neg I$	hip
3	J	hip
4	$\neg I \vee F$	comut, 2
5	$I \to F$	cond, 3
6	$J \to F$	sh, 1, 5
7	F	mp, 3, 6

□

5.4.3 Exercícios

Exercício 5.22 Converta os argumentos verbais dados abaixo para argumentos formais, ou seja, converta as sentenças para a forma simbólica e identifique a conclusão (consequência lógica) indicada no argumento.

1. Ou esta pintura é de Rembrandt ou é de Vermeer. Não é de Rembrandt. Então, deve ser de Vermeer.
2. Café e chá contêm cafeína. Então, o chá contém cafeína.
3. Se o prêmio foi pago, o seguro está em vigor. Mas o prêmio não foi pago. Então, o seguro não está em vigor.
4. Se as pessoas podem viver em Vênus, então elas podem viver em Marte. Se eles podem viver em Marte, então eles podem viver em Júpiter. Portanto, se as pessoas podem viver em Vênus, elas podem viver em Júpiter.
5. A casa será vendida até agosto, ou não será vendida este ano. Não será vendido até agosto. Então, não será vendido este ano.
6. Se George não estiver atrasado para a reunião, ele apresentará o orador. Mas George estava atrasado para a reunião. Então, ele não apresentará o orador.
7. Rotterdam está na Holanda ou na Europa. Rotterdam está na Holanda. Então, Roterdã não está na Europa.
8. O cão não vai latir, se a criança não o assustar. A criança não vai assustar. Então, o cachorro não vai latir.
9. Se chover, então as ruas estão molhadas. Se congelar, as ruas ficam escorregadias. Está chovendo ou congelando. Então, as ruas estão molhadas ou escorregadias.

10. Se chover e congelar, as ruas estarão molhadas e escorregadias. Está chovendo ou congelando. Então, as ruas estão molhadas ou escorregadias.

Exercício 5.23 Converta os argumentos verbais dados abaixo para argumentos formais, ou seja, converta as sentenças para a forma simbólica e identifique a conclusão (consequência lógica) indicada no argumento.

1. Se George ou Liz foram à festa, Tom e Susan ficaram chateados. Liz, como se viu, não foi, mas Tom e Susan ainda estavam chateados. Portanto, George de fato foi à festa.

2. Se Al não está cantando, Bo não está dançando. Ou Bo ou Clyde está dançando. Então, se Clyde não está dançando, então Al está cantando.

3. A orquestra não tocará ambos Stravinski e Mozart hoje à noite. Eles vão, como sabemos, tocar Mozart hoje à noite. Devemos concluir, portanto, que eles não vão tocar Stravinski hoje à noite.

4. Não é verdade que tanto você não pode ir nos passeios de crianças quanto você não pode ir nos passeios de adultos. Você, naturalmente, não pode ir nos passeios infantis. Portanto, você pode ir nos passeios de adultos.

5. Sua carteira de motorista não será revogada se ele não violar a lei. Mas ele deve ter violado a lei, porque sua licença foi revogada.

6. Se esta escola irá sobreviver, deve aumentar sua anuidade (a fim de compensar as despesas). Mas, se esta escola quiser sobreviver, não pode aumentar sua anuidade (para se manter competitiva). Então, essa escola definitivamente não vai sobreviver.

7. Se essa criatura não tiver dentes, ela não morderá. Ai! Bem, não é o caso que não morde. Então, não é o caso que não tem dentes.

8. Eles venceram a batalha e é falso que eles não venceram a guerra. Então, eles venceram a guerra, e não é verdade que eles não venceram a batalha.

9. Se algum número N for o maior número possível, então N é o maior número possível (por hipótese) e N não é o maior número possível (desde que você possa adicionar 1 a ele). Então, é falso que algum número N seja o maior número possível.

10. Paris, Londres ou Roma sediarão a Convenção do Vinho este ano. Se Paris o fizer, os vinhos franceses vencerão. Se Londres, então, os vinhos britânicos vão ganhar. Se Roma o fizer, os vinhos italianos vencerão. Os vinhos britânicos não vão ganhar este ano. Então, vinhos franceses ou italianos ganharão este ano.

Exercício 5.24 Converta os argumentos verbais dados abaixo para argumentos formais, ou seja, converta as sentenças para a forma simbólica e identifique a conclusão (consequência lógica) indicada no argumento.

1. Robert conhece o latim se e somente se ele não sabe grego. Ele sabe grego. Então, Robert não sabe latim.

2. Beth irá, se James pedir a ela. Se Mateus pedir a ela, Beth também irá. Se James não pedir a ela, Mateus pedirá. Então, Beth irá.

3. A música não é de Vivaldi, a menos que o estilo seja barroco. Se o estilo é romântico, então não é barroco. Então, se a música é de Vivaldi, então não é romântico.

4. Mateus terá comido, se Beth cozinhou. Mateus não comeu a menos que James também tenha comido. Então, James comeu apenas se Beth cozinhou.

5. Laura dará a palestra a menos que ninguém apareça. Felizmente, as pessoas apareceram. Então, ela deu a palestra.

6. Se o anfitrião conhece o senador, então o senador será convidado. Se a anfitriã gostar do senador, ele também será convidado. Mas nem o anfitrião o conhece nem a anfitriã gosta dele. Então, o senador não será convidado.

7. Eles não venderão a casa somente se puderem pagar a hipoteca. Mas eles estão vendendo a casa. Então, eles não podem pagar a hipoteca.

8. Samantha não correrá a menos que o tempo não esteja quente. Assim, ela correr quando o tempo está quente não vai acontecer.

9. Este cilindro é quadrado apenas se não for redondo. Mas, mesmo que o cilindro seja quadrado, ele ainda precisa ser redondo. Mas qualquer tolo sabe que não pode ser tanto redondo quanto não redondo. Então, esse cilindro não pode ser quadrado.

10. Se a demanda por esses produtos aumentar, o preço subirá. Além disso, a demanda por esses produtos aumentará somente se o emprego subir. No entanto, a demanda não aumentou. Então, o preço ou o emprego não está subindo.

11. Alguns funcionários devem ser demitidos, se o orçamento for reduzido. Não haverá um aumento salarial a menos que o orçamento não seja congelado. O orçamento será reduzido ou congelado. Então, alguns funcionários devem ser demitidos ou não haverá um aumento salarial.

12. Se a pressão for muito baixa, o motor não funcionará. E, se a pressão for muito alta, o motor não funcionará. Então, se o motor funcionar, a pressão não é nem muito baixa nem muito alta.

13. O proprietário pode despejar o inquilino apenas se o inquilino não tiver satisfeito os termos do contrato de arrendamento. O inquilino não satisfez os termos do contrato, a menos que o aluguel seja pago. Então, se o aluguel não for pago, o senhorio pode despejar o inquilino.

14. Desde que Albert saiba cálculo, ele conhece álgebra. Mas ele não conhece cálculo se não conhece a trigonometria. Então, ele não sabe trigonometria a menos que ele não saiba álgebra.

15. Joe não conhece cálculo, mas conhece álgebra. Mas ele não conhece a trigonometria sem conhecer o cálculo. Então, Joe sabe álgebra, mas ele não sabe trigonometria.

5.5 Exercícios de Revisão

Exercício 5.25 Para cada sentença abaixo, indique se ela é uma tautologia, uma contradição, ou um contingência. Justifique utilizando tabelas-verdade ou dedução natural.

1. Fumaça $\rightarrow$ Fumaça
2. Fumaça $\rightarrow$ Fogo
3. (Fumaça $\rightarrow$ Fogo) $\rightarrow$ ($\neg$Fumaça $\rightarrow$ $\neg$Fogo)
4. Fumaça $\vee$ Fogo $\vee$ $\neg$Fogo
5. ((Fumaça $\wedge$ Calor) $\rightarrow$ Fogo) $\iff$ ((Fumaça $\rightarrow$ Fogo) $\vee$ (Calor $\rightarrow$ Fire))
6. (Fumaça $\rightarrow$ Fogo) $\rightarrow$ ((Fumaça $\wedge$ Calor) $\rightarrow$ Fogo)
7. Grande $\vee$ Bobo $\vee$ (Grande $\rightarrow$ Bobo)
8. (Grande $\wedge$ Bobo) $\vee$ $\neg$Bobo

Exercício 5.26 Para cada fórmula bem formada abaixo, determine se ela é uma tautologia, uma contradição, ou uma contingência. Justifique utilizando tabelas-verdade ou dedução natural.

1. $((P \rightarrow (Q \vee R)) \wedge \neg(Q \vee R)) \rightarrow \neg P$
2. $((P \vee Q) \rightarrow \neg R) \wedge (\neg R \vee (Q \vee P))$

Exercício 5.27 A afirmação abaixo está correta?[5] Demonstre por quê sim, ou por quê não.

$$\bot \vDash \alpha \qquad \text{para qualquer fórmula bem formada } \alpha$$

[5] Lembre-se que $\bot$ significa "falso".

Exercício 5.28 Para cada um dos argumentos verbais abaixo, determine se o argumento é válido, ou seja, traduza o argumento para a forma simbólica e utilize dedução natural para verificar o argumento.

1. "Se a temperatura e a pressão são constantes, então não chove. A temperatura permaneceu constante. Logo, se choveu, então a preção não permaneceu constante."

2. "Homens sempre comem quando estão com fome. João sempre veste seu melhor terno para comer. João não está com fome. Portanto, João não está usando seu melhor terno."

Exercício 5.29 Considere o texto abaixo:

> "Há apenas dois formatos de fotos: redonda e quadrada. Fotos são ou coloridas ou então em preto-e-branco. Deixe eu te falar sobre a foto que eu achei ontem. Se a foto for quadrada, então ela é em preto-e-branco. Se ela for redonda, ela é uma foto digital colorida. Se a foto for digital ou em preto-e-branco, então ela é um retrato. Se ela for um retrato, então é uma foto do meu amigo."

1. Construa uma base de conhecimento, i.e., um conjunto de fórmulas, que represente as declarações sobre a foto que o autor do texto anterior encontrou. Use as variáveis proposicionais A, B, C, D, E, F, e G, conforme definidas abaixo:

Símbolo	Significado
A	A foto é colorida
B	A foto é em preto-e-branco
C	A foto é quadrada
D	A foto é redonda
E	A foto é digital
F	A foto é um retrato
G	A foto mostra o meu amigo

   a) Há apenas dois formatos de fotos: redonda e quadrada.

   b) Fotos são ou coloridas ou então em preto-e-branco.

   c) Se a foto for quadrada, então ela é em preto-e-branco.

   d) Se ela for redonda, ela é uma foto digital colorida.

   e) Se a foto for digital ou em preto-e-branco, então ela é um retrato.

   f) Se ela for um retrato, então é uma foto do meu amigo.

2. Usando o sistema de dedução natural, é possível responder à pergunta do autor: "A foto que eu achei ontem é mostra o meu amigo?"

Exercício 5.30 Considere o texto abaixo:

> "Se o unicórnio é mítico, então ele é imortal, mas se ele não é mítico, então ele é um mamífero mortal. Se o unicórnio é imortal ou um mamífero, então ele tem chifre. O unicórnio é mágico se ele tem chifre."

É possível provar que o unicórnio é mítico? É possível provar que ele é mágico? Que ele tem chifre?

Exercício 5.31 Suponha que desejamos jogar *Campo Minado* em um tabuleiro de n_r linhas por n_c colunas, i.e., um tabuleiro de $n_r \times n_c$ células, que contém m minas invisíveis espalhadas aleatóriamente.

- Considere que a variável proposicional X_{ij} seja verdadeira se houver uma mina na célula (i, j) do tabuleiro. Escreva uma fórmula em lógica proposicional que descreva o fato de que existem exatamente duas minas nas células adjacentes à célula $(1, 1)$ do tabuleiro. Escreva a *fbf* utilizando proposições do tipo X_{ij}

- Generalize a fórmula obtida no item anterior explicando como seria possível construir uma *fbf* descrevendo o fato de que existem exatamente k minas nas células adjacente uma célula (i, j) qualquer.

Exercício 5.32 Assumindo que uma pessoa mentirosa sempre mente e uma pessoa honesta sempre diz a verdade, formalize os argumentos abaixo, e use os métodos de demonstração lógica para resolver os enigmas.

1. Vamos ouvir Alceo, Safo e Catulo.

   Alceo "Os únicos honestos aqui somos Catulo e eu."
   Safo "Catulo é um mentiroso."
   Catulo "Safo fala a verdade. Ou talvez seja Alceo que mentiu."

   Quem é honesto? Quem é mentiroso?

2. Vamos ouvir Anaximandro, Parménides e Heráclito.

   Anaximandro "Não se deve confiar em Heráclito."
   Parménides "Anaximandro e Heráclito nunca mentem."
   Heráclito "Parménides falou a verdade."

   Quem é honesto? Quem é mentiroso?

Exercício 5.33 Usando lógica proposicional e dedução natural (raciocínio em linguagem natural não é aceito), resolva o seguinte enigma.
No senado Romano:

Marco Antônio "Foi Cassius ou Brutus (ou ambos)."

Cassius "Não fui eu. Marco Antônio está mentindo."

Brutus "Não fui eu."

Assumindo que um mentiroso sempre mente, e um Romano honesto sempre fala a verdade, e que apenas um deles está dizendo a verdade, quem é o culpado? Quem está dizendo a verdade?

Exercício 5.34 Existem três baús (nas cores vermelho, verde e azul) com as inscrições:

Vermelho "O tesouro está aqui."

Verde "O tesouro não está aqui."

Azul "O tesouro não está no baú vermelho."

Sabendo que apenas um baú contém o tesouro, e que no máximo uma das inscrições está correta, é possível decidir qual baú contém o tesouro?
Usando os átomos abaixo:

- R – "O baú vermelho contém o tesouro."
- G – "O baú verde contém o tesouro."
- B – "O baú azul contém o tesouro."
- RR – "A inscrição no baú vermelho está correta."
- GG – "A inscrição no baú verde está correta."
- BB – "A inscrição no baú azul está correta."

1. Escreva uma base de conhecimento, i.e., um conjunto de fórmulas, que codifique o enigma.
2. Usando dedução natural demonstre que as seguintes fórmula são consequências lógicas da sua base de conhecimento.
   a) $\neg(RR \wedge \neg BB \wedge \neg GG)$
   b) $\neg(\neg RR \wedge BB \wedge \neg GG)$
   c) $\neg(\neg RR \wedge \neg BB \wedge \neg GG)$
   d) $(\neg RR \wedge \neg BB \wedge GG)$
3. Com os resultados acima é possível determinar qual baú contém o tesouro? Caso seja, em qual está? Justifique sua resposta.

Capítulo 6

Demonstração de Correção de Algoritmos

Sumário

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Capítulo 7

Lógica de Predicados

Sumário

7.1 Introdução

A principal característica que distingue a lógica de predicados dos outros ramos mais complexos da lógica é que a lógica proposicional não lida com os conceitos de relações lógicas e de propriedades que aparecem como partes menores das sentenças. Isso significa que a lógica proposicional é um sistema relativamente rudimentar com um poder de expressão muito limitado, onde a única relação possível de estabelecer entre as várias sentenças é a de identidade – ou sua negação, a diferença. Considere, por exemplo, o argumento abaixo.

$$\begin{array}{ll} A: & \text{Todo humano e mortal.} \\ B: & \text{Sócrates é humano.} \\ \hline \therefore C: & \text{Sócrates é mortal.} \end{array}$$

O silogismo[1] acima é evidentemente válido, entretanto, se formalizarmos esse silogismo na lógica proposicional teremos $A, B \vdash C$, que obviamente não é um argumento válido.

A validade do argumento acima deriva do fato de que a *premissa menor*, B, torna Sócrates uma instância do sujeito da *premissa maior*, A – qualquer coisa que se aplique a todo humano, se aplica a um humano em particular.

Para reconhecer a validade do argumento acima é preciso reconhecer que o sujeito da premissa B é uma instância do sujeito da premissa A. Entretanto, na lógica proposicional as sentenças (proposições atômicas) são consideradas

[1] Aristóteles usava a palavra *silogismo* para indicar um raciocínio perfeito, uma inferência que fosse evidentemente verdadeira.

indivisíveis e as relações logica e propriedades que envolvem partes de declarações, como seus sujeitos e predicados, não são levados em consideração.

Buscando aumentar o poder de expressividade, várias outras lógicas foram propostas. Aquela que é considerada a *sucessora direta* da Lógica de Predicados Clássica é a *Lógica de Predicados de Primeira Ordem Clássica*, também chamada de *Lógica de Primeira Ordem*, de *Cálculo de Predicados de Primeira Ordem* ou, simplesmente, de *Lógica de Predicados*.

7.2 Cálculo de Predicados

Em termos intuitivos (não formais) a "visão de mundo" que serve de base para a lógica de predicados pode ser resumida como se segue:

1. Existe um *domínio de discurso* U que contém todas as entidades que nos interessam.
2. Existem métodos para selecionar uma entidade em particular dentro do domínio.
   a) Podemos utilizar nomes para determinadas entidades. Por exemplo, 'Sócrates' pode ser usado para designar um indivíduo em particular. Esses nomes correspondem a contantes (ou, a funções com zero argumentos). Por exemplo, podemos utilizar as letras *so* para representar 'Sócrates'.
   b) Podemos designar um indivíduo por uma descrição como 'o pai de Sócrates'. Aqui, 'o pai de ...' é é uma função que recebe um argumento, i.e., um elemento arbitrário de U, e *aponta* para outro elemento de U. Por exemplo, podemos usar $f(x)$ para representar "o pai de x", assim, se *so* representa 'Sócrates', então $f(so)$ representa "O pai de Sócrates".
   c) Para permitir formas mais flexíveis de expressão, podemos usar variáveis (com ou sem índices subscritos), tais como x, y, z, x_1, y_1, z_1, etc. para representar entidades arbitrárias de U em expressões mais complexas.
3. As entidades de U podem ser classificadas e relacionadas entre si utilizando predicados e relações:
   a) Podemos identificar subconjuntos de U através de predicados unários que representam características. Por exemplo: $H(x)$ é um predicado verdadeiro para os elementos de U que forem humanos, e.g $H(\text{Sócrates})$ – esse predicado pode ser usado para identificar o subconjunto de U que representa os humanos, i.e., $\{x \in U \mid H(x)\}$; $M(x)$ é verdadeiro para os elementos de U que são mortais; $A(x)$ é verdadeiro para os elementos que são animais; $C(x)$ é verdadeiro para os elementos que são cavalos; etc.
   b) Os elementos de U podem participar de várias relação entre si. Por exemplo: "x é mais velho que y", pode ser representado por $O(x, y)$; ou "x é o primeiro elemento da lista xs", representado por $\mathit{First}(x, \mathit{xs})$.
4. É muito importante perceber a diferença entre o **predicado** $\mathit{First}(x, \mathit{xs})$, "$x$ é o primeiro elemento da lista *xs*", e a **função** $\mathit{first}(\mathit{xs})$, que retorna, para cada lista *xs*, o seu primeiro elemento. A função *first* simplesmente designa um elemento do domínio U, enquanto o predicado *First*, estabelece uma relação entre dois elementos de U. Um predicado sempre estabelece um fato, i.e., sempre pode ser avaliado como verdadeiro ou falso. Da mesma forma que na lógica proposicional, os fatos atômicos podem ser combinados para formar fatos mais complexos utilizando os operadores lógico "não", "e", "ou", "se ..., então ...", etc.
5. Além de poder estabelecer fatos sobre entidades conhecidas, como 'Sócrates', podemos também estabelecer fatos sobre entidades indefinidas, como por exemplo: "para todo $x \in U$: $H(x) \rightarrow M(x)$", "não existe x: $H(x) \wedge C(x)$". Esses fatos são chamados de fatos *quantificados*.

Considerando a interpretação introduzida acima, o argumento do início deste capítulo poderia ser traduzido como segue.

Todo humano é mortal.	$\Longrightarrow$	*para-todo* x:	$H(x) \rightarrow M(x)$
Sócrates é humano.	$\Longrightarrow$		$H(so)$
Portanto, Sócrates é mortal.	$\Longrightarrow$	$\therefore$	$M(so)$

Considere ainda o exemplo a seguir.

Exemplo 7.1

Seja o argumento dado abaixo:

Todo cavalo é um animal.

Portanto, toda cabeça de cavalo é uma cabeça de animal.

Considerando uma interpretação em que:

$A(x)$: x é um animal.

$C(x)$: x é um cavalo.

$Hd(x, y)$: x é a cabeça de y.

Podemos reescrever o argumento como:

$$\frac{\text{para-todo } x : C(x) \to A(x)}{\therefore \text{ para-todo } x : ((\text{existe } y : (C(y) \wedge Hd(x, y)) \to (\text{existe } y : (A(y) \wedge Hd(x, y)))))}$$

Os conceitos apresentados nesta seção serão vistos com mais detalhes nas seções seguintes.

7.2.1 Predicados e Variáveis

Considere a seguinte sentença:

Todo estudante é mais jovem que algum instrutor. (7.1)

Na lógica proposicional, nós poderíamos representar essa sentença com uma proposição atômica P. Entretanto, esta representação não captura a estrutura lógica mais detalhada da sentença. Sobre o que é esta sentença? Ela é sobre *ser um estudantes*, sobre *ser um instrutor*, e sobre *ser mais jovem do que alguém*. Estes três conceitos são todos propriedades de algum tipo, então nós precisamos de um mecanismo para expressar estas propriedades, bem como de formas de representar seus relacionamentos e dependências.

Na lógica de primeira ordem, nós utilizamos **predicados** para representar propriedades (e relações) das entidades (elementos) do domínio. Por exemplo, nós podemos escrever $S(paulo)$ para representar que Paulo é um estudante e $L(ana)$ para representar que Ana é uma instrutora. Também podemos escrever $J(paulo, ana)$ para representar que Paulo é mais jovem do que Ana. Os símbolos S, L e J são chamados de predicados. Sempre é necessário definir precisamente o significado dos predicados para evitar ambiguidades. Por exemplo, $J(paulo, ana)$ pode representar tanto que "*paulo* é mais jovem que *ana*", quanto o contrário.

Frequentemente, quando estamos descrevendo fatos e situações, não estamos preocupados em designar entidades em particular (como Paulo e Ana), mas queremos descrever estes fatos ou situações de modo genérico, sem dar nomes à entidades envolvidas. Nestes casos é conveniente o uso de *variáveis*. Variáveis são escritas como u, v, w, x, y, z, ...ou x_1, y_3, z_5, ...e podemos pensar nelas como *marcadores de lugar* para valores concretos (como Paulo, Ana ou Sócrates). Usando variáveis é possível definir os significados dos predicados S, L e J formalmente:

$S(x)$: x é um estudante
$L(x)$: x é um instrutor
$J(x, y)$: x é mais jovem que y.

Note que os nomes das variáveis não são importantes, desde que sejam utilizados consistentemente. Poderíamos definir exatamente o mesmo significado para L escrevendo

$L(y)$: y é um instrutor

ou ainda

$L(z)$: z é um instrutor.

A possibilidade de usar variáveis ainda não é suficiente para capturar todo o significado da sentença (7.1). Temos que transmitir o significado de "**Todo** estudante x é mais jovem que **algum** instrutor y." Neste ponto precisamos introduzir a ideia de quantificadores para representar expressões como "todo", "para todo", "algum", etc. Ou seja, quantificadores expressa a ideia de quantidade. Na lógica de primeira ordem dispomos de dois quantificadores: (a) o quantificador

universal $\forall$ (lido "para todo"); e (b) o quantificador existencial $\exists$ (lido "existe" ou "para algum"). Os quantificadores sempre vêm conectados a alguma variável, como em $\forall x$ ("para todo x") ou $\exists z$ ("existe z", ou "para algum z").

Utilizando os quantificadores, agora é possível escrever a sentença (7.1) em uma forma completamente simbólica:

$$\forall x \left(S(x) \rightarrow \exists y \left(L(y) \wedge J(x, y)\right)\right)$$

Na verdade, a codificação, ao ser traduzida de volta para o português resulta em um texto diferente do original, mas que preserva o mesmo significado. A re-tradução fica da seguinte forma:

> *Para todo x, se x é um estudantes, então existe algum y tal que y é instrutor e x é mais jovem que y.*

Predicados diferentes podem ter números de argumentos diferentes. No caso acima, os predicados S e L têm apenas um argumento – e são chamados de *predicados unário*; enquanto o predicado J tem dois argumentos – e é chamado de *predicado binário*. Na lógica de primeira ordem, predicados podem ter qualquer quantidade finita de argumentos. A quantidade de argumentos que um predicado recebe é chamada de *aridade* do predicado.

Abaixo temos outro exemplo.

Exemplo 7.2

Considere a sentença:

> *Nem todos os pássaros podem voar.*

Para representá-la nós escolhemos os predicados unários P e V com os significados:
$P(x)$: x é um pássaro
$V(x)$: x pode voar.
Assim, a sentença pode ser traduzida como:

$$\neg \left(\forall x \left(P(x) \rightarrow V(x)\right)\right)$$

que significa: "Não é verdade que para todo x se x é pássaro, então x pode voar." Alternativamente, poderíamos ter adotado a codificação:

$$\exists x \left(P(x) \wedge \neg V(x)\right)$$

que significa: "Existe algum x tal que x é pássaro e x não pode voar."

Embora a primeira tradução apresentada seja mais próxima da estrutura da sentença original, as duas traduções são semanticamente equivalente, i.e., tem o mesmo significado. Veremos o porquê desta equivalência nas próximas seções.

Ser capaz de codificar fatos complexos expressos em português como fórmulas na lógica de predicados é importante — e.g., em design de software com UML ou em especificações formais para sistemas críticos — e é necessário ter muito mais cuidados ao fazer a codificação do que na lógica proposicional. Entretanto, uma vez que essa tradução tenha sido realizada, o nosso principal objetivo passa a ser o raciocínio simbólico ou semântico sobre a informação expressa por estas fórmulas. Ou seja, uma vez que a tradução tenha sido realizada nós *abstraímos* a interpretação e nos concentramos apenas na manipulação simbólica das fórmulas.

7.2.2 Quantificadores

Quando construímos frases para descrever situações, frequentemente utilizamos pronomes ou substantivos para nos referirmos a elementos conhecidos, por exemplo:

a) "Aquele carro é um Corolla."

b) "Antônio é o gerente do setor de vendas."

c) "Esta pessoa é mais alta do que a média."

Na linguagem da Lógica de Primeira Ordem, estes elementos conhecidos e definidos são representados por constantes. Assim, as sentenças acimas poderiam ser traduzidas para as seguintes fórmulas, com suas respectivas interpretações:

a) "Aquele carro é um Corolla."
Fórmula: $\text{Carro}(a) \wedge \text{Modelo}(a, c)$
Interpretação:

- a: aquele carro
- c: Corolla
- $\text{Carro}(x)$: x é um carro
- $\text{Modelo}(x, y)$: o modelo de x é y

b) "Antônio é o gerente do setor de vendas."
Fórmula: $\text{Gerente}(a, v)$
Interpretação:

- a: Antônio
- v: setor de vendas
- $\text{Gerente}(x, y)$: x é o gerente de (do setor/seção) y

c) "Esta pessoa é mais alta do que a média."
Fórmula: $\text{Altura}(p, h) \wedge h > h_m$ *Interpretação:*

- p: esta pessoa
- h_m: altura média das pessoas
- $\text{Altura}(p, h)$: a altura da pessoa p é h

Entretanto, frequentemente desejamos falar sobre situações que se aplicam não apenas a um objeto conhecido, mas a "todos" os objetos com alguma característica, ou a "algum" objeto de uma categoria, ou que algo é verdadeiro para "nenhum" objeto, ou (equivalentemente) que é falso para todos, ou que "nem todos" os objetos de algum tipo tem certas características, e assim por diante. Para expressar esse tipo de ideia mais geral, utilizamos elementos lógicos chamados de quantificadores. Uma fórmula lógica construída com o uso de quantificadores é chamada de uma *fórmula quantificada*. Os dois quantificadores mais comuns são o quantificador **universal**, que expressa a ideia de "para todo", e o quantificador **existencial**, que expressa a ideia de "existe" (no sentido de "existe ao menos um"). O símbolo tradicional para o quantificador universal "$\forall$", uma letra "A" girada, e para o quantificador existencial é "$\exists$", uma letra "E" girada.

Utilizando os quantificadores universal e existencial é possível construir fórmulas como $\forall x.P(x)$ (lê-se "para todo x $P(x)$ é verdadeiro") e $\exists x.P(x)$ (lê-se "existe x tal que $P(x)$ é verdadeiro"). Também é possível construir fórmulas mais complexas tais como $\forall x.(P(x) \rightarrow Q(x))$ ou $\exists x.(P(x) \wedge Q(x))$, etc.

É necessário tomar algum cuidado ao construir fórmulas quantificadas. A *semântica*, i.e., o significado, dos quantificadores universal e existencial nem sempre é exatamente como estamos acostumados a pensar quando construímos sentenças em linguagem natural. A semântica formal dos quantificadores na lógica pode ser interpretado da seguinte forma:

Fórmula Universalmente Quantificada – Uma fórmula do tipo $\forall x.\alpha[x]$ será verdadeira a menos que exista um valor possível para x que torne $\alpha[x]$ falso.

Fórmula Existencialmente Quantificada – Uma fórmula do tipo $\exists x.\alpha[x]$ sera falsa a menos que exista um valor possível para x que torne $\alpha[x]$ verdadeira.

Essa semântica gera duas situações que frequentemente não são consideradas pelas pessoas que utilizam os quantificadores:

1. No uso do quantificador universal, frequentemente a sentença será verdadeira mesmo que não exista nenhum elemento no domínio de discurso que satisfaça à fórmula construída. Por exemplo, considere a sentença "Todos os professores venusianos sabem falar Latim." Esta sentença poderia ser traduzida para a forma simbólica com a seguinte fórmula e interpretação:

   Fórmula: $\forall x.\,(\text{Venusiano}(x) \wedge \text{Professor}(x) \rightarrow \text{Fala}(x, \textit{latim}))$
   Interpretação:

- Venusiano(x): x é venusiano
- Professor(x): x é professor
- *latim*: latim

À primeira vista, pelo senso comum, poderíamos pensar que esta sentença é falsa, pois não existe nenhum professor venusiano[2], logo, não faria sentido fazer afirmações sobre quais línguas eles falam. Entretanto, é exatamente por não haver professores venusianos que a sentença é verdadeira. Como não existem professores venusianos, não existe nenhum valor para a variável x que faça a fórmula "Venusiano(x) $\wedge$ Professor(x) $\rightarrow$ Fala(x, *latim*)" ser falsa. Portanto, a fórmula quantificada será verdadeira.

2. No uso do quantificador existencial, frequentemente a sentença será verdadeira mesmo que todos os elementos do domínio de discurso, para aquela categoria, satisfaçam à fórmula construída. Por exemplo, considere a sentença "Existe algum animal vertebrado com sistema nervoso."

7.3 Linguagem Formal da Lógica de Predicados

Nesta seção daremos regras sintáticas para a formação de fórmulas da lógica de predicados. Por causa do poder da lógica de predicados, a linguagem é muito mais complexa que a da lógica proposicional.

Com já mencionado em Definição 5.3, uma *fórmula bem formada*, abreviada como FBF, ou simplesmente *fórmula*, é uma sequencia finita de símbolos de um determinado alfabeto que faz parte de uma linguagem formal.

A primeira coisa a notar é que existem dois tipos de coisas envolvidas em uma fórmula da lógica de predicados. O primeiro tipo denota os objetos sobre os quais estamos falando: indivíduos como a e p (referindo-se a "André" e "Paulo") são exemplos, assim como variáveis como x e v. Símbolos de função também nos permitem referir objetos: assim, $m(a)$ e $g(x, y)$ também são objetos. Expressões na lógica de predicados que denotam objetos são chamadas de **termos**.

O outro tipo de coisas na lógica de predicados denota valores-verdade, ou valores lógicos; expressões desse tipo são **fórmulas**: $Y(x, m(x))$ é uma fórmula, embora x e $m(x)$ sejam termos.

Um **vocabulário** da lógica de predicados consiste em três conjuntos: um conjunto de símbolos de predicado $\mathcal{P}$, um conjunto de símbolos de função $\mathcal{F}$ e um conjunto de símbolos constantes $\mathcal{C}$. Cada símbolo de predicado e cada símbolo de função vem com uma *aridade*, i.e., o número de argumentos esperados. De fato, as constantes podem ser consideradas como funções que não aceitam nenhum argumento (e até deixamos de lado os parênteses de argumentos) – portanto, para simplificar a formulação, vamos considerar que as constantes residem no conjunto $\mathcal{F}$ junto com as funções "verdadeiras" que levam argumentos. A partir de agora, vamos descartar o conjunto $\mathcal{C}$, já que é conveniente fazer isso, e estipular que constantes são funções de aridade-0, chamadas de *funções nulárias*.

7.3.1 Termos

Os termos da nossa linguagem são compostos de variáveis, símbolos constantes e funções aplicadas a eles. As funções podem ser aninhadas, como em $m(m(x))$ ou $g(m(a), c)$.

Definição 7.1 (Termos)

Termos são definidos como se segue:

- Qualquer variável é um termo.
- Se $c \in \mathcal{F}$ é uma função nulária, então c é um termo.
- Se $t_1, t_2, \ldots, t_n$ são termos e $f \in \mathcal{F}$ tem aridade $n > 0$, então $f(t_1, t_2, \ldots, t_n)$ é um termo.
- Nada mais é um termo.

Na forma de Backus Naur (BNF) podemos escrever:

$$t ::= x \mid c \mid f(t, \ldots, t)$$

[2]Pelo menos, até onde seja do nosso conhecimento.

onde x varia sobre um conjunto de variáveis var, c sobre símbolos de função nulos em $\mathcal{F}$ e f sobre aqueles elementos de $\mathcal{F}$ com aridade $n > 0$.

É importante notar que:

- os primeiros blocos de construção de termos são constantes (funções nulas) e variáveis;
- termos mais complexos são construídos a partir de símbolos de função usando tantos termos previamente construídos quanto requerido por tais símbolos de função; e
- a noção de termos depende do conjunto F. Se você alterá-lo, você altera o conjunto de termos.

Exemplo 7.3

Suponha que n, f e g sejam símbolos de função, respectivamente nulário, unário e binário. Então $g(f(n), n)$ e $f(g(n, f(n)))$ são termos, mas $g(n)$ e $f(f(n), n)$ não são (eles violam as aridades). Suponha $0, 1, \ldots$ são nulários, s é unário e $+$, $-$ e $\times$ são binários. Então $\times(-(2, +(s(x), y)), x)$ é um termo. Normalmente, os símbolos binários são escritos de notação infixa em vez de prefixa; assim, o termo geralmente é escrito $(2 - (s(x) + y)) \times x$.

7.3.2 Fórmulas

Uma fórmula é um objeto sintático que pode receber um significado semântico por meio de uma interpretação. Dois usos principais de fórmulas estão na lógica proposicional e na lógica de predicados.

A definição de uma fórmula na lógica de primeira ordem é relativa à *assinatura*[3] da teoria em questão. Essa assinatura especifica os símbolos constantes, os símbolos de relação, e os símbolos de função da teoria em questão, juntamente com as aridades dos símbolos de função e relação.

A escolha dos conjuntos $\mathcal{P}$ e $\mathcal{F}$ para símbolos de predicados e de funções, respectivamente, é guiada pelo que pretendemos descrever. Por exemplo, se trabalharmos em um banco de dados representando as relações entre nossos parentes, poderemos considerar $\mathcal{P} = \{M, F, S, D\}$, referindo-se a *ser homem*, *ser mulher*, *ser filho de ...* e *ser filha de ...*. Naturalmente, F e M são predicados unários (eles aceitam um argumento), enquanto D e S são binários. Da mesma forma, podemos definir $\mathcal{F} = \{\textit{mae-de}, \textit{pai-de}\}$.

Nós já sabemos quais são os termos sobre $\mathcal{F}$. Sabendo disso, podemos agora definir as fórmulas da lógica de predicados.

Definição 7.2 (Fórmulas)

Definimos o conjunto de fórmulas sobre $(\mathcal{F}, \mathcal{P})$ indutivamente, usando o conjunto de termos já definido sobre $\mathcal{F}$:

- Se $P \in \mathcal{P}$ é um símbolo de predicado de aridade $n \geq 1$, e se $t_1, t_1, \ldots, t_n$ são termos sobre $\mathcal{F}$, então $P(t_1, t_2, \ldots, t_n)$ é uma fórmula.
- Se ϕ é uma fórmula, então $(\neg\phi)$ é uma fórmula.
- Se ϕ e ψ são fórmulas, então $(\phi \wedge \psi)$, $(\phi \vee \psi)$, e $(\phi \rightarrow \psi)$ são fórmulas.
- Se ϕ é uma fórmula e x é uma variável, então $(\forall x.\phi)$ e $(\exists x.\phi)$ são fórmulas.
- Nada mais é uma fórmula.

Observe como os argumentos dos predicados são sempre termos. Isso também pode ser visto na forma Backus Naur (BNF) para lógica de predicado:

$$\phi ::= P(t_1, t_2, \ldots, t_n) \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi) \mid (\forall x.\phi) \mid (\exists x.\phi)$$

onde $P \in \mathcal{P}$ é um símbolo predicado de aridade $n \leq 1$, t_i são termos sobre $\mathcal{F}$ e x é uma variável. Lembre-se de que cada ocorrência de ϕ no lado direito de $::=$ significa qualquer fórmula já construída por essas regras.

[3] Na lógica uma assinatura lista e descreve os símbolos não lógicos de uma linguagem formal. As assinaturas desempenham o mesmo papel na matemática do que as assinaturas de tipo na programação de computadores. Eles raramente são explicitados em mais tratamentos filosóficos da lógica.

Definição 7.3 (Precedência)

Por conveniência, mantemos as prioridades vinculativas usuais definidas para a Lógica Proposicional e acrescentamos que $\forall v$ e $\exists v$ tem a mesma precedência de $\neg$. Assim, a ordem é:

1. $\neg$, $\forall v$ e $\exists v$ tem a maior precedência;
2. em seguida $\wedge$;
3. em seguida $\vee$;
4. em seguida $\rightarrow$, que é associativo à direita.

Também omitimos frequentemente os colchetes em torno dos quantificadores, desde que isso não introduza ambiguidades.

Exemplo 7.4

Considere traduzir a sentença abaixo para a lógica de predicados.

> Todo filho do meu pai é meu irmão.

A escolha de *design* aqui é se representamos "pai" como um predicado ou como um símbolo de função.

- *Como um predicado.* Nós escolhemos uma constante m para "eu", então m é um termo, e nós escolhemos $\{F, P, I\}$ como o conjunto de predicados com os significados:
  $F(x, y)$: x é um filho de y
  $P(x, y)$: x é o pai de y
  $I(x, y)$: x é um irmão de y
  Assim, a representação simbólica da sentença acima fica:
  $$\forall x. \forall y. \, (P(x, m) \wedge F(y, x) \rightarrow I(y, m)) \tag{7.2}$$
  Que diz: "Para todo x e todo y, se x é o meu pai (i.e., pai de 'eu') e y é um filho de x, então y é meu irmão (i.e., irmão de 'eu')."
- *Como uma função.* Mantemos m, F e I conforme definido acima e escrevemos p para a função que, dado um argumento, retorna o pai correspondente. Note que isso funciona apenas porque os pais são únicos e sempre definidos, portanto, é realmente uma função em oposição a uma mera relação.
  A codificação simbólica da sentença com esse *design* fica:
  $$\forall x. \, (F(x, f(m)) \rightarrow I(x, m)) \tag{7.3}$$
  Que significa: "Para todo x, se x é um filho do meu pai, então x é mau irmão."; É uma formulação menos complexa porque inclui apenas um quantificador.

Especificações formais requerem conhecimento específico do domínio. Especialistas em domínio geralmente não explicitam todo esse conhecimento, portanto, um especificador pode perder restrições importantes para um modelo ou uma implementação. Por exemplo, a especificação em (7.2) e (7.3) pode parecer correta, mas e quando os valores de x e m são iguais? Se o domínio de parentesco não é de conhecimento comum, então um especificador pode não perceber que um homem não pode ser seu próprio irmão. Assim, (7.2) e (7.3) não estão completamente corretos!

7.3.3 Variáveis Livres e Vinculadas

O uso de variáveis e quantificadores nos permite expressar as noções de *todos...* e *alguns...*. Intuitivamente, verificar se $\forall x.Q(x)$ é verdadeiro equivale a substituir x por qualquer um dos seus valores possíveis e verificar se Q é válido para cada um deles. Há dois sentidos importantes e diferentes em que tais fórmulas podem ser "verdadeiras". Primeiro, se dermos significados concretos a todos os símbolos de predicado e função envolvidos, temos um *modelo* e podemos *verificar* se

uma fórmula é verdadeira para esse modelo em particular. Por exemplo, se uma fórmula codifica um comportamento requerido de um circuito de hardware, então gostaríamos de saber se é verdade para o modelo do circuito. Segundo, às vezes, gostaríamos de garantir que certas fórmulas sejam verdadeiras *para todos os modelos possíveis*. Considere $P(c) \wedge \forall y.\,((P(y) \to Q(y)) \to Q(c)$ para uma constante c; claramente, essa fórmula deve ser verdadeira, não importa o modelo que estamos olhando. É esse segundo tipo de verdade que é o foco principal da Seção 7.5.

Toda variável que apareça em uma fórmula e que não esteja dentro do *escopo* de algum quantificador é chamada de uma **variável livre**. Considere a definição do conjunto de variáveis presentes em um termo:

Definição 7.4 (Conjunto de Variáveis do Termo)

Seja t um termo da Lógica de Predicados de Primeira Ordem. Definimos o *conjunto de variáveis do termo* t, denotado por $\mathsf{Var}(t)$, da seguinte forma:

1. Se t é uma constante, então $\mathsf{Var}(t) = \emptyset$;
2. Se $t = x$ para alguma variável x, então $\mathsf{Var}(t) = \{x\}$;
3. Se $t = f(t_1, \ldots, t_n)$ para algum símbolo de função f e termos $t_1, \ldots, t_n$, então $\mathsf{Var}(t) = \mathsf{Var}(t_1) \cup \cdots \cup \mathsf{Var}(t_n)$.

Podemos definir o conjunto de variáveis livre de uma fórmula ϕ da seguinte forma:

Definição 7.5 (Conjunto de Variáveis Livres)

Seja ϕ uma fórmula bem formada. Definimos o *conjunto de variáveis livres* de ϕ, denotado por $\mathsf{FV}(\phi)$, da seguinte forma:

1. Se $\phi = R(t_1, \ldots, t_n)$, então $\mathsf{FV}(\phi) = \{x \mid x \in \mathsf{Var}(t_i), 0 < i \leq n\}$;
2. Se $\phi = \neg\psi$, então $\mathsf{FV}(\phi) = \mathsf{FV}(\psi)$;
3. Se $\phi = (\psi \wedge \eta)$ ou $\phi = (\psi \vee \eta)$ ou $\phi = (\psi \to \eta)$, então $\mathsf{FV}(\phi) = \mathsf{FV}(\psi) \cup \mathsf{FV}(\eta)$;
4. Se $\phi = \forall x.\psi$ ou $\phi = \exists x.\psi$, então $\mathsf{FV}(\phi) = \psi \setminus \{x\}$.

Dizemos que uma fórmula ϕ na Lógica de Primeira Ordem é uma **fórmula fechada** sse $\mathsf{FV}(\phi) = \emptyset$.

Todas as variáveis que aparecem em uma fórmula e que não são variáveis livre são chamadas de **variáveis ligadas** ou **variáveis vinculadas** na fórmula em questão.

7.3.4 Tradução para Forma Simbólica

Ao se traduzir uma determinada informação para a Lógica de Primeira Ordem é necessário identificar quais são os objetos sobre os quais estamos falando e quais são os atributos ou qualidades de objetos, e quais são as relações entre os objetos. Objetos serão representados na forma simbólica por constantes ou símbolos funcionais, enquanto atributos, qualidade e relacionamentos serão representados por predicados.

Exemplo 7.5

Digamos que queremos traduzir a sentença "João é simpático e gosta de Maria" para a forma simbólica, uma possível tradução seria:

- Identificamos "João" e "Maria" como os objetos sobre os quais estamos falando. Iremos representá-los pelos símbolos constantes j e m, respectivamente.
- Identificamos que "simpático" é um atributo de "João". Iremos representá-lo pelo predicado $S/1$. Assim, quando escrevermos $S(j)$ isso deverá ser lido como "João é simpático".

- Identificamos que "gostar" é uma relação que existe entre "João" e "Maria". Iremos representá-la por $G/2$. Assim, quando escrevermos $G(j, m)$ isso deverá ser lido como "João gosta de Maria".

Considerando as definições acima, a tradução completa ficaria:

$$S(j) \wedge G(j, m)$$

A escolha de nomes para os símbolos constantes, os símbolos funcionais e os predicados é livre no processo de tradução, mas para fórmulas que vão ser lidas por pessoas (ao invés daquelas que serão lidas por computadores) é recomendado usar nomes cujos significados sejam fáceis de identificar.

Exemplo 7.6

Traduza a frase "Alice é extrovertida e Beto é tímido, mas os dois estão namorando." para a forma simbólica da Lógica de Primeira Ordem.

- Os objetos do discurso são "Alice" e "Beto", que representaremos por a e b, respectivamente.
- Os atributos são "extrovertida" e "tímido", que representaremos por $E/1$ e $T/1$, respectivamente.
- O relacionamento entre os objetos é "estão namorando", que representaremos por $N/2$.

Assim, a tradução da frase fica:

$$E(a) \wedge T(b) \wedge N(a, b)$$

Exemplo 7.7

Traduza a frase "Carlos e seu pai são engenheiros, eles trabalham na mesma empresa." para a forma simbólica da Lógica de Primeira Ordem.

- Os objetos do discurso são "Carlos" e "seu pai". O pai de Carlos não recebe um nome (embora certamente tenha um). A melhor forma de tratar este tipo de situação é usando símbolos funcionais. Então usaremos o símbolo constante c para representar "Carlos", e usaremos $p/1$ para representar "pai de", de modo que $p(c)$ significa "pai de carlos".
- Os objetos possuem o atributo de serem engenheiros, que representaremos por $E/1$.
- A segunda parte da frase pode ser traduzida de duas formas diferentes:

  1. Podemos criar um predicado, $M/2$, que signifique "trabalham na mesma empresa", de modo que $M(c, p(c))$ signifique que "Carlos e o pai de Carlos trabalham na mesma empresa". Esta opção gera a seguinte forma simbólica:
  $$E(c) \wedge E(p(c)) \wedge M(c, p(c))$$
  2. Alternativamente, podemos definir um símbolo constante, e, para representar a empresa e definir um predicado, $T/2$, que signifique "trabalha em", de modo que $T(c, e)$ significa "Carlos trabalha na empresa e". Esta forma gera a seguinte forma simbólica:
  $$E(c) \wedge E(p(c)) \wedge T(c, e), \wedge T(p(c), e)$$

  As duas formas estão corretas. A forma mais indicada depende da situação em que a forma simbólica será usada. Se for necessário falar mais coisas sobre a empresa onde Carlos e seu pai trabalham, a segunda forma é mais indicada, caso contrário a primeira atenderá.

Esta forma de tradução resolve diversos casos, mas não todos. Frequentemente, queremos falar sobre situações em que não existem objetos definidos. Ou seja, não queremos falar sobre um (ou um conjunto finito) de objetos conhecidos, mas sim de vários (ou todos) os objetos que se enquadram em uma categoria

Por exemplo considere a frase "Todas as rosas têm espinhos." Nesta frase não estamos falando de uma rosa específica, nem mesmo de um conjunto de rosas, mas de todas as rosas possíveis.

Como não há objetos determinados, não podemos criar constantes para representá-los. O que fazemos então é utilizar quantificadores e predicados para "selecionar" os objetos sobre os quais queremos falar. Se definirmos que $R/1$ significa "é uma rosa", e que x é uma variável, então $R(x)$ significa que "x é uma rosa". Usando os quantificadores podemos gerar novas expressões como $\forall x R(x)$ (todo x é uma rosa), $\exists x R(x)$ (ao menos um x é uma rosa), $\neg\forall x R(x)$ (nem todo x é uma rosa), $\neg\exists x R(x)$ (nenhum um x é uma rosa).

Entretanto, a tradução para a forma simbólica usando quantificadores exige cuidado. Quando se usa o quantificador universal, é muito fácil escrever fórmula mais "fortes" do que deveriam ser. Considere novamente a frase "Todas as rosas tem espinhos.". Vamos definir $R/1$ como "é uma rosa" e $E/1$ como "tem espinhos". Como estas definições, poderíamos imaginar que a tradução completa da frase seria $\forall x(R(x) \wedge E(x))$, mas não é o caso. A fórmula $\forall x(R(x) \wedge E(x))$ deve ser lida como "todo x é uma rosa e tem espinhos" ou, de modo mais natural, "todas as coisas são rosas com espinhos". Não é o que queremos dizer. Na verdade, o que queremos dizer é algo mais paracido com "se alguma coisa é uma rosa, então esta coisa tem espinhos". Esta ideia é traduzida como:

$$\forall x(R(x) \rightarrow E(x))$$

Por outro lado, quando se usa o quantificador existencial é muito fácil escrever fórmulas mais "fracas" do que deveriam ser. Considere a frase "Existem aves que não voam." Se definirmos que $A/1$ significa "é ava" e que $V/1$ significa "voa", e considerando o que discutimos sobre o quantificador universal, talvez imaginássemos que a tradução completa da frase seria $\exists x(A(x) \rightarrow \neg V(x))$. Novamente não é o caso. Esta fórmula deve ser lida como "existe alguma coisa que, se esta coisa é uma ave, então esta coisa não voa." O problema aqui está no condicional, i.e., no "se ..., então ...". Quando temos um condicional, i.e., $\alpha \rightarrow \beta$, se o antecedente (α) for falso então todo o condicional será **verdadeiro**. Por essa razão, na fórmula $\exists x(A(x) \rightarrow \neg V(x))$, se houver alguma coisa que não seja uma ave, este valor de x fará com que o condicional seja verdadeiro e portanto fará com que toda a fórmula seja verdadeira, independentemente de qualquer outra coisa. Certamente não é isso o que queremos dizer. O que queremos dizer está mais na linha de "existe alguma coisa que é uma ava e essa coisa não voa." Esta ideia é traduzida como:

$$\exists x(A(x) \wedge V(x))$$

De modo geral, ao fazer a traduções o quantificador universal aparece junto com o condicional, i.e., $\forall x(\alpha \rightarrow \beta)$, e o quantificador existencial aparece junto com a conjunção, i.e., $\exists x(\alpha \wedge \beta)$.

Exemplo 7.8

Traduza a frase "Nem tudo que reluz é ouro." para a forma simbólica da Lógica de Primeira Ordem.

Vamos definir:

- $R/1$ significa "reluz"
- $O/1$ significa "é ouro"

Assim a tradução fica:

$$\neg\forall x(R(x) \rightarrow O(x))$$

ou, aplicando equivalência de operadores seguida de DeMorgan:

$$\exists x(R(x) \wedge \neg O(x))$$

Exemplo 7.9

Traduza a frase "Todo estudante cursa ao menos uma disciplina." para a forma simbólica da Lógica de Primeira Ordem.

Vamos definir:

- $E(x)$ significa "x é um estudante"

- $D(y)$ significa "y é uma disciplina"
- $C(x, y)$ significa "x cursa y"

Assim a tradução fica:

$$\forall x(E(x) \rightarrow \exists y(D(y) \wedge C(x, y))$$

No exemplo anterior é possível mover o quantificador existencial para fora dos parênteses sem alterar o significado da fórmula:

$$\forall x \exists y(E(x) \rightarrow (D(y) \wedge C(x, y))$$

Entretanto, deve-se tomar cuidado para não "cruzar" os quantificadores. A expressão $\forall x \exists y(E(x) \rightarrow (D(y) \wedge C(x, y))$ significa "todo estudante cursa alguma disciplina", note que não é necessário que seja a mesma disciplina para todos os estudantes, cada estudante pode cursar uma disciplina diferente. Já a fórmula $\exists y \forall x(E(x) \rightarrow (D(y) \wedge C(x, y))$ significa que "existe alguma disciplina que é cursada por todos os estudantes. Neste caso todos os estudantes teriam que cursar a mesma disciplina. Uma diferença significativa.

7.3.5 Interpretações e Validade

Em todas as traduções que fizemos na seção anterior, iniciamos definindo o significado dos símbolos (constantes, funcionais, predicados) que iríamos usar. O significado atribuído aos símbolos em uma fórmula (ou conjunto de fórmulas) é chamado de uma **interpretação**. Formalmente, uma *interpretação* é definida como um domínio, D, e mapeamentos para todos os símbolos funcionais e predicados.

Definição 7.6 (Interpretação)

Uma **interpretação** da Lógica de Primeira Ordem consiste em um domínio D não vazio e mapeamentos para símbolos funcionais e predicados. Todo símbolo funcional de aridade n é mapeado para uma função de D^n para D, e todo símbolo predicado aridade n é mapeado para uma função de D^n para o conjunto composto pelos valores verdadeiro e falso.

O domínio D é o conjunto que contém os valores de todas as variáveis nas fórmulas da Lógica de Primeira Ordem e é chamado *domínio da interpretação*.

Para uma dada interpretação, a tabela-verdade de qualquer fórmula é definida pelas seguintes regras.

1. As tabelas verdadeiras para conectivos proposicionais aplicam-se para avaliar o valor de $\alpha \wedge \beta$, $\alpha \vee \beta$, $\alpha \rightarrow \beta$ e $\neg\alpha$.
2. $\forall x\alpha$ é verdadeiro se α for verdadeiro para qualquer elemento de D como valor de x nas ocorrências livres de x em α. Caso contrário, $\forall x\alpha$ é falso.
3. $\exists x\alpha$ é verdadeiro se α for verdadeiro para pelo menos um elemento de D como valor de x nas ocorrências livres de x em α. Caso contrário, $\exists x\alpha$ é falso.

7.3.6 Exercícios

Exercício 7.1 Indique quais das sequências de símbolos abaixo representam fórmulas bem formadas da lógica de predicados de 1ª ordem. Os símbolos a e b são constantes; $f/1$ e $g/2$ são símbolos funcionais e $P/1$, $R/2$ e $Q/3$ são predicados.

1. $Q(a)$
2. $P(y)$
3. $P(g(b))$
4. $\neg R(x, a)$
5. $Q(x, P(a), b)$
6. $P(g(f(a), g(x, f(x))))$
7. $Q(f(a), f(f(x)), f(g(f(z), g(a, b))))$
8. $R(a, R(a, a))$
9. $R(a, g(a, a))$
10. $g(a, g(a, a))$

11. $\forall x.\neg P(x)$
12. $\neg R(P(a), x)$
13. $\exists a.R(a, a)$
14. $\exists x.Q(x, f(x), b) \rightarrow \forall x.R(a, x)$
15. $\exists x.P(R(a, x))$
16. $\forall R(x, a)$
17. $a \rightarrow P(b)$
18. $R(x, b) \rightarrow \exists y.Q(y, y, y)$
19. $R(x, b) \vee \neg\exists y.g(y, b)$
20. $\neg y \vee P(y)$
21. $\neg\neg P(a)$
22. $\neg\forall x.\neg P(x)$
23. $\forall x.\exists y.(R(x, y) \rightarrow R(y, x))$
24. $\forall x.\exists y.(R(x, y) \rightarrow (R(y, x) \vee (f(a) = g(a, x))))$

Exercício 7.2 Qual é o valor lógico de cada uma das fórmulas abaixo, na lógica de predicados, na interpretação em que o domínio é o conjunto dos números inteiros, $O(x)$ é "x é ímpar", $L(x)$ é "$x < 10$" e $G(x)$ é "$x > 9$"?

1. $\exists x.O(x)$
2. $\forall x.[L(x) \rightarrow O(x)]$
3. $\exists x.[L(x) \wedge G(x)]$
4. $\forall x.[L(x) \vee G(x)]$
5. $\forall x.[L(x) \rightarrow \neg G(x)]$
6. $\forall x.[\neg L(x) \rightarrow G(x)]$

Exercício 7.3 Usando os símbolos predicados mostrados abaixo e os quantificadores apropriados, transcreva cada sentença em português dada abaixo como uma fórmula na lógica de predicados.

- $B(x)$ é "x é uma abelha"
- $F(x)$ é "x é uma flor"
- $L(x, y)$ é "x ama y"

1. Todas as abelha amam todas as flores.
2. Algumas abelhas amam todas as flores.
3. Todas as abelhas amam algumas flores.
4. Toda abelha odeia[4] alguma flor.
5. Todas as abelhas odeiam apenas flores.
6. Apenas abelhas amam flores.
7. Toda abelha ama apenas flores.
8. Nenhuma abelha ama apenas flores.
9. Algumas abelhas amam algumas flores.
10. Algumas abelhas amam apenas flores.
11. Toda abelha odeia todas as flores.
12. Nenhuma abelha odeia todas as flores.

Exercício 7.4 Considere as seguintes definições:

- $A(x)$: x é um aluno
- $P(x)$: x é um professor
- $D(x)$: x é uma disciplina
- $M(x, y)$: x é o professor de (da disciplina) y
- $T(x)$: x é inteligente
- $C(x, y)$: x cursa (a disciplina) y
- $R(x, y)$: x admira y
- $L(x, y)$: x ama y
- $V(x, y)$: x foi aprovado em (na disciplina) y
- p: Paulo
- c: Carlos
- j: Joana
- f: Flávia
- g: Geometria
- a: Álgebra

Utilizando estas definições codifique (traduza para a forma simbólica) as sentenças abaixo.

(a) Todo aluno é inteligente.

(b) Alguns professores são inteligentes.

[4]Para este exercício considere que "odeia" e "não ama" são sinônimos.

(c) Todo aluno que cursa Álgebra é inteligente.

(d) Nenhum professor que ministra Álgebra é inteligente.

(e) Apenas os professores que ministram Álgebra são inteligentes.

(f) Todos os professores admiram os alunos inteligentes.

(g) Todo professor admira algum aluno inteligente.

(h) Todos os alunos inteligente admiram os professores de Álgebra.

(i) Ninguém admira os professores de geometria.

(j) Paulo admira os alunos que foram aprovados em Álgebra.

(k) Alguns professores medíocres[5] não admiram nenhum aluno.

(l) Carlos foi aprovado em Álgebra mas não em Geometria.

(m) Uma condição necessária para cursar Álgebra é ter sido aprovado em Geometria.

(n) Flávia admira os alunos que foram aprovados em Álgebra e em Geometria.

(o) Algum aluno que cursa Geometria ama Joana.

(p) Paulo cursa as mesmas disciplinas que Flávia.

(q) Todos os professores admiram Flávia.

(r) Nenhum aluno admira apenas os professores de Geometria.

(s) Todo aluno admira os professores da disciplina que cursa.

(t) Joana admira todos os alunos do professor de Geometria.

(u) Joana ama algum aluno que cursa Álgebra e que já foi aluno do professor de Geometria.

(v) Existe um professor que admira apenas os alunos que já foram aprovados em todas as disciplinas.

(w) Alguns alunos admiram apenas os professores que admiram a si mesmos.

(x) Todos os professores inteligentes admiram os alunos que já foram aprovados em Álgebra e Geometria.

(y) Alguns alunos de Geometria admiram os alunos de Álgebra que Carlos admira.

(z) Todos os alunos de Geometria admiram os alunos de Álgebra que o professor de Geometria admira.

7.4 Raciocínio na Lógica de Predicados

Que tipo de racínio a lógica de predicados deve permitir? Para ter uma ideia de como responder a esta pergunta vamos considerar o argumento a seguir.

Nenhum livro é líquido. Dicinários são livros. Portanto, nenhum dicionário é líquido. (7.4)

Os predicados que nós escolhemos são:

$L(x)$**:** x é um livro

$Q(x)$**:** x é líquido

$D(x)$**:** x é um dicionário.

[5] Considere que "medíocre" é o mesmo que "não inteligente".

Com os predicados acima é possível traduzir o argumento (7.4) como:

$$\neg\exists x\,(L(x) \wedge Q(x)), \forall x\,(D(x) \rightarrow L(x)) \vdash \neg\exists x\,(D(x) \wedge Q(x))$$

Evidentemente, nós precisamos construir uma teoria de provas que nos permita demonstrar a validade do argumento formal acima. Ou seja, verificar que o argumento formal realmente expressa o argumento (7.4) em uma forma simbólica.

A lógica de predicados estende a lógica proposicional não apenas com quantificadores, mas também com o conceito de *símbolos funcionais*. Considere a seguinte sentença:

Toda criança é mais jovem do que sua mãe. (7.5)

Usando os predicados

$C(x)$: x é uma criança

$M(x, y)$: x é a mãe de y

$J(x, y)$: x é mais jovem que y

podemos traduzir a sentença (7.5) como

$$\forall x\,(C(x) \rightarrow \forall y\,(M(y, x) \rightarrow J(x, y)))$$

Que, numa tradução reversa literal, significa "Para todo x, se x é uma criança, então para todo y se y é a mãe de x, então x é mais jovem do que y." Não é muito elegante dizer "para todas as mães de x", uma vez que nós sabemos que cada indivíduo possui uma e apenas uma mãe.[6] A deselegância de se codificar 'mãe' como um predicado se torna ainda mais evidente se considerarmos a seguinte sentença:

Ana e Paulo possuem a mesma avó materna. (7.6)

Assumindo as constantes a para Ana e p para Paulo, e os mesmos predicados da sentença (7.5), podemos traduzir a sentença (7.6) como:

$$\forall x \forall y \forall u \forall v\,(M(x, y) \wedge M(y, a) \wedge M(u, v) \wedge M(v, p) \rightarrow x = u)$$

Esta formula diz que, se y é a mãe de Ana e x é a mãe da mãe de Ana, e se v é a mãe de Paulo e se u é a mãe da mãe de Paulo, então x e u são a mesma pessoa. Note que nós usamos um predicado especial da lógica de primeira ordem, a *igualdade*; que é um predicado binário e é escrito como o símbolo $=$. Diferentemente do outros predicados, a igualdade é escrita entre os argumento, então escrevemos $x = y$ ao invés de escrever $=(x, y)$.

Os símbolos funcionais da lógica de predicados oferecem uma forma de evitar essa codificação deselegante, uma vez que eles permitem representar a mãe de y de forma mais direta. Ao invés de escrever $M(x, y)$ para representar que x é a mãe de y, nós simplesmente escrevemos $m(y)$ para designar a mãe de y. O símbolo m é um símbolo funcional: ele recebe um argumento e retorna a mãe deste argumento. Usando o símbolo m, as sentenças (7.5) e (7.6) possuem codificações bem mais simples do que as vistas anteriormente. A sentença "Toda criança é mais jovem que sua mãe." agora pode ser codificada como:

$$\forall x\,(C(x) \rightarrow J(x, m(x)))$$

Note que foi necessária apenas uma variável (e um quantificador) ao invés de dois. E a sentença "Ana e Paulo possuem a mesma avó materna", recebe uma codificação ainda mais simples, podendo ser escrita como:

$$m(m(a)) = m(m(p))$$

Sempre é possível fazer as codificações sem recorrer a símbolos funcionais, utilizando variáveis e predicados como visto anteriormente. Entretanto, o uso de símbolos variáveis geralmente leva a formas mais limpas e sucintas de codificação. Apesar disso, só podem ser usados em situações onde nós queremos designar uma única entidade (elemento) do domínio. Acima, foi possível usar o símbolo m para designar "mãe" devido ao fato de todo indivíduo possuir uma mãe unicamente identificada, de modo que podemos falar da mãe de y sem correr o risco de gerar qualquer ambiguidade.

[6] Estamos assumindo que a sentença fala de mãe biológica, não mãe adotiva, madrasta, etc.

Por esta razão, não podemos ter um símbolo funcional $i(\cdot)$ para "irmão". Pode não fazer sentido falar do irmão de x, pois x pode não ter irmãos ou ter vários irmãos. Assim, "irmão" tem que ser codificado como um predicado binário.

Para exemplificar melhor este ponto, se Maria tem vários irmão, então a sentença "Ana gosta do irmão de Maria" é ambígua pois não especifica de qual dos irmão de Maria estamos falando. Por outro lado, a sentença "Ana gosta de um dos irmãos de Maria" está correta e pode ser codificada como:

$$\exists x\,(I(x,m) \wedge G(a,x))$$

onde, $I(x,y)$ significa "x é irmão de y", $G(x,h)$ significa "x gosta de y", e a e m significam Ana e Maria, respectivamente. Alternativamente, a sentença "Ana gosta de todos os irmão de Maria", pode ser codificada como:

$$\forall x\,(I(x,m) \rightarrow G(a,x))$$

Diferentes símbolos funcionais podem ter diferentes números de argumentos. Funções podem inclusive ter zero argumentos, e neste caso são chamadas de **constantes**. Os símbolos a e m nos exemplos acima são constantes representando Ana e Maria, respectivamente. Em um domínio envolvendo estudantes, cursos e notas, podemos ter uma função binária $n(x,y)$ que designa a nota obtida pelo estudante x no curso y.

7.5 Dedução no Cálculo de Predicados

Da mesma forma que ocorre com o cálculo proposicional, o objetivo do cálculo de predicados é permitir que argumentos sejam representados de modo formal, através da linguagem simbólica da lógica. Como vimos na Seção 7.1, o cálculo de predicados busca trazer mais poder de expressividade para a lógica, de modo que conceitos e relações que não podem (ou que são muito difíceis de) ser representados no cálculo proposicional possam passar a ser representados.

Apesar de estarmos lidando com um tipo diferente de lógica, com um poder expressivo maior, o conceito de dedução, ou de modo mais geral, de *argumeto* não muda em relação ao que foi apresentado para o cálculo proposicional:

> A forma geral de um argumento em uma linguagem natural consistem em apresentar premissas para fundamentar uma conclusão. Em um argumento dedutivo típico, as premissas se destinam fornecer uma garantia da validade da conclusão, enquanto em argumentos indutivos se considera que elas fornecem razões para apoiar a provável validade da conclusão.

Assim, o que queremos é entender que regras, ou que condições, são suficientes para termos certeza que um determinado conjunto de premissas justifica a nossa confiança de que a conclusão é válida.

7.5.1 Argumentos Válidos

O conceito de *consequência lógica* para o cálculo de predicados, mantém a mesma ideia que tem para o cálculo proposicional, ou seja, uma conclusão é consequência lógica de um conjunto de premissas, se esse conjunto de premissas acarreta essa conclusão, ou seja, se o conjunto de premissas fornece "motivos" suficientes para justificar a validade da conclusão.

Entretanto, no cálculo de predicados não falamos mais em funções de valoração, e sim em interpretações, por isso a definição formal de consequência lógica precisa ser atualizada.

Definição 7.7 (Consequência Lógica para Cálculo de Predicados)

Dizemos que uma *fórmula bem formada* β do cálculo de predicados é uma **consequência lógica** de um conjunto de fórmulas $\Gamma = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, quando o conjunto de premissas Γ **acarreta** a conclusão β. Existem dois tipos de consequência lógica:

Consequência Semântica Dizemos que a fórmula β é uma *consequência semântica* do conjunto de fórmulas $\Gamma = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, escrito

$$\alpha_1, \alpha_2, \ldots, \alpha_n \vDash \beta$$

se e somente se não existir nenhuma **interpretação** para as que faça todas as fórmulas $\alpha_i \in \Gamma$ serem verdadeiras mas que faça β ser falsa.

Consequência Sintática Dizemos que a fórmula β é uma *consequência sintática* do conjunto de fórmulas $\Gamma =$

$\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, escrito

$$\alpha_1, \alpha_2, \ldots, \alpha_n \vdash \beta$$

se e somente se não existir existir uma "prova" (por algum sistema de prova) de que dadas as premissas $\alpha_1, \alpha_2, \ldots, \alpha_n$ é possível derivar a fórmula β.

Uma vez que o número de interpretações para um dado argumento formal é infinito, não é trivial demonstrar que uma dada conclusão é consequência semântica das premissas. Por esse motivo, o estudo de consequência lógica na lógica de predicados de primeira ordem costuma se focar principalmente no estudo de consequências sintáticas. O sistema de provas que estudaremos para o cálculo de predicado será, a exemplo do que fizemos para o cálculo proposicional, o Sistema de Dedução Natural.

7.5.2 Dedução Natural

Uma forma possível de se pensar as proposições da lógica proposicional é visualizá-las como predicados de aridade zero na lógica de primeira ordem. Se assumirmos essa visão, não é difícil verificar que toda a lógica proposicional pode ser representada através da lógica de primeira ordem. Assim, se o nosso sistema de prova deve preservar validade, tudo o que era válido para o cálculo proposicional deve continuar válido para o cálculo de predicados.

Mais especificamente, no sistema de dedução natural para o cálculo de predicados continuam válidas as regras de inferência básicas apresentadas na Tabela 5.3, ou seja, *modus ponens*, *modus tolens*, *conjunção*, *simplificação*, *adição* e *reapresentação*, bem como todas as regras de inferência derivadas apresentadas na Tabela 5.4. Também continuam válidos o uso de *sub-provas* e do *método dedutivo* (ou exportação).

Entretanto, apenas repetir as regras de inferência do sistema de dedução natural do cálculo proposicional não é suficiente para construir um sistema de dedução natural para o cálculo de predicados. Precisamos de regras de inferência que sejam capazes de lidar com quantificadores, regras que sejam capazes de inserir ou de remover quantificadores nas fórmulas. Evidentemente, essas regras não podem permitir a inserção ou remoção indiscriminada dos quantificadores, haverá algumas restrições, mas vamos ver as restrições quando falarmos de cada regra. Existem quatro regras para lidar com quantificadores:

Particularização Universal: permite remover o quantificador universal de uma fórmula.

Generalização Universal: permite inserir o quantificador universal à frente de uma fórmula.

Generalização Existencial: permite inserir o quantificador existencial à frente de uma fórmula.

Particularização Existencial: permite remover o quantificar existencial de uma fórmula.

Discutiremos mais detalhadamente estas regras nas seções abaixo.

Particularização Universal

Nós podemos pensar no quantificador universal ($\forall$) como uma generalização da conjunção ($\wedge$). Imagine que nosso domínio seja o conjunto $\{1, 2, 3, 4\}$. Considerando este domínio, a fórmula $\forall x.P(x)$ tem exatamente o mesmo significado que $P(1) \wedge P(2) \wedge P(3) \wedge P(4)$. De modo semelhante, podemos considerar a regra de inferência da dedução natural para remover o quantificador universal, i.e., a *particularização universal*, como uma generalização da regra de inferência para remover a conjunção.

Na regra de inferência de simplificação, se temos uma fórmula no formato $\alpha \vee \beta$, escolhemos uma das duas partes e a introduzimos como parte do argumento. Na regra de inferência de *particularização universal* faremos algo parecido. Se temos uma fórmula no formato $\forall x.P(x)$, iremos "escolher" qualquer termo t válido no nosso domínio (que seja livre em x) e introduzir a fórmula $P(t)$, como parte do argumento. Esta regra é representada baixo:

$$\frac{\forall x.\phi}{\phi[t/x]} \qquad \text{(pu)}$$

A notação "$\phi[t/x]$" significa a fórmula obtida substituindo-se todas as ocorrências livres de x, na fórmula ϕ, por t.

Exemplo 7.10

Demonstre o argumento formal $\forall_x.(H(x) \rightarrow M(x)), H(s) \vdash M(s)$.

1	$\forall x.(H(x) \to M(x))$	hip
2	$H(s)$	hip
3	$H(s) \to M(s)$	pu [s/x], 1
4	$M(s)$	mp, 2, 3

□

Generalização Universal

Inserir o quantificador universal em uma fórmula não é tão simples quanto removê-lo. Se usarmos uma estratégia análoga à que usamos para a particularização universal e tentarmos buscar uma generalização da regra de inferência que insere conjunções, teremos um problema. A regra de inferência que insere conjunções é a seguinte:

$$\frac{\alpha, \beta}{\alpha \wedge \beta} \qquad \text{(conj)}$$

Ou seja, generalizando esta regra de inferência, teríamos que ter $P(x)$ provado para todos os valores de x possíveis para que pudéssemos escrever $\forall_x.P(x)$. Como o número de interpretações possíveis é infinito e os domínios dessas interpretações também podem ser infinitos, é impossível provar explicitamente uma fórmula para todos os valores do domínio.

Mas nem tudo está perdido. Existem outras formas pelas quais nós realizamos prova que são válidas para todos os valores. A título de exemplo, considere o teorema abaixo e sua prova.

Teorema: Todo número inteiro positivo par é a soma de dois números inteiros positivos ímpares cuja diferença é no máximo 2.

Prova: Seja n um número inteiro positivo par qualquer. Como n é par, então $n = 2k$ para algum k inteiro positivo.

- Se k é ímpar, então $n = k + k$, e os 2 k's satisfazem ao teorema.
- Se k é par, então $n = (k - 1) + (k + 1)$, e os números $k - 1$ e $k + 1$ satisfazem ao teorema.

□

O teorema acima, e a sua prova, são válidos para todos os números inteiros positivos pares, mesmo que nós não tenhamos realizado a prova para todos eles — o que seria impossível.

Se olharmos para esta prova, ela considera um número inteiro positivo par arbitrário n, e demonstra que ele pode ser representado como a soma de dois números inteiros positivos ímpares cuja diferença é no máximo 2. O que torna a prova genérica é o fato de não haver **nenhuma** propriedade especial sobre n, isto é, ele não aparece na declaração do teorema ou em qualquer outro lugar fora da prova. Não fizemos nenhuma suposição sobre n – exceto a de que ele é um valor de interesse para o teorema (inteiro positivo par). Como não existe nenhuma propriedade ou característica que torne n diferente de qualquer outro número, a prova que construímos para n pode ser considerada válida para qualquer, e todos, os números.

A estratégia de prova exemplificada acima sugere que para provar uma fórmula da forma $\forall_x.\phi$, podemos provar a fórmula ϕ para alguma variável arbitrária, porém **nova**, x_0 no lugar de x. Ou seja, provamos a fórmula $\phi[x_0/x]$, onde x_0 é uma variável nova. A representação da regra de *generalização universal* é dada abaixo:

$$\begin{array}{|l|}\hline x_0 \\ \quad \vdots \\ \quad \phi[x_0/x] \\ \hline \end{array}$$

$$\frac{\qquad\qquad}{\forall x.\phi} \qquad \text{(gu)}$$

A expressão "variável nova" significa que é uma variável que não foi utilizada em nenhum ponto anterior da prova, mas também que ela não será usada depois que $\phi[x_0/x]$ tiver sido provado. Ela é uma variável "local" para esta parte da prova. Para evidenciar o uso local dessas variáveis usaremos o mesmo tipo de marcação que usamos em sub-provas, porém com o nome da variável escrito no início do bloco da sub-prova.

Exemplo 7.11

Demonstre o argumento: $\forall x.(P(x) \vee Q(x)), \forall x.(\neg P(x)) \vdash \forall x.Q(x)$.

1	$\forall x.(P(x) \vee Q(x))$	hip
2	$\forall x.(\neg P(x))$	hip
3	$x_0 \mid P(x_0) \vee Q(x_0)$	pu $[x_0/x]$, 1
4	$\mid \neg P(x_0)$	pu $[x_0/x]$, 2
5	$\mid Q(x_0)$	sd, 3, 4
6	$\forall x.Q(x)$	ge $[x/x_0]$, 3–5

□

Generalização Existencial

Assim como existe um paralelo entre o quantificador universal e a conjunção, também existe um paralelo entre o quantificador existencial ($\exists$) e a disjunção ($\vee$). Consideremos novamente como exemplo o domínio $\{1, 2, 3, 4\}$. Neste domínio a fórmula $\exists x.P(x)$ tem exatamente o mesmo significado de $P(1) \vee P(2) \vee P(3) \vee P(4)$. Ou seja, o quantificador existencial funciona como se fosse uma "grande disjunção" abrangendo todos os valores do domínio.

Para introduzir o operador de disjunção no sistema de dedução natural da lógica proposicional, temos a seguinte regra de inferência:

$$\frac{\phi}{\phi \vee \psi} \quad - \text{ ou } - \quad \frac{\psi}{\phi \vee \psi}$$

Ou seja, dado que provamos uma fórmula, ϕ, por exemplo, podemos fazer a disjunção dela com qualquer outra. Isso nos leva a concluir que, se temos que uma fórmula é verdadeira para um dos valores do domínio, podemos simplesmente inserir o quantificador existencial, pois o efeito é o mesmo que o de uma "grande disjunção". O que nos leva à seguinte regra para a *generalização existencial*:

$$\frac{\phi[t/x]}{\exists x.\phi} \quad \text{(ge)}$$

Exemplo 7.12

Demonstre o argumento: $A(s) \wedge B(s), \forall x.(A(x) \wedge B(x) \rightarrow C(x)) \vdash \exists x.(A(x) \wedge C(x))$

1	$A(s) \wedge B(s)$	hip
2	$\forall x.(A(x) \wedge B(x) \rightarrow C(x))$	hip
3	$A(s) \wedge B(s) \rightarrow C(s)$	pu [s/x], 2
4	$C(s)$	mp, 1, 3
5	$A(s)$	simp, 1
6	$A(s) \wedge C(s)$	conj, 5, 4
7	$\exists x.(A(x) \wedge C(x))$	ge [x/s], 6

□

Particularização Existencial

Para remover o quantificador existencial, entretanto, temos um pouco mais de dificuldade. O fato da fórmula $\exists x.P(x)$ ser verdadeira, garante que existe algum valor de x que faz $P(x)$ ser verdadeiro. Entretanto, não sabemos qual é esse valor e não podemos arbitrar um valor qualquer que já tenha aparecido na prova como sendo "o escolhido".

O que fazemos então é criar uma sub-prova onde "instanciamos" a fórmula, ou seja, para a fórmula $\exists x.\phi$, criaremos uma variável **nova** x_0, e abriremos uma sub-prova com a fórmula $\phi[x_0/x]$. Novamente, a variável x_0 será uma variável local à sub-prova, e o desenvolvimento da sub-prova não deve ser "vazado" para fora dela, mas a conclusão da sub-prova, i.e., a última fórmula acrescentada à sub-prova – e que não deve conter a variável x_0, pode ser usada no restante da prova. Esse mecanismo constitui a regra de inferência de *particularização existencial*, e é representado abaixo:

$$\dfrac{\exists x.\phi, \quad \begin{array}{|l|l|} \hline x_0 & \phi[x_0/x] \\ & \vdots \\ & \Theta \\ \hline \end{array}}{\Theta} \text{ (pu)}$$

Exemplo 7.13

Demonstre o argumento: $\forall x.(P(x) \vee Q(x)), \exists x.(\neg P(x)) \vdash \exists x.Q(x)$

1		$\forall x.(P(x) \vee Q(x))$	hip
2		$\exists x.(\neg P(x))$	hip
3	x_0	$\neg P(x_0)$	sup
4		$P(x_0) \vee Q(x_0)$	pu $[x_0/x]$, 1
5		$Q(x_0)$	sd, 3, 4
6		$\exists x.Q(x)$	ge $[x/x_0]$, 5
7		$\exists x.Q(x)$	pe, 2, 3–6

7.5.3 Exercícios

Exercício 7.5 Para cada um dos argumentos formais abaixo em lógica de 1ª ordem, demonstre a validade do argumento por dedução natural, ou apresente uma interpretação na qual ele é falso.

(a) $\exists x.(A(x) \wedge B(x)) \vdash \exists x.A(x)$

(b) $\exists y.(B(y) \to C(y)), \forall x.B(x) \vdash \exists x.C(x)$

(c) $\forall z.(\neg A(z) \vee B(z)) \vdash \forall z.(A(z) \to B(z))$

(d) $\exists x.A(x) \to \forall x.B(x) \vdash \forall x.(A(x) \to B(x))$

(e) $\forall x.\forall y.A(x,y) \vdash \neg\forall x.\neg A(x,x)$

(f) $\exists x.\forall y.R(x,y) \vdash \forall y.\exists x.R(x,y)$

(g) $\vdash \exists x.(P(x) \to \forall y.P(y))$

(h) $\vdash \forall x.(P(x) \to \exists y.P(y))$

(i) $\exists x.P(x) \wedge \forall x.Q(x) \vdash \exists x.(P(x) \wedge Q(x))$

(j) $\forall x.A(x,x) \vdash \forall x.\exists y.A(x,y)$

(k) $\exists x.(A(x) \vee B(x)) \vdash \exists x.A(x) \vee \exists x.B(x)$

(l) $\exists x.A(x) \vee \exists x.B(x) \vdash \exists x.(A(x) \vee B(x))$

(m) $\exists x.(A(x) \wedge B(x)) \vdash \exists x.A(x) \wedge \exists x.B(x)$

(n) $\forall x.A(x) \vee \forall x.B(x) \vdash \forall x.(A(x) \vee B(x))$

(o) $\forall x.(A(x) \vee B(x)) \vdash \forall x.A(x) \vee \forall x.B(x)$

(p) $\neg\neg\forall x.A(x) \vdash \forall x.\neg\neg A(x)$

(q) $\vdash \forall x.\exists y.x = y$

(r) $\vdash \forall x.\forall y.\forall v.\forall w.((x = v \wedge y = w) \rightarrow (R(x, y) \rightarrow R(v, w)))$

(s) $\exists x.A(x), \forall x.(A(x) \rightarrow A(f(x))), \forall x.(A(x) \rightarrow A(g(x))) \vdash \exists x.A(f(g(x)))$

(t) $\forall x.(P(x) \rightarrow \exists y.Q(x, y)) \vdash \exists x.P(x) \rightarrow \exists y.Q(x, y)$

(u) $\exists x.Q(x), \forall x.(P(x) \rightarrow \neg Q(x)) \vdash \exists x.\neg P(x)$

Exercício 7.6 Para cada um dos argumentos formais abaixo em lógica de 1ª ordem, demonstre a validade do argumento por dedução natural, ou apresente uma interpretação na qual ele é falso.

(a) $A(u) \rightarrow \forall x.B(x) \vdash \forall x.(A(u) \rightarrow B(x))$

(b) $\forall x.(A(x) \vee B(u)) \vdash \forall x.A(x) \vee B(u)$

(c) $\exists x.(A(x) \rightarrow B(u)) \vdash \forall x.A(x) \rightarrow B(u)$

(d) $\exists x.(A(u) \rightarrow B(x)) \vdash A(u) \rightarrow \exists x.B(x)$

(e) $\forall x.(A(x) \rightarrow B(u)) \vdash \exists x.A(x) \rightarrow B(u)$

7.6 Exercícios de Revisão

Capítulo 8

Lógica Relacional

Sumário

8.1 Introdução

8.2 Sintaxe

8.3 Semântica

8.4 Avaliação

8.5 Satisfabilidade

8.6 Exemplo – Sorority World

8.7 Exemplo – Mundo dos Blocos

8.8 Exemplo – Aritmética Modular

8.9 Propriedades Lógicas

8.10 Logical Entailment

8.11 Lógica Relacional e Lógica Proposicional

8.12 Exercícios

Capítulo 9

Noções de Modelagem e Programação Lógica

Sumário

Apêndice A

Sistemas de Numeração

Sumário

A.1 Introdução

Um *sistema de numeração*, ou *sistema numeral*, é um sistema de escrita para expressar números, ou seja, é uma notação matemática para representar números de um determinado conjunto (normalmente o conjunto $\mathbb{Z}$ dos números inteiros, ou o conjunto $\mathbb{R}$ dos números reais) utilizando símbolos (normalmente chamados dígitos) de modo consistente e bem definido. Dependendo do sistema de numeração utilizado, a sequência de símbolos "11" pode ser interpretada como o número três (em binário), ou o número nove (em octal), ou o número onze (em decimal), e assim por diante.

Um sistema de numeração deve ser capaz de:

- Representar um conjunto de números úteis, por exemplo: todos os números inteiros, ou todos os números racionais.
- Atribuir a cada número uma representação única, ou ao menos uma representação padrão.

- Expressar tanto a estrutura algébrica quanto a estrutura aritimética dos números.

Por exemplo, a representação comum, em decimal, dos números inteiros atribui a cada número inteiro uma representação única como uma sequência finita de algarismos, começando com um algarismo diferente de zero. Entretanto, quando a representação decimal é usada para os números racionais ou para os números reais, a representação deixa de ser padronizada, permitindo um número infinito de representações. Por exemplo, no número $3,21$ também pode ser representado como $3,210$, $3,21000$, $3,209999\ldots$, etc.

A.2 Principais Sistemas de Numeração

O sistema de numeração mais comumente usado nos dias de hoje é conhecido como *"Números Hindo-Arábicos"*. Dois matemáticos indianos são considerados os criadores deste sistema. Aryabhata de Kusumapura desenvolveu a notação posição-valor no século V, e um século depois Brahmagupta introduziu o símbolo para zero [SK11]. O sistema de numeração e o conceito de zero desenvolvido pelos Hindus na Índia lentamente se espalharam para outros países vizinhos devido às suas atividades comerciais e militares com a Índia. Os árabes adotaram e modificaram este sistema de numeração e o conceito de zero. Os árabes traduziram os textos hindus sobre números e os difundiram no ocidente devido à suas conexões comerciais com a Europa. Os europeus também introduziram modificações no sistema, e o chamaram de *"Números Arábicos"*, uma vez que tinham tomado conhecimento deste sistema através dos árabes. Assim, o sistema de numeração mais utilizado no mundo moderno é uma versão modificada do sistema desenvolvido na Índia no século V.

O sistema de numeração mais simples possível é o sistema de numeração unário, no qual cada número natural é representado pela quantidade correspondente de repetições de um determinado símbolo. Por exemplo, se o símbolo escolhido for / (barra), então o número sete será representado por ///////. O sistema de numeração unário é incapaz de representar o conceito de zero e só é útil para números pequenos. Entretanto, este sistema desempenha um papel importante em teoria da computação. A *Codificação Gama de Elias* [Eli75], que é comumente usada em compressão de dados, é capaz de expressar números de tamanho arbitrário utilizando o sistema de numeração unário para indicar o tamanho de números em binário.

A notação unária pode ser abreviada se forem introduzidos símbolos para certos valores específicos. Tipicamente, esses valores são potências de 10; por exemplo, se / significar um, + significar dez, e * significar cem, então o número trezentos e quatro pode ser representado como ***//// e o número cento e vinte e três pode ser representado como *++///, sem a necessidade de se utilizar o zero. Esta notação é chamada de *notação sinal-valor*. O sistema de numeração dos antigos Egípcios era destes tipo, e o sistema de numeração romano era uma versão modificada desta ideia.

Uma evolução do sistema *notação sinal-valor* são os sistemas que empregam abreviações especiais para repetições de símbolos; por exemplo, usando as primeiras nove letras do alfabeto para essas abreviações, com A significando "um repetição", B significando "duas repetições", e assim por diante, podemos escrever C*D/ para representar o número trezentos e quatro. Um sistema com estas características (mas com símbolos diferentes) é usado ao se escrever numerais em Chinês e em outras línguas do leste asiático.

Um sistema ainda mais elegante é o *sistema posicional*, também chamado de *notação posição-valor*. Considerando um sistema baseado em dez (base 10), um conjunto de 10 diferentes símbolos, ou dígitos, são utilizados e a posição do símbolo no numeral (da direita para a esquerda, com a posição mais à direita sendo identificada como a posição zero) é usado para significar a potência de 10 pelo qual o dígito deve ser multiplicado. Por exemplo, para o numeral 304, o dígito 4 ocupa a posição 0, o dígito 0 ocupa a posição 1 e o dígito 3 ocupa a posição 2, assim o valor do numeral é dado por $304 = 3 \cdot 10^2 + 0 \cdot 10^1 + 4 \cdot 10^0$. Note que o zero, que não é necessário nos outros sistemas, é de importância crucial no sistema posicional, pois ele permite "pular" uma potência. O sistema de numeração Hindu-Arábico é um sistema de numeração posicional de base 10.

As operações aritméticas são muito mais simples em sistemas posicionais do que nos sistemas anteriores (i.e., sistemas aditivos); além disso, os sistemas aditivos precisam de vários símbolos diferentes para representar as diversas potências de 10, enquanto que os sistemas posicionais precisam apenas de 10 símbolos (supondo que esteja sendo usada a base 10).

Em computação o principal sistema de numeração é baseado no sistema posicional de base 2, i.e., *sistema de numeração binário*, que utiliza dois dígitos, 0 e 1.

A.3 Sistemas de Numeração Posicionais

Sistemas de numeração posicionais, ou notação posicional, ou notação posição-valor, é um método de representação ou codificação e números. A notação posicional é diferente das outras notações por utilizar o mesmo símbolo para diferentes

ordens de magnitude (as unidades, as dezenas, as centenas, os milhares, etc.). Esta notação simplificou bastante a forma de executar as operações aritiméticas, o que levou à sua popularização relativamente rápida por todo o mundo.

Com o uso de um *ponto raiz*[1] (vírgula decimal), esta notação pode ser extendida para incluir números fracionários, como as expansões numéricas dos números reais. O sistema de numeração Hindo-Arábico é um exemplo de um sistema de numeração posicional baseado no número 10.

A.3.1 Base do Sistema de Numeração

Uma "base de numeração" é um conjunto de símbolos utilizado para escrever numerais. Em princípio qualquer conjunto finito de símbolos pode ser utilizado como base de numeração, entretanto, usualmente os dígitos do sistema de numeração hindo-arábicos e as letras do alfabeto latino são utilizadas. A "base" ou "radical" é o número de dígitos, incluíndo o zero, que um sistema posicional de numeração usa para representar números. Por exemplo, no sistema decimal a base é 10 porque ele usa 10 dígitos, de 0 a 9.

O símbolo mais alto de um sistema de numeração posicional tem o valor da base menos um. Quando o número que se quer representar é maior do que o mais alto dígito da base são acrescentados mais dígitos. Assim, na base 10, o número seguinte ao 9 é escrito colocando-se o "1" seguido do "0", que formam o número dez. Em binário a base é 2, assim o número que se segue ao 1 (o dígito mais alto da base 2) também é formado colocando-se o "1" seguido do "0" para formar o número dois.

Na notação posicional de base 10 (decimal), existem 10 dígitos e temos:

$$2506 = 2 \cdot 10^3 + 5 \cdot 10^2 + 0 \cdot 10^1 + 6 \cdot 10^0$$

Na notação posicional de base 16 (hexadecimal), existem 16 dígitos e temos:

$$171\mathrm{B} = 1 \cdot 16^3 + 7 \cdot 16^2 + 1 \cdot 16^1 + \mathrm{B} \cdot 16^0$$

onde o B representa o valor onze.

De modo geral, em um sistema posicional de base b, há b dígitos e temos:

$$a_k \ldots a_2 a_1 a_0 = a_k \cdot b^k + \ldots + a_2 \cdot b^2 + a_1 \cdot b^1 + a_0 \cdot b^0$$

Note que $a_k a_{k-1} \ldots a_2 a_1 a_0$ representa a sequência dos dígitos a_i e não a sua multiplicação.

As vezes o número da base aparece subescrito após o número representado. Por exemplo, 23_8 indica que o número 23 está expresso na base 8; 1111011_2 indica que o número 1111011 está na base 2; 173_{12} está na base 12; e $7\mathrm{B}_{16}$ está na base 16. Esta é a notação que usaremos neste texto. Caso a base não seja explicitamente indicada, assumiremos que a base é 10.

Ao descrever bases em notação matemática, a letra b é geralmente usada como um símbolo para representar a base, assim, para um sistema binário $b = 2$. Evidentemente, a base b também pode ser indicada pela fraze "base b". Assim números binário são "base 2", números octais são "base 8", número decimais são "base 10", e assim por diante.

As bases númericas mais comumente utilizadas são:[2]

Base	—	**Nomenclatura**	—	**Dígitos**
Base 2	—	binário	—	0 1
Base 8	—	octal	—	0 1 2 3 4 5 6 7
Base 10	—	decimal	—	0 1 2 3 4 5 6 7 8 9
Base 12	—	duodecimal	—	0 1 2 3 4 5 6 7 8 9 A B
Base 16	—	hexadecimal	—	0 1 2 3 4 5 6 7 8 9 A B C D E F
Base 20	—	vigesimal	—	0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J

A.3.2 Exponenciação

Sistemas de numeração posicionais funcionam baseados em exponenciação da base. O valor de um dígito é obtido multiplicando-se o valor do algarismo pelo valor de sua posição. O valor da posição é dado pelo valor da base multiplicado pela n-ésima potência, onde n é o número de outros dígitos entre um dado dígito e o ponto raiz. Se um determinado dígito está à esquerda do ponto raiz, então o valor de n é positivo ou zero; se o dígito está à esquerda do ponto raiz, então n é negativo.

[1] Também chamado de *ponto de origem*. Em língua portuguesa, espanhol, alemão e outras, o "ponto" é na verdade uma vírgula.

[2] Note que alguns altores não incluem o "I" na base vigesimal, alegando que o "I" é muito parecido com o "1"; esses altores consideram que os dígitos da base vigesimal são: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, G, H, J e K.

Como um exemplo, considere o número 465_b (note que $b \geq 7$, pois o algarismo mais alto do número é 6), assim:

$$465_b = 4 \cdot b^2 + 6 \cdot b^1 + 5 \cdot b^0$$

Se $b = 10$, então o valor seria:

$$\begin{aligned} 465_{10} &= 4 \cdot 10^2 + 6 \cdot 10^1 + 5 \cdot 10^0 \\ &= 400 + 60 + 5 \\ &= 465_{10} \end{aligned}$$

Entretanto, se o número estiver na base 7, então:

$$\begin{aligned} 465_7 &= 4 \cdot 7^2 + 6 \cdot 7^1 + 5 \cdot 7^0 \\ &= 4 \cdot 49 + 6 \cdot 7 + 5 \cdot 1 \\ &= 196 + 42 + 5 \\ &= 243_{10} \end{aligned}$$

Para qualquer base b, $10_b = b$. Isso pode ser demonstrado pelo fato de que $10_b = 1 \cdot b^1 + 0 \cdot b^0$, logo $10_b = 1 \cdot b$.

Números que não são inteiros possuem dígitos à direita do ponto raiz. Para cada posição além (à direita) deste ponto, a potência n diminui de 1. Por exemplo, o número $12,35_b$ é igual a:

$$12,35_b = 1 \cdot b^1 + 2 \cdot b^0 + 3 \cdot b^{-1} + 5 \cdot b^{-2}$$

Assim, se $b = 8$ temos:

$$\begin{aligned} 12,35_8 &= 1 \cdot 8^1 + 2 \cdot 8^0 + 3 \cdot 8^{-1} + 5 \cdot 8^{-2} \\ &= 1 \cdot 8 + 2 \cdot 1 + 3 \cdot \frac{1}{8} + 5 \cdot \frac{1}{64} \\ &= 8 + 2 + \frac{3}{8} + \frac{5}{64} \\ &= 10,453125_{10} \end{aligned}$$

Mas, se $b = 12$ temos:

$$\begin{aligned} 12,35_{12} &= 1 \cdot 12^1 + 2 \cdot 12^0 + 3 \cdot 12^{-1} + 5 \cdot 12^{-2} \\ &= 1 \cdot 12 + 2 \cdot 1 + 3 \cdot \frac{1}{12} + 5 \cdot \frac{1}{144} \\ &= 12 + 2 + \frac{3}{12} + \frac{5}{144} \\ &= 14,28472222\ldots_{10} \end{aligned}$$

A.3.3 Dígitos e Numerais

Um dígito é o que é usado como uma posição na notação posição-valor, e um núméral é uma sequência de um ou mais dígitos. A distinção entre um digito e um numeral é mais pronunciada ao se considerar diferentes bases de numeração.

Um numeral não-zero com mais de uma posição de dígito representará números diferentes em bases de numeração diferentes, mas de modo geral, os dígitos manterão o mesmo significado. Por exemplo, considere o numeral "23"; na base 8 este numeral representa o número dezenove, ao passo que na base 10 eles representam o número vinte e três. Entretanto, tanto na base 8 quanto na base 10 o dígito "2" representa "dois" e o dígito "3" representa "três".

O sistema de numeração mais comum nos sistemas de computação atuais é o sistema binário (base 2), mas também se utilizam os sistemas octal (base 8) e hexadecimal (base 16) uma vez que estes sistemas são congruentes ao sistema de numeração binário. Em binário, apenas os dígitos "0" e "1" são usados. Em octal, os dígitos de "0" a "7" são usados. Em hexadecimal, dezesseis dígitos são usados; os dígitos de "0" a "9" e de "A" a "F", com "A" representando dez, "B" representando onze, e assim por diante.

A.3.4 Exercícios

Exercício A.1 Escreva em ordem crescente os primeiros 25 números em cada uma das bases indicadas abaixo. Escreva os números na forma de uma tabela usando as bases como cabeçalhos das colunas.

(a) decimal
(b) binário
(c) quinário
(d) octal
(e) undecimal
(f) duodecimal
(g) hexadecimal
(h) vigesimal

Exercício A.2 Quantos números diferentes de no máximo três (3) dígitos é possível escrever em cada uma das bases indicadas abaixo:

(a) Base 2: _____
(b) Base 4: _____
(c) Base 5: _____
(d) Base 8: _____
(e) Base 10: _____
(f) Base 12: _____
(g) Base 16: _____
(h) Base 20: _____

Exercício A.3 Quanto dígitos são necessários para se escrever o numero 1111 em cada uma das bases abaixo?

(a) Base 2: _____
(b) Base 4: _____
(c) Base 5: _____
(d) Base 8: _____
(e) Base 10: _____
(f) Base 12: _____
(g) Base 16: _____
(h) Base 20: _____

Exercício A.4 Indique qual a menor base possível para os numerais indicados abaixo. Por exemplo, para 547_b, temos $b \geq 8$; e para $5A7_b$, temos $b \geq 11$.

(a) 0_b
(b) 1101_b
(c) 777_b
(d) 1024_b
(e) 666_b
(f) $1EE7_b$
(g) $A5A_b$
(h) $B1A_b$

Exercício A.5 Considere os seguintes símbolos: $\square, \diamond, \triangle, \star, \bowtie, \odot$. Considerando que esta sequência de símbolos forme uma base numérica, e que os símbolos foram dados em ordem aritmética, ou seja, o primeiro símbolo tem valor zero, o segundo tem valor um, etc. Determine o valor de cada um dos numerais abaixo (na base definida acima) usando o método de expoentes:

(a) $\diamond\square\star$
(b) $\star\square\square\odot$
(c) $\bowtie \diamond \star \square$
(d) $\odot\square\triangle\triangle$

Exercício A.6 Considere a base $\mathcal{L}$ formada pelas 26 letras do alfabeto latino ocidental, sendo $Z = 0, A = 1, B = 2, C = 3, \ldots, X = 24$ e $Y = 25$. Determine o valor de cada um dos numerais abaixo (na base $\mathcal{L}$) usando o método de expoentes:

(a) YOU
(b) $SHALL$
(c) NOT
(d) $PASS$

A.4 Conversão entre Bases Numéricas

É possível converter numeros de uma base de numeração b_1 para outra base b_2 diretamente, entretanto essa conversão pode ser bastante difícil se a pessoa realizando a conversão não estiver acostumada a realizar operações aritiméticas diretamente nas bases b_1 e b_2. Como a maioria das pessoas está mais acostumada a lidar com a base 10, é mais simples converter da base b_1 para a base 10 e da base 10 para a base b_2. A exceção a esta regra é quando as bases envolvidas são congruentes, ou seja, quando ambas são potências de uma base comum, como é o caso da conversão de base 8 para a base 16, por exemplo, pois ambas são potências de 2 (i.e., $8 = 2^3$ e $16 = 2^4$).

Um ponto importante a se ter em mente é a distinção entre a representação (o numeral) e o número. Não confunda a representação com o número. Por exemplo, cada um dos exemplos abaixo é uma representação diferente para o mesmo número:

- 11111111_2
- 100110_3
- 3333_4
- 2010_5
- 377_8
- 255_{10}
- 193_{12}
- FF_{16}
- CF_{20}

A.4.1 Conversão de Base b para Base 10

De modo geral, para um número expresso na base b por um numeral de n dígitos, i.e., para um númeral na forma $a_n \dots a_2 a_1 a_0$ na base b, o valor do número na base 10 pode ser calculado pela fórmula abaixo:

$$[a_n \dots a_2 a_1 a_0]_b = \left[\sum_{i=0}^{n} a_i \cdot b^i \right]_{10} \tag{A.1}$$

Ou seja, cada dígito a_i é multiplicado pela base b elevada à posição i do dígito. As posições são atribuídas da direita para a esquerda, iniciando em zero (0) para o dígito mais à direita. Expandindo o lado direito da equação (A.1) obtemos a equação na forma abaixo:

$$\begin{aligned} [a_n \dots a_2 a_1 a_0]_b &= \left[\sum_{i=0}^{n} a_i \cdot b^i \right]_{10} \\ &= \left[a_n \cdot b^n + \ldots + a_2 \cdot b^2 + a_1 \cdot b^1 + a_0 \cdot b^0 \right]_{10} \end{aligned} \tag{A.2}$$

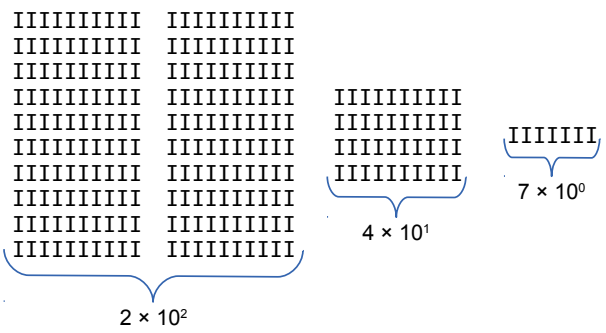
Abaixo temos alguns exemplos do uso da equação (A.2).

Exemplo A.1 (Conversão de 241_5 para base 10)

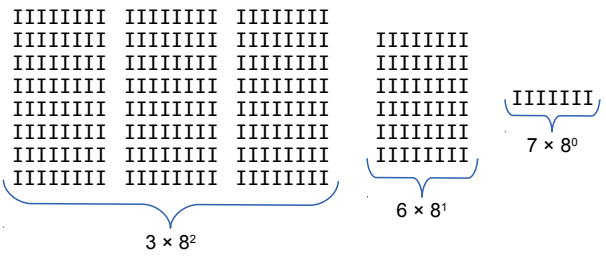
$$\begin{aligned} 241_5 &= 2 \cdot 5^2 + 4 \cdot 5^1 + 1 \cdot 5^0 \\ &= 2 \cdot 25 + 4 \cdot 5 + 1 \cdot 1 \\ &= 50 + 20 + 1 \\ &= 71_{10} \end{aligned}$$

Exemplo A.2 (Conversão de 193_{12} para base 10)

$$\begin{aligned} 193_{12} &= 1 \cdot 12^2 + 9 \cdot 12^1 + 3 \cdot 12^0 \\ &= 1 \cdot 144 + 9 \cdot 12 + 3 \cdot 1 \\ &= 144 + 108 + 3 \\ &= 255_{10} \end{aligned}$$



(a) Elementos agrupados em base 10.



(b) Elementos agrupados em base 8.

Figura A.1: Coleção de 247 elementos organizados em grupos (bases) de 10 e de 8 elementos.

Exemplo A.3 (Conversão de 23333_4 para base 10)

$$\begin{aligned} 23333_4 &= 2 \cdot 4^4 + 3 \cdot 4^3 + 3 \cdot 4^2 + 3 \cdot 4^1 + 3 \cdot 4^0 \\ &= 2 \cdot 256 + 3 \cdot 64 + 3 \cdot 16 + 3 \cdot 4 + 3 \cdot 1 \\ &= 512 + 192 + 48 + 12 + 3 \\ &= 767 \end{aligned}$$

Exemplo A.4 (Conversão de 1337_8 para base 10)

$$\begin{aligned} 1337_8 &= 1 \cdot 8^3 + 3 \cdot 8^2 + 3 \cdot 8^1 + 7 \cdot 8^0 \\ &= 1 \cdot 512 + 3 \cdot 64 + 3 \cdot 8 + 7 \cdot 1 \\ &= 512 + 192 + 24 + 7 \\ &= 735_{10} \end{aligned}$$

A.4.2 Conversão de Base 10 para Base b

Como seria de se esperar, o processo de conversão de um numeral na base 10 para uma base b qualquer, segue a mecânica inversa à da conversão da base b para a base 10, ou seja, ao invés de efetuarmos uma série de multiplicações pelas diversas potências da base, fazemos uma série de divisões pela base.

O processo se baseia no fato de que, quando fazemos a divisão inteira de um número a por um número b, obtendo um quociente q e um resto r, podemos escrever a seguinte igualdade:

$$a = b \cdot q + r \tag{A.3}$$

A equação (A.3) nos permite visualizar um algoritmo para converter número em base 10 para números na base b através de divisões sucessivas.

Assim, se tomarmos um número a na base 10 e o dividirmos pela base b, obteremos um quociente q_0 que representa a quantidade de vezes que o número a "repete" a base b e um resto r_0 que representa a quantidade de "unidades" de a que sobraram "dentro" de uma repetição de b.

Se repetirmos o processo utilizando q_0 como dividendo, então obteremos um novo quociente q_1 que representará quantas vezes o número q_0 repete a base b. Como cada unidade de q_0 já representava grupos de b elementos, cada unidade de q_1 representará grupos de b^2 elementos.

Cada divisão sucessiva aumenta a base em uma potência, formando grupos cada vez maiores, e em cada divisão sucessiva, o resto indica a quantidade de elementos ou grupos que sobraram do agrupamento. A Figura A.1 ilustra este processo de reagrupamento das unidades que compõem o número, mostrando o mesmo número de unidade arranjado em grupos (bases) de 10 e de 8 elementos.

Se repetirmos o processo de divisões sucessivas pela base, até atigirmos uma divisão em que o quociente seja zero (0), os restos destas diviões nos indicarão quantos grupos de cada potência sucessiva da base podemos formar. Os exemplos abaixo ilustram este processo.

Exemplo A.5 (Conversão de 247_{10} para base 8)

#	base	quociente	resto
247	8	30	7
30	8	3	6
3	8	0	3

Tomando os restos sucessivos, da última divisão para a primeira temos: $247_{10} = 376_8$.

Exemplo A.6 (Conversão de 255_{10} para base 12)

#	base	quociente	resto
255	12	21	3
21	12	1	9
1	12	0	1

$\therefore 255_{10} = 193_{12}$

Exemplo A.7 (Conversão de 777_{10} para base 5)

#	base	quociente	resto
777	5	155	2
155	5	31	0
31	5	6	1
6	5	1	1
1	5	0	1

$\therefore 777_{10} = 11102_5$

Exemplo A.8 (Conversão de 1775_{10} para base 16)

#	base	quociente	resto
1775	16	110	$15 \rightarrow F$
110	16	6	$14 \rightarrow E$
6	16	0	6

$\therefore 1775_{10} = 6EF_{16}$

A.4.3 Conversão entre Bases Congruentes

Neste texto, usaremos o termo *bases congruentes* para designar bases, b_1 e b_2 sempre que a base b_2 for uma potência da base b_1, por exemplo: as bases 2 e 4 (2^2); 2 e 8 (2^3); 3 e 9 (3^2); 4 e 16 (4^2); e 5 e 25 (5^2) são todas congruentes duas a duas.

Uma característica interessante das bases congruentes é que sempre é possível converter um dígito da base mais alta em um número inteiro de dígitos da base mais baixa. Por exemplo:

- cada dígito em um numeral em base 8 corresponde a exatamente três digitos em um numeral em base 2;
- cada dígito em um numeral em base 16 corresponde a exatamente dois dígitos em um numeral em base 4;
- cada dígito em um numeral em base 16 corresponde a exatamente quatro dígitos em um numeral em base 2;
- cada dígito em um numeral em base 25 correspondem a exatamente dois dígitos em um numeral em base 5;

Tabela A.1: Conversões entre as bases congruentes 2–8, 3–9, 4–16.

(a) Base 8 vs. Base 2

B.8	B.2
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

(b) Base 9 vs. Base 3

B.9	B.3
0	00
1	01
2	02
3	10
4	11
5	12
6	20
7	21
8	22

(c) Base 16 vs. Base 4

B.16	B.4	B.16	B.4
0	00	8	20
1	01	9	21
2	02	A	22
3	03	B	23
4	10	C	30
5	11	D	31
6	12	E	32
7	13	F	33

De modo geral, se $b_2 = b_1^n$, então cada dígito da base b_2 corresponderá a exatamente n dígitos na base b_1. Desta forma o processo de conversão da base b_2 para a base b_1 se torna bastante simples, basta converter diretamente cada dígito do numeral na base b_2 para n dígitos no numeral na base b_1.

Exemplo A.9 (Conversão de base congruente maior para menor)

- $777_8 = \overbrace{111}^{7}\overbrace{111}^{7}\overbrace{111}^{7}{}_2$
- $753_8 = \overbrace{111}^{7}\overbrace{101}^{5}\overbrace{011}^{3}{}_2$
- $46_8 = \overbrace{100}^{4}\overbrace{110}^{6}{}_2$
- $\mathrm{B4D}_{16} = \overbrace{1011}^{\mathrm{B}}\overbrace{0100}^{4}\overbrace{1101}^{\mathrm{D}}{}_4$
- $8567_9 = \overbrace{22}^{8}\overbrace{12}^{5}\overbrace{20}^{6}\overbrace{21}^{7}{}_3$
- $\mathrm{C4F3}_{16} = \overbrace{30}^{\mathrm{C}}\overbrace{10}^{4}\overbrace{33}^{\mathrm{F}}\overbrace{03}^{3}{}_4$

Para efeito de demonstração, a Tabela A.1 mostra conversões entre dígitos de algumas bases congruentes. Tabelas semelhantes podem ser construídas entre outras bases congruentes como 2 e 16, ou 5 e 25, por exemplo.

Evidentemente a operação inversa, ou seja, dados dois númerais nas bases b_1 e b_2, onde $b_2 = b_1^n$, é possível converter o numeral da base b_1 "diretamente" para a base b_2, formando-se grupos de n dígitos (da direita para a esquerda) do numeral na base b_1 e convertendo cada grupo para o dígito correspondente na base b_2.

Exemplo A.10 (Conversão de base congruente menor para maior)

- $10110011101_2 = \overbrace{2}^{10}\overbrace{6}^{110}\overbrace{3}^{011}\overbrace{5}^{101}{}_8$
- $010011111110_2 = \overbrace{4}^{0100}\overbrace{\mathrm{F}}^{1111}\overbrace{\mathrm{E}}^{1110}{}_{16}$
- $13312211_4 = \overbrace{7}^{13}\overbrace{\mathrm{D}}^{31}\overbrace{\mathrm{A}}^{22}\overbrace{5}^{11}{}_{16}$
- $2122111_3 = \overbrace{2}^{2}\overbrace{5}^{12}\overbrace{7}^{21}\overbrace{4}^{11}{}_9$
- $11110_2 = \overbrace{3}^{11}\overbrace{6}^{110}{}_8$
- $12212101_3 = \overbrace{\mathrm{E}}^{12}\overbrace{\mathrm{W}}^{212}\overbrace{\mathrm{J}}^{101}{}_{27}$

Note que a base 27, usada acima, utiliza o dígito 0 e as letras de A até Z, sendo A = 1, B = 2, C = 3, ..., Z = 26.

A.4.4 Exercícios

Exercício A.7 Converta os numerais em binário dados abaixo para a base 10.

(a) 1101_2
(b) 1010_2
(c) 10101_2
(d) 11010_2
(e) 110011_2
(f) 101010_2
(g) 1111011_2
(h) 1101111_2
(i) 1001101_2
(j) 10011100_2
(k) 01001010_2
(l) 11010110_2

Exercício A.8 Converta os numerais em quaternário dados abaixo para a base 10.

(a) 32_4
(b) 13_4
(c) 123_4
(d) 232_4
(e) 1332_4
(f) 3113_4

Exercício A.9 Converta os numerais em quinário dados abaixo para a base 10.

(a) 32_5
(b) 14_5
(c) 143_5
(d) 234_5
(e) 4332_5
(f) 3413_5

Exercício A.10 Converta os numerais em octal dados abaixo para a base 10.

(a) 32_8
(b) 73_8
(c) 543_8
(d) 274_8
(e) 1337_8
(f) 3456_8

Exercício A.11 Converta os numerais em duodecimal dados abaixo para a base 10.

(a) 32_{12}
(b) $A3_{12}$
(c) $B43_{12}$
(d) $2AB_{12}$
(e) $1BB7_{12}$
(f) $357B_{12}$

Exercício A.12 Converta os numerais em hexadecimal dados abaixo para a base 10.

(a) 32_{16}
(b) AE_{16}
(c) $BA3_{16}$
(d) $2AB_{16}$
(e) $1BBF_{16}$
(f) $3D7B_{16}$

Exercício A.13 Converta os numerais dados abaixo de suas respectivas bases para a base 10.

(a) 101010_2
(b) 101010_3
(c) 1021_4
(d) 1025_6
(e) 2165_8
(f) $1FA2_{16}$
(g) $E1A_{16}$
(h) 707_8
(i) 100110000_2
(j) 11011_5
(k) $1BA2_{12}$
(l) $C1A_{20}$

Exercício A.14 Converta os numerais abaixo da base 10 para a binário, octal e hexadecimal.

(a) 10
(b) 31
(c) 100
(d) 120
(e) 127
(f) 254
(g) 1025
(h) 4095
(i) 65535

Exercício A.15 Converta os numerais abaixo da base 10 para as bases 5 e 7.

(a) 15
(b) 45
(c) 153
(d) 211
(e) 272
(f) 549
(g) 1023
(h) 4949
(i) 65535

Exercício A.16 Converta os numerais abaixo da base 10 para hexadecimal.

(a) 365
(b) 5267
(c) 4612
(d) 170742
(e) 161051
(f) 465
(g) 3420
(h) 6341
(i) 32575

Exercício A.17 Converta os números abaixo de binário para octal.

(a) 110101010_2
(b) 100100010_2
(c) 10100000_2
(d) 101001_2
(e) 10100101000_2
(f) 111101001111_2
(g) 1100100110_2
(h) 10110100100_2
(i) 111001111110_2

Exercício A.18 Converta os números abaixo de binário para hexadecimal.

(a) 1110100001010_2
(b) 1010110010001001_2
(c) 11100111111_2
(d) 00101001011111_2
(e) 000101011111_2
(f) 1111100110100_2
(g) 1111010111_2
(h) 1011111010_2
(i) 110010110_2

Exercício A.19 Converta os números abaixo de octal para hexadecimal.

(a) 50533_8
(b) 44772_8
(c) 7675_8
(d) 761_8
(e) 3530_8
(f) 43160_8
(g) 5571_8
(h) 552_8
(i) 174_8

Exercício A.20 Converta os números abaixo de hexadecimal para octal.

(a) $\mathrm{BB7}_{16}$
(b) $\mathrm{FE4}_{16}$
(c) $\mathrm{2C5}_{16}$
(d) $\mathrm{1B37E}_{16}$
(e) $\mathrm{062FC}_{16}$
(f) $\mathrm{5E68}_{16}$
(g) $\mathrm{A2}_{16}$
(h) $\mathrm{DA572}_{16}$
(i) $\mathrm{540F}_{16}$

A.5 Operações Aritméticas na Base b

Os algoritmos convencionais para adição, subtração, multiplicação e divisão que são ensinados no ensino fundamental para o sistema de numeração posicional de base 10 também funcionam para qualquer outra base. Basta que se ajuste adequadamente a forma de representar os números.

A.5.1 Adição

Adição é uma das operações básicas da álgebra. Na sua forma mais simples, adição combina dois números (termos, somandos ou parcelas), em um único número, a soma. Adicionar mais números corresponde a repetir a operação. Uma adição é representada por:

$$a + b = c \tag{A.4}$$

Caso os númerais a e b na Equação (A.4) possuam apenas um dígito, a adição será feita seguindo o procedimento "padrão" de acrescentar b unidades ao numeral a. Por exemplo, na base hexadecimal temos $\mathrm{A} + 9 = 13$, ou seja, foram acrescentadas 9_{16} unidades, ao numeral A_{16}, e a soma foi 13_{16}. Por exemplo:

- $1_2 + 1_2 = 11_2$
- $3_4 + 1_4 = 10_4$
- $6_8 + 5_8 = 13_8$
- $7_8 + 1_8 = 10_8$
- $A_{12} + 7_{12} = 15_{12}$
- $B_{12} + 9_{12} = 18_{12}$
- $E_{16} + A_{16} = 18_{16}$
- $F_{16} + 7_{16} = 16_{16}$
- $H_{20} + A_{20} = 17_{20}$

Para adições envolvendo numerais com mais do que um dígito, utilizamos o algoritmo de adição manual – que é o mesmo aprendido no ensino fundamental –, que consiste em escrever as duas parcelas alinhadas à direita de modo que os dígitos das posições correspondentes dos dois numerais fiquem alinhados. Em seguida, realiza-se a soma dos dígitos dois-a-dois das duas parcelas, iniciando-se com os dígitos da posição zero, em seguida com os da posição um, e assim por diante. Caso a soma de algum par de dígitos resulte em um numeral com mais do que um dígito, o dígito de mais baixa ordem da soma é escrito no resultado e os demais digitos são promovidos para a próxima etapa da soma.

Note que só faz sentido efetuar o algoritmo de adição se os dois numerais sendo somados estiverem na mesma base.

Exemplo A.11 (Adição dos numerais 442_8 e 5473_8)

1. Escrevemos as duas parcelas alinhadas à direita:

$$\begin{array}{cccccc} & & 4 & 4 & 2 \\ + & 5 & 4 & 7 & 3 \\ \hline \end{array}$$

2. Somamos os dígitos da posição 0; na base 8: $2 + 3 = 5$

$$\begin{array}{cccccc} & & 4 & 4 & 2 \\ + & 5 & 4 & 7 & 3 \\ \hline & & & & 5 \end{array}$$

3. Somamos os dígitos da posição 1; na base 8: $4 + 7 = 13$ — uma vez que a soma possui mais de um dígito, escrevemos o dígito de mais baixa ordem (3) no resultado e promovemos os demais dígito para a próxima etapa da soma.

$$\begin{array}{cccccc} & & \overset{1}{4} & 4 & 2 \\ + & 5 & 4 & 7 & 3 \\ \hline & & & 3 & 5 \end{array}$$

4. Somamos os dígitos da posição 2 (incluindo os promovidos da etapa anterior); na base 8: $1+4 = 5$ e $5+4 = 11$ — novamente, escrevemos o dígito de mais baixa ordem e promovemos os demais para a próxima etapa.

$$\begin{array}{cccccc} & 1 & \overset{1}{4} & 4 & 2 \\ + & 5 & 4 & 7 & 3 \\ \hline & & 1 & 3 & 5 \end{array}$$

5. Somamos os dígitos da posição 3; na base 8: $1 + 5 = 6$

$$\begin{array}{cccccc} & 1 & \overset{1}{4} & 4 & 2 \\ + & 5 & 4 & 7 & 3 \\ \hline & 6 & 1 & 3 & 5 \end{array}$$

6. Portanto, $442_8 + 5473_8 = 6135_8$.

Exemplo A.12 (Adição dos numerais $\mathrm{FA84}_{16}$ e $\mathrm{1ADC7}_{16}$)

1. Escrevemos as duas parcelas alinhadas à direita:

$$\begin{array}{rccccc} & 1 & \mathrm{A} & \mathrm{D} & \mathrm{C} & 7 \\ + & & \mathrm{F} & \mathrm{A} & 8 & 4 \\ \hline \end{array}$$

2. Somamos os dígitos da posição 0; na base 16: $7 + 4 = B$

$$\begin{array}{rccccc} & 1 & \mathrm{A} & \mathrm{D} & \mathrm{C} & 7 \\ + & & \mathrm{F} & \mathrm{A} & 8 & 4 \\ \hline & & & & & \mathrm{B} \end{array}$$

3. Somamos os dígitos da posição 1; na base 16: $\mathrm{C} + 8 = 14$ — escrevemos o 4 e promovemos o 1.

$$\begin{array}{rccccc} & & & \overset{1}{\mathrm{D}} & & \\ & 1 & \mathrm{A} & \mathrm{D} & \mathrm{C} & 7 \\ + & & \mathrm{F} & \mathrm{A} & 8 & 4 \\ \hline & & & & 4 & \mathrm{B} \end{array}$$

4. Somamos os dígitos da posição 2; na base 16: $1 + \mathrm{D} = \mathrm{E}$ e $\mathrm{E} + \mathrm{A} = 18$ — escrevemos o 8 e promovemos o 1.

$$\begin{array}{rccccc} & & \overset{1}{\mathrm{A}} & \overset{1}{\mathrm{D}} & & \\ & 1 & \mathrm{A} & \mathrm{D} & \mathrm{C} & 7 \\ + & & \mathrm{F} & \mathrm{A} & 8 & 4 \\ \hline & & & 8 & 4 & \mathrm{B} \end{array}$$

5. Somamos os dígitos da posição 3; na base 16: $1 + \mathrm{A} = \mathrm{B}$ e $\mathrm{B} + \mathrm{F} = 1\mathrm{A}$ — escrevemos o A e promovemos o 1.

$$\begin{array}{rccccc} & \overset{1}{1} & \overset{1}{\mathrm{A}} & \overset{1}{\mathrm{D}} & & \\ & 1 & \mathrm{A} & \mathrm{D} & \mathrm{C} & 7 \\ + & & \mathrm{F} & \mathrm{A} & 8 & 4 \\ \hline & & \mathrm{A} & 8 & 4 & \mathrm{B} \end{array}$$

6. Somamos os dígitos da posição 4; na base 16: $1 + 1 = 2$.

$$\begin{array}{rccccc} & \overset{1}{1} & \overset{1}{\mathrm{A}} & \overset{1}{\mathrm{D}} & & \\ & 1 & \mathrm{A} & \mathrm{D} & \mathrm{C} & 7 \\ + & & \mathrm{F} & \mathrm{A} & 8 & 4 \\ \hline & 2 & \mathrm{A} & 8 & 4 & \mathrm{B} \end{array}$$

A.5.2 Subtração

Subtração é uma operação matemática que indica quanto é um valor numérico resultante se, de um valor inicial (minuendo), for removido outro valor numérico (subtraendo). Uma subtração é representada por:

$$a - b = c \tag{A.5}$$

Caso os numerais a e b na Equação (A.5) possuam apenas um dígito e o numeral a repesente um número maior do que b, então a subtração $a - b$ poderá ser calculada simplesmente "contando-se" quanta unidades a possui a mais do que b. Por exemplo:

- $1_2 - 0_2 = 1_2$
- $3_4 - 1_4 = 2_4$
- $6_8 - 5_8 = 1_8$
- $7_8 - 1_8 = 6_8$
- $A_{12} - 7_{12} = 3_{12}$
- $B_{12} - 5_{12} = 6_{12}$
- $E_{16} - A_{16} = 4_{16}$
- $F_{16} - 9_{16} = 6_{16}$
- $H_{20} - 8_{20} = A_{20}$

Entretanto, se o minuendo e/ou o subtaendo forem compostos por mais de um dígito, deve-se utilizar o algoritmos manual de subtração. Este algoritmo é, em essência, o mesmo que é aprendido no ensino fundamental, exceto pelo fato de que as subtrações dígito-a-dígito devem ser efetuadas na mesma base do minuando e do subtraendo. (Note que só faz sentido subtrair dois númerais se eles estiverem na mesma base.)

O algoritmo de subtração manual pode ser resumido da seguinte forma:

1. Escreve-se o minuendo e abaixo dele, alinhados à direita, o subtraendo. Ambos os numerais devem ser escrito de tal modo que o dígito na posição zero do minuendo esteja diretamente acima do dígito da posição zero do subtraendo, o dígito da posição um do minuendo esteja diretamente acima do dígito da posição um do subtraendo, e assim por diante.

2. Cada dígito do minuendo e do subtraendo constituem uma etapa da subtração, assim o dígito zero corresponderá à etapa zero, o dígito um à etapa um, etc.
3. Para cada etapa da subtração:
   a) Se o valor do minuendo para esta etapa for maior do que o valor do subtraendo, então realize a subtração simples entre os dois valores e escreva o resultado.
   b) Se o valor do minuendo para esta etapa for menor do que o valor do subtraendo, então uma unidade de potência deverá ser subtraída da etapa porterior (do dígito à esquerda) e adicionada ao valor desta etapa, após adicionar esta unidade de potência deve-se fazer a subtração simples e escrever o resultado.

Exemplo A.13 (Subração dos numerais 1337_8 e 235_8)

1. Escrever o minuendo e o subtraendo alinhados à direita.

	1	3	3	7
−		2	3	5

2. Subtrair os valores da etapa 0; na base 8: $7 - 5 = 2$.

	1	3	3	7
−		2	3	5
				2

3. Subtrair os valores da etapa 1; na base 8: $3 - 3 = 0$.

	1	3	3	7
−		2	3	5
			0	2

4. Subtrair os valores da etapa 2; na base 8: $3 - 2 = 1$.

	1	3	3	7
−		2	3	5
		1	0	2

5. Subtrair os valores da etapa 3; na base 8: $1 - 0 = 1$. Note que a ausência de dígitos à esquerda é interpretada como zero.

	1	3	3	7
−		2	3	5
	1	1	0	2

Exemplo A.14 (Subração dos numerais $1A7CD_{16}$ e $38F5_{16}$)

1. Escrever o minuendo e o subtraendo alinhados à direita.

	1	A	7	C	D
−		3	8	F	5

2. Subtrair os valores da etapa 0; na base 16: $D - 5 = 8$.

	1	A	7	C	D
−		3	8	F	5
					8

3. Subtrair os valores da etapa 1; em hexadecimal $C < F$ portanto não é possível fazer a subtração simples nesta etapa; é necessário subtrair uma unidade de potência do dígito da etapa à esquerda, ou seja, é preciso

"pegar emprestado" uma unidade do dígito 7 na posição 2 e adicionar esta unidade de potência ao dígito C na posição 1. Só depois podemos fazer a subtração em hexadecimal $1C - F = D$.

	1	A	$\not{7}^{6}$	^{1}C	D
$-$		3	8	F	5
				D	8

4. Subtrair os valores da etapa 2; uma vez que $6 < 8$, não é possível fazer a subtração simples; é necessário "pegar emprestado" uma unidade do dígito A na posição 3 e adicionar esta unidade de potência ao dígito 6 na posição 2. Só depois podemos fazer a subtração em hexadecimal $16 - 8 = E$.

	1	$\not{A}^{9}$	$\not{7}^{16}$	^{1}C	D
$-$		3	8	F	5
			E	D	8

5. Subtrair os valores da etapa 3; em hexadecimal: $9 - 3 = 6$.

	1	$\not{A}^{9}$	$\not{7}^{16}$	^{1}C	D
$-$		3	8	F	5
		6	E	D	8

6. Subtrair os valores da etapa 3; em hexadecimal: $1 - 0 = 1$.

	1	$\not{A}^{9}$	$\not{7}^{16}$	^{1}C	D
$-$		3	8	F	5
	1	6	E	D	8

A.5.3 Multiplicação

Na sua forma mais simples a multiplicação é uma forma simples de se adicionar uma quantidade finita de números iguais. O resultado da multiplicação de dois números é chamado produto. Os números sendo multiplicados são chamados de coeficientes, fatores ou operandos, e individualmente de multiplicando e multiplicador. A multiplicação é representada por:

$$a \cdot b = c \quad \text{ou} \quad a \times b = c$$

A.5.4 Divisão

Divisão é a operação matemática inversa da multiplicação. O ato de dividir por um elemento de um conjunto só faz sentido quando a multiplicação por aquele elemento for uma função bijetora. No conjunto dos número inteiro a propriedade de bijetividade não é satisfeita para o zero (0), assim não se define divisão por zero. A divisão é representada por:

$$a \div b = c \quad \text{ou} \quad \frac{a}{b} = c$$

A.5.5 Exercícios

Exercício A.21 Efetue as adições indicadas abaixo com os número em suas respectivas bases.

(a) $01010110_2 + 1000110_2$

(b) $1110000_2 + 100111_2$

(c) $100110000_2 + 100111101_2$

(d) $000001001_2 + 111001_2$

(e) $1211030_4 + 32000_4$

(f) $1331331_4 + 3301122_4$

(g) $3120322_4 + 1121000_4$

(h) $23301_4 + 1231030_4$

(i) $505052_8 + 4034_8$

(j) $4306_8 + 77166_8$

(k) $43254_8 + 132042_8$

(l) $327104_8 + 373277_8$

(m) $6A41_{12} + 310_{12}$

(n) $6A5B50_{12} + BA342_{12}$

(o) $75A6B4_{12} + 786625_{12}$

(p) $798A2_{12} + 749220_{12}$

(q) $4DE058_{16} + 564DF_{16}$

(r) $85DE_{16} + F1365E_{16}$

(s) $\text{E2C1}_{16} + \text{A123}_{16}$

(t) $\text{64FA4D}_{16} + \text{E5AFFE}_{16}$

Exercício A.22 Efetue as subtrações indicadas abaixo com os número em suas respectivas bases.

(a) $100101101_2 - 10110010_2$

(b) $10011_2 - 101110_2$

(c) $1100100_2 - 1011_2$

(d) $110000110_2 - 10111000_2$

(e) $3121020_4 - 1212131_4$

(f) $123020_4 - 122302_4$

(g) $3132320_4 - 2202122_4$

(h) $232021_4 - 230001_4$

(i) $515377_8 - 3550213_8$

(j) $2204420_8 - 57743_8$

(k) $5512_8 - 601411_8$

(l) $4560407_8 - 252111_8$

(m) $\text{6870B}_{12} - \text{B2B917}_{12}$

(n) $86712_{12} - 1423_{12}$

(o) $\text{41308A}_{12} - 96599_{12}$

(p) $\text{9BA29B}_{12} - \text{48A50}_{12}$

(q) $\text{66BA94}_{16} - \text{49FEC}_{16}$

(r) $\text{A27A42}_{16} - \text{A7FD72}_{16}$

(s) $\text{C7C83}_{16} - \text{1855BC}_{16}$

(t) $\text{9337C}_{16} - \text{30BDE4}_{16}$

Exercício A.23 Efetue as multiplicações indicadas abaixo com os número em suas respectivas bases.

(a) $1100_2 \cdot 11101_2$

(b) $1011010_2 \cdot 10100_2$

(c) $10010_2 \cdot 110100_2$

(d) $101001_2 \cdot 10100_2$

(e) $32210_4 \cdot 12033_4$

(f) $30101_4 \cdot 20121_4$

(g) $3212_4 \cdot 13013_4$

(h) $1033_4 \cdot 21132_4$

(i) $562_8 \cdot 6606_8$

(j) $73042_8 \cdot 10764_8$

(k) $7140_8 \cdot 3071_8$

(l) $7422_8 \cdot 145_8$

(m) $4564_{12} \cdot \text{2B4}_{12}$

(n) $942_{12} \cdot \text{B10}_{12}$

(o) $\text{7B5}_{12} \cdot 278_{12}$

(p) $\text{17A2}_{12} \cdot \text{A35}_{12}$

(q) $\text{9A}_{16} \cdot 20_{16}$

(r) $89_{16} \cdot 109_{16}$

(s) $0787_{16} \cdot 36_{16}$

(t) $\text{5B0}_{16} \cdot \text{B9}_{16}$

Exercício A.24 Para cada uma das equações abaixo, determine qual é a base b_i na qual os numerais estão representados.

(a) $\text{3B8}_{b_1} + 608_{b_1} = \text{9C1}_{b_1}$

(b) $\text{93B23}_{b_2} + \text{9A2F8}_{b_2} = \text{10DDHB}_{b_2}$

(c) $134140_{b_3} + 3124410_{b_3} = 3314100_{b_3}$

(d) $6124536_{b_4} + 2253134_{b_4} = 11411003_{b_4}$

A.6 Números em Complemento de Dois

Em computação, a expressão *complemento para dois* ou *complemento de dois* pode significar tanto uma operação matemática sobre números em binário quanto um tipo de representação de números em binário com sinal baseada nesta operação. A notação de complemento de dois é amplamente usada nas arquiteturas dos dispositivos computacionais modernos. Nesta representação dígito mais significativo, i.e., o dígito mais à esquerda, é o que informa o sinal do número. Se este dígito for 0 o número é positivo, e se for 1 é negativo.

O complemento de dois de um número x de n-bits[3] é definido como o complemento em relação à 2^n; em outras palavras, é o resultado de subtrair x de 2^n. O mesmo resultado pode ser conseguido tomando-se o *complemento de um* do número x, escrito como $\bar{x}$, e em seguida somando-se 1 a $\bar{x}$. O *complemento de um* de um número é calculado invertendo-se todos os bits de um número, ou seja, trocando-se todos os 0's por 1's e vice-versa. O complemento de dois de um número x se comporta como o negativo de x na maioria das operações aritiméticas, de modo que números positivos e negativos podem coexistir de modo natural, sem a necessidade de sinais.

Na notação de complemento de dois, números negativos são representados pelo complemento de dois de seu valor absoluto. Este sistema é o método mais comum de representar números inteiros com sinal em sistemas digitais. Um número de n-bits na notação de complemento de dois pode representar todos os inteiros na faixa de $-(2^{n-1})$ até $+(2^{n-1}-1)$.

O sistema de complemento de complemento de dois tem a vantagem de que as operações aritméticas fundamentais de adição, subtração e mutiplicação são idênticas àquelas para números binários sem sinal, desde que todos os números envolvidos sejam representados com a mesma quantidade de bits e que qualquer *overflow*[4] ou *underflow*[5] que exceda estes bits seja descartado. Esta propriedade torna o sistema ao mesmo tempo mais simples de implementar e capaz de manipular facilmente aritmética de alta precisão.

Para identificar que um número está representado em complemento de dois, anexaremos o subscrito $\bar{2}$ (o número dois com uma barra acima) ao número. Por exemplo: $00101101_{\bar{2}}$, $10010110_{\bar{2}}$, etc. Desta forma, poderemos diferenciar entre a notação em complemento de dois e a notação binária comum.

A.6.1 Conversão em Complemento de Dois

Podemos falar de conversões em complemento de dois em quatro situações mais comuns:

1. Conversão de complemento de dois para decimal;
2. Conversão de decimal para complemento de dois;
3. Conversão de binário (com sinal) para complemento de dois;
4. Conversão de complemento de dois para binário (com sinal);

Falaremos sobre estas conversões nas seções abaixo.

Conversão de Complemento de Dois para Decimal

O sistema numérico de complemento de dois codifica números negativos e positivos como números binários. O "peso" de cada bit é uma potência de dois exceto para o bit mais significante (o mais a esquerda), cujo peso é o negativo da potência de dois correspondente.

No sistema numérico de complemento de dois, o valor w de um inteiro de n bits representado por $a_{n-1}a_{n-2}\ldots a_0$ (onde cada a_i, para i de $n-1$ até 0, representa um bit) é dado pela fórmula abaixo:

$$w = -a_{n-1} \cdot 2^{n-1} + \sum_{i=0}^{n-2} a_i \cdot 2^i \tag{A.6}$$

Desta forma, o bit mais significante determina o sinal do número e é comumente chamado de "bit de sinal". Diferentemente do representação "sinal-e-magnitude", o bit de sinal também tem um "peso" associado que a potência 2^{n-1} mostrada na Equação (A.6), porém este peso é sempre negativo. Usando n bits em complemento de dois é possível representar todos os números inteiros de $-(2^{n-1})$ até $2^{n-1}-1$.

Exemplo A.15 (Converter $00101101_{\bar{2}}$ para decimal)

[3] Um digito em binário é chamado de um *bit*. Uma sequência de 4 bits é chamada de um *nibble*. Uma sequência de 8 bits é chamada de um *byte* ou *octeto*.

[4] O termo *overflow* descreve a condição que ocorre quando uma operação aritmética produz um resultado que tem magnitude maior do que aquela que um dado registrador ou espaço de armazenamento pode armazenar, i.e., o resultado exigiria mais bits do que estão disponíveis no espaço de armazenamento.

[5] O termo *underflow* descreve uma condição na qual o resultado de um cálculo é menor do que pode ser armazenado no espaço de armazenamento disponível; é uma condição análoga ao *overflow*, porém para valores negativos.

Seguindo a Equação (A.6):

$$\begin{aligned} 00101101_{\bar{2}} &= -0 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \\ &= -0 \cdot 128 + 0 \cdot 64 + 1 \cdot 32 + 0 \cdot 16 + 1 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 \\ &= -0 + 0 + 32 + 0 + 8 + 4 + 0 + 1 \\ &= 45 \end{aligned}$$

$\therefore 00101101_{\bar{2}} = 45_{10}$

Exemplo A.16 (Converter $11101001_{\bar{2}}$ para decimal)

Seguindo a Equação (A.6):

$$\begin{aligned} 11101001_{\bar{2}} &= -1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \\ &= -1 \cdot 128 + 1 \cdot 64 + 1 \cdot 32 + 0 \cdot 16 + 1 \cdot 8 + 0 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 \\ &= -128 + 64 + 32 + 0 + 8 + 0 + 0 + 1 \\ &= -23 \end{aligned}$$

$\therefore 11101001_{\bar{2}} = -23_{10}$

Conversão de Decimal para Complemento de Dois

Não se conhece um algoritmo para conversão direta de números em decimal com sinal para complemento de dois. O processo normalmente utilizado para se realizar a conversão manualmente é primeiro converter o número em decimal para binário com sinal e em seguida converter o número binário com sinal para complemento de dois.

Conversão para Complemento de Dois

Em notação de complemento de dois, um número inteiro não negativo é representado simplesmente por sua notação em binária; neste caso o bit mais significativo é zero (0). Entretanto, deve-se prestar atenção para o fato de que a faixa de valores representada não é a mesma que na notação em binário. Por exemplo, um número de 8 bits em binário pode representar os valores com magnitude de 0 a 255 (11111111_2). Entretanto, um número de 8 bits em complemento de dois pode representar apenas os número não-negativos de 0 a 127 (0111111_2), porque todas as outras combinações de bits, com o bit mais significativo como 1, representam os valores negativos de -1 a -128.

A operação de complemento de dois é *elemento inverso*[6], ou *oposto aditivo*[7], de modo que os números negativos são representados pelo complemento de dois do valor absoluto do número. Como dito, a operação de complemento de dois de um número de n bits é definida como o complemento em relação a 2^n, o complemento de dois de um número binário negativo pode ser obtido subtraindo-se o valor absoluto deste número de 2^n em binário, ou seja, um número binário de $n + 1$ bits, com o primeiro bit 1 e os demais 0.

Exemplo A.17 (Converter -101001_2 para complemento de dois com 8 bits)

O valor absoluto do número a ser convertido é 101001_2.
Este valor será subtraído de 2^8 (i.e., 100000000_2).

[6] *Elemento inverso* é aquele cuja utilização numa operação binária matemática bem definida resulta no elemento neutro específico dessa operação.
[7] *Oposto aditivo* de um número x é o número y tal que $x + y = 0$. Em notação convencional, o oposto aditivo de um número x é escrito $-x$.

$$\begin{array}{cccccccccc} & \not{1}^{0} & \not{0}^{1} & \not{0}^{1} & \not{0}^{1} & \not{0}^{1} & \not{0}^{1} & \not{0}^{1} & \not{0}^{1} & {}^{1}0 \\ - & & & & 1 & 0 & 1 & 0 & 0 & 1 \\ \hline & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \end{array}$$

Portanto, a representação em complemento de dois com 8 bits do número -101001_2 é $11010111_{\bar{2}}$

Exemplo A.18 (Converter -1110100_2 para complemento de dois com 8 bits)

O valor absoluto do número a ser convertido é 1110100_2.
Este valor será subtraído de 2^8 (i.e., 100000000_2).

$$\begin{array}{cccccccccc} & \not{1}^{0} & \not{0}^{1} & \not{0}^{1} & \not{0}^{1} & \not{0}^{1} & \not{0}^{1} & {}^{1}0 & 0 & 0 \\ - & & & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ \hline & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{array}$$

Portanto, a representação em complemento de dois com 8 bits do número -101001_2 é $11010111_{\bar{2}}$

Conversão para Complemento de Dois via Complemento de Um

Uma forma alternativa de realizar a conversão de um número em binário para complemento de dois é utilizar uma conversão intermediária para *complemento de um*. O processo para obtenção do complemento de dois com n bits de um número binário x via complemento de um é descrito abaixo:

1. Toma-se o valor absoluto de x denotado por $|x|$.
2. Caso $|x|$ possua menos do que n bits, $|x|$ deve ser reescrito como x', acrescentando-se zeros à esquerda até que x' fique com n bits.
3. Calcula-se o complemento de um de x', que será denotado por $\bar{x'}$.
   a) O complemento de um de x' é calculado invertendo-se (negando-se) cada um dos bits de x'.
4. Soma-se 1 ao valor de $\bar{x'}$, despresando-se qualquer possível bit de *overflow* resultante da soma. Os n bits obtidos representam o complemento de dois de x.

Exemplo A.19 (Converter -101001_2 para complemento de dois com 8 bits)

1. O valor absoluto do número é 101001;
2. Ajustando para 8 bits temos: 00101001;
3. Tomando o complemento de um: 11010110;
4. Somando 1 ao complemento de um:

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

Portanto, a representação em complemento de dois com 8 bits do número -101001_2 é $11010111_{\bar{2}}$

Exemplo A.20 (Converter -1110100_2 para complemento de dois com 8 bits)

1. O valor absoluto do número é 1110100;
2. Ajustando para 8 bits temos: 01110100;
3. Tomando o complemento de um: 10001011;
4. Somando 1 ao complemento de um:

$$\begin{array}{cccccccccc} & & 1 & 0 & 0 & 0 & 1 & \overset{1}{0} & \overset{1}{1} & 1 \\ + & & & & & & & & & 1 \\ \hline & & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{array}$$

Portanto, a representação em complemento de dois com 8 bits do número -1110100_2 é $10001100_{\bar{2}}$

A.6.2 Operações Aritméticas em Complemento de Dois

Adição

A adição de dois números em complemento de dois não requer qualquer pocessamento especial, mesmo que os operando tenham sinais opostos: o sinal do resultado é determinado autommaticamente como resultado da própria operação de adição.

Exemplo A.21 (Adicionando 15 e -5)

Considerando a notação de complemento de dois com 8 bits, temos:
$15_{10} = 00001111_{\bar{2}}$
$-5_{10} = 11111011_{\bar{2}}$
Somando os dois termos:

$$\begin{array}{cccccccccc} & \not{1} & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \\ & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & \\ + & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & \\ \hline & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & \end{array}$$

Temos, como esperado, que: $00001010_{\bar{2}} = 10_{10}$.

Este processo impõe a restrição de que o número seja limitado a 8 bits. Devido a essa restrição o vai-um para o 9º bit do Exemplo A.21 foi ignorado, o que resultou na soma correta. Caso a restrição não existisse o 9º bit teria sido considerado e o resultado seria um valor negativo (incorreto).

Os dois últimos bits da linha de vai-um (da direita para a esquerda) contém informações de grande importância: eles determinam se ocorreu ou não *overflow*, i.e., se o resultado foi um número grande demais para ser armazenado na quantidade de bits disponível. Se os dois últimos bits da linha de vai-um foram diferentes entre si, ocorreu *overflow*.

Em outras palavras, se os dois últimos bits da linha de vai-um forem 11 ou 00[8], o resultado é válido; porém, se os dois últimos bits forem 01 ou 10, então ocorreu *overflow*. Considere o exemplo abaixo:

Exemplo A.22 (Adicionando 7 e 3 em complemento de dois com 4 bits)

Considerando a notação de complemento de dois com 4 bits, temos:

[8]Dizer que os bits de vai-um são 00 é o mesmo que dizer que não houve "vai-um" nestes bits.

$7_{10} = 0111_{\bar{2}}$
$3_{10} = 0011_{\bar{2}}$
Somando os dois termos:

$$\begin{array}{cccccc} \not{0} & 1 & 1 & 1 & \\ & 0 & 1 & 1 & 1 \\ + & 0 & 0 & 1 & 1 \\ \hline & 1 & 0 & 1 & 0 \end{array}$$

Temos: $1010_{\bar{2}} = -6_{10}$, que é um resultado inválido.

No caso ilustrado no Exemplo A.22, os dois bits de vai-um mais a esquerda são 01, o que significa que houve um *overflow* na adição, ou seja, o resultado da adição $7 + 3$ está fora da faixa permitida para número em complemento de doi de quatro bits, que é de -8 a 7.

Subtração

Utilizando-se a notação de complemento de dois é possível simplificar o cálculo da operação de subtração transformando-a em uma adição. O procedimento de baseia na identidade mostrada abaixo:

$$x - y \equiv x + (-y)$$

Assim, ao invés de efetuarmos a subtração $x - y$, efetuamos a soma dos números x e $-y$, ambos expressos em complemento de dois.

Exemplo A.23 (Subtração $15 - 5$ em complemento de dois com 8 bits)

A subtração $15 - 5$ é transformada na adição $15 + (-5)$;
$15_{10} = 00001111_{\bar{2}}$
$-5_{10} = 11111011_{\bar{2}}$
Efetuando a soma:

$$\begin{array}{cccccccccc} \not{1} & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \\ & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ + & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ \hline & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{array}$$

Temos então o resultado: $00001010_{\bar{2}} = 10_{10}$.

Exemplo A.24 (Subtração $15 - (-5)$ em complemento de dois com 8 bits)

A subtração $15 - 5$ é transformada na adição $15 + (+5)$;
$15_{10} = 00001111_{\bar{2}}$
$5_{10} = 00000101_{\bar{2}}$
Efetuando a soma:

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

Temos então o resultado: $00010100_{\bar{2}} = 20_{10}$.

Exemplo A.25 (Subtração $15 - 35$ em complemento de dois com 8 bits)

A subtração $15 - 5$ é transformada na adição $15 + (-35)$;

$15_{10} = 00001111_{\bar{2}}$
$-35_{10} = 11011101_{\bar{2}}$
Efetuando a soma:

$$\begin{array}{cccccccccc} & & & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & & \\ & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & \\ + & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & \\ \hline & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & \end{array}$$

Temos então o resultado: $11101100_{\bar{2}} = -20_{10}$.

A.6.3 Exercícios

Exercício A.25 Converta os números abaixo de decimal para complemento de dois com 5 bits.

(a) 12
(b) 10
(c) -7
(d) -10
(e) 3
(f) -9
(g) 15
(h) -4
(i) -15
(j) -3
(k) 11
(l) -1

Exercício A.26 Complete a tabela abaixo com o número em complemento de dois com 4 bits equivalentes aos número decimais.

Decimal	Complemento de 2
7	
6	
5	
4	
3	
2	
1	
0	

Decimal	Complemento de 2
-1	
-2	
-3	
-4	
-5	
-6	
-7	
-8	

Exercício A.27 Converta os número em binário abaixo para complemento de dois em 12 bits.

(a) -1011101_2
(b) -11010111_2
(c) -1011101111_2
(d) -10101001100_2
(e) -1011110111_2
(f) -11011011_2

Exercício A.28 Converta os número em complemento de dois de 10 bit abaixo para binário.

(a) $1101110111_{\bar{2}}$
(b) $0101000001_{\bar{2}}$
(c) $0101101001_{\bar{2}}$
(d) $1101001110_{\bar{2}}$
(e) $1001101001_{\bar{2}}$
(f) $0111101000_{\bar{2}}$
(g) $1101111101_{\bar{2}}$
(h) $1010000000_{\bar{2}}$
(i) $0010101101_{\bar{2}}$
(j) $0110001110_{\bar{2}}$
(k) $1011001010_{\bar{2}}$
(l) $0010100000_{\bar{2}}$

Exercício A.29 Converta os número em complemento de dois de 10 bit abaixo para decimal.

(a) $0010100101_{\bar{2}}$
(b) $0010101100_{\bar{2}}$
(c) $1110110101_{\bar{2}}$
(d) $0000001011_{\bar{2}}$
(e) $1000010011_{\bar{2}}$
(f) $1100111010_{\bar{2}}$
(g) $1110101010_{\bar{2}}$
(h) $1000110111_{\bar{2}}$
(i) $1000110011_{\bar{2}}$

(j) $1011111000_{\bar{2}}$ (k) $0111101110_{\bar{2}}$ (l) $0101010011_{\bar{2}}$

Exercício A.30 Efetue as operações aritméticas indicadas abaixo em complemento de dois de 8 bits.

(a) $101111_2 + 1000_2$
(b) $1101001_2 - 11100_2$
(c) $001_2 - 101111_2$
(d) $110_2 + 100_2$
(e) $100_2 - 11_2$
(f) $10000_2 + 11101_2$
(g) $111111_2 + 1111111_2$
(h) $1001_2 - 10000_2$
(i) $110101_2 - 101000_2$
(j) $101101_2 + 101101_2$

Exercício A.31 Efetue as operações aritméticas indicadas abaixo em complemento de dois de 8 bits.

(a) $-103_{10} - 118_{10}$
(b) $-98_{10} + 25_{10}$
(c) $36_{10} - 124_{10}$
(d) $104_{10} - 82_{10}$
(e) $-26_{10} - (-63_{10})$
(f) $-2_{10} + 78_{10}$
(g) $83_{10} - 29_{10}$
(h) $41_{10} + 110_{10}$
(i) $118_{10} - 113_{10}$
(j) $-42_{10} - 12_{10}$

A.7 Exercícios de Revisão

Apêndice B

Indução Matemática

Sumário

Indução matemática é o raciocínio segundo o qual se estende uma dada propriedade a todos os elementos de um conjunto, permitindo, deste modo, demonstrar a verdade de um número possivelmente infinito de proposições. É o método por excelência do raciocínio lógico-matemático.

A idéia central desse método consiste em (i) provar que o enunciado é verdadeiro para um valor inicial, e então (ii) provar que o processo usado para ir de um valor para o próximo é valido. Se estas duas coisas são provadas, então a prova para qualquer elemento do conjunto pode ser obtida através da repetição desse processo.

Para entender por que os dois passos descritos acima são suficientes, é útil pensar no efeito dominó. Se você tem uma longa fila de dominós em pé e você puder assegurar que:

1. O primeiro dominó cairá.

2. Sempre que um dominó cair, seu próximo vizinho também cairá.

Então, podemos concluir que todos os dominós cairão.

Indução matemática não deve ser interpretada como uma forma de *raciocínio indutivo*, que é considerado não-rigoroso em matemática. A indução matemática é uma forma rigorosa de raciocínio dedutivo. O método de indução pode ser estendido para provar afirmações sobre *estruturas bem-fundadas* mais gerais, tais como listas e árvores, por exemplo. Essa generalização, conhecida como *indução estrutural*, é amplamente usada na lógica matemática e em ciência da computação. Indução matemática, nesse sentido estendido, está intimamente relacionada com *recursividade*, um dos conceitos básicos de teoria da computação.

Recursão, também conhecida como *estratégia de dividir para conquistar*, é um método que quebra um problema grande (difícil) em parte menores e geralmente mais simples de resolver. Se você puder demonstrar que qualquer problema grande pode ser subdividido em outros menores, e que os menores problemas possíveis (i.e., aqueles que não podem mais ser subdivididos) podem ser resolvidos, você tem um método para resolver problemas de qualquer tamanho. Evidentemente, nós podemos provar isso usando indução.

Vejamos um exemplo simples. Suponha que lhe foram dadas as coordenadas dos vértices de um polígono simples, i.e., um polígono cujos vértices são distintos e cujos lados não se intercruzam, e você quer dividir o polígono em triângulos. Se você puder escrever um programa que quebra qualquer polígono grande (i.e., com quatro lados ou mais) em dois polígonos menores, então você sabe que pode triangular o polígono todo. Divide-se o polígono original (grande) em dois menores, e então repetidamente aplica-se o processo para os polígonos menores obtidos no passo anterior. Uma aplicação deste problema é o algoritmo para "renderização" de objetos em computação gráfica.

B.1 Primeiro Princípio de Indução — Indução Fraca

A forma mais simples e mais comum de indução matemática prova que um enunciado vale para todos os números naturais n e consiste de dois passos:

1. O *caso base*: mostrar que o enunciado vale para $n = 1$.
2. O *passo indutivo*: mostrar que, **se** o enunciado vale para $n = k1$, **então** o mesmo enunciado vale para $n = k + 1$.

A suposição, na etapa indutiva, de que o enunciado vale para algum k é chamado a *hipótese de indução.* Para executar o passo indutivo, deve-se assumir a hipótese de indução e, em seguida, usa essa suposição para provar a afirmação para $k + 1$.

Definição B.1 (Princípio de Indução Matemática Fraca)

Seja P uma propriedade arbitrária para os números naturais. Escrevemos $P(n)$ para representar que a propriedade P é verdadeira para o número n.
O *princípio de indução fraca*[a] diz que se pudermos provar que:

1. $P(0)$ é verdadeiro; e
2. Supondo-se $P(k)$, podemos mostrar que $P(k + 1)$ é verdadeiro.

Então podemos afirmar que $P(n)$ é verdadeiro para todo $n \in \mathbb{N}$.

[a] Também chamado "Primeiro Princípio de Indução".

Exemplo B.1 (Soma de pares)

Podemos usar o princípio de indução fraca para provar que a sentença abaixo é verdadeira para todo número natural n.

$$P(n) : \ 0 + 2 + 4 + \cdots + 2n = n^2 + n \tag{B.1}$$

A prova de que a senteça é verdadeira para todo $n \in \mathbb{N}$ é dada abaixo e segue a forma geral de demonstrações por indução, i.e., primeiro provaremos o caso base e, em seguida, o caso indutivo.

1. *Caso base:* $n = 0$. Substituindo n por 0 em $P(n)$ temos:

$$\begin{aligned} P(0) : \ 0 &= 0^2 + 0 \\ \therefore \ 0 &= 0 \end{aligned}$$

2. *Caso indutivo:* Supomos que P é verdadeiro para algum valor $k \geq 0$, i.e., maior ou igual ao caso base, e a partir desta suposição (i.e., da *hipótese indutiva*) devemos demonstrar que P também é verdadeiro para $k+1$. Ou seja, queremos mostrar que **se** $P(k)$ for verdadeiro, **então** $P(k + 1)$ também será. Logo, supomos

$$P(k) : \ 0 + 2 + 4 + \cdots + 2k = k^2 + k \tag{B.2}$$

e devemos provar que

$$P(k + 1) : \ 0 + 2 + 4 + \cdots + 2k + 2(k + 1) = (k + 1)^2 + (k + 1) \tag{B.3}$$

também é verdadeiro.

$$\begin{aligned} \underbrace{0+2+4+\cdots+2k}_{\text{lado esq. de (B.2)}}+2(k+1) &= (k+1)^2+(k+1) \\ (k^2+k)+2(k+1) &= (k^2+2k+1)+(k+1) \qquad \triangleright \text{ substituindo (B.2)} \\ k^2+3k+2 &= k^2+3k+2 \end{aligned}$$

□

Exemplo B.2 (Soma dos n primeiros números naturais)

Indução matemática pode ser usada para provar que a conjectura abaixo vale para todos os números naturais n.

$$0+1+2+\cdots+n=\frac{n(n+1)}{2} \tag{B.4}$$

Essa conjectura estabelece uma fórmula para a soma dos números naturais menores ou iguais ao número n. A prova de que a conjectura é verdadeira para todos $n \in \mathbb{N}$ é dada abaixo.

Chamemos a conjectura de $P(n)$, ou seja, diremos que $P(n)$ é verdadeiro se a igualdade na Equação (B.4) for verdadeira.

1. *Caso base*: Mostrar que o enunciado vale para $n=0$.

   $P(0)$ representa a declaração:

   $$0=\frac{0\cdot(0+1)}{2}$$

   No lado esquerdo da equação, o único termo é 0.

   No lado direito da equação temos $\frac{0\cdot(0+1)}{2}$ que também reduz a 0.

   Os dois lados são iguais, então a afirmação é verdadeira para $n=0$, o que prova o caso base.

2. *Passo indutivo*: Mostrar que *se* $P(k)$ é verdadeiro, então, $P(k+1)$ também será verdadeiro, para qualquer k. Isso pode ser feito da seguinte forma.

   Supomos que $P(k)$ é verdadeiro (por algum valor não especificado de k, ou seja, para qualquer k). Em seguida, devemos mostrar que $P(k+1)$ é verdadeiro, ou seja:

   Supomos que

   $$0+1+2+\cdots+k=\frac{n(n+1)}{2} \tag{B.5}$$

   é verdadeiro e, em seguida, tentamos mostrar que

   $$0+1+2+\cdots+k+(n+1)=\frac{(n+1)((n+1)+1)}{2} \tag{B.6}$$

   também é verdadeiro.

   Usando a hipótese de indução de que $P(k)$ é verdadeiro, podemos reescrever a Equação (B.6) da seguinte forma:

   $$\frac{n(n+1)}{2}+(n+1)=\frac{(n+1)((n+1)+1)}{2} \tag{B.7}$$

   Aplicando sucessivas simplificações algébricas à Equação (B.7), temos:

$$\frac{n(n+1)+2(n+1)}{2} = \frac{(n+1)(n+2)}{2}$$
$$n(n+1)+2(n+1) = (n+1)(n+2)$$
$$n^2+n+2n+2 = n^2+2n+n+2$$
$$n^2+3n+2 = n^2+3n+2$$

Logo, podemos verificar que os dois lados da Equação (B.7) são iguais, demonstrando assim que $P(k+1)$ é verdadeiro.

Uma vez que tanto o caso base quanto o passo indutivo foram provados, podemos afirmar que foi provado por indução matemática que $P(n)$ vale para todos os números naturais n.

□[a]

[a] O símbolo □, muitas vezes escrito como "Q.E.D.", significa *"Quod erat demonstrandum"* – é uma expressão em latim que significa *"como se queria demonstrar"*.

A parte em que se faz a prova do passo indutivo pode variar bastante, dependendo da natureza exata da propriedade a ser provada, mas a estrutura básica da prova por indução é sempre a mesma.

B.1.1 Iniciando em $n_0 \neq 0$

A escolha entre $n = 0$, $n = 1$, ou outro valor, no caso base é específico para cada contexto. Se 0 é considerado um número natural, como é comum nas áreas de análise combinatória e lógica matemática, então $n = 0$. Se, por outro lado, 1 é tomado como o primeiro número natural, então o caso base é dada por $n = 1$. Também deve ser considerada a prova em questão. Pode ser que a conjectura que se deseja provar somente seja válida para valores de $n \geq n_0$, neste caso o caso base passa a ser $n = n_0$.

Se nós queremos provar uma dada conjectura, não para todos os números naturais, mas apenas para todos os números maiores ou iguais a um determinado número n_0, então:

1. Mostrando que o enunciado vale quando $n = n_0$.
2. Mostrando que, *se* o enunciado vale para $n = k$, onde $k \geq b$, então o mesmo enunciado vale para $n = k + 1$.

Poderemos afirmar que $P(n)$ é verdadeiro para todo $n \in \mathbb{N}, n \geq b$.

Exemplo B.3 ($n^2 > 2n, n \geq 3$)

Podemos usar esta variação da indução fraca para para mostrar que $n^2 > 2n$ para todo $n \geq 3$.
Definiremos a propriedade a ser demonstrada como $P(n) : n^2 > 2n$.

1. *Caso base*: Mostrar que o enunciado vale para $n = n_0$, i.e., mostrar que $P(n_0)$ é verdadeiro.
   Como $n_0 = 3$, $P(3)$ eleva a:
   $$\begin{aligned} 3^2 &> 2 \cdot 3 \\ \therefore\ 9 &> 6 \end{aligned}$$
   Como a inequação é verdadeira, podemos afirmar que $P(3)$ é verdadeiro, prova o caso base.
2. *Passo indutivo*: Mostrar que *se* $P(k)$ é verdadeiro para algum valor de $k \geq n_0$, então, $P(k+1)$ também será verdadeiro.
   Supomos que $P(k)$ é verdadeiro. Em seguida, devemos mostrar que $P(k+1)$ é verdadeiro, ou seja:

Supomos que

$$P(k) : k^2 > 2k \tag{B.8}$$

é verdadeiro e, em seguida, tentamos mostrar que

$$P(k+1) : (k+1)^2 > 2(k+1) \tag{B.9}$$

também é.

Desenvolvendo a Equação (B.9), temos:

$$\begin{array}{lcll} (k+1)^2 & = & k^2+2k+1 & \text{(expandindo o termo à esquerda)} \\ (k+1)^2 & > & 2k+2k+1 & \text{(pela hipótese de indução, i.e., } k^2>2k\text{)} \\ (k+1)^2 & > & 2k+6+1 & \text{(já que } k \geq 3\text{)} \\ (k+1)^2 & > & 2k+2 & \text{(pois se } 2k+7>2k+2\text{)} \\ (k+1)^2 & > & 2(k+1) & \end{array}$$

Uma vez que tanto o caso base quanto o passo indutivo foram provados, podemos afirmar que $P(n)$ vale para todo $n \geq 3$.

□

O próximo exemplo apresenta uma demonstração que se destaca por dois aspectios. Primeiramente, ela é uma demonstração menos numérica do que as apresentadas anteriormente. Em segundo lugar porque ela é uma demonstração *por casos* ou seja, uma demonstração que analisa quais situações diferentes podem ocorrer e demonstra, para cada uma das possíveis situações diferentes, que a propriedade desejada é verdadeira.

Exemplo B.4 (Caixa Eletrônico)

Suponha que um caixa eletrônico tem apenas notas de dois e cinco Reais. Você pode digitar o valor que quer, e o caixa eletrônico vai descobrir como as dividir o valor em nos números adequados de notas de dois e cinco Reais. *Conjectura*: o caixa eletrônico pode gerar qualquer valor $n \geq 4$ reais.

Representaremos a propriedade "o caixa eletrônico pode gerar o valor n" por $G(n)$. Portanto, queremos demonstrar que $G(n)$ é verdadeiro para todo valor $n \geq 4$.

1. *Caso Base*: $n = 4$. Podem ser geradas com 2 notas de R$ 2,00. Portanto, $G(4)$ é verdadeiro.
2. *Passo Indutivo*: Como *hipótese indutiva* supomos que $G(k)$ é verdadeiro para $k \geq 4$. Devemos demonstrar que $G(k+1)$ segue da hipótese indutiva. Efetuaremos a demonstração supondo dois casos distintos:

   a) A saída para k contém ao menos uma nota de R$ 5,00.
   Neste caso, para gerar $k+1$ basta substituir a nota de R$ 5,00 por 3 notas de R$ 2,00 (i.e., R$ 6,00).

   b) A saída não contém nenhuma nota de R$ 5,00.
   Neste caso, já que $k \geq 4$, devem haver ao menos 2 notas de R$ 2,00. Para gerar $k+1$ basta substituir 2 notas de R$ 2,00 (i.e., R$ 4,00) por uma nota de R$ 5,00.

   Como $G(k+1)$ é verdadeira nos dois casos possíveis, podemos afirmar que $G(k+1)$ é verdadeira sempre que $G(k)$ for verdadeira.

Uma vez que tanto o caso base quanto o passo indutivo foram provados, podemos afirmar que $G(n)$ vale para todo $n \geq 4$.

□

B.2 Segundo Princípio de Indução — Indução Forte

O Princípio da Indução Matemática afirma que se $P(n_0)$ é verdadeiro e, para $k \geq n_0$, $P(k)$ é verdadeiro, então $P(k+1)$ é verdadeira. No entanto, às vezes é preciso "voltar" mais do que um passo para obter $P(k+1)$. Por exemplo, pode ser

que o valor de $P(k+1)$ seja promado pela combinação dos valores de $P(k)$ e $P(k-1)$. Neste tipo de situação é onde a forma forte da indução matemática é útil.

Indução forte, ou *indução completa*, ou *segundo princípio de indução*, diz que na segunda etapa pode-se supor que não só o enunciado vale para $n = k$, mas também que ela é verdadeira para todo n menor ou igual a k.

Definição B.2 (Princípio da Indução Matemática Forte)

Sejam a e b números naturais tais que $a \leq b$, e seja $P(n)$ um predicado (propriedade) definido sobre todos os números naturais $n \geq a$.

Suponha que as seguintes afirmações são verdadeiras:

1. $P(a), P(a+1), \ldots, P(b)$ são todas afirmações verdadeiras. (*Caso Base*)
2. Para qualquer número natural $k > b$, **se** $P(i)$ é verdadeira para todos os números naturais i, tais que $a \leq i < k$, então $P(k)$ é verdadeira. (*Passo Indutivo*)

Então a afirmação $P(n)$ é verdadeira para todos os números naturais $n \geq a$.

Exemplo B.5 (Postagem com selos de 4 e 5 centavos)

Conjectura: qualquer valor de postagem acima de 12 centavos pode ser gerado apenas com selos de 4 e 5 centavos.

Representaremos a propriedade "é possível gerar, apenas com selos de 4 e 5 centavos, o valor de postagem de n centavos" por $G(n)$. Desejamos provar que $G(n)$ é verdadeiro para todo $n \geq 12$.

1. *Caso Base*: Consideraremos como casos bases $n = 12$, $n = 13$, $n = 14$ e $n = 15$.
   a) $n = 12$, gerado por 3 selos de 4 centavos.
   b) $n = 13$, gerado por 2 selos de 4 centavos e 1 selo de 5 centavos.
   c) $n = 14$, gerado por 1 selo de 4 centavos e 2 selos de 5 centavos.
   d) $n = 15$, gerado por 3 selos de 5 centavos.

   Portanto, $G(n)$ é verdadeiro para todos os casos base.
2. *Passo Indutivo*: Seja $k \geq 15$.

   Assumiremos que $G(r)$ é verdadeiro para todo r, tal que $12 \leq r \leq k$. (*Hipótese Indutiva*)

   Para formar o valor de postagem $k+1$, ou seja, para demonstrar que $G(k+1)$ é verdadeiro, gera-se o valor de postagem para $k-3$ (que pode se gerado, de acordo com a hipótese de indução) e acrescenta-se um selo de 4 centavos.

□

Exemplo B.6 (Enésimo Número de Fibonacci)

Os *Números de Fibonacci*, ou Sequência de Fibonacci, são uma sequência definida recursivamente pela fórmula abaixo:

$$F(n) = \begin{cases} 0 & \text{se } n = 0 \\ 1 & \text{se } n = 1 \\ F(n-1) + F(n-2) & \text{se } n > 1 \end{cases} \tag{B.10}$$

E tem aplicações em biologia, arquitetura, análise combinatória, música e outras áreas. Os primeiros 11 números

de Fibonacci, i.e., $n = 0$ até $n = 10$, são: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55.

A *proporção áurea* é uma constante irracional que é frequentemente encontrada em campos tão diversos quanto artes e biologia. A proporção áurea é denotada pela letra grega φ (*phi*), com valor aproximado de $1,618$:

$$\varphi = \frac{1+\sqrt{5}}{2} \tag{B.11}$$

Interessantemente, existe uma relação entre os Números de Fibonacci e a Proporção Áurea.

Conjectura: Para todo número natural n:

$$F(n) = \frac{\varphi^n - \left(\frac{-1}{\varphi}\right)^n}{\sqrt{5}} \tag{B.12}$$

Chamando (B.12) de $A(n)$, queremos demonstrar que $A(n)$ é verdadeiro para todo $n \in \mathbb{N}$.

1. *Caso Base.* $n = 0$ e $n = 1$.

   a) $F(0) = \dfrac{\varphi^0 - \left(\frac{-1}{\varphi}\right)^0}{\sqrt{5}} = \dfrac{1-1}{\sqrt{5}} = 0$

   b) $F(1) = \dfrac{\varphi^1 - \left(\frac{-1}{\varphi}\right)^1}{\sqrt{5}} = \dfrac{\varphi - \frac{-1}{\varphi}}{\sqrt{5}} = \dfrac{\varphi^2+1}{\varphi\sqrt{5}}$

   Substituindo (B.11), temos:

$$F(1) = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^2 + 1}{\left(\frac{1+\sqrt{5}}{2}\right)\sqrt{5}} = \frac{\frac{1+2\sqrt{5}+\sqrt{5}^2}{2^2} + 1}{\frac{\sqrt{5}+\sqrt{5}\sqrt{5}}{2}} = \frac{\frac{1+2\sqrt{5}+5}{4} + 1}{\frac{\sqrt{5}+5}{2}}$$

$$F(1) = \frac{\frac{1+2\sqrt{5}+5+4}{4}}{\frac{\sqrt{5}+5}{2}} = \frac{\frac{2\sqrt{5}+10}{4}}{\frac{\sqrt{5}+5}{2}} = \frac{\frac{\sqrt{5}+5}{2}}{\frac{\sqrt{5}+5}{2}} = 1$$

   Portanto, $A(n)$ é verdadeiro para os dois casos base.

2. *Passo Indutivo.* Seja $k > 1$.

   Assumiremos que $A(r)$ é verdadeiro para todo r, tal que $0 \leq r \leq k$. (*Hipótese Indutiva*)

   Devemos provar que $A(k+1)$ é verdadeiro.

$$F(k+1) = \frac{\varphi^{(k+1)} - \left(\frac{-1}{\varphi}\right)^{(k+1)}}{\sqrt{5}} \tag{B.13}$$

   Mas de (B.10) temos que:

$$F(k+1) = F(k) + F(k-1) \tag{B.14}$$

Pela hipótese indutiva, (B.14) pode ser desenvolvida para:

$$\begin{aligned}
F(k+1) &= \frac{\varphi^k - \left(\frac{-1}{\varphi}\right)^k}{\sqrt{5}} + \frac{\varphi^{(k-1)} - \left(\frac{-1}{\varphi}\right)^{(k-1)}}{\sqrt{5}} \\
F(k+1) &= \frac{\varphi^k - \left(\frac{-1}{\varphi}\right)^k + \varphi^{(k-1)} - \left(\frac{-1}{\varphi}\right)^{(k-1)}}{\sqrt{5}} \\
F(k+1) &= \frac{\left(\varphi^k + \varphi^{(k-1)}\right) - \left[\left(\frac{-1}{\varphi}\right)^k + \left(\frac{-1}{\varphi}\right)^{(k-1)}\right]}{\sqrt{5}} \\
F(k+1) &= \frac{\varphi^{(k-1)}(\varphi+1) - \left(\frac{-1}{\varphi}\right)^{(k-1)}\left(\frac{-1}{\varphi}+1\right)}{\sqrt{5}}
\end{aligned} \tag{B.15}$$

Para tornar o lado equerdo de (B.15) igual ao lado esquerdo de (B.13) devemos demonstrar que $\varphi + 1 = \varphi^2$ e que $\frac{-1}{\varphi} + 1 = \left(\frac{-1}{\varphi}\right)^2$.

$$\begin{aligned}
\varphi^2 &= \left(\frac{1+\sqrt{5}}{2}\right)^2 = \frac{\left(1+\sqrt{5}\right)^2}{2^2} = \frac{1+2\sqrt{5}+\sqrt{5}^2}{2^2} \\
\varphi^2 &= \frac{6+2\sqrt{5}}{4} = \frac{3+\sqrt{5}}{2} \\
\varphi^2 &= \frac{1+\sqrt{5}}{2} + 1 = \varphi + 1
\end{aligned} \tag{B.16}$$

$$\begin{aligned}
\left(\frac{-1}{\varphi}\right)^2 &= \frac{1}{\varphi^2}\left(\times\frac{\varphi-1}{\varphi-1}\right) = \frac{\varphi-1}{\varphi^2(\varphi-1)} \\
\left(\frac{-1}{\varphi}\right)^2 &= \frac{\varphi-1}{(\varphi+1)(\varphi-1)} = \frac{\varphi-1}{\varphi^2-1} \\
\left(\frac{-1}{\varphi}\right)^2 &= \frac{\varphi-1}{(\varphi+1)-1} = \frac{\varphi-1}{\varphi} = \frac{-1}{\varphi} + 1
\end{aligned} \tag{B.17}$$

Substituindo (B.16) e (B.17) na equação (B.15), temos:

$$\begin{aligned}
F(k+1) &= \frac{\varphi^{(k-1)}\varphi^2 - \left(\frac{-1}{\varphi}\right)^{(k-1)}\left(\frac{-1}{\varphi}\right)^2}{\sqrt{5}} \\
F(k+1) &= \frac{\varphi^{(k+1)} - \left(\frac{-1}{\varphi}\right)^{(k+1)}}{\sqrt{5}}
\end{aligned} \tag{B.18}$$

O que prova o passo indutivo.

Tanto os casos base quanto o passo indutivo foram provados, portanto podemos afirmar que $A(n)$ vale para todo $n \in \mathbb{N}$.

□

B.2.1 Indução Transfinita

Em matemática, e em especial na teoria dos conjuntos, a indução transfinita é uma técnica matemática rigorosa que permite provar propriedades para todos números ordinais[1] – ou, de forma mais geral, para qualquer conjunto bem ordenado – a partir de etapas finitas. É uma generalização da indução finita.

Seja $P(\alpha)$ uma propriedade definida para todos os ordinais α. Suponha que sempre que $P(\beta)$ é verdadeiro para todos $\beta < \alpha$, então $P(\alpha)$ também é verdadeiro. Então, a indução transfinita nos diz que P é verdadeiro para todos os ordinais.

Ou seja, se $P(\alpha)$ é verdadeira sempre que $P(\beta)$ é verdadeiro para todo $\beta < \alpha$, então $P(\alpha)$ é verdadeira para todos os α. Ou, em termos mais práticos: para provar a propriedade P para todos os ordinais α, pode-se supor que P já é conhecido para todos os $\beta < \alpha$.

Normalmente a prova é dividida em três casos:

1. *Caso Zero*: Prove que $P(0)$ é verdadeira.
2. *Caso Sucessor*: Prove que, para qualquer sucessor ordinal $\alpha + 1$, $P(\alpha + 1)$ decorre de $P(\alpha)$ e, se necessário, de $P(\beta)$ para todo $\beta < \alpha$.
3. *Caso Limite*: Prove que, para qualquer ordinal limite γ, $P(\gamma)$ segue a partir de $P(\beta)$ para todo $\beta < \gamma$.

Observe que o segundo e terceiro casos são idênticas, exceto para o tipo de ordinal considerado. Eles não precisam ser formalmente provados em separado, mas, na prática, as provas são geralmente tão diferentes que exigem apresentações separadas.

Observe também que a rigor, não é necessário provar o *caso zero* (i.e., a base na indução transfinita), porque este é um caso especial *vazio* da proposição de que se $P(\beta)$ é verdadeiro para todo $\beta < \alpha$, então P é verdadeiro para α. Isto é *trivialmente verdadeiro*, porque não há valores para $\beta < \alpha$ que poderiam servir como contra-exemplos. Entretanto, na prática, é comum apresentar a prova para o *caso zero* como uma forma de enfatizar a corretude da prova.

B.3 Exercícios

Exercício B.1 (Aquecimento) Mostre que as equações abaixo são verdadeiras para todo $n \in \mathbb{N}$.

1. $0^1 + 1^1 + 2^1 + 3^1 + \cdots + (n-1)^1 = \dfrac{n^2}{2} - \dfrac{n}{2}$
2. $0^2 + 1^2 + 2^2 + 3^2 + \cdots + (n-1)^2 = \dfrac{n^3}{3} - \dfrac{n^2}{2} + \dfrac{n}{6}$
3. $0^3 + 1^3 + 2^3 + 3^3 + \cdots + (n-1)^3 = \dfrac{n^4}{4} - \dfrac{n^3}{2} + \dfrac{n^2}{4}$
4. $0^4 + 1^4 + 2^4 + 3^4 + \cdots + (n-1)^4 = \dfrac{n^5}{5} - \dfrac{n^4}{2} + \dfrac{n^3}{3} - \dfrac{n}{30}$
5. $0^2 + 1^2 + 2^2 + \cdots + n^2 = \dfrac{n(n+1)(2n+1)}{6}$

Exercício B.2 Mostre que as equações abaixo são verdadeiras para todo $n \in \mathbb{N}, n \geq 1$.

1. $1 + 3 + \cdots + (2n-1) = n^2$
2. $\dfrac{1}{1 \cdot 2} + \dfrac{1}{2 \cdot 3} + \cdots + \dfrac{1}{n(n+1)} = \dfrac{n}{n+1}$
3. $1^3 + 2^3 + \cdots + n^3 = \left[\dfrac{n(n+1)}{2}\right]^2$

[1] Os números ordinais são números usados para assinalar uma posição numa sequência ordenada: primeiro, segundo, terceiro, quarto, quinto, sexto etc. Em matemática, os números ordinais são uma extensão dos números naturais criada para incluir sequências infinitas por Georg Cantor em 1897.

4. $1 - 2^2 + 3^2 - 4^2 + \cdots + (-1)^{n-1}n^2 = (-1)^{n-1} \cdot \dfrac{n(n+1)}{2}$

5. $1^2 + 3^2 + \cdots + (2n-1)^2 = \dfrac{1}{3}\left[n(2n-1)(2n+1)\right]$

Exercício B.3 Mostre que a soma de três números naturais consecutivos é sempre divisível por 3.
Sugestão: Considere a sentença aberta

$$P(n):\ n + (n+1) + (n+2) \text{ é divisível por 3,}$$

e mostre, por indução, que ela é verdadeira para todo $n \in \mathbb{N}$.

Exercício B.4 Mostre que a soma dos cubos de três números naturais consecutivos é sempre divisível por 9.

Exercício B.5 Dada a sentença aberta em $\mathbb{N}$:

$$P(n):\ 1 + 2 + \cdots + n = \frac{n(n+1)}{2} + 1,$$

mostre que:

1. Qualquer que seja $n \in \mathbb{N}$, **se** $P(n)$ é verdadeira, **então** P(n + 1) é verdadeira.
2. $P(n)$ não é verdadeira para nenhum valor de $n \in \mathbb{N}$.

Exercício B.6 Mostre, por indução, a validade das fórmulas abaixo:

1. $1 \cdot 2^0 + 2 \cdot 2^1 + 3 \cdot 2^2 + \cdots + n \cdot 2^{n-1} = 1 + (n-1)2^n$
2. $\left(1 + \dfrac{1}{1}\right)\left(1 + \dfrac{1}{2}\right)\left(1 + \dfrac{1}{3}\right)^2 \cdots \left(1 + \dfrac{1}{n-1}\right)^{n-1} = \dfrac{n^{n-1}}{(n-1)!}$
3. $1 \cdot 1! + 2 \cdot 2! + 3 \cdot 3! + \cdots + n \cdot n! = (n+1)!$

Exercício B.7 Mostre que a equação abaixo é verdadeira para todo $n \in \mathbb{N}, n \geq 1$.

$$1 + \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{3}} + \cdots + \frac{1}{\sqrt{n}} \leq 2\sqrt{n}$$

Exercício B.8 Mostre que a equação abaixo é verdadeira para todo $n \in \mathbb{N}, n \geq 1$.

$$2! \cdot 4! \cdot 6! \cdots (2n)! \geq ((n+1)!)^n$$

Onde $x!$ denota a fatorial de x.

Exercício B.9 Mostre que a equação abaixo é verdadeira para todo $n \in \mathbb{N}, n \geq 1$.

$$\underbrace{\sqrt{2 + \sqrt{2 + \sqrt{2 + \cdots \sqrt{2}}}}}_{\times n} = 2\cos\frac{\pi}{2^{n+1}}$$

Exercício B.10 Demonstre que qualquer valor postal maior ou igual a dois centavos pode ser obtido usando-se somente selos com valor de 2 e 3 centavos.

Exercício B.11 Demonstre, usando o princípio de indução forte, que todo número natural $n \geq 2$ é um número primo ou pode ser expresso como o produto de números primos.

Exercício B.12 Em qualquer grupo de n pessoas, $n \in \mathbb{Z}^+$, cada uma deve apertar a mão de todas as outras pessoas. Encontre uma fórmula que forneça o número de apertos de mão, e demonstre que a fórmula é verdadeira, usando indução.

Exercício B.13 Escreva os cinco primeiros valores para as sequências (definidas por recursão) abaixo.

1. $A(1) = 5$
   $A(n) = A(n-1) + 5$, para $n > 1$

2. $B(1) = 1$
   $B(n) = \frac{1}{B(n-1)}$, para $n > 1$

3. $C(1) = 1$
   $C(n) = C(n-1) + n^2$, para $n > 1$

4. $S(1) = 1$
   $S(n) = S(n-1) + \frac{1}{n}$, para $n > 1$

5. $T(1) = 1$
   $T(n) = n \cdot T(n-1)$, para $n > 1$

6. $P(1) = 1$
   $P(n) = n^2 \cdot P(n-1) + (n-1)$, para $n > 1$

7. $M(1) = 2$
   $M(2) = 2$
   $M(n) = 2M(n-1) + M(n-2)$, para $n > 2$

8. $D(1) = 3$
   $D(2) = 5$
   $D(n) = (n-1)D(n-1) + (n-2)D(n-2)$, para $n > 2$

9. $W(1) = 2$
   $W(2) = 3$
   $W(n) = W(n-1) \cdot W(n-2)$, para $n > 2$

10. $R(1) = 1$
    $R(2) = 2$
    $R(3) = 3$
    $R(n) = R(n-1) + 2R(n-2) + 3R(n-3)$, para $n > 3$

Exercício B.14 Suponha que a exponenciação é definida pela equação

$$x^j \cdot x = x^{j+1}$$

para qualquer $j \in \mathbb{Z}^+$. Use indução para provar que $x^n \cdot x^m = x^{n+m}$ para $m, n \in \mathbb{Z}^+$

Exercício B.15 Seja $F(n)$ o n-ésimo termo da série de Fibonacci, conforme definido no Exemplo B.6. Mostre que as equações abaixo são verdadeiras para todo $n \in \mathbb{N}, n \geq 1$.

1. $F(n-1) \cdot F(n+1) = F(n)^2 + (-1)^n$

2. $\sum_{i=0}^{n} F(i)^2 = F(n) \cdot F(n+1)$

3. $F(1) + F(2) + \cdots + F(n) = F(n+2) - 1$

4. $F(2) + F(4) + \cdots + F(2n) = F(2n+1) - 1$

5. $F(n+3) = 2F(n+1) + F(n)$

Exercício B.16 Uma cadeia de 0s e 1s deve ser processada e convertida para uma cadeia de paridade par acrescentando-se um bit de paridade no final da cadeia. O bit de paridade é um bit que faça com que o número total de bits 1 da cadeia final seja par. Por exemplo:

- Para a cadeia 0010101 o bit de paridade seira 1, e a cadeia final seria 00101011, com 4 bits 1.
- Para a cadeia 0010010 o bit de paridade seria 0, e a cadeia final seria 00100100, com 2 bits 1.

O bit de paridade é inicialmente 0. Quando um caracter 0 é processado, o bit de paridade permanece inalterado. Quando um caracter 1 é processado, o bit de paridade muda de 0 para 1 ou de 1 para 0.

Prove que o número de 1s numa cadeia final, incluindo o bit de paridade, é sempre par.
(*Sugestão:* considere várias possibilidades.)

Exercício B.17 Sejam o consjunto $T = \{a, b\}$, e a linguagem T_1 sobre T definida do seguinte modo:

i. $a \in T_1$

ii. Se $u, v \in T_1$, então $ubv \in T_1$.

Considerando as definições acima, responda os itens abaixo:

1. Escreva 5 cadeias que pertencem a T_1.
2. Se $w \in T_1$ é possível ter dois a's consecutivos em w? Demonstre a sua resposta usando indução.
3. Demonstre que se $w \in T_1$ o número de a's em w é igual ao número de b's mais 1.

Exercício B.18 Sejam o consjunto $T = \{a, b\}$, e a linguagem T_2 sobre T definida do seguinte modo:

i. $a \in T_2$

ii. Se $w \in T_2$, então $wbw \in T_2$.

Considerando as definições acima, responda os itens abaixo:

1. Escreva 4 cadeias que pertencem a T_2.
2. Dê 3 exemplos de cadeias que pertencem a T_1 (Exercício B.17 mas não pertencem a T_2.
3. Se $w \in T_2$ é possível ter dois b's consecutivos em w? Demonstre a sua resposta usando indução.
4. Demonstre que, se $w \in T_2$, o número de a's em w é igual ao número de b's mais 1.
5. Demonstre que, se $w \in T_2$, então o número de a's em w é igual a 2^n para algum número natural n.

Apêndice C

Definições Recursivas e Indução Estrutural

Sumário

Se é difícil definir um objeto explicitamente, pode ser fácil definir esse objeto em termos de si mesmo, i.e., o termo corrente poderia ser definido a partir dos termos anteriores). Este processo é chamado de recursão.

A recursão, também conhecida como *estratégia de dividir para conquistar*, é um método que quebra um problema grande (difícil) em parte menores e geralmente mais simples de resolver. Se você puder demonstrar que qualquer problema grande pode ser subdividido em outros menores, e que os menores problemas possíveis (i.e., aqueles que não podem mais ser subdivididos) podem ser resolvidos, você tem um método para resolver problemas de qualquer tamanho. Evidentemente, nós podemos provar isso usando indução. A recursão pode ser usada para definir seqüências, conjuntos, relações, funções, e vários outros tipos de estruturas.

Vejamos um exemplo simples. Suponha que lhe foram dadas as coordenadas dos vértices de um polígono simples, i.e., um polígono cujos vértices são distintos e cujos lados não se intercruzam, e você quer dividir o polígono em triângulos. Se você puder escrever um programa que quebra qualquer polígono grande (i.e., com quatro lados ou mais) em dois polígonos menores, então você sabe que pode triangular o polígono todo. Divide-se o polígono original (grande) em dois menores, e então repetidamente aplica-se o processo para os polígonos menores obtidos no passo anterior. Uma aplicação deste problema é o algoritmo para "renderização" de objetos em computação gráfica.

C.1 Funções Definidas Recursivamente

Seja f uma função com o domínio $\mathbb{N}$. Para definir a função f usamos duas etapas:

1. *Caso Base*: Especifique o valor da função a zero, $f(0) = a$. (Para algum valor a.)
2. *Caso Recursivo*: Defina uma regra para encontrar o valor da função $f(n)$ a partir de $f(0)$, $f(1)$, ..., $f(n-1)$.

Essa definição é chamada de *definição recursiva* ou *definição indutiva*.

Exemplo C.1

Se f é definida recursivamente por

$$\begin{aligned} f(0) &= 3, \\ f(n) &= 2f(n-1) + 3, \text{ para } n > 0 \end{aligned}$$

encontre $f(1)$, $f(2)$, $f(3)$ e $f(4)$.
Solução: Pela definição de f.

$$\begin{aligned} f(1) &= 2f(0)+3 = 2\cdot 3+3 = 6+3 = 9 \\ f(2) &= 2f(1)+3 = 2\cdot 9+3 = 18+3 = 21 \\ f(3) &= 2f(2)+3 = 2\cdot 21+3 = 42+3 = 45 \\ f(4) &= 2f(3)+3 = 2\cdot 45+3 = 90+3 = 93 \end{aligned}$$

Exemplo C.2

Dê uma definição recursiva para a^n, onde $a \in \mathbb{R}$, $a \neq 0$ e $n \in \mathbb{N}$.
Solução: Especificando as duas etapas de uma definição recursiva:

1. *Caso Base*: Especificando o valor da função a^n em zero: $a^0 = 1$.
2. *Caso Recursivo*: Definindo uma regra para calcular o valor da função em n, à partir dos valores para $0, 1, 2, \ldots, n-1$. A regra é $a^n = a \cdot a^{n-1}$.

Poderíamos ter escrito o mesmo de modo mais compacto usando a notação abaixo:

$$a^n = \begin{cases} 1, & \text{se } n = 0 \\ a \cdot a^{n-1}, & \text{se } n > 0 \end{cases}$$

Exemplo C.3

Dar uma definição recursiva de $\sum_{k=0}^{n} a^k$
Solução: Usando as duas etapas:

1. *Caso Base*: Especifique o valor da função em zero, $S(0) = a^0$.
2. *Caso Indutivo*: Defina a regra para encontrar o valor da função $S(n)$ a partir de $S(0)$, $S(1)$, ..., $S(n-1)$. A regra é $S(n) = S(n-1) + a^{n+1}$, para $n > 0$.

C.2 Alfabetos, Palavras e Linguagens

A noção de conjunto permite definir *linguagem*, um dos conceitos fundamentais em computação. Antes de apresentar a definição de linguagem, entretanto, é necessário introduzir os conceitos de *alfabeto* e de *cadeia de caracteres*. Nesta seção faremos apenas um estudo superficial do assunto. O estudo detalhado destes e outros conceitos correlatos é realizado em algumas disciplinas como Linguagens Formais, Compiladores e Teoria da Computação.

Definição C.1 (Alfabeto)

Um *alfabeto* é um conjunto finito. Os elementos de um alfabeto são usualmente denomidados *símbolos* ou *caracteres*.

Portanto, o conjunto vazio *é* um alfabeto, e um conjunto infinito *não é* um alfabeto.

Definição C.2 (Palavra, Cadeia de Caracteres, Sentença)

Uma *palavra*, ou *cadeia de caracteres*, ou *sentença*, sobre um alfabeto é uma sequência finita de símbolos do alfabeto justapostos, i.e., colocados um após o outro.

Como a definição apenas especifica "sequência finita", uma cadeia sem símbolos é uma palavra válida. O símbolo ε denota a *cadeia vazia*, ou *palavra vazia*, ou *sentença vazia*.

Seja Σ um alfabeto, então $\Sigma^\star$ denota o conjunto de todas as palavras possíveis sobre Σ.

Exemplo C.4 (Alfabeto, Palavra)

1. Os conjuntos $\emptyset$ e $\{a, b, c\}$ são alfabetos.
2. O conjunto $\mathbb{N}$ não é um alfabeto.
3. ε é uma palavra sobre o alfabeto $\{a, b, c\}$.
4. ε é a única palavra sobre o alfabeto $\emptyset$.
5. $a, e, i, o, u, ai, oi, ui$ e uai são palavras sobre o alfabeto Vogais.
6. $\{a, b\}^\star = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$.
7. $\emptyset^\star = \{\varepsilon\}$.

Definição C.3 (Linguagem Formal)

Uma *linguagem formal*, ou simplesmente *linguagem*, é um conjunto de palavras sobre um alfabeto.

Exemplo C.5 (Linguagem Formal)

Suponha o alfabeto $\Sigma = \{a, b\}$. Então:

1. O conjunto $\emptyset$ é uma linguagem sobre Σ.
2. O conjunto $\{\varepsilon\}$ também é uma linguagem (finita) sobre Σ.
3. O conjunto $\{b, ba, baa, baaa, \ldots\}$ é uma linguagem sobre Σ.
4. O conjunto $\{\varepsilon, ab, aabb, dab\}$ *não* é uma linguagem sobre Σ.
5. O conjunto de *palíndromos* (palavras que tem a mesma leitura tanto da esquerda para direita quanto da direita para a esquerda) sobre Σ é uma linguagem (infinita) sobre Σ.

C.3 Definição Indutiva de Conjuntos e Estruturas

Conjuntos e outras estruturas, como listas e árvores, por exemplo, também podem ser definidos recursivamente.

Para conjuntos, no caso base uma coleção inicial de elementos é especificada. No caso recursivo são dadas as regras para a formação de novos elementos a partir dos antigos (aqueles que já se sabe estarem no conjunto).

Exemplo C.6

Considere o conjunto $S \subset \mathbb{Z}$ definido por:

1. *Caso Base*: $3 \in S$.
2. *Caso Indutivo*: Se $x \in S$ e $y \in S$, então $x + y \in S$.

Assim, os elementos de S são 3, 6, 9, 12, 15, ... Pois:

$$\begin{aligned} 3 &\Leftarrow \text{caso base} \\ 6 &\Leftarrow 3+3 \\ 9 &\Leftarrow 6+3, 3+6 \\ 12 &\Leftarrow 6+6, 3+9, 9+3 \\ 15 &\Leftarrow 3+12, 6+9, 9+6, 12+3 \\ &\dots \end{aligned}$$

Seja Σ um alfabeto, conforme Definição C.1. Iremos agora dar uma definição indutiva de Σ^* (o conjunto de todas as palavras sobre Σ).

Definição C.4 (Conjunto Σ^*)

Seja Σ um alfabeto. O conjunto Σ^*, de todas as cadeias sobre Σ, é recursivamente definido por:

1. *Caso base*: $\varepsilon \in \Sigma^*$, onde ε é a cadeia vazia.
2. *Caso indutivo*: Seja w uma palavra tal que $w \in \Sigma^*$ e seja x um símbolo tal que $x \in \Sigma$, então $wx \in \Sigma^*$. Onde wx denota a cadeia formada acrescentando-se o símbolo x ao final da palavra w.

Exemplo C.7

Se $\Sigma = \{0, 1\}$, então após a primeira aplicação da etapa indutiva, os elementos 0, 1 de Σ^* são formadas. Pela aplicação da etapa indutiva uma vez mais, temos 00, 01, 10, 11 como os elementos de Σ^*. Aplicando a etapa indutiva uma terceira vez temos 000, 001, 010, 011, 100, 101, 110 e 111, e assim por diante.

Exemplo C.8

Define-se um conjunto E, recursivamente, como se segue:

1. $0 \in E$
2. se $n \in E$, então $n + 2 \in E$
3. se $n \in E$, então $-n \in E$

Usando esta definição, podemos ver que uma vez que $0 \in E$ por 1, segue-se a partir de 2 que $0 + 2 = 2 \in E$ e logo $2 + 2 = 4 \in E$, $4 + 2 = 6 \in E$, ..., na verdade qualquer número par não negativas pertence a E. Além disso, por 3, $-2, -4, -6, \dots \in E$.

Existe alguma coisa mais em E? A definição não diz isso explicitamente, mas uma condição *implícita* de uma definição recursiva é que a única maneira de as coisas pertencerem a E é como consequência de 1, 2 ou 3. Então, podemos afirmar que E é exatamente o conjunto dos números inteiros pares.

Exemplo C.9

Define-se um conjunto S, de cadeias de a's e b's (i.e., $S \subseteq \{a, b\}^*$) recursivamente como se segue:

1. $\varepsilon \in S$, onde ε é a string vazia,
2. se $u \in S$, então $aub \in S$,
3. se $u \in S$, então $bua \in S$,
4. se $u, v \in S$, então $u.v \in S$, onde $u.v$ denota a concatenação das cadeias u e v.

Usando esta definição, podemos ver que, uma vez que $\varepsilon \in S$ por 1, segue-se, a partir de 2, que $a\varepsilon b = ab \in S$, então $aabb \in S$, também por 2, e $baba \in S$ por 3. Da mesma forma, $b\varepsilon a = ba \in S$ por 3, assim $abab \in S$ por 2 e $bbaa \in S$ por 3. Além disso, uma vez que $ab \in S$ e $ba \in S$, temos $abba \in S$, bem como a $baab \in S$ por 4.

Observe que cada cadeia em S tem um número igual de a's e b's. É possível provar esta propriedade por indução sobre as regras que definem cadeias que pertencem a S.

Duas palavras (cadeias) podem ser combinadas pela operação de *concatenação*. A operação de concatenação é denotada por $u.v = w$, onde u, v e w são palavras sobre um mesmo alfabeto Σ, ou seja, $u, v, w \in \Sigma^*$, e esta operação indica que a palavra w é formada por todos os símbolos da palavra u seguidos por todos os símbolos da palavra v. Por exemplo, $acaca.ababa = acacaababa$.

Definição C.5 (Concatenação de Palavras)

Seja Σ um conjunto de símbolos. Podemos definir a concatenação dos duas palavras recursivamente como se segue.

1. *Caso base*: Se $w \in \Sigma^*$, então $w.\varepsilon = w$, onde ε é a palavra vazia.
2. *Caso indutivo*: Se $w_1 \in \Sigma^*$, $w_2 \in \Sigma^*$, e $x \in \Sigma$, então $w_1.(w_2x) = (w_1.w_2)x$.

Exemplo C.10

Seja $\Sigma = \{a, b, c\}$, e sejam $w_1 = babaca$ e $w_2 = acaca$, então a concatenação das palavras w_1 e w_2, denotada por $w_1.w_2$ pode ser determinada pela aplicação sucessiva da definição (indutiva) de concatenação de cadeias como

se seque:

$$\begin{aligned} w_1.w_2 &= babaca.acaca \\ &= babaca.(acaca) \\ &= (babaca.acac)a \\ &= ((babaca.aca)c)a \\ &= (((babaca.ac)a)c)a \\ &= ((((babaca.a)c)a)c)a \\ &= (((((babaca.\varepsilon)a)c)a)c)a \\ &= (((((babaca)a)c)a)c)a \\ &= ((((babacaa)c)a)c)a \\ &= (((babacaac)a)c)a \\ &= ((babacaaca)c)a \\ &= (babacaacac)a \\ &= babacaacaca \end{aligned}$$

Uma propriedade de cadeias (palavras) que aparece de modo bastante intuitivo é a propriedade de *comprimento* da cadeia. O aluno provavelmente responderia que o comprimento de ε (cadeia vazia) é zero, e que o comprimento da cadeia abc é 3. Iremos agora formalizar esta propriedade através de uma definição recursiva.

Definição C.6 (Comprimento de Palavra)

Sejam Σ um alfabeto. A função $\mathrm{l} : \Sigma^* \to \mathbb{N}$ que mapeia cada palavra em Σ^* no seu respectivo comprimento pode ser definida como:

1. *Caso base:* $\mathrm{l}(\varepsilon) = 0$, onde ε é a palavra vazia.
2. *Caso indutivo:* $\mathrm{l}(wx) = \mathrm{l}(w) + 1$, se $w \in \Sigma^*$ e $x \in \Sigma$.

Exemplo C.11

Sejam $\Sigma = \{a, b, c\}$ e $w_1 = babaca$. Usando a definição recursiva dada para o comprimento de palavras na Definição C.6 podemos calcular o comprimento da palavra w_1.

$$\begin{aligned} \mathrm{l}(w_1) &= \mathrm{l}(babaca) \\ &= \mathrm{l}(babac) + 1 \\ &= \mathrm{l}(baba) + 1 + 1 \\ &= \mathrm{l}(bab) + 1 + 1 + 1 \\ &= \mathrm{l}(ba) + 1 + 1 + 1 + 1 \\ &= \mathrm{l}(b) + 1 + 1 + 1 + 1 + 1 \\ &= \mathrm{l}(\varepsilon) + 1 + 1 + 1 + 1 + 1 + 1 \\ &= 0 + 1 + 1 + 1 + 1 + 1 + 1 \\ &= 6 \end{aligned}$$

C.4 Indução Estrutural

Para provar resultados sobre conjuntos recursivamente definidos, pode-se usar indução matemática e também pode-se usar uma forma mais conveniente de indução conhecida como indução estrutural.

Uma demonstração usando indução estrutural tem a seguinte forma:

1. *Caso base:* Mostrar que o resultado é válido para todos os elementos especificados no caso base da definição recursiva do conjunto em questão.
2. *Caso indutivo:* Mostrar que, se a afirmação é verdadeira para cada um dos elementos utilizados para a construção de novos elementos no caso intutivo da definição do conjunto, então o resultado é válido para estes novos elementos.

Exemplo C.12

Use indução estrutural para provar que $\mathrm{l}(v.w) = \mathrm{l}(v) + \mathrm{l}(w)$, onde $v, w \in \Sigma^*$ e $l : \Sigma^* \to \mathbb{N}$ é a função de comprimento de palavras definida na Definição C.6.
Solução: Seja $P(w) : \mathrm{l}(v.w) = \mathrm{l}(v) + \mathrm{l}(w)$, onde $v, w \in \Sigma^*$. Sabemos que $\mathrm{l}(\varepsilon) = 0$, e que $\mathrm{l}(wx) = \mathrm{l}(w) + 1$ quando $w \in \Sigma^*$ e $x \in \Sigma$.

1. *Caso base*: Para $w = \varepsilon$, temos que mostrar que $P(\varepsilon)$ é verdadeiro, ou seja, temos que mostrar que

$$\mathrm{l}(v.\varepsilon) = \mathrm{l}(v) + \mathrm{l}(\varepsilon)$$

Pela definição de concatenação $v.\varepsilon = v$, e pela definição de comprimento $\mathrm{l}(\varepsilon) = 0$, logo:

$$\begin{aligned} \mathrm{l}(v.\varepsilon) &= \mathrm{l}(v) + \mathrm{l}(\varepsilon) \\ \mathrm{l}(v) &= \mathrm{l}(v) + 0 \\ \therefore\ \mathrm{l}(v) &= \mathrm{l}(v) \end{aligned}$$

Que demonstra que o caso base é verdadeiro.

2. *Caso indutivo*: Supomos que $P(u)$ é verdadeiro para uma palavra $u \in \Sigma^*$ arbitrária. Devemos mostrar que esta suposição implica que $P(ux)$ é verdadeiro para qualquer $x \in \Sigma$. Ou seja, supomos que a equação $\mathrm{l}(v.u) = \mathrm{l}(v) + \mathrm{l}(u)$ é verdadeira (***hipótese de indução***), e devemos mostrar que $\mathrm{l}(v.ux) = \mathrm{l}(v) + \mathrm{l}(ux)$ é verdadeira.

$$\begin{aligned} \mathrm{l}(v.ux) &= \mathrm{l}(v) + \mathrm{l}(ux) && \\ \mathrm{l}(v.u) + 1 &= \mathrm{l}(v) + \mathrm{l}(u) + 1 && \triangleright \text{pela definição de } l \\ \mathrm{l}(v) + \mathrm{l}(u) + 1 &= \mathrm{l}(v) + \mathrm{l}(u) + 1 && \triangleright \text{pela hipótese de indução} \end{aligned}$$

O que demonstra que, de fato, $\mathrm{l}(v.ux) = \mathrm{l}(v) + \mathrm{l}(ux)$.

□

A operação *reverso* aplicada a uma palavra (cadeia) mapeia esta cadeia em uma outra palavra, sobre o mesmo alfabeto, que possui exatamente os mesmos símbolos da cadeia original, mas na ordem inversa. Por exemplo, $\mathrm{r}(babaca) = acabab$. Onde $\mathrm{r}(w)$ denota a operação reverso aplicada à palavra w.

Definição C.7 (Reverso de Palavras)

Seja Σ um conjunto de símbolos e Σ^* o conjunto de palavras formadas por símbolos Σ. Podemos definir $\mathrm{r}(w)$, o reverso de uma palavra w, como se segue.

1. Caso base: $\mathrm{r}(\varepsilon) = \varepsilon$.
2. Caso indutivo: $\mathrm{r}(wx) = x\,\mathrm{r}(w)$, onde $x \in \Sigma$ e $w \in \Sigma^*$.

Exemplo C.13

Mostre, passo-a-passo, como é obtido o reverso da palavra $abacca$ de acordo com a operação para a operação reverso dada na Definição C.7.
Solução:

$$\begin{array}{rclcrclcrcl} \mathrm{r}(abacca) & = & a\,\mathrm{r}(abacc) & = & ac\,\mathrm{r}(abac) & = & acc\,\mathrm{r}(aba) \\ & = & acca\,\mathrm{r}(ab) & = & accab\,\mathrm{r}(a) & = & accaba\,\mathrm{r}(\varepsilon) \\ & = & accaba \end{array}$$

Exemplo C.14

Use indução estrutural para provar que $\mathrm{r}(u.v) = \mathrm{r}(v).\mathrm{r}(u)$
Solução: É possível efetuar a demonstração de duas formas diferentes:

1. Considerar u como uma palavra arbitrária e usar indução sobre v.
2. Considerar v como uma palavra arbitrária e usar indução sobre u.

Optaremos por seguir a primeira forma, i.e., usar indução sobre v. O aluno pode tentar refazer o exemplo, como exercício, usando indução sobre u.

1. *Caso base:* $\mathrm{r}(u.\varepsilon) = \mathrm{r}(\varepsilon).\mathrm{r}(u) \Longrightarrow \mathrm{r}(u) = \varepsilon.\mathrm{r}(u) \Longrightarrow \therefore\ \mathrm{r}(u) = \mathrm{r}(u)$
2. *Caso indutivo:* Supondo que $\mathrm{r}(u.v) = \mathrm{r}(v).\mathrm{r}(u)$ seja verdadeiro, desejamos mostrar que $\mathrm{r}(u.vx) = \mathrm{r}(vx).\mathrm{r}(u)$.

$$\begin{array}{rcll} \mathrm{r}(u.vx) & = & x\,\mathrm{r}(u.v) & \triangleright \text{ pela definição de } \mathrm{r}(wx) \\ & = & x\,\mathrm{r}(v).\mathrm{r}(u) & \triangleright \text{ pela hipótese de indução} \\ & = & \mathrm{r}(vx).\mathrm{r}(u) & \triangleright \text{ pela definição de } \mathrm{r}(wx) \end{array}$$

□

C.5 Exercícios

Exercício C.1 Escreva os cinco primeiros valores para as sequências (definidas por recursão) abaixo.

1. $A(1) = 5$
   $A(n) = A(n-1) + 5$, para $n > 1$
2. $B(1) = 1$
   $B(n) = \frac{1}{B(n-1)}$, para $n > 1$
3. $C(1) = 1$
   $C(n) = C(n-1) + n^2$, para $n > 1$
4. $S(1) = 1$
   $S(n) = S(n-1) + \frac{1}{n}$, para $n > 1$
5. $T(1) = 1$
   $T(n) = n \cdot T(n-1)$, para $n > 1$

6. $P(1) = 1$
   $P(n) = n^2 \cdot P(n-1) + (n-1)$, para $n > 1$
7. $M(1) = 2$
   $M(2) = 2$
   $M(n) = 2M(n-1) + M(n-2)$, para $n > 2$
8. $D(1) = 3$
   $D(2) = 5$
   $D(n) = (n-1)D(n-1) + (n-2)D(n-2)$, para $n > 2$
9. $W(1) = 2$
   $W(2) = 3$
   $W(n) = W(n-1) \cdot W(n-2)$, para $n > 2$
10. $R(1) = 1$
    $R(2) = 2$
    $R(3) = 3$
    $R(n) = R(n-1) + 2R(n-2) + 3R(n-3)$, para $n > 3$

Exercício C.2 Suponha que a exponenciação é definida pela equação

$$x^j \cdot x = x^{j+1}$$

para qualquer $j \in \mathbb{Z}^+$. Use indução para provar que $x^n \cdot x^m = x^{n+m}$ para $m, n \in \mathbb{Z}^+$

Exercício C.3 Uma cadeia de 0s e 1s deve ser processada e convertida para uma cadeia de paridade par acrescentando-se um bit de paridade no final da cadeia. O bit de paridade é um bit que faça com que o número total de bits 1 da cadeia final seja par. Por exemplo:

- Para a cadeia 0010101 o bit de paridade seira 1, e a cadeia final seria 00101011, com 4 bits 1.
- Para a cadeia 0010010 o bit de paridade seria 0, e a cadeia final seria 00100100, com 2 bits 1.

O bit de paridade é inicialmente 0. Quando um caracter 0 é processado, o bit de paridade permanece inalterado. Quando um caracter 1 é processado, o bit de paridade muda de 0 para 1 ou de 1 para 0.

Prove que o número de 1s numa cadeia final, incluindo o bit de paridade, é sempre par.
(*Sugestão:* considere várias possibilidades.)

Exercício C.4 Sejam o consjunto $T = \{a, b\}$, e a linguagem T_1 sobre T definida do seguinte modo:

1. $a \in T_1$
2. Se $u, v \in T_1$, então $ubv \in T_1$.

Considerando as definições acima, responda os itens abaixo:

1. Escreva 5 cadeias que pertencem a T_1.
2. Se $w \in T_1$ é possível ter dois a's consecutivos em w? Demonstre a sua resposta usando indução.
3. Demonstre que se $w \in T_1$ o número de a's em w é igual ao número de b's mais 1.

Exercício C.5 Sejam o consjunto $T = \{a, b\}$, e a linguagem T_2 sobre T definida do seguinte modo:

1. $a \in T_2$
2. Se $w \in T_2$, então $wbw \in T_2$.

Considerando as definições acima, responda os itens abaixo:

1. Escreva 4 cadeias que pertencem a T_2.
2. Dê 3 exemplos de cadeias que pertencem a T_1 (Exercício C.4) mas não pertencem a T_2.

3. Se $w \in T_2$ é possível ter dois b's consecutivos em w? Demonstre a sua resposta usando indução.
4. Demonstre que, se $w \in T_2$, o número de a's em w é igual ao número de b's mais 1.
5. Demonstre que, se $w \in T_2$, então o número de a's em w é igual a 2^n para algum número natural n.

Apêndice D

Mini Tutorial de Python

Sumário

D.1 Instalação do Python

D.2 Executando Código Python

D.3 Variáveis e Scripts

D.4 Condicioais e Laços

D.5 Funções

D.6 Coleções

D.7 Importando Bibliotecas

D.8 Entrada e Saída de Arquivos

Apêndice E

Lógica Digital

Sumário

E.1 Conceitos Básicos

E.2 Aritmética Digital

E.3 Álgebra Booleana

E.4 Circuitos Lógicos

E.5 Simplificação de Expressões Booleanas

E.6 Minimização de Funções Booleanas

E.7 Flip-Flops e Multivibradores

E.8 Registradores de Deslocamento

E.9 Contadores

Soluções dos Problemas Propostos

Problema .

- ☑ *Fernando cozinho demais a couve-flor.*
- ☐ *Não existe força que possa parar você.* — Não é proposição por usar o indicial "você".
- ☐ *Estava muito escuro lá.* — Não é proposição porque usa o indicial "lá".
- ☐ *Meus amigos foram à mata colher cogumelos.* — Não é proposição porque usa o indicial "meus (amigos)".
- ☑ *Renata se esgueirou para dentro da cozinha.*
- ☑ *Paula se grudou na parede recém pintada.*
- ☑ *Se Fred Astaire não fosse um dançarino, Greta Garbo não seria uma atriz.*
- ☐ *Se ao menos as crianças soubessem mais do que seus pais!* — Não é proposição porque é uma exclamação.
- ☐ *Será que o Henrique algum dia vai gostar de garotas?* — Não é proposição porque é uma pergunta.
- ☑ *Floriano Peixoto é uma mulher.*

Problema .

(a) ***(Simples)*** *Tiago é um homem extraordinariamente agradável que odeia todas as mulheres que usam chapéus grandes.*

(b) ***(Composta)*** *Tiago convidou Suzana para sair* ***e*** *ela aceitou.*

(c) ***(Composta)*** ***Se*** *Suzana não chegar na hora,* [então] *Tiago ficará preocupado.*

(d) ***(Simples)*** *Suzana foi bastante pontual.*

(e) ***(Composta)*** *Tiago* ***não*** *acreditou no que viu.*

(f) ***(Simples)*** *Suzana estava usando o maior chapéu que Tiago já viu na vida.*

(g) ***(Simples)*** *Tiago começou a pedir para Suzana tirar aquela coisa horrível que deve ter sido uma das coisas mais feias já produzidas por um ser humano.*

(h) ***(Composta)*** *Suzana concordou em tirar o chapéu* ***se e somente se*** *Tiago tirar a gravata borboleta* ***e*** *as botas de cowboy.*

Problema .

S – *Tiago convida Suzana para sair.*

A – *Suzana aceita sair com Tiago.*

C – *Suzana usa um chapéu grande.*

L – *Suzana quer ensinar uma lição ao Tiago.*

Problema . *Para cada sentenças serão apresentados abaixo a chave de simbolização e a forma simbólica correspondente à sentença.*

(Ra) *João é magro e José é alto.*
M – *João é magro.*
A – *José é alto.*
Fórmula: $M \wedge A$

(Rb) *Mário foi ao cinema, João foi ao teatro e Marcelo ficou em casa.*
C – *Mário vai ao cinema.*
T – *João vai ao teatro.*
H – *Marcelo fica em casa.*
Fórmula: $C \wedge T \wedge H$

(Rc) *Maria foi à praia ou ao mercado.*
P – *Maria vai à praia.*
M – *Maria vai ao mercado.*
Fórmula: $P \vee M$

(Rd) *Mário foi ao cinema ou Marcelo ficou em casa.*
C – *Mário vai ao cinema.*
H – *Marcelo fica em casa.*
Fórmula: $C \vee H$

(Re) *A Lua não é o satélite da Terra.*
L – *A Lua é o satélite da Terra.*
Fórmula: $\neg L$

(Rf) *Se a chuva continuar a cair, então o rio vai transbordar.* C – *A chuva continua a cair.*
T – *O rio transborda.*
Fórmula: $C \to T$

(Rg) *Se João estudar, será aprovado.*
E – *João estuda.*
A – *João é aprovado.*
Fórmula: $E \to A$

(Rh) *João será aprovado se e somente se estudar.* A – *João é aprovado.*
E – *João estuda.*
Fórmula: $A \leftrightarrow E$

Problema .

a)

A	B	$A \wedge B$	$A \wedge B \to A$
0	*0*	*0*	*1*
0	*1*	*0*	*1*
1	*0*	*0*	*1*
1	*1*	*1*	*1*

Tautologia.

b)

P	Q	$\neg P$	$P \wedge Q$	$\neg P \wedge (P \wedge Q)$
0	*0*	*1*	*0*	*0*
0	*1*	*1*	*0*	*0*
1	*0*	*0*	*0*	*0*
1	*1*	*0*	*1*	*0*

Contradição.

c)

P	Q	$\neg P$	$\neg Q$	$\neg P \vee \neg Q$	$Q \vee (\neg P \vee \neg Q)$	$P \vee (Q \wedge (\neg P \vee \neg Q))$
0	*0*	*1*	*1*	*1*	*0*	*0*
0	*1*	*1*	*0*	*1*	*1*	*1*
1	*0*	*0*	*1*	*1*	*0*	*1*
1	*1*	*0*	*0*	*0*	*0*	*1*

Contingência.

d)

A	B	$A \wedge B$	$B \to A \wedge B$	$A \to (B \to A \wedge B)$
0	*0*	*0*	*1*	*1*
0	*1*	*0*	*0*	*1*
1	*0*	*0*	*1*	*1*
1	*1*	*1*	*1*	*1*

Tautologia.

e)

P	$\neg P$	$0 \to \neg P$	$P \to (0 \to \neg P)$
0	*1*	*1*	*1*
1	*0*	*1*	*1*

Tautologia.

Bibliografia

[Ale08] Edigard de Alencar Filho. *Iniciação à Lógica Matemática*. 21ª. Nobel, 2008.

[Bak80] Jacobus W. de Bakker. *Mathematical Theory of Program Correctness*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1980. ISBN: 0135621321 (ver p. 14).

[Ber+01] Beatrice Berard et al. *Systems and Software Verification: Model-Checking Techniques and Tools*. New York, NY, USA: Springer-Verlag, 2001, p. 196 (ver p. 14).

[Bla15] J. Anthony Blair. "What Is Informal Logic?" Em: *Reflections on Theoretical Issues in Argumentation Theory*. Springer, 2015, pp. 27–42 (ver p. 5).

[Boo58] George Boole. *An Investigation of the Laws of Thought*. Dover Publications, 1958, p. 424. URL: http://www.gutenberg.org/etext/15114 (ver pp. 4, 81).

[CL73] Chin-liang Chang e Richard Char-Tung Lee. *Symbolic logic and mechanical theorem proving*. Academic Press, 1973, p. 331 (ver p. 14).

[CSM08] Dan Cryan, Sharron Shatil e Bill Mayblin. *Logic: A Graphic Guide*. Icon Books, 2008 (ver p. 82).

[Dij75] Edsger W. Dijkstra. "Guarded commands, nondeterminacy and formal derivation of programs". Em: *Commun. ACM* 18.8 (1975), pp. 453–457. ISSN: 0001-0782. DOI: http://doi.acm.org/10.1145/360933.360975 (ver p. 14).

[Dij76] Edsger W. Dijkstra. *A Discipline of Programming*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1976, p. 217 (ver p. 14).

[Eem+14] Frans H. van Eemeren et al. "Argumentation and Artificial Intelligence". Em: *Handbook of Argumentation Theory*. Springer Netherlands, 2014. Cap. 11, pp. 615–675. DOI: 10.1007/978-90-481-9473-5_11 (ver p. 13).

[Eli75] Peter Elias. "Universal codeword sets and representations of the integers". Em: *Information Theory, IEEE Transactions on* 21.2 (1975), pp. 194–203. ISSN: 0018-9448. DOI: 10.1109/TIT.1975.1055349 (ver p. 148).

[Fre79] Gottlob Frege. *Begriffsschrift, eine der arithmetischen Nachgebildete Formelsprache des reinen Denkens*. – Tradução para o inglês: *"Concept Script, a formal language of pure thought modelled upon that of arithmetic"*, por S. Bauer-Mengelberg em: Jean van Heijenoort (ed.), *"From Frege to Gödel: A Source Book in Mathematical Logic, 1879-1931"*, Cambridge, MA: Harvard University Press, 1967. Louis Nebert, 1879, p. 88 (ver p. 4).

[Gal86] Jean H. Gallier. *Logic for Computer Science: Foundations of Automatic Theorem Proving*. – Disponível para download em http://www.cis.upenn.edu/~jean/gbooks/logic.html. Harper & Row Publishers, 1986, p. 511. URL: http://www.cis.upenn.edu/~jean/gbooks/logic.html (ver p. 14).

[Ger06] Judith L. Gersting. *Fundamentos Matemáticos para a Ciência da Computação*. 5ª. LTC, 2006.

[GH09] Steven Givant e Paul Halmos. *Introduction to Boolean Algebras*. Undergraduate Texts in Mathematics. Springer, 2009 (ver p. 82).

[Hoa69] Charles A. R. Hoare. "An Axiomatic Basis for Computer Programming". Em: *Commun. ACM* 12.10 (1969), pp. 576–580. ISSN: 0001-0782. DOI: http://doi.acm.org/10.1145/363235.363259 (ver p. 14).

[HR04] Michael Huth e Mark Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. 2nd. Cambridge University Press, 2004, p. 440 (ver p. 14).

[Loc99] John Locke. *Ensaio Acerca do Entendimento Humano*. Nova Cultural, 1999 (ver p. 3).

[Men04] Paulo Blauth Menezes. *Matemática Discreta para Computação e Informática*. 1ª. Sagra Luzzatto, 2004.

[RSW06] Frank Ruskey, Carla D. Savage e Stan Wagon. "The Search for Simple Symmetric Venn Diagrams". Em: *Notices of the AMS* 53.11 (dez. de 2006), pp. 1304–1311 (ver p. 20).

[Rus15] Bertrand Russell. *História da Filosofia Ocidental*. Nova Fronteira, 2015 (ver p. 14).

[Sch03] Arthur Schopenhauer. *Como Vencer um Debate Sem Precisar Ter Razão*. Topbooks, 2003 (ver p. 11).

[SK11] D.E. Smith e L.C. Karpinski. *The Hindu-Arabic Numerals*. Cornell University Library historical math monographs. Ginn, 1911. URL: http://books.google.com.br/books?id=wEw6AAAAMAAJ (ver p. 148).

[Ski71] Burrhus Frederic Skinner. *Beyond Freedom and Dignity*. New York, USA: Knopf, 1971 (ver p. 6).

[Smu82] Raymond M. Smullyan. *The Lady or the Tiger? and Other Logic Puzzles*. Dover Publications, 1982 (ver p. 82).

[Smu90] Raymond M. Smullyan. *What is the name of this book?* Dover Publications, 1990 (ver p. 82).

[Ten02] Robert D. Tennent. *Specifying Software*. 1st. Cambridge University Press, 2002, p. 250 (ver p. 14).

[Ven80] John Venn. "On the Diagrammatic and Mechanical Representation of Propositions and Reasonings". Em: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 9 (1880), pp. 1–18 (ver p. 20).

[Wal07] Douglas N. Walton. *Dialog theory for critical argumentation*. Vol. 5. John Benjamins Publishing, 2007 (ver p. 8).

[WR25] Alfred North Whitehead e Bertrand Russell. *Principia Mathematica*. 2nd. Vol. 3 vols. Cambridge University Press, 1925 (ver p. 4).

[WRM08] Douglas Walton, Christopher Reed e Fabrizio Macagno. *Argumentation Schemes*. Cambridge University Press, 2008 (ver p. 12).