Lógica Computacional 1

Flávio L. C. de Moura
Departamento de Ciência da Computação
Universidade de Brasília[1]

14 de setembro de 2025

[1]flaviomoura@unb.br

Capítulo 1

Introdução

Este material foi desenvolvido para dar suporte à disciplina Lógica Computacional 1 do Departamento de Ciência da Computação da Universidade de Brasília.

A lógica consiste no estudo do raciocínio, ou seja, consiste no estudo dos caminhos que nos permitem concluir quando um determinado fato é verdadeiro. Na antiguidade, a lógica consistia em um ramo da Filosofia. Depois passou a ser também um ramo da Matemática, e mais recentemente, pode ser vista também como um ramo da Computação.

No contexto computacional, o estudo da lógica é particularmente importante porque é o fundamento matemático dos programas de computador. De fato, a lógica é utilizada tanto em projetos de *hardware* quanto de *software*.

A primeira parte deste material apresenta os conceitos básicos da Lógica Proposicional (LP), que será dividida em três etapas:

1. A Lógica Proposicional Minimal (LPM);

2. A Lógica Proposicional Intuicionista (LPI), e;

3. A Lógica Proposicional Clássica (LPC).

O estudo dessas etapas é feito de forma incremental, onde a compreensão de uma etapa nos permite avançar para a próxima. Utilizaremos o sistema de Dedução Natural para construirmos a demonstração dos sequentes, que são os objetos a serem provados, e portanto, verdadeiros.

A segunda parte, consiste no estudo da Lógica de Primeira Ordem (LPO), também conhecida como Lógica de Predicados. A LPO pode ser vista como a "lógica padrão"utilizada, ainda que informalmente, em Matemática e Computação.

Chamaremos os caminhos trilhados até uma conclusão de demonstração. Assim, podemos definir uma demonstração como sendo um objeto que nos permite concluir se uma determinada afirmação é verdadeira. As provas, ou demonstrações, tanto na LP quanto na LPO serão desenvolvidas em dois níveis: inicialmente serão feitas em papel e lápis (provas informais), e posteriormente, em computador (provas formais). A construção de provas é um tema que costuma ser espinhoso, de forma que aqui tentaremos facilitar o processo de familiarização com este tema partindo de provas simples, para em seguida explorarmos situações mais complexas. Para as provas formais, isto é, as provas desenvolvidas em computador utilizaremos o assistente de provas Rocq, que é um *software* de código aberto que pode ser facilmente instalado em qualquer sistema operacional.

Apesar da LP possuir limitações de expressividade, ela será útil para que possamos entender a dinâmica da construção de provas, mas a lógica efetivamente usada no dia a dia do matemático ou do cientista da computação é a Lógica de Primeira Ordem (LPO), que nos permitirá expressar propriedades de algoritmos de forma mais natural. Durante esta caminhada, estudaremos um assunto fundamental que está presente em diversos contextos: *indução*. Intuitivamente, o conceito de indução é bastante simples, mas a sua aplicação em situações específicas costuma gerar muita dúvida.

A construção de provas mecânicas, ou seja, provas feitas em computador, é uma atividade que tem despertado interesse crescente nas últimas décadas em função da forma como a computação tem se infiltrado no nosso dia a dia. Mas aqui precisamos de uma pequena pausa para explicarmos o que queremos dizer com provas feitas por computador. Esta explicação se faz necessária porque existem pelo menos duas abordagens distintas no que se refere a este assunto: os provadores automáticos de teoremas por um lado, e os assistentes de prova por outro.

Um provador automático de teoremas é um programa munido de uma heurística que recebe um teorema como argumento e tenta, de forma automática, encontrar uma prova para o teorema dado [RV02, KSUV15, McC10]. Um assistente de provas por outro lado, consiste em um programa que requer a orientação/interação do usuário para poder construir uma prova. Ou seja, o usuário vai guiando o sistema na construção de prova, enquanto o sistema verifica se cada passo dado/sugerido pelo usuário está correto. São exemplos de assistentes de prova o PVS[ORS92], o Isabelle/HOL[NPW02], o Lean[dMU21] e o Rocq[Tea21]. Neste material trabalharemos com o assistente de provas Rocq, que é um sistema de código aberto e que pode ser instalado em sistemas Linux, MacOs e Windows, ou até mesmo ser executado via browser[APJ17].

Existem materiais muito interessantes que servem como tutoriais do Rocq, como por exemplo, [PCG+14], ou [Chl17]. Este material não é um tutorial do Rocq, mas a sua utilização enriquecerá bastante nosso estudo. Nosso foco é o estudo da lógica proposicional e de primeira ordem, assim como a sua utilização/aplicação no estudo de algoritmos. Para isto utilizaremos o Rocq como ferramenta de apoio mostrando como um assistente de provas pode ser útil nesta caminhada. Aqui é importante observar também que um assistente de provas é basicamente uma linguagem de programação juntamente com uma linguagem de especificação, ou seja, além da linguagem de programação existem uma camada lógica adicional, chamada de linguagem de especificação, que nos permite expressar os lemas e teoremas, por exemplo. A camada lógica do Rocq é baseada em um formalismo conhecido como *cálculo de construções indutivas* [Pau14] que é muito mais expressivo do que a lógica de primeira ordem que estudaremos aqui. Neste sentido, utilizaremos apenas uma pequena parte do poder de computacional do Rocq.

Não assumimos nenhum conhecimento prévio de Rocq, e sua utilização é opcional. Ou seja, é perfeitamente possível utilizar apenas a parte teórica deste material. A ideia aqui é que você possa reproduzir os temas abordados em Rocq a partir do zero: simplesmente abra o Rocq com a interface de sua preferência, e siga as orientações das atividades propostas. Nem tudo que será abordado aqui terá uma versão correspondente em Rocq, já que alguns temas serão puramente teóricos e foram pensados para serem feitos apenas em papel e lápis. E mesmo a etapa de construção de provas em um assistente de provas deve ser precedida de um esboço em papel e lápis. As atividades a serem realizadas no assistente de provas têm um arquivo correspondente de apoio cujo *link* é fornecido junto com o texto da atividade.

No contexto de algoritmos e desenvolvimento de *software* é comum a utilização de testes como método de validação. Ou seja, o programa (ou *software*) é executado com diversas entradas distintas, e se nenhum problema é encontrado, o programa é considerado bom o suficiente para ser utilizado. De fato, a primeira coisa que fazemos após implementar um algoritmo é testá-lo para diversas entradas, e caso alguma resposta seja incorreta, uma revisão da implementação é feita para corrigir o erro, e então novos testes são realizados. Este processo é repetido até que o programador sinta confiança na implementação, mas depois de todos estes testes é possível dizer que o programa é correto? Certamente não! Pensando no caso particular da implementação de um algoritmo de ordenação listas de naturais ou inteiros (ou qualquer estrutura munida de uma ordem total), sabemos que existe uma infinidade de listas de inteiros que podem ser utilizadas nos testes, e portanto não é possível testar todas elas. Em se tratando de programas utilizados em sistemas críticos (aviação, medicina, sistemas bancários, etc), por menores

que sejam as chances de erros, falhas não são toleradas. O que fazer então para garantir a correção de um programa? Uma abordagem possível consiste em utilizar a lógica para **provar** a correção do programa! Uma prova de uma propriedade de um programa fornece a garantia de que o programa satisfaz a propriedade provada **sempre**! Esta é a abordagem que utilizaremos aqui, e que tem se mostrado cada vez mais importante para o desenvolvimento da Matemática[HAB+15, Gon08, ADGR07, AH14] e Computação[Ler09, Pau15, NNdMA10]. Para concluir esta seção e começarmos a colocar a mão na massa, listamos três exemplos famosos de erros em sistemas computacionais:

1. **Therac-25**: Uma máquina de radioterapia controlada por computador causou a morte de pelo menos 6 pacientes entre 1985 e 1987 por overdose de radiação.

2. **Pentium FDIV**: Um erro na construção da unidade de ponto flutuante do processador Pentium da Intel causou um prejuízo de aproximadamente 500 milhões de dólares para a empresa que se viu forçada a substituir os processadores que já estavam no mercado em 1994.

3. **Ariane 5**: Um foguete que custou aproximadamente 7 bilhões de dólares para ser construído explodiu no seu primeiro voo em 1996 devido ao reuso sem verificação apropriada de partes do código do seu predecessor.

Já deixamos claro que vamos **provar** muita coisa aqui. Mas o que é uma prova? Uma resposta possível "é um argumento feito para convencer alguém"[Smu09]. O problema deste argumento é que pessoas diferentes podem ter compreensões distintas sobre o argumento, de forma que o argumento pode ser uma prova para uma pessoa, mas não para a outra... estranho, não? Uma definição geral e abstrata para a noção de prova não é uma tarefa fácil, mas forneceremos uma definição precisa em um contexto mais restrito, a saber, o da lógica simbólica[HR04, vD13].

Capítulo 2

A Lógica Proposicional (LP)

Linguagens naturais, como o Português por exemplo, são ambíguas. De fato, considere a seguinte afirmação:

> Eu vi a moça com o binóculo.

Quem estava com o binóculo? Ou ainda,

> O professor da Maria acabou a aula fazendo apontamentos no seu caderno.

Em qual caderno o professor estava fazendo apontamentos, no da Maria ou no do dele mesmo? A primeira coisa que faremos para evitarmos ambiguidades consiste em restringir a linguagem com a qual trabalharemos.

A Lógica Proposicional é baseada na noção de **proposição**, que consiste em uma afirmação (ou sentença) que pode ser qualificada como verdadeira ou falsa, mas nunca ambos. Por exemplo, são proposições:

- $2+2 = 4$.
- $1+3 < 0$.
- 2 é um número primo.
- João tem 20 anos e Maria tem 22 anos.

Mas nem toda sentença é uma proposição. De fato, a sentença "Feche a porta!", ou ainda a pergunta "Qual é o seu nome?"não podem ser qualificadas como verdadeira ou falsa, e portanto não são proposições. As proposições também são chamadas de sentenças lógicas.

Nos exemplos acima podemos observar dois tipos de proposições:

1. **Proposições atômicas**: são sentenças que não podem ser divididas em proposições menores como é o caso de:

   - 2+2 = 4.
   - 1+3 < 0.
   - 2 é um número primo.

2. **Proposições compostas**: são sentenças formadas pela composição de uma ou mais proposições atômicas como é o caso de:

   - João tem 20 anos e Maria tem 22 anos.
   - Se está chovendo então o chão está molhado.

Como dito anteriormente, a lógica é o estudo do raciocínio, mais especificamente, neste curso estamos interessados no raciocínio dedutivo. Ou seja, estamos interessados no processo de inferência onde a conclusão é garantida se as premissas forem verdadeiras. Vejamos dois exemplos:

Premissa 1: Se está chovendo então o chão está molhado.
Premissa 2: Está chovendo.
Conclusão: O chão está molhado.

Premissa 1: Se $x = 4$ então a terra é o centro do universo.
Premissa 2: $x = 4$
Conclusão: a terra é o centro do universo.

Estruturalmente, os dois exemplos são iguais, o que muda é o conteúdo das proposições. Se abstrairmos o conteúdo das proposições, e escrevermos a implicação "Se p então q"na forma $p \to q$, teremos algo da seguinte forma:

Premissa 1: $p \to q$
Premissa 2: p
Conclusão: q

Como veremos, o conteúdo das proposições será irrelevante porque estamos interessados nos passos da inferência. A linguagem da LP nos permite estudar as regras de inferência sem considerar o conteúdo das proposições. As estruturas abstratas construídas na linguagem da LP são chamadas de *fórmulas bem formadas*, ou simplesmente, fórmulas. Utilizaremos letras latinas minúsculas, aqui chamadas de *variáveis proposicionais*, para representar proposições atômicas. Assim, variáveis proposicionais são fórmulas. Adicionalmente, utilizaremos a implicação para construirmos fórmulas mais complexas, ou seja, para representarmos proposições compostas. Assumiremos um conjunto enumerável $\mathbb{P}$ de variáveis proposicionais.

2.1 O Fragmento Implicacional da Lógica Proposicional

O Fragmento Implicacional da Lógica Proposicional (FILP) contém exatamente as fórmulas que são construídas utilizando apenas variáveis proposicionais e a implicação como conectivo lógico. Podemos representar as fórmulas deste fragmento pela seguinte gramática:

$$\varphi ::= p \mid (\varphi \to \varphi) \tag{2.1}$$

onde $p \in \mathbb{P}$, e o construtor $\varphi \to \varphi$ diz que uma fórmula implicacional é construída a partir de duas fórmulas já construídas anteriormente (não necessariamente iguais). Por exemplo, se p e q denotam variáveis proposicionais então podemos concluir que $p \to q$ é uma fórmula, e neste caso, chamamos p de *antecedente*, e q de *sucedente* da implicação. Utilizando esta nova fórmula, podemos construir uma nova implicação a partir dela, e por exemplo, p, obtendo $(p \to q) \to p$ ou $p \to (p \to q)$, e assim por diante. Como indicado pela gramática (2.1), utilizaremos letras gregas minúsculas para representar as fórmulas da LP.

A implicação lógica possui algumas diferenças em relação à implicação que costumamos utilizar no dia a dia em linguagem natural. Por exemplo, não precisa existir uma relação de causa e efeito entre o antecedente e o consequente de uma implicação lógica. No primeiro exemplo acima, a causalidade existe entre o antecedente (está chovendo) e o consequente (o chão está molhado) da implicação, ou seja, o fato de estar chovendo é causa do chão estar molhado. No segundo exemplo, não existe relação de causa e efeito entre o antecedente ($x = 4$) e o consequente (a terra é o centro do universo). A implicação lógica é definida apenas pela relação de dependência entre os valores de verdade do antecedente e do consequente, sintetizada na seguinte tabela:

φ	ψ	$\varphi \to \psi$
V	V	V
V	F	F
F	V	V
F	F	V

A tabela acima é conhecida como *tabela verdade* da implicação e fornece todas as possíveis relações de valores de verdade entre o antecedente e o consequente da implicação. Ou seja, nos fornece a semântica ou o significado da implicação. Existem outras formas de expressar a implicação $\varphi \to \psi$ que podem facilitar a compreensão destas relações:

1. φ é condição suficiente para ψ, ou seja, se *varphi* for verdadeira (V) então ψ também será verdadeira. Basta que φ seja verdadeira para que ψ também seja.

2. ψ é condição necessária para φ, ou seja, *psi* segue de φ.

Nosso objetivo agora é raciocinar sobre as fórmulas construídas a partir da gramática (2.1). Mais especificamente, queremos obter (ou derivar) novas informações a partir de informações conhecidas. Tudo isto em um contexto abstrato onde os símbolos proposicionais utilizados podem representar qualquer informação que corresponda a uma proposição. Utilizaremos a notação de *sequentes* para separar as informações (fórmulas) dadas, ou conhecidas, da nova informação (fórmula) que queremos derivar ou concluir. Chamaremos as fórmulas dadas de *premissas*, e a fórmula a ser derivada de *conclusão*, assim um sequente é formado por duas partes: um conjunto finito de fórmulas (que são as premissas), digamos Γ, e uma fórmula que é a conclusão, digamos φ, que escrevemos como $\Gamma \vdash \varphi$. Assim, se $\varphi_1, \varphi_2, \ldots, \varphi_n$ são as premissas de um sequente, e se ψ é a sua conclusão, então escrevemos $\varphi_1, \varphi_2, \ldots, \varphi_n \vdash \psi$ para representar o sequente que tem ψ como conclusão, e o conjunto $\{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ de premissas. O conjunto $\{\varphi_1, \varphi_2, \ldots, \varphi_n\}$, isto é, a primeira componente do sequente $\varphi_1, \varphi_2, \ldots, \varphi_n \vdash \psi$ também pode ser chamado de *contexto* ao longo do texto, e normalmente será escrito sem as chaves que usualmente são

usadas para representar conjuntos. Este é um abuso de linguagem usado para deixar a notação mais leve. Assim, se Γ denota um conjunto finito de fórmulas, ao invés de $\Gamma \cup \{\varphi\} \vdash \psi$, escreveremos simplesmente $\Gamma, \varphi \vdash \psi$, onde Γ, φ deve então ser lido como a união Γ com o conjunto unitário $\{\varphi\}$.

O conceito de prova agora será definido de forma mais precisa. Concretamente, uma prova (ou uma derivação) de um sequente da forma $\Gamma \vdash \psi$ é uma sequência de passos dedutivos, e um passo dedutivo consiste na aplicação de uma *regra de inferência* que possui a seguinte forma:

$$\frac{\Gamma_1 \vdash \gamma_1 \ \Gamma_2 \vdash \gamma_2 \ldots \Gamma_k \vdash \gamma_k}{\Gamma \vdash \psi}$$

onde $k \geq 0$. Quando $k = 0$ e $\Gamma = \{\psi\}$ a regra corresponde a um *axioma*:

$$\frac{}{\{\psi\} \vdash \psi}\ (\mathrm{Ax})$$

Uma prova (*i.e.* uma sequência de passos dedutivos) pode ser representada por meio de uma estrutura de árvore, onde os nós são anotados com sequentes. A raiz da árvore é anotada com o sequente que queremos provar, digamos, $\Gamma \vdash \psi$, e as folhas são axiomas. Quais são as regras de inferência que podem ser utilizadas no fragmento implicacional da lógica proposicional? Além do axioma apresentado acima, temos duas regras para a implicação. Antes de apresentá-las devemos lembrar que o sistema dedutivo que utilizaremos se chama *dedução natural*, e as regras deste sistema são divididas em dois tipos: *introdução* e *eliminação*. A regra de eliminação da implicação é conhecida pelo nome *modus ponens* e tem a seguinte estrutura:

$$\frac{\Gamma_1 \vdash \varphi \to \psi \qquad \Gamma_2 \vdash \varphi}{\Gamma_1 \cup \Gamma_2 \vdash \psi}\ (\to_e)$$

ou seja, para construirmos uma prova de um sequente com a forma $\Gamma \vdash \psi$ utilizando esta regra, precisamos construir duas outras provas: uma do sequente $\Gamma \vdash \varphi \to \psi$, e outra do sequente $\Gamma \vdash \psi$. Ou seja, na leitura desta regra de baixo para cima (*i.e.* da conclusão para as premissas) reduzimos o problema de provar $\Gamma \vdash \psi$ a dois outros problemas (potencialmente) mais simples. Esta regra também pode ser lida de cima para baixo (*i.e.* das premissas para a conclusão), e neste caso precisamos de uma prova de uma implicação, a saber $\Gamma \vdash \varphi \to \psi$, e de uma prova do antecedente desta implicação, a saber $\Gamma \vdash \psi$ para construirmos uma prova da conclusão da implicação, ou seja, uma prova de $\Gamma \vdash \psi$.

A regra de introdução é bastante intuitiva e, em certo sentido, nos fornece uma definição da implicação:

$$\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi}\ (\to_{\mathrm{i}})$$

ou seja, na leitura de baixo para cima, para construirmos a prova de uma implicação precisamos construir uma prova do sucedente assumindo que temos uma prova do antecedente. Na leitura de cima para baixo,

precisamos transformar uma prova do antecedente em uma prova do sucedente.

O interesse computacional do fragmento implicacional está diretamente relacionado ao algoritmo de inferência de tipos em linguagens funcionais[Hin97]. O fundamento teórico destas linguagens é o cálculo λ[Bar84] desenvolvido por Alonzo Church em 1936 [Chu36, Chu40]. Para mais detalhes veja o Capítulo 1 de [AdM17].

Como primeiro exemplo, considere o sequente $\vdash (p \to q) \to (q \to r) \to p \to r$. A primeira observação a ser feita aqui é que a implicação é associativa à direita, ou seja, $\varphi \to \psi \to \gamma$ deve ser lido como $\varphi \to (\psi \to \gamma)$, e não como $(\varphi \to \psi) \to \gamma$. Portanto, o sequente que queremos provar deve ser lido como $\vdash (p \to q) \to ((q \to r) \to (p \to r))$. Vejamos como aplicar as regras $(\to_{\mathrm{i}}), (\to_{\mathrm{e}})$ e (Ax) apresentadas anteriormente para construirmos a árvore de derivação do sequente $\vdash (p \to q) \to (q \to r) \to p \to r$. Nosso objetivo é construir uma árvore com raiz $\vdash (p \to q) \to (q \to r) \to p \to r$, cuja folha sejam axiomas. Assim, iniciaremos a prova de baixo para cima, isto é, da raiz para as folhas da árvore. Como a fórmula a ser provada é uma implicação, a utilização da regra $(\to_{\mathrm{i}})$ parece ser uma boa ideia:

$$\dfrac{?}{\vdash (p \to q) \to (q \to r) \to p \to r}\,(\to_{\mathrm{i}}) \quad \Longleftrightarrow \quad \dfrac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi}\,(\to_{\mathrm{i}})$$

Aqui temos o passo chave para aplicarmos as regras de Dedução Natural: precisamos instanciar as variáveis Γ, φ e ψ da regra $(\to_{\mathrm{i}})$. Neste caso, Γ é instanciada como o conjunto vazio, φ é instanciada com a fórmula $p \to q$, e ψ é instanciada com a fórmula $(q \to r) \to p \to r$. Esta instanciação nos permite saber o que colocar no antecedente da regra $(\to_{\mathrm{i}})$:

$$\dfrac{\dfrac{?}{p \to q \vdash (q \to r) \to p \to r}\,(\to_{\mathrm{i}})}{\vdash (p \to q) \to (q \to r) \to p \to r}\,(\to_{\mathrm{i}}) \quad \Longleftrightarrow \quad \dfrac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi}\,(\to_{\mathrm{i}})$$

Agora podemos repetir o processo com o sequente $p \to q \vdash (q \to r) \to p \to r$. Como o consequente é uma implicação, vamos novamente aplicar a regra $(\to_{\mathrm{i}})$ instanciando Γ com o conjunto unitário contendo a fórmula $p \to q$, *varphi* com a fórmula $q \to r$ e ψ com a fórmula $p \to r$:

$$\dfrac{\dfrac{\dfrac{?}{p \to q \vdash (q \to r) \to p \to r}\,(\to_{\mathrm{i}})}{\vdash (p \to q) \to (q \to r) \to p \to r}\,(\to_{\mathrm{i}})}{} \quad \Longleftrightarrow \quad \dfrac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi}\,(\to_{\mathrm{i}})$$

E assim, podemos avançar mais um passo na construção da árvore:

$$\dfrac{\dfrac{\dfrac{?}{p \to q, q \to r \vdash p \to r}\,(\to_{\mathrm{i}})}{p \to q \vdash (q \to r) \to p \to r}\,(\to_{\mathrm{i}})}{\vdash (p \to q) \to (q \to r) \to p \to r}\,(\to_{\mathrm{i}}) \quad \Longleftrightarrow \quad \dfrac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi}\,(\to_{\mathrm{i}})$$

Mais uma vez, a fórmula a ser provada é uma implicação, a saber $p \to r$. Então podemos aplicar a regra $(\to_{\mathrm{i}})$ mais uma vez:

$$\dfrac{\dfrac{\dfrac{\dfrac{?}{p \to q, q \to r \vdash p \to r}\,(\to_{\mathrm{i}})}{p \to q \vdash (q \to r) \to p \to r}\,(\to_{\mathrm{i}})}{\vdash (p \to q) \to (q \to r) \to p \to r}\,(\to_{\mathrm{i}})}{} \quad \Longleftrightarrow \quad \dfrac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi}\,(\to_{\mathrm{i}})$$

Instanciando Γ com o conjunto $p \to q, q \to r$, φ com p e ψ com r, podemos adicionar um novo nó à árvore de derivação:

$$\dfrac{\dfrac{\dfrac{\dfrac{?}{p \to q, q \to r, p \vdash r}}{p \to q, q \to r \vdash p \to r}\,(\to_{\mathrm{i}})}{p \to q \vdash (q \to r) \to p \to r}\,(\to_{\mathrm{i}})}{\vdash (p \to q) \to (q \to r) \to p \to r}\,(\to_{\mathrm{i}})$$

Agora precisamos construir uma prova de r tendo como premissas as fórmulas $p \to q, q \to r$ e p. Como a fórmula r não é uma implicação, não temos mais como aplicar a regra $(\to_{\mathrm{i}})$. Este sequente também não tem a forma do axioma, e portanto não podemos aplicar a regra (Ax). Assim, vamos tentar aplicar a regra $(\to_e)$ de eliminação da implicação. A instanciação a ser feita é relativamente simples: $\Gamma_1 \cup \Gamma_2$ será instanciado com o conjunto $p \to q, q \to r, p$ e ψ será instanciado com r:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{?}{p \to q, q \to r, p \vdash r}\,(\to_e)}{p \to q, q \to r \vdash p \to r}\,(\to_{\mathrm{i}})}{p \to q \vdash (q \to r) \to p \to r}\,(\to_{\mathrm{i}})}{\vdash (p \to q) \to (q \to r) \to p \to r}\,(\to_{\mathrm{i}}) \iff \dfrac{\Gamma_1 \vdash \varphi \to \psi \qquad \Gamma_2 \vdash \varphi}{\Gamma_1 \cup \Gamma_2 \vdash \psi}\,(\to_e)$$

As premissas da regra $(\to_e)$ exigem um pouco mais de atenção. Observe que a árvore de prova é bifurcada em dois ramos: um para $\Gamma_1 \vdash \varphi \to \psi$ e outro para $\Gamma_2 \vdash \varphi$. Adicionalmente, a fórmula φ não aparece no consequente da regra. Como fazer então esta instanciação? A única fórmula que temos que nos permite concluir r é $q \to r$, então parece uma boa ideia instanciar φ com q. Adicionalmente, nosso objetivo é chegar em um axioma, e portanto se instanciarmos Γ_1 com $q \to r$ conseguiremos concluir o ramo esquerdo da árvore com um axioma. No ramo direito, Γ_2 será instanciado com $p \to q, p$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{q \to r \vdash q \to r}\,(\mathrm{Ax}) \qquad \dfrac{?}{p \to q, p \vdash q}}{p \to q, q \to r, p \vdash r}\,(\to_e)}{p \to q, q \to r \vdash p \to r}\,(\to_{\mathrm{i}})}{p \to q \vdash (q \to r) \to p \to r}\,(\to_{\mathrm{i}})}{\vdash (p \to q) \to (q \to r) \to p \to r}\,(\to_{\mathrm{i}}) \iff \dfrac{\Gamma_1 \vdash \varphi \to \psi \qquad \Gamma_2 \vdash \varphi}{\Gamma_1 \cup \Gamma_2 \vdash \psi}\,(\to_e)$$

Por fim, a prova do sequente $p \to q, p \vdash q$ consiste em mais uma aplicação da regra $(\to_e)$. Você consegue dizer como são as instanciações neste caso?

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{q \to r \vdash q \to r}\,(\mathrm{Ax}) \qquad \dfrac{\dfrac{}{p \to q \vdash p \to q}\,(\mathrm{Ax}) \qquad \dfrac{}{p \vdash p}\,(\mathrm{Ax})}{p \to q, p \vdash q}\,(\to_e)}{p \to q, q \to r, p \vdash r}\,(\to_e)}{p \to q, q \to r \vdash p \to r}\,(\to_{\mathrm{i}})}{p \to q \vdash (q \to r) \to p \to r}\,(\to_{\mathrm{i}})}{\vdash (p \to q) \to (q \to r) \to p \to r}\,(\to_{\mathrm{i}})$$

O exemplo que acabamos de fazer contém as ideias mais importantes que utilizaremos ao longo de todo o curso: como instanciar e aplicar as regras de Dedução Natural.

Fragmento Implicacional da Lógica Proposicional

$$\dfrac{}{\varphi \vdash \varphi}\,(\mathrm{Ax})$$

Regras de introdução	Regras de eliminação
$\dfrac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi}\,(\to_i)$	$\dfrac{\Gamma_1 \vdash \varphi \to \psi \qquad \Gamma_2 \vdash \varphi}{\Gamma_1 \cup \Gamma_2 \vdash \psi}\,(\to_e)$

Agora veja se compreendeu o processo resolvendo os exercícios a seguir com as regras $(\to_{\mathrm{i}}), (\to_{\mathrm{e}})$ e (Ax):

Exercício 1. *Prove o sequente* $\vdash (p \to q \to r) \to q \to p \to r$.

Exercício 2. *Prove o sequente* $\vdash (p \to q \to r) \to (p \to q) \to p \to r$.

Exercício 3. *Prove o sequente* $\vdash (p \to q) \to (p \to r) \to (q \to r \to t) \to p \to t$.

Exercício 4. *Prove o sequente* $\vdash (q \to r \to t) \to (p \to q) \to p \to r \to t$.

Exercício 5. *Prove o sequente* $\vdash (p \to p \to q) \to p \to q$.

Exercício 6. *Prove o sequente* $\vdash (p \to q) \to p \to p \to q$

2.2 O fragmento implicacional no Rocq

As provas construídas na seção anterior baseiam-se na aplicação das regras $(\to_{\mathrm{i}})$, $(\to_e)$ e (Ax), que é um processo puramente sintático. Ou seja, uma vez escolhida a regra a ser aplicada, basta instanciar o sequente da parte inferior da regra para inferir o sequente da parte superior, quando a regra é aplicada de baixo para cima. O mesmo raciocínio vale quando aplicamos as regras são aplicadas de cima para baixo. Portanto, podemos pensar em implementar essas regras em uma linguagem de programação qualquer, e nosso trabalho seria indicar, a cada passo, qual regra deveria ser aplicada. A implementação das regras teria a vantagem de garantir a aplicação correta de cada uma, uma vez que a instanciação é única e a aplicação da regra é um passo puramente sintático. Nessa seção, veremos como colocar esta ideia na prática, mas ao invés de implementar as regras, vamos utilizar um sistema que já possui uma implementação das regras. Trata-se do assistente de provas Rocq (`https://rocq-prover.org`). O Rocq é uma ferramenta de código aberto e que pode ser facilmente instalada em qualquer sistema operacional.

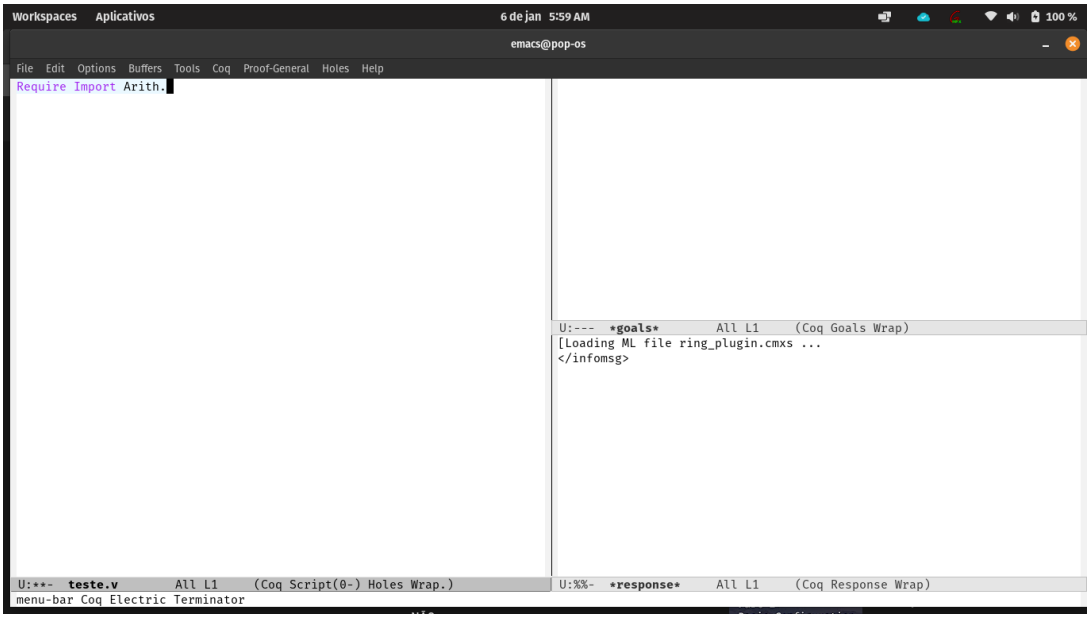
O Rocq é um sistema que possui uma implementação das lógicas que estudaremos neste curso. Um assistente de provas, como o nome sugere, é uma ferramenta que, sob nossa orientação, nos permitirá construir uma prova. Ou seja, nos permitirá refazer uma prova manuscrita para o computador. Mas qual é a vantagem de refazer uma prova no computador? A grande vantagem é ter a garantia de que a prova está efetivamente correta. Provas manuscritas são mais susceptíveis a erros.

No entanto, como comentado na Introdução, um assistente de provas não é adequado para encontrar uma nova prova. Ou seja, não deve ser utilizado de forma não planejada para eventualmente conseguir completar uma prova. A utilização eficiente consiste em primeiro ter uma prova manuscrita (ou pelo menos na cabeça) para, em seguida, reconstruí-la no assistente de prova. Provas feitas em assistentes de prova são bastante confiáveis, e amplamente aceitas na comunidade científica. Um caso interessante, que levou o matemático americano Thomas Hales ao mundo dos assistentes de prova, ocorreu em 1998 quando ele submeteu um artigo de aproximadamente 250 páginas para publicação contendo uma prova

da Conjectura de Kepler. Esta conjectura foi formulada em 1611 pelo matemático Johannes Kepler e permaneceu sem solução até então, ou seja, por quase 400 anos! Uma comissão de especialistas foi montada para analisar o artigo de Hales e concluiu, após quase 2 anos, que tinha 99% de certeza de que a prova apresentada estava correta. Conta-se, que o então diretor do periódico onde se pleiteava a publicação, disse a Hales que se ele fornecesse alguma nova evidência de que a prova estava correta, o artigo seria publicado. Mas o que pode fornecer uma evidência maior da prova de um teorema do que o próprio artigo? Sim, uma prova formal! Em 2003, Halles começou a trabalhar no projeto Flyspeck que tinha como objetivo formalizar sua prova utilizando os assistentes de prova Isabelle[NPW02] e HOL/Light[1], mas o artigo acabou sendo publicado em 2005[Hal05]. O projeto tinha uma previsão inicial de durar 20 anos, mas em 2014, isto é 11 anos depois, o grupo do projeto Flyspeck anunciou que tinha completado a formalização da prova da Conjectura de Kepler[HAB+15].

Voltando ao Rocq, é importante observarmos que ele é uma ferramenta de pesquisa, e não foi desenvolvido exclusivamente para o ensino. Grandes projetos foram feitos nele como a formalização do Teorema das 4 cores[Gon08], do Teorema de Feit-Thompson[GAA+13] e a verificação de um compilador C[Ler09]. Por não ter o ensino como foco principal, veremos que os comandos não estão em uma relação biunívoca com as regras de inferência estudadas aqui, mas ainda assim o Rocq nos será bastante útil. Assumiremos que você tem o Rocq instalado em sua máquina.

Utilizaremos `verbatim` para apresentar o código do Rocq. Uma sessão típica do Rocq possui três janelas:



A janela da esquerda é a janela de especificação, ou seja, onde escrevemos as definições, os lemas e as provas. As especificações serão escritas em uma caixa azul:

janela de especificação

A janela do canto superior direito nos mostra o *status* atual da prova; e a do canto inferior direito nos mostra mensagens do sistema. O texto correspondente a estas janelas aparecerá em uma caixa verde:

[1] `https://www.cl.cam.ac.uk/~jrh13/hol-light/`

janela de prova ou mensagens do sistema

Existem diversas razões para a utilização do Rocq, mas aqui apresentaremos duas: a primeira é que o Rocq implementa o sistema de dedução natural e portanto, é o candidato natural para explorar computacionalmente este sistema dedutivo; segundo, que o Rocq possui uma comunidade muito grande e sempre disponível para ajudar com as dúvidas[2]. Apresentaremos as regras do fragmento implicacional da LP e faremos uma analogia entre o sistema de dedução natural que apresentamos aqui e o Rocq sempre que possível, mas como veremos, esta analogia não é feita via uma correspondência direta entre as regras em dedução natural e as *regras* do Rocq que são chamadas de *táticas*. De fato, as táticas são desenvolvidas para realizarem vários passos de prova, incluindo simplificações, de só uma vez porque isto facilita o processo de construção de provas em sistemas mais complexos.

A regra de introdução da implicação ($\to_{\mathrm{i}}$) é simulada por meio da tática `intro`. Por exemplo, para provarmos uma implicação, digamos:

```
============================
 phi -> psi
```

podemos utilizar a tática `intro` que vai mover o antecedente `phi` da implicação para as hipóteses, reduzindo assim, o problema de provar `phi -> psi` ao problema de provar `psi` assumindo `phi`:

```
H : phi
  ============================
  psi
```

A regra de *eliminação da implicação* ($\to_e$), isto é, a regra *modus ponens* corresponde à tática `apply`. Considere uma aplicação da regra ($\to_e$) como a seguir:

$$\dfrac{\dfrac{}{\varphi \to \psi, \varphi \vdash \varphi \to \psi}\,(\mathrm{Ax}) \quad \dfrac{}{\varphi \to \psi, \varphi \vdash \varphi}\,(\mathrm{Ax})}{\varphi \to \psi, \varphi \vdash \psi}\,(\to_e)$$

A situação correspondente no Rocq é dada a seguir:

```
  H1 : phi -> psi
  H2 : phi
  ============================
  psi
```

Podemos aplicar a tática `apply H1` que vai reduzir a prova de `psi` a uma prova de `phi` já que o consequente da implicação `phi -> psi` coincide com a fórmula que queremos provar. Em seguida, concluímos a prova com a utilização da tática `assumption` porque `phi` é a hipótese `H2`.

[2]Existem diversos canais disponíveis para dúvidas sobre o Rocq como, por exemplo, `https://coq.discourse.group/` e `https://proofassistants.stackexchange.com/`.

Agora vamos refazer aqui os exemplos da seção anterior. Iniciaremos com o $\vdash (p \to q) \to ((q \to r) \to (p \to r))$. Precisamos declarar as variáveis `p`, `q` e `r` e, em seguida enunciarmos o lema a ser provado:

```
Parameter p q r: Prop.

Lemma ex1: (p -> q) -> ((q -> r) -> (p -> r)).
Proof.
```

A janela de prova correspondente é:

```
1 goal (ID 1)

============================
(p -> q) -> (q -> r) -> p -> r
```

Repetindo os passos da prova construída na seção anterior, devemos aplicar 3 vezes a regra $(\to_{\mathrm{i}})$. Na primeira aplicação, a fórmula `p -> q` é colocada no contexto. Em outras palavras, introduzimos a fórmula `p -> q`:

```
Parameter p q r: Prop.

Lemma ex1: (p -> q) -> ((q -> r) -> (p -> r)).
Proof.
  intro H1.
```

```
  1 goal (ID 2)

  H1 : p -> q
  ============================
  (q -> r) -> p -> r
```

Em seguida, introduzimos a fórmula `q -> r`:

```
Parameter p q r: Prop.

Lemma ex1: (p -> q) -> ((q -> r) -> (p -> r)).
Proof.
  intro H1. intro H2.
```

```
 1 goal (ID 3)

 H1 : p -> q
 H2 : q -> r
 ============================
 p -> r
```

E então, introduzimos a fórmula `p`:

```
Parameter p q r: Prop.

Lemma ex1: (p -> q) -> ((q -> r) -> (p -> r)).
Proof.
  intro H1. intro H2. intro H3.
```

```
1 goal (ID 4)

  H1 : p -> q
  H2 : q -> r
  H3 : p
  ============================
  r
```

Alternativamente, podemos compor os 3 passos de introdução de uma única vez usando a variante `intros`:

```
Parameter p q r: Prop.

Lemma ex1: (p -> q) -> ((q -> r) -> (p -> r)).
Proof.
  intros H1 H2 H3.
```

```
1 goal (ID 4)

  H1 : p -> q
  H2 : q -> r
  H3 : p
  ============================
  r
```

Agora podemos provar `r` via a regra ($\rightarrow_e$) utilizando a hipótese `H2` que tem `r` como consequente. Isto corresponde a utilizar a tática `apply H2`, que resulta na obrigação de provar `q`:

```
Parameter p q r: Prop.

Lemma ex1: (p -> q) -> ((q -> r) -> (p -> r)).
Proof.
  intros H1 H2 H3. apply H2.
```

```
1 goal (ID 5)

  H1 : p -> q
  H2 : q -> r
  H3 : p
  ============================
  q
```

Agora utilizamos novamente a regra ($\rightarrow_e$) com a hipótese `H1`, ou seja, aplicamos a tática `apply H1`, e concluímos com `assumption` já que `p` corresponde à hipótese `H3`:

```
Parameter p q r: Prop.

Lemma ex1: (p -> q) -> ((q -> r) -> (p -> r)).
Proof.
  intros H1 H2 H3. apply H2. apply H1.
  assumption.
Qed.
```

Ferramentas de Inteligência Artificial (IA) também podem resolver estes problemas facilmente:

IA

Pergunta: Construa uma prova para o sequente $\vdash (p \to q) \to ((q \to r) \to (p \to r))$ no assistente de provas Rocq.

Resposta:

```
Theorem example : forall (p q r : Prop), (p -> q) -> ((q -> r) -> (p -> r)).
Proof.
 intros p q r.
 intros Hpq Hqr Hp.
 apply Hqr.
 apply Hpq.
 apply Hp.
Qed.
```

2.3 A Lógica Proposicional Minimal

Agora vamos ampliar o poder de expressividade do fragmento implicacional da LP adicionando novos construtores gramaticais e novas regras de inferência. Assim, a nova gramática terá, além da implicação, a constante ($\bot$), a negação ($\neg$), a conjunção ($\wedge$) e a disjunção ($\vee$):

$$\varphi ::= p \mid \bot \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \to \varphi) \tag{2.2}$$

A constante $\bot$ é utilizada para representar a negação: $\neg\varphi$ é o mesmo que $\varphi \to \bot$. Ou seja, temos duas maneiras distintas de escrever a negação, e portanto a gramática acima possui redundâncias. De fato, veremos que existem outras redundâncias na gramática (2.2), mas elas são úteis porque simplificam a escrita das fórmulas.

A gramática (2.2) define as fórmulas da LP, e a partir dela consideraremos 3 sublógicas da LP: a minimal, a intuicionista e a clássica. Nesta seção estudaremos a lógica proposicional minimal (LPM), que assim como no fragmento implicacional visto anteriormente, possui uma regra de introdução e uma regra de eliminação para cada um dos conectivos lógicos. Ou seja, uma regra de introdução e uma de eliminação para cada um dos construtores recursivos da gramática (2.2).

Apesar da gramática apresentada acima não incluir a bi-implicação, este é um conectivo bastante utilizado, e pode ser escrito em função dos outros conectivos: $\varphi \leftrightarrow \psi$ é o mesmo que $(\varphi \to \psi) \wedge (\psi \to \varphi)$.

As regras da negação são análogas às regras da implicação, uma vez que uma negação, digamos $(\neg\varphi)$ é definida como $(\varphi \to \bot)$.

$$\frac{\Gamma, \varphi \vdash \bot}{\Gamma \vdash \neg\varphi}\ (\neg_i) \qquad\qquad \frac{\Gamma_1 \vdash \neg\varphi \qquad \Gamma_2 \vdash \varphi}{\Gamma_1 \cup \Gamma_2 \vdash \bot}\ (\neg_e)$$

Veremos posteriormente que apenas com a negação e implicação podemos expressar todos os outros conectivos apresentados na gramática (2.2), que portanto é uma gramática redundante. No entanto, esta redundância é interessante porque nos permite expressar fórmulas complexas de forma compacta.

Exercício 7. *Seja φ uma fórmula da LP. Prove o sequente $\varphi \vdash \neg\neg\varphi$ na LPM.*

A prova do exercício anterior ocorre com alta frequência como parte de outras provas, e por isto ela será promovida ao *status* de *regra derivada*, ou seja, regras que podem ser provadas a partir das regras básicas:

$$\frac{\Gamma \vdash \varphi}{\Gamma \vdash \neg\neg\varphi}\ (\neg\neg_i)$$

Talvez você esteja esperando agora a derivação da eliminação da dupla negação para se juntar a regra anterior, mas infelizmente isto não é possível neste momento porque o poder de expressividade que temos até agora com as regras $(\to_e)$, $(\to_\mathrm{i})$, $(\neg_\mathrm{i})$ e $(\neg_e)$ não é suficiente para provarmos a eliminação da dupla negação.

Ainda não temos expressividade suficiente para provar uma regra geral de eliminação da dupla negação, mas podemos provar a dupla eliminação de fórmulas negadas. Ou seja, provaremos o seguinte sequente: $\neg\neg\neg\varphi \vdash \neg\varphi$.

$$\dfrac{(\neg\neg_i)\ \dfrac{\dfrac{}{\neg\neg\neg\varphi, \varphi \vdash \varphi}\,(\mathrm{Ax})}{\neg\neg\neg\varphi, \varphi \vdash \neg\neg\varphi} \qquad \dfrac{}{\neg\neg\neg\varphi, \varphi \vdash \neg\neg\neg\varphi}\,(\mathrm{Ax})}{\dfrac{\neg\neg\neg\varphi, \varphi \vdash \bot}{\neg\neg\neg\varphi \vdash \neg\varphi}\,(\neg_i)}\,(\neg_e)$$

Voltaremos a falar da eliminação da dupla negação na seção sobre lógica proposicional clássica.

Considere o sequente $\varphi \to \psi, \neg\psi \vdash \neg\varphi$. Como a fórmula do consequente é uma negação, vamos aplicar a regra de introdução da negação na construção de uma prova de baixo para cima, isto é, da raiz para as folhas da árvore:

$$\dfrac{\dfrac{?}{\varphi \to \psi, \neg\psi, \varphi \vdash \bot}}{\varphi \to \psi, \neg\psi \vdash \neg\varphi}\ (\neg_{\mathrm{i}})$$

Agora, precisamos construir uma prova do absurdo, e portanto podemos tentar utilizar a regra $(\neg_e)$. Para isto precisamos escolher uma fórmula do contexto para fazer o papel de φ da regra 8 da Tabela **??**. A princípio temos três opções: $\varphi \to \psi$, $\neg\psi$ e φ. A boa escolha neste caso é $\neg\psi$ porque podemos facilmente provar ψ a partir deste contexto:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\varphi \to \psi \vdash \varphi \to \psi}\ (\mathrm{Ax}) \quad \dfrac{}{\varphi \vdash \varphi}\ (\mathrm{Ax})}{\varphi \to \psi, \varphi \vdash \psi}\ (\to_e) \qquad \dfrac{}{\neg\psi \vdash \neg\psi}\ (\mathrm{Ax})}{\varphi \to \psi, \neg\psi, \varphi \vdash \bot}\ (\neg_e)}{\varphi \to \psi, \neg\psi \vdash \neg\varphi}\ (\neg_{\mathrm{i}})$$

Depois de concluída a prova é fácil entender o que queríamos dizer com *boa escolha* acima: Uma boa escolha é um caminho que vai nos permitir concluir uma prova. Mas como fazer uma boa escolha? Isto depende do problema a ser resolvido. Em alguns casos pode ser simples, mas em outros, bastante complicado. O ponto importante a compreender é que existem caminhos possíveis distintos na construção de provas da lógica proposicional, e muito deste processo depende da nossa criatividade.

O sequente que acabamos de provar ocorre com certa frequência em outras provas, assim como a regra derivada $(\neg\neg_i)$. As regras que são obtidas a partir das regras da Tabela **??** são chamadas de *regras derivadas*. Este é o caso da regra conhecida como *modus tollens* (MT) obtida a partir do sequente do exemplo anterior, onde cada antecedente é generalizado como uma premissa da regra:

$$\dfrac{\Gamma_1 \vdash \varphi \to \psi \qquad \Gamma_2 \vdash \neg\psi}{\Gamma_1 \cup \Gamma_2 \vdash \neg\varphi}\ (\mathrm{MT})$$

Exercício 8. *Sejam φ e ψ fórmulas da LP. Prove o sequente $\varphi \to \psi \vdash (\neg\neg\varphi) \to (\neg\neg\psi)$ na LPM.*

Exercício 9. *Sejam φ e ψ fórmulas da LP. Prove o sequente $\neg\neg(\varphi \to \psi) \vdash (\neg\neg\varphi) \to (\neg\neg\psi)$ na LPM.*

A *regra de introdução da conjunção*, denotada por $(\wedge_i)$, nos diz o que precisamos fazer para construir uma prova de um sequente que possui uma conjunção na conclusão, isto é, um sequente da forma $\Gamma \vdash \varphi_1 \wedge \varphi_2$, onde Γ é um conjunto finito de fórmulas da LP, e φ_1 e φ_2 são fórmulas da LP. A regra $(\wedge_i)$ é dada pela seguinte regra de inferência:

$$\frac{\Gamma_1 \vdash \varphi_1 \qquad \Gamma_2 \vdash \varphi_2}{\Gamma_1 \cup \Gamma_2 \vdash \varphi_1 \wedge \varphi_2}\ (\wedge_i)$$

ou seja, uma prova de $\Gamma \vdash \varphi_1 \wedge \varphi_2$ é construída a partir de uma prova de $\Gamma \vdash \varphi_1$ e de uma prova de $\Gamma \vdash \varphi_2$.

Existem duas regras de eliminação para a conjunção já que podemos extrair qualquer uma das componentes de uma conjunção:

$$\frac{\Gamma \vdash \varphi_1 \wedge \varphi_2}{\Gamma \vdash \varphi_1}\ (\wedge_{e_1}) \qquad\qquad \frac{\Gamma \vdash \varphi_1 \wedge \varphi_2}{\Gamma \vdash \varphi_2}\ (\wedge_{e_2})$$

Estas duas regras podem ser representadas de forma mais concisa da seguinte forma:

$$\frac{\Gamma \vdash \varphi_1 \wedge \varphi_2}{\Gamma \vdash \varphi_{i\in\{1,2\}}}\ (\wedge_e)$$

Usaremos o nome $(\wedge_e)$ para designar a utilização da regra de eliminação da conjunção quando não quisermos especificar qual das regras $(\wedge_{e_1})$ ou $(\wedge_{e_2})$ foi utilizada.

Com as regras da conjunção já podemos fazer um exercício interessante: provar a comutatividade da conjunção, isto é, queremos construir uma prova para o sequente $\varphi \wedge \psi \vdash \psi \wedge \varphi$, onde φ e ψ são fórmulas quaisquer da LP. A construção da prova é feita inicialmente de baixo para cima com a aplicação da regra $(\wedge_i)$:

$$\frac{\dfrac{?}{\varphi \wedge \psi \vdash \psi} \qquad \dfrac{?}{\varphi \wedge \psi \vdash \varphi}}{\varphi \wedge \psi \vdash \psi \wedge \varphi}\ (\wedge_i)$$

Concluímos com a regra de eliminação da conjunção e o axioma:

$$\frac{\dfrac{\dfrac{}{\varphi \wedge \psi \vdash \varphi \wedge \psi}\ (\text{Ax})}{\varphi \wedge \psi \vdash \psi}\ (\wedge_e) \qquad \dfrac{\dfrac{}{\varphi \wedge \psi \vdash \varphi \wedge \psi}\ (\text{Ax})}{\varphi \wedge \psi \vdash \varphi}\ (\wedge_e)}{\varphi \wedge \psi \vdash \psi \wedge \varphi}\ (\wedge_i)$$

Exercício 10. *Prove que a conjunção é associativa, isto é, prove o sequente* $(\varphi \wedge \psi) \wedge \rho \dashv\vdash \varphi \wedge (\psi \wedge \rho)$ *na LPM, onde* φ, ψ *e* ρ *são fórmulas quaisquer da LP.*

Exercício 11. *Sejam φ e ψ fórmulas quaisquer da LP. Prove os sequentes $\neg\neg(\varphi \wedge \psi) \dashv\vdash (\neg\neg\varphi) \wedge (\neg\neg\psi)$ na LPM.*

Uma vez que uma bi-implicação corresponde a uma conjunção de duas implicações, ela pode ser decomposta com a regra de eliminação da conjunção:

$$\frac{\Gamma \vdash \varphi_1 \leftrightarrow \varphi_2}{\Gamma \vdash \varphi_1 \rightarrow \varphi_2}\ (\wedge_{e_1}) \qquad \frac{\Gamma \vdash \varphi_1 \leftrightarrow \varphi_2}{\Gamma \vdash \varphi_2 \rightarrow \varphi_1}\ (\wedge_{e_2})$$

Vejamos agora as regras para a disjunção. A *regra de introdução da disjunção* nos permite construir a prova de uma disjunção a partir da prova de qualquer uma das suas componentes:

$$\frac{\Gamma \vdash \varphi_1}{\Gamma \vdash \varphi_1 \vee \varphi_2}\ (\vee_{i_1}) \qquad \frac{\Gamma \vdash \varphi_2}{\Gamma \vdash \varphi_1 \vee \varphi_2}\ (\vee_{i_2})$$

Como no caso da regra de eliminação da conjunção podemos representar estas duas regras de forma mais compacta:

$$\frac{\Gamma \vdash \varphi_{i\in\{1,2\}}}{\Gamma \vdash \varphi_1 \vee \varphi_2}\ (\vee_i)$$

A *regra de eliminação da disjunção* nos permite construir a prova de uma fórmula, digamos γ, a partir de uma disjunção. Para isto, precisamos de duas provas distintas de γ, cada uma assumindo uma das componentes da disjunção separadamente:

$$\frac{\Gamma_1 \vdash \varphi_1 \vee \varphi_2 \qquad \Gamma_2, \varphi_1 \vdash \gamma \qquad \Gamma_3, \varphi_2 \vdash \gamma}{\Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \vdash \gamma}\ (\vee_e)$$

Assim, para que tenhamos uma prova de γ a partir de Γ precisamos de uma prova de γ a partir de φ e das fórmulas em Γ e de outra prova de γ a partir de φ_2 e das fórmulas em Γ. Observe como os contextos mudam em cada um dos sequentes que compõem esta regra.

A Tabela abaixo resume todas as regras da LPM, isto é, as regras de introdução e eliminação dos conectivos lógicos apresentados na gramática 2.2.

Regras da Lógica Proposicional Minimal

$$\frac{}{\varphi \vdash \varphi}\ (\text{Ax})$$

Regras de introdução	Regras de eliminação
$\dfrac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi}\ (\to_{\text{i}})$	$\dfrac{\Gamma_1 \vdash \varphi \to \psi \qquad \Gamma_2 \vdash \varphi}{\Gamma_1 \cup \Gamma_2 \vdash \psi}\ (\to_e)$
$\dfrac{\Gamma, \varphi \vdash \bot}{\Gamma \vdash \neg\varphi}\ (\neg_i)$	$\dfrac{\Gamma_1 \vdash \neg\varphi \qquad \Gamma_2 \vdash \varphi}{\Gamma_1 \cup \Gamma_2 \vdash \bot}\ (\neg_e)$
$\dfrac{\Gamma_1 \vdash \varphi_1 \qquad \Gamma_2 \vdash \varphi_2}{\Gamma_1 \cup \Gamma_2 \vdash \varphi_1 \wedge \varphi_2}\ (\wedge_i)$	$\dfrac{\Gamma \vdash \varphi_1 \wedge \varphi_2}{\Gamma \vdash \varphi_{i\in\{1,2\}}}\ (\wedge_e)$
$\dfrac{\Gamma \vdash \varphi_{i\in\{1,2\}}}{\Gamma \vdash \varphi_1 \vee \varphi_2}\ (\vee_i)$	$\dfrac{\Gamma_1 \vdash \varphi_1 \vee \varphi_2 \qquad \Gamma_2, \varphi_1 \vdash \gamma \qquad \Gamma_3, \varphi_2 \vdash \gamma}{\Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \vdash \gamma}\ (\vee_e)$

Vamos mostrar que a disjunção é comutativa, ou seja, queremos construir uma prova para o sequente $\varphi \vee \psi \vdash \psi \vee \varphi$. A ideia aqui é utilizarmos a regra $(\vee_{\text{e}})$. Para isto podemos instanciar Γ com o conjunto unitário contendo a fórmula $\varphi \vee \psi$. Em função da estrutura da regra $(\vee_{\text{e}})$, precisamos construir duas provas distintas de $\psi \vee \varphi$: uma a partir de φ, e outra a partir de ψ. Podemos fazer isto com a ajuda da regra $(\vee_i)$:

$$\dfrac{(\text{Ax})\ \dfrac{}{\varphi \vee \psi \vdash \varphi \vee \psi} \qquad \dfrac{\dfrac{}{\varphi \vdash \varphi}\ (\text{Ax})}{\varphi \vdash \psi \vee \varphi}\ (\vee_i) \qquad \dfrac{\dfrac{}{\psi \vdash \psi}\ (\text{Ax})}{\psi \vdash \psi \vee \varphi}\ (\vee_i)}{\varphi \vee \psi \vdash \psi \vee \varphi}\ (\vee_{\text{e}})$$

Exercício 12. *Sejam φ, ψ e ρ fórmulas quaisquer da LP. Prove que a disjunção é associativa, isto é, prove o sequente $(\varphi \vee \psi) \vee \rho \dashv\vdash \varphi \vee (\psi \vee \rho)$ na LPM.*

Considere o sequente $\varphi \to \psi \vdash \neg\psi \to \neg\varphi$. Inicialmente, devemos observar que a fórmula que queremos provar é uma implicação, e portanto, o mais natural é tentar aplicar a regra $(\to_{\text{i}})$, e em seguida aplicar (MT) (na construção de baixo para cima) para poder completar a prova:

$$\dfrac{\dfrac{(\text{Ax})\ \dfrac{}{\varphi \to \psi \vdash \varphi \to \psi} \qquad \dfrac{}{\neg\psi \vdash \neg\psi}\ (\text{Ax})}{\varphi \to \psi, \neg\psi \vdash \neg\varphi}\ (\text{MT})}{\varphi \to \psi \vdash \neg\psi \to \neg\varphi}\ (\to_{\text{i}})$$

A prova que acabamos de fazer é outro caso que aparece com frequência, e corresponde a uma regra conhecida como *contrapositiva*:

$$\frac{\Gamma \vdash \varphi \to \psi}{\Gamma \vdash \neg\psi \to \neg\varphi}\ (\text{CP})$$

Exercício 13. *Sejam φ e ψ fórmulas da LP. Prove o sequente $\varphi \to \neg\psi \vdash \psi \to \neg\varphi$ na LPM.*

Exercício 14. *Sejam φ e ψ fórmulas da LP. Prove o sequente $\vdash ((((\varphi \to \psi) \to \varphi) \to \varphi) \to \psi) \to \psi$ na LPM.*

Exercício 15. *Sejam φ e ψ fórmulas da LP. Prove o sequente $\varphi, \neg\varphi \vdash \neg\psi$ na LPM.*

Neste exemplo, veremos que é possível fazer a introdução de uma implicação sem precisar descartar uma hipótese, se tivermos uma prova do consequente da implicação que queremos construir. Ou seja, se temos uma prova de ψ então podemos construir uma prova de $\varphi \to \psi$, qualquer que seja a fórmula φ. Em outras palavras, queremos construir uma prova para o sequente $\psi \vdash \varphi \to \psi$:

$$\dfrac{\dfrac{}{\psi, \varphi \vdash \psi}\,(\text{Ax})}{\psi \vdash \varphi \to \psi}\,(\to_{\text{i}})$$

Como este raciocínio aparece com frequência nas provas, vamos colocá-lo como uma regra derivada:

$$\dfrac{\Gamma \vdash \psi}{\Gamma \vdash \varphi \to \psi}\,(\to_i)\ \emptyset$$

Regras derivadas da LPM

$\dfrac{\Gamma \vdash \varphi}{\Gamma \vdash \neg\neg\varphi}\,(\neg\neg_i)$	$\dfrac{\Gamma_1 \vdash \varphi \to \psi \qquad \Gamma_2 \vdash \neg\psi}{\Gamma_1 \cup \Gamma_2 \vdash \neg\varphi}\,(\text{MT})$	$\dfrac{\Gamma \vdash \psi}{\Gamma \vdash \varphi \to \psi}\,(\to_i)\ \emptyset$
$\dfrac{\Gamma \vdash \varphi \to \psi}{\Gamma \vdash \neg\psi \to \neg\varphi}\,(\text{CP})$	$\dfrac{\Gamma \vdash \varphi \to \neg\psi}{\Gamma \vdash \psi \to \neg\varphi}\,(\text{CP})'$	

Exercício 16. *Sejam φ, ψ e γ fórmulas da LP. Construa uma prova para o sequente $\vdash (\varphi \to \psi) \to \varphi \to \gamma \to \psi$ na LPM.*

Exercício 17. *Sejam φ e γ fórmulas da LP. Construa uma prova para o sequente $\varphi, \neg\varphi \vdash \neg\gamma$ na LPM.*

Exercício 18. *Seja φ uma fórmula da LP. Construa uma prova para o sequente $\neg\neg\neg\varphi \vdash \neg\varphi$ na LPM.*

Exercício 19. *Sejam φ e ψ fórmulas da LP. Construa uma prova para o sequente $\neg(\varphi \vee \psi) \vdash (\neg\varphi) \wedge (\neg\psi)$ na LPM.*

Exercício 20. *Sejam φ e ψ fórmulas da LP. Construa uma prova para o sequente $(\neg\varphi) \wedge (\neg\psi) \vdash \neg(\varphi \vee \psi)$ na LPM.*

Exercício 21. *Sejam φ, ψ e δ fórmulas da LP. Construa uma prova para o sequente $\varphi \to \psi \vdash (\delta \vee \varphi) \to (\delta \vee \psi)$ na LPM.*

Exercício 22. *Sejam φ e ψ fórmulas da LP. Construa uma prova para o sequente $\varphi \to \psi \vdash \neg(\varphi \wedge \neg\psi)$ na LPM.*

Exercício 23. *Sejam φ e ψ fórmulas da LP. Construa uma prova para o sequente $\varphi \wedge \psi \vdash \neg(\neg\varphi \vee \neg\psi)$ na LPM.*

Exercício 24. *Sejam φ e γ fórmulas da LP. Construa uma prova para o sequente $(\neg\varphi) \vee (\neg\gamma) \vdash \neg(\varphi \wedge \gamma)$ na LPM.*

Exercício 25. *Sejam φ e γ fórmulas da LP. Construa uma prova para o sequente $\neg\neg(\varphi \wedge \gamma) \vdash (\neg\neg\varphi) \wedge (\neg\neg\gamma)$ na LPM.*

Exercício 26. *Sejam φ e γ fórmulas da LP. Construa uma prova para o sequente $(\neg\neg\varphi) \wedge (\neg\neg\gamma) \vdash \neg\neg(\varphi \wedge \gamma)$ na LPM.*

Exercício 27. *Sejam φ, ψ e γ fórmulas da LP. Prove o sequente $\varphi \vee (\psi \wedge \gamma) \vdash (\varphi \vee \psi) \wedge (\varphi \vee \gamma)$ na LPM.*

Exercício 28. *Sejam φ, ψ e γ fórmulas da LP. Prove o sequente $(\varphi \vee \psi) \wedge (\varphi \vee \gamma) \vdash \varphi \vee (\psi \wedge \gamma)$ na LPM.*

Exercício 29. *Sejam φ, ψ e γ fórmulas da LP. Prove o sequente $\varphi \wedge (\psi \vee \gamma) \vdash (\varphi \wedge \psi) \vee (\varphi \wedge \gamma)$ na LPM.*

Exercício 30. *Sejam φ, ψ e γ fórmulas da LP. Prove o sequente $(\varphi \wedge \psi) \vee (\varphi \wedge \gamma) \vdash \varphi \wedge (\psi \vee \gamma)$ na LPM.*

Exercício 31. *Seja φ uma fórmula da LP. Prove o sequente $\vdash \neg\neg(\varphi \vee \neg\varphi)$ na LPM.*

Exercício 32. *Seja φ uma fórmula da LP. Prove o sequente $\vdash \neg(\varphi \wedge \neg\varphi)$ na LPM.*

Exercício 33. *Sejam φ e γ fórmulas da LP. Construa uma prova para o sequente $(\varphi \to \gamma) \wedge \neg(\varphi \wedge \gamma) \vdash \neg\varphi$ na LPM.*

Exercício 34. *Sejam φ, ψ e γ fórmulas da LP. Prove o sequente $\varphi \leftrightarrow \psi, \psi \to \gamma, \neg\gamma \vdash (\neg\varphi) \wedge (\neg\psi)$ na LPM.*

Exercício 35. *Sejam φ e ψ fórmulas da LP. Construa uma prova para o sequente $\neg\neg(\psi \to \varphi) \vdash (\neg\varphi) \to (\neg\psi)$ na LPM.*

2.4 Dedução Natural em notação padrão

O sistema de dedução natural visto anteriormente utiliza uma notação de sequentes. Nesta notação, o contexto aparece no mesmo nível da fórmula que está sendo provada. A desvantagem desta notação é ter que explicitar o contexto em cada nó da árvore de prova. Nesta seção, vamos apresentar uma notação equivalente à anterior onde o contexto não aparece de forma explícita[HR04, dSdMF06, vD08, AdM17]. Para motivarmos a vantagem da nova notação, considere a prova da associatividade da conjunção, isto é, do sequente $(\phi \wedge \psi) \wedge \varphi \vdash \phi \wedge (\psi \wedge \varphi)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(\phi \wedge \psi) \wedge \varphi \vdash (\phi \wedge \psi) \wedge \varphi}\,(\text{Ax})}{(\phi \wedge \psi) \wedge \varphi \vdash \phi \wedge \psi}\,(\wedge_e)}{(\phi \wedge \psi) \wedge \varphi \vdash \phi}\,(\wedge_e) \qquad \dfrac{\dfrac{\dfrac{}{(\phi \wedge \psi) \wedge \varphi \vdash (\phi \wedge \psi) \wedge \varphi}\,(\text{Ax})}{(\phi \wedge \psi) \wedge \varphi \vdash \phi \wedge \psi}\,(\wedge_e)}{(\phi \wedge \psi) \wedge \varphi \vdash \psi} \quad \dfrac{}{(\phi \wedge \psi) \wedge \varphi \vdash (\phi \wedge \psi) \wedge \varphi}\,(\text{Ax})}{(\phi \wedge \psi) \wedge \varphi \vdash \varphi}\,(\wedge_e)}{(\phi \wedge \psi) \wedge \varphi \vdash (\psi \wedge \varphi)}\,(\wedge_i)}{(\phi \wedge \psi) \wedge \varphi \vdash \phi \wedge (\psi \wedge \varphi)}\,(\wedge_i)$$

Observe que o contexto, isto é, o antecedente de cada um dos sequentes desta prova é o mesmo, e como o que muda ao longo da prova é o consequente dos sequentes, podemos colocar nosso foco na parte que é modificada e remover os contextos da prova deixando-a mais limpa e compacta:

$$\dfrac{\dfrac{\dfrac{(\phi \wedge \psi) \wedge \varphi}{(\phi \wedge \psi)}\,(\wedge_e)}{\phi}\,(\wedge_e) \qquad \dfrac{\dfrac{\dfrac{(\phi \wedge \psi) \wedge \varphi}{\phi \wedge \psi}\,(\wedge_e)}{\psi} \quad \dfrac{(\phi \wedge \psi) \wedge \varphi}{\varphi}\,(\wedge_e)}{(\psi \wedge \varphi)}\,(\wedge_i)}{\phi \wedge (\psi \wedge \varphi)}\,(\wedge_i)$$

A remoção do contexto em provas como a do exemplo anterior é trivial já que ele não muda e é dado no enunciado. Mas será que é possível sempre remover os contextos das provas de uma forma sistemática? Sim! Mas então como lidar com as regras $(\neg_i)$, $(\to_{\mathrm{i}})$ e $(\vee_{\mathrm{e}})$ que mudam o contexto? Vejamos um exemplo:

Considere novamente o sequente $\varphi \to \psi, \neg\psi \vdash \neg\varphi$ que foi provado em um exemplo anterior utilizando a notação de sequentes. Se simplesmente apagarmos os contextos, obtemos a seguinte árvore de derivação:

$$\dfrac{\dfrac{\dfrac{\varphi \to \psi \qquad \varphi}{\psi} \qquad \neg\psi}{\bot}\,(\neg_e)}{\neg\varphi}\,(\neg_{\mathrm{i}})$$

Agora vamos inferir o contexto em cada nó da árvore considerando o comportamento das regras da Tabela **??**. Vamos fazer isto das folhas para a raiz considerando apenas a árvore de derivação acima. Como a árvore possui três folhas, cada uma contendo uma fórmula distinta, o contexto das folhas é o conjunto contendo as três fórmulas, ou seja, o conjunto $\{\varphi \to \psi, \neg\psi, \varphi\}$. Como a regra $(\to_e)$ preserva o mesmo contexto, o contexto da fórmula ψ, na linha 2, é também o conjunto $\{\varphi \to \psi, \neg\psi, \varphi\}$. Da mesma forma, a regra $(\neg_e)$ preserva o contexto, e portanto o contexto da fórmula $\bot$ é também o conjunto $\{\varphi \to \psi, \neg\psi, \varphi\}$. Já a regra $(\neg_{\mathrm{i}})$ adiciona uma fórmula ao contexto (leitura de baixo para cima), ou remove uma fórmula do contexto, se a leitura for feita de cima para baixo. Neste caso, a fórmula removida é φ, e portanto o contexto da fórmula $\neg\varphi$ (raiz da árvore) é o conjunto $\{\varphi \to \psi, \neg\psi\}$ como esperado. Na árvore de derivação acima, não existe nenhuma informação explícita de que a fórmula φ é eliminada do contexto. Podemos inferir isto porque conhecemos o comportamento da regra $(\neg_{\mathrm{i}})$, mas em uma árvore maior onde várias fórmulas sejam removidas do contexto a leitura da árvore de derivação pode se tornar bastante confusa. Para facilitar a leitura de qual fórmula é eliminada, e em qual passo, podemos marcar as fórmulas que são eliminadas:

$$\dfrac{\dfrac{\dfrac{\varphi \to \psi \qquad [\varphi]}{\psi} \qquad \neg\psi}{\bot}\,(\neg_e)}{\neg\varphi}\,(\neg_{\mathrm{i}})$$

e agora fica claro que a fórmula φ não faz parte do contexto original do sequente a ser provado. Note que os colchetes são colocados **apenas nas folhas** que contêm fórmulas que não fazem parte do contexto dado pelo problema. Em outras palavras, os colchetes são utilizados para marcar fórmulas que são hipóteses temporárias, ou seja, que em determinado momento serão removidas do contexto. Em uma situação geral, podem existir diversas fórmulas para serem descartadas, e por regras distintas. Para que fique claro o escopo de cada fórmula, precisamos de um mecanismo para nos informar quando as fórmulas marcadas com os colchetes são **removidas** (ou **descartadas**) do contexto. No exemplo acima, isto ocorre ao aplicarmos a regra $(\neg_{\mathrm{i}})$. Então, utilizaremos um símbolo qualquer para registrar este fato. Neste exemplo utilizamos como símbolo a letra u:

$$\dfrac{\dfrac{\dfrac{\varphi \to \psi \qquad [\varphi]^u}{\psi} \qquad \neg\psi}{\bot}\,(\neg_e)}{\neg\varphi}\,(\neg_i)\ u$$

Agora sabemos que hipótese $[\varphi]^u$ foi descartada durante a aplicação da regra $(\neg_i)\ u$ na árvore de derivação.

A seguir, veremos exemplos mais complexos onde fórmulas idênticas podem exigir marcas distintas, mas antes disto compare as regras de dedução natural para a lógica proposicional minimal com notação de sequente, e com a notação padrão na Tabela **??**. Observe como o mecanismo de descarte simula a mudança de contexto antes e depois de uma aplicação das regras $(\vee_{\mathrm{e}})$, $(\to_{\mathrm{i}})$ e $(\neg_{\mathrm{i}})$. Como a notação padrão (contextos implícitos) é mais compacta, a partir deste momento não utilizaremos mais a notação

com sequentes. A intenção de iniciar este trabalho utilizando a notação com sequentes foi para permitir uma explicação mais fácil e natural para o processo de descarte de hipóteses, que sempre gera muitas dúvidas entre os alunos. Por exemplo, se a razão do descarte não está clara, é comum aparecerem árvores de derivação com descarte de hipóteses feito em regras como $(\wedge_i)$, $(\wedge_e)$, $(\vee_i)$, $(\to_e)$ e $(\neg_e)$. Se você acha que está tudo bem uma árvore de derivação conter descarte de hipóteses nas regras citadas na frase anterior, volte para o início deste capítulo e reinicie o estudo do sistema de dedução natural antes de prosseguir :-)

Regras da LPM

	Notação com sequentes	Notação padrão
1	$\dfrac{\Gamma_1 \vdash \varphi_1 \qquad \Gamma_2 \vdash \varphi_2}{\Gamma_1 \cup \Gamma_2 \vdash \varphi_1 \wedge \varphi_2}\ (\wedge_i)$	$\dfrac{\varphi_1 \qquad \varphi_2}{\varphi_1 \wedge \varphi_2}\ (\wedge_i)$
2	$\dfrac{\Gamma \vdash \varphi_1 \wedge \varphi_2}{\Gamma \vdash \varphi_{i\in\{1,2\}}}\ (\wedge_e)$	$\dfrac{\varphi_1 \wedge \varphi_2}{\varphi_{i\in\{1,2\}}}\ (\wedge_e)$
3	$\dfrac{\Gamma \vdash \varphi_{i\in\{1,2\}}}{\Gamma \vdash \varphi_1 \vee \varphi_2}\ (\vee_i)$	$\dfrac{\varphi_{i\in\{1,2\}}}{\varphi_1 \vee \varphi_2}\ (\vee_i)$
4	$\dfrac{\Gamma \vdash \varphi_1 \vee \varphi_2 \qquad \Gamma, \varphi_1 \vdash \gamma \qquad \Gamma, \varphi_2 \vdash \gamma}{\Gamma \vdash \gamma}\ (\vee_e)$	$\dfrac{\varphi_1 \vee \varphi_2 \qquad \begin{array}{c}[\varphi_1]^u \\ \vdots \\ \gamma\end{array} \qquad \begin{array}{c}[\varphi_2]^v \\ \vdots \\ \gamma\end{array}}{\gamma}\ (\vee_e)\ u, v$
5	$\dfrac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi}\ (\to_i)$	$\dfrac{\begin{array}{c}[\varphi]^u \\ \vdots \\ \psi\end{array}}{\varphi \to \psi}\ (\to_i)\ u$
6	$\dfrac{\Gamma \vdash \varphi \to \psi \qquad \Gamma \vdash \varphi}{\Gamma \vdash \psi}\ (\to_e)$	$\dfrac{\varphi \to \psi \qquad \varphi}{\psi}\ (\to_e)$
7	$\dfrac{\Gamma, \varphi \vdash \bot}{\Gamma \vdash \neg\varphi}\ (\neg_i)$	$\dfrac{\begin{array}{c}[\varphi]^u \\ \vdots \\ \bot\end{array}}{\neg\varphi}\ (\neg_i)\ u$
8	$\dfrac{\Gamma \vdash \neg\varphi \qquad \Gamma \vdash \varphi}{\Gamma \vdash \bot}\ (\neg_e)$	$\dfrac{\neg\varphi \qquad \varphi}{\bot}\ (\neg_e)$

A partir deste ponto, utilizaremos apenas a notação padrão.

2.5 A Lógica Proposicional Intuicionista

A Lógica Proposicional Intuicionista (LPI) tem como sistema dedutivo as regras da LPM juntamente com a regra de eliminação do absurdo intuicionista, também conhecida como regra da explosão:

$$\frac{\bot}{\varphi}\ (\bot_e)$$

Assim, as provas em dedução natural na LPI são construídas utilizando as regras da tabela abaixo:

Regras derivadas da LPI

1	$\dfrac{\varphi_1 \qquad \varphi_2}{\varphi_1 \wedge \varphi_2}\ (\wedge_i)$
2	$\dfrac{\varphi_1 \wedge \varphi_2}{\varphi_{i\in\{1,2\}}}\ (\wedge_e)$
3	$\dfrac{\varphi_{i\in\{1,2\}}}{\varphi_1 \vee \varphi_2}\ (\vee_i)$
4	$\dfrac{\varphi_1 \vee \varphi_2 \qquad \begin{array}{c}[\varphi_1]^u \\ \vdots \\ \gamma\end{array} \qquad \begin{array}{c}[\varphi_2]^v \\ \vdots \\ \gamma\end{array}}{\gamma}\ (\vee_e)\ u, v$
5	$\dfrac{\begin{array}{c}[\varphi]^u \\ \vdots \\ \psi\end{array}}{\varphi \to \psi}\ (\to_i)\ u$
6	$\dfrac{\varphi \to \psi \qquad \varphi}{\psi}\ (\to_e)$
7	$\dfrac{\begin{array}{c}[\varphi]^u \\ \vdots \\ \bot\end{array}}{\neg\varphi}\ (\neg_i)\ u$
8	$\dfrac{\neg\varphi \qquad \varphi}{\bot}\ (\neg_e)$
9	$\dfrac{\bot}{\varphi}\ (\bot_e)$

Exercício 36. *Sejam φ e ψ fórmulas da LP. Construa uma prova para o sequente $(\neg\varphi) \vee \psi \vdash \varphi \to \psi$ na LPI.*

Exercício 37. *Seja φ uma fórmula da LP. Construa uma prova para o sequente $\vdash \neg\neg(\neg\neg\varphi \to \varphi)$ na LPI.*

Exercício 38. *Seja φ uma fórmula da LP. Construa uma prova para o sequente $\vdash \neg\neg((\neg\varphi) \vee \varphi)$ na LPI.*

Exercício 39. *Sejam φ e ψ fórmulas da LP. Construa uma prova para o sequente $(\neg\neg\varphi) \to (\neg\neg\psi) \vdash \neg\neg(\varphi \to \psi)$ na LPI.*

Exercício 40. *Sejam φ e ψ fórmulas da LP. Construa uma prova para o sequente $\vdash \neg\neg(((\varphi \to \psi) \to \varphi) \to \varphi)$ na LPI.*

2.6 A Lógica Proposicional Clássica

Vamos caminhar na direção de mais uma extensão, agora da LPI para a Lógica Proposicional Clássica (LPC). Iniciamos com a lógica proposicional minimal, depois a estendemos para a lógica proposicional intuicionista, e agora vamos estendê-la com a regra da eliminação do absurdo clássico, também conhecida como prova por contradição, obtendo assim a LPC. A tabela abaixo apresenta as regras da LPC:

Regras da Lógica Clássica

1	$\dfrac{\varphi_1 \qquad \varphi_2}{\varphi_1 \wedge \varphi_2}\ (\wedge_i)$
2	$\dfrac{\varphi_1 \wedge \varphi_2}{\varphi_{i\in\{1,2\}}}\ (\wedge_e)$
3	$\dfrac{\varphi_{i\in\{1,2\}}}{\varphi_1 \vee \varphi_2}\ (\vee_i)$
4	$\dfrac{\varphi_1 \vee \varphi_2 \qquad \begin{array}{c}[\varphi_1]^u \\ \vdots \\ \gamma\end{array} \qquad \begin{array}{c}[\varphi_2]^v \\ \vdots \\ \gamma\end{array}}{\gamma}\ (\vee_e)\ u, v$
5	$\dfrac{\begin{array}{c}[\varphi]^u \\ \vdots \\ \psi\end{array}}{\varphi \to \psi}\ (\to_i)\ u$
6	$\dfrac{\varphi \to \psi \qquad \varphi}{\psi}\ (\to_e)$
7	$\dfrac{\begin{array}{c}[\varphi]^u \\ \vdots \\ \bot\end{array}}{\neg\varphi}\ (\neg_i)\ u$
8	$\dfrac{\neg\varphi \qquad \varphi}{\bot}\ (\neg_e)$
9	$\dfrac{\bot}{\varphi}\ (\bot_e)$
10	$\dfrac{\begin{array}{c}[\neg\varphi]^u \\ \vdots \\ \bot\end{array}}{\varphi}\ (PBC)\ u$

Exercício 41. *Mostre que a regra* $(\bot_e)$ *é uma regra derivada na LPC. Ou seja, ela pode ser provada com as regras da LPM juntamente com (PBC).*

Exercício 42. *Acabamos de caracterizar a LPC como sendo a LPI juntamente com a regra de prova por contradição (PBC), mas outras caracterizações são possíveis. De fato, podemos adicionar qualquer das regras a seguir à LPI para obter a LPC:*

$$\frac{\neg\neg\varphi}{\varphi}\ (\neg\neg_e) \qquad \frac{}{\varphi \vee (\neg\varphi)}\ (\text{LEM}) \qquad \frac{}{((\varphi \to \psi) \to \varphi) \to \varphi}\ (\text{LP})$$

Para justificar o argumento acima, mostre que as regras (PBC), $(\neg\neg_e)$, (LEM) *e* (LP) *são equivalentes.*

Exercício 43. *Sejam φ e ψ fórmulas da LP. Prove $\varphi \wedge \psi \dashv\vdash \neg(\neg\varphi \vee \neg\psi)$.*

Exercício 44. *Sejam φ e ψ fórmulas da LP. Prove $\varphi \to \psi \dashv\vdash (\neg\varphi) \vee \psi$.*

Exercício 45. *Sejam φ e ψ fórmulas da LP. Prove o sequente $(\neg\varphi) \to \psi \vdash (\neg\psi) \to \varphi$.*

Exercício 46. *Sejam φ e ψ fórmulas quaisquer da LP. Prove os sequentes $\neg\neg(\varphi \vee \psi) \dashv\vdash (\neg\neg\varphi) \vee (\neg\neg\psi)$.*

Exercício 47. $\varphi \leftrightarrow \neg\varphi \vdash \bot$

Exercício 48. $(\neg\varphi) \to (\neg\psi) \vdash \neg\neg(\psi \to \varphi)$

Exercício 49. *Sejam φ e ψ fórmulas quaisquer da LP. Mostre que o sequente $\varphi \to \psi \vdash (\neg\varphi) \vee \psi$ não possui uma derivação intuicionista.*

Exercício 50. *Sejam φ e ψ fórmulas quaisquer da LP. Mostre que o sequente $\neg((\neg\varphi) \wedge (\neg\psi)) \vdash \varphi \vee \psi$ não possui uma derivação intuicionista.*

Exercício 51. *Sejam φ e ψ fórmulas quaisquer da LP. Mostre que o sequente $\neg((\neg\varphi) \vee (\neg\psi)) \vdash \varphi \wedge \psi$ não possui uma derivação intuicionista.*

2.6.1 A Semântica da LPC

Considere o seguinte problema:

Em uma ilha moram apenas dois tipos de pessoas: as honestas, que sempre falam a verdade; e as desonestas, que sempre mentem. Um viajante, ao passar por esta ilha encontra três moradores chamados A, B e C. O viajante pergunta para o morador A:
"Você é honesto ou desonesto?" A responde algo incompreensível, e o viajante pergunta para B: "O que ele disse?" B então responde "Ele disse que é desonesto". Neste momento C se manifesta: "Não acredito nisto! Isto é uma mentira!". Questão: C é honesto ou desonesto?

Para resolver este problema pense no que ocorre se um morador desta ilha, digamos X, disser "Eu sou desonesto"? Isto nos levaria a uma contradição! De fato, se X for honesto então ele disse a verdade, e portanto é desonesto. Por outro lado, se X é desonesto então ele mentiu, e portanto é honesto. Assim, como A não poderia ter dito que é desonesto, podemos concluir que B é desonesto! E portanto, C é honesto! Vamos construir uma prova de que este raciocínio está em correto usando a teoria que estudamos? O ponto de partida é construir um sequente que corresponda ao enunciado deste problema. Que variáveis proposicionais vamos precisar? Certamente precisamos de variáveis que nos permitam caracterizar quando um morador é ou não honesto. Assim, utilizaremos três variáveis proposicionais com a seguinte semântica:

- a: o morador A é honesto;
- b: o morador B é honesto;
- c: o morador C é honesto.

Desta forma, a negação de qualquer destas variáveis significa que o morador correspondente é desonesto. Agora precisamos representar o que foi dito por cada um dos moradores por meio de uma fórmula da lógica proposicional. Considere o que disse o morador B: "Ele disse que é desonesto", quer dizer, o morador B disse que o morador A disse que era desonesto. Como codificar este fato por meio de uma fórmula da LP? Vamos iniciar considerando uma situação geral e mais simples. Digamos que um morador X tenha dito Y, isto é, "X disse Y". Que fórmula da LP corresponde a este fato? Suponha que a variável x codifica a proposição "X é honesto". Então observe que, se X for honesto então o que ele disse é verdade, ou seja, tanto x quanto Y são verdade. Por outro lado, se X for desonesto então Y é falso, e tanto x quanto Y são falsos. Assim, podemos concluir que as variáveis x e Y são equivalentes, no sentido que ou ambas são verdadeiras, ou ambas são falsas. Assim, podemos representar a afirmação "X disse Y"pela fórmula $x \leftrightarrow Y$. Voltando então ao nosso problema original, podemos agora representar o fato de que o morador B disse que o morador A disse que era desonesto pela fórmula $b \leftrightarrow (a \leftrightarrow (\neg a))$. O morador C por sua vez, disse que B mentiu, o que corresponde a fórmula $c \leftrightarrow (\neg b)$. Com isto podemos montar o sequente a ser provado: $b \leftrightarrow (a \leftrightarrow (\neg a)), c \leftrightarrow (\neg b) \vdash c$.

Exercício 52. *Prove o sequente $b \leftrightarrow (a \leftrightarrow (\neg a)), c \leftrightarrow (\neg b) \vdash c$ construído no exemplo anterior.*

Exercício 53. *Considere uma ilha onde moram apenas dois tipos de pessoas: as honestas, e que portanto sempre falam a verdade; e as desonestas, que sempre mentem. Um viajante, ao passar por esta ilha encontra três moradores chamados A, B e C. O viajante pergunta para o morador A: "Quantos, dentre vocês três, são desonestos?" A responde algo incompreensível, e o viajante pergunta para B: "O que ele disse?" B então responde "Ele disse que exatamente dois de nós somos desonestos". Neste momento C se manifesta: "Não acredito nisto! Isto é uma mentira!". Questão: C é honesto ou desonesto?*

Exercício 54. *Em uma ilha moram apenas dois tipos de habitantes: os honestos, que sempre falam a verdade; e os desonestos, que sempre mentem. Você encontra dois habitantes desta ilha, digamos João e José. João diz que José é desonesto. José diz "Nem João nem eu somos desonestos". Você consegue determinar qual dos dois é honesto e qual é desonesto?*

No exemplo anterior, utilizamos a associação do valor de verdade (verdadeiro ou falso) de uma variável proposicional para resolver um problema. Esta abordagem está relacionado com a semântica da lógica proposicional clássica que nos fornece os meios para concluir quando uma fórmula é verdadeira ou falsa. A gramática

$$\varphi ::= p \mid \bot \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \to \varphi) \tag{2.3}$$

define como são as fórmulas da LP, a partir de seis construtores:

1. O primeiro denota uma variável proposicional, e caracteriza uma fórmula atômica, i.e. uma fórmula que não pode ser subdividida em uma fórmula menor.

2. O segundo construtor é uma constante que denota o absurdo ($\bot$), que também é uma fórmula atômica. O absurdo é utilizado para representar uma fórmula que tem valor de verdade "falso (F)". É importante observar que podemos associar a qualquer fórmula da LP apenas dois valores de verdade, a saber: verdadeiro (T) ou falso (F).

3. O terceiro construtor denota a negação e nos permite construir uma nova fórmula a partir de uma fórmula dada. Assim, dada uma fórmula φ, podemos construir a sua negação $(\neg\varphi)$. A semântica da negação é a que conhecemos intuitivamente: se uma fórmula φ é verdadeira (T) então sua negação é falsa (F), e vice-versa. Normalmente, representamos este fato via a seguinte tabela:

φ	$(\neg\varphi)$
T	F
F	T

4. O quarto construtor denota a conjunção e nos permite construir uma nova fórmula a partir de duas fórmulas dadas. Assim, dadas duas fórmulas φ_1 e φ_2, podemos construir a sua conjunção $(\varphi_1 \wedge \varphi_2)$. A semântica da conjunção também é a usual, isto é, a conjunção $(\varphi_1 \wedge \varphi_2)$ é verdadeira somente quando φ_1 e φ_2 são simultaneamente verdadeiras:

φ_1	φ_2	$(\varphi_1 \wedge \varphi_2)$
T	T	T
T	F	F
F	T	F
F	F	F

Aqui é importante observar que a leitura da construção da conjunção na gramática 2.2 não diz que suas componentes são iguais (apesar da utilização do mesmo símbolo φ nas duas componenetes). Lembre-se que a leitura desta construção em 2.2 é: dadas duas fórmulas (não necessariamente iguais!), podemos construir a sua conjunção. Alternativamente, poderíamos ter escrito a gramática 2.2 da seguinte forma equivalente:

$$\varphi, \psi ::= p \mid \bot \mid (\neg\varphi) \mid (\varphi \wedge \psi) \mid (\varphi \vee \psi) \mid (\varphi \to \psi) \tag{2.4}$$

5. O quinto construtor denota a disjunção e, como no caso anterior, nos permite construir uma nova fórmula a partir de duas fórmulas dadas. Assim, dadas duas fórmulas φ_1 e φ_2, podemos construir a sua disjunção $(\varphi_1 \vee \varphi_2)$, cuja semântica é dual à semântica da conjunção: a disjunção $(\varphi_1 \vee \varphi_2)$ é falsa somente quando φ_1 e φ_2 são simultaneamente falsas.

φ_1	φ_2	$(\varphi_1 \vee \varphi_2)$
T	T	T
T	F	T
F	T	T
F	F	F

6. O sexto construtor é a implicação. Assim, dadas duas fórmulas φ_1 e φ_2, podemos construir a sua implicação $(\varphi_1 \to \varphi_2)$ com a semântica dada na tabela abaixo.

φ_1	φ_2	$(\varphi_1 \to \varphi_2)$
T	T	T
T	F	F
F	T	T
F	F	T

O sentido usual da implicação assume implicitamente uma relação de causa e efeito, ou causa e consequência no sentido de que o antecedente φ_1 é o que gera o consequente φ_2 como em "Se eu não beber água então ficarei desidratado". No entanto, o sentido da implicação na lógica é um pouco diferente pois tem como fundamento a *preservação da verdade*, que não necessariamente possui uma relação de causa e efeito. Por exemplo, a proposição "Se 2+2=4 então o dia tem 24 horas"é verdadeira, mas não existe relação causal entre a igualdade 2+2=4 e o fato de o dia ter 24 horas de duração.

Uma gramática como 2.2 (ou 2.4) nos fornece as regras sintáticas para a construção das fórmulas da LP. São quatro construtores recursivos (negação, conjunção, disjunção e implicação) também chamados de conectivos lógicos, e dois não recursivos.

Apesar da gramática apresentada acima não incluir a bi-implicação, este é um conectivo bastante utilizado. De fato, a bi-implicação, já utilizada em exemplos anteriores, pode ser reescrita em usando a implicação e conjunção. Como exercício construa a tabela verdade da bi-implicação e observe que $\varphi \leftrightarrow \psi$ é verdadeira somente quando φ e ψ possuem o mesmo valor de verdade. Adicionalmente, dizemos que duas fórmulas φ e ψ são **equivalentes** quando a fórmula $\varphi \leftrightarrow \psi$ é uma tautologia:

Tautologia	Uma fórmula que é sempre verdadeira, independentemente dos valores de verdade associados às suas variáveis.
Contradição	Uma fórmula que é sempre falsa, independentemente dos valores de verdade associados às suas variáveis.
Contingência	Uma fórmula que pode ser tanto verdadeira quanto falsa dependendo dos valores de verdade associados às suas variáveis.

As tautologias e as contradições são particularmente importantes, e possuem símbolos especiais para representá-las. Nas gramáticas 2.2 e 2.4 já vimos que a constante $\bot$ é o representante das contradições. As tautologias, por sua vez, podem ser representadas pelo símbolo $\top$.

Agora chegamos em um momento chave do nosso estudo. Considere um sequente arbitrário, digamos $\Gamma \vdash \varphi$, onde Γ é um conjunto finito de fórmulas da LP. Podemos então perguntar: é possível provar este sequente? Ou em outras palavras, qualquer sequente possui uma prova? A resposta certamente é não. Se tudo pudesse ser provado então não teríamos razão para estudar a lógica proposicional. Como então é possível separar os sequentes que têm prova dos que não podem ser provados? Para responder esta pergunta precisamos inicialmente compreender a noção de **consequência lógica**. Dizemos que uma fórmula φ é consequência lógica da fórmula ψ, notação $\psi \models \varphi$, se φ for verdadeira sempre que ψ for verdadeira. Este conceito pode ser facilmente estendido para um conjunto Γ de fórmulas, de forma que $\Gamma \models \varphi$, isto é, φ é consequência lógica do conjunto Γ se φ for verdadeira sempre que as fórmulas em Γ forem verdadeiras. Agora podemos enunciar dois teoremas importantes que nos permitirão responder à questão anterior:

Teorema 1 (Correção da LP)**.** *Sejam Γ um conjunto, e φ uma fórmula da lógica proposicional. Se $\Gamma \vdash \varphi$ então $\Gamma \models \varphi$.*

A prova deste teorema é por indução em $\Gamma \vdash \varphi$. Não detalharemos aqui esta prova, que pode ser encontrada por exemplo em [AdM17].

Teorema 2 (Completude da LP)**.** *Sejam Γ um conjunto, e φ uma fórmula da lógica proposicional. Se $\Gamma \models \varphi$ então $\Gamma \vdash \varphi$.*

A prova do teorema de completude da LP é um pouco mais complexa do que a prova de correção, e também pode ser encontrada em [AdM17]. Note que este lema responde a nossa pergunta anterior: um sequente tem prova exatamente quando seu consequente for consequência lógica do seu antecedente.

Referências Bibliográficas

[ADGR07] Jeremy Avigad, Kevin Donnelly, David Gray, and Paul Raff. A formally verified proof of the prime number theorem. *ACM Transactions on Computational Logic*, 9(1):2–es, December 2007.

[AdM17] M. Ayala-Rincón and F. L. C. de Moura. *Applied Logic for Computer Scientists - Computational Deduction and Formal Proofs*. UTCS. Springer, 2017.

[AH14] Jeremy Avigad and John Harrison. Formally verified mathematics. *Communications of the ACM*, 57(4):66–75, April 2014.

[APJ17] Emilio Jesús Gallego Arias, Benoît Pin, and Pierre Jouvelot. jsCoq: Towards Hybrid Theorem Proving Interfaces. *Electronic Proceedings in Theoretical Computer Science*, 239:15–27, January 2017.

[Bar84] H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. Number v. 103 in Studies in Logic and the Foundations of Mathematics. North-Holland ; Sole distributors for the U.S.A. and Canada, Elsevier Science Pub. Co, Amsterdam ; New York : New York, N.Y, rev. ed edition, 1984.

[Chl17] A. Chlipala. *Certified Programming with Dependent Types*. MIT Press, 2017.

[Chu36] A. Church. An Unsolvable Problem of Elementary Number Theory. *American Journal of Mathematics*, 58(2):345–363, 1936.

[Chu40] A. Church. A Formulation of the Simple Theory of Types. *journal of Symbolic Logic*, 5:56–68, 1940.

[dMU21] Leonardo de Moura and Sebastian Ullrich. The Lean 4 Theorem Prover and Programming Language. In André Platzer and Geoff Sutcliffe, editors, *Automated Deduction – CADE 28*, Lecture Notes in Computer Science, pages 625–635, Cham, 2021. Springer International Publishing.

[dSdMF06] F. S. C. da Silva, A. C. V. de Melo, and M. Finger. *Lógica Para Computação*. THOMSON PIONEIRA, 2006.

[GAA+13] Georges Gonthier, Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, François Garillot, Stéphane Le Roux, Assia Mahboubi, Russell O'Connor, Sidi Ould Biha, Ioana Pasca, Laurence Rideau, Alexey Solovyev, Enrico Tassi, and Laurent Théry. A Machine-Checked Proof of the Odd Order Theorem. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *Interactive Theorem Proving*, volume 7998, pages 163–179. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[Gon08] G. Gonthier. A computer-checked proof of the Four Colour Theorem. Technical report, Microsoft Research Cambridge, 2008.

[HAB+15] T. Hales, M. Adams, G. Bauer, D. Tat Dang, J. Harrison, T. Le Hoang, C. Kaliszyk, V. Magron, S. McLaughlin, T. Tat Nguyen, T. Quang Nguyen, T. Nipkow, S. Obua, J. Pleso, J. Rute, A. Solovyev, A. Hoai Thi Ta, T. N. Tran, D. Thi Trieu, J. Urban, K. Khac Vu, and R. Zumkeller. A formal proof of the Kepler conjecture. *ArXiv e-prints*, January 2015.

[Hal05] Thomas Hales. A Proof of the Kepler Conjecture. *Annals of Mathematics*, 162(3):1065–1185, 2005.

[Hin97] J. Roger Hindley. *Basic Simple Type Theory*. Number 42 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1997.

[HR04] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning About Systems*. Cambridge University Press, New York, NY, USA, 2004.

[KSUV15] Cezary Kaliszyk, Stephan Schulz, Josef Urban, and Jiří Vyskočil. System Description: E.T. 0.1. In Amy P. Felty and Aart Middeldorp, editors, *Automated Deduction - CADE-25*, volume 9195, pages 389–398. Springer International Publishing, Cham, 2015.

[Ler09] Xavier Leroy. Formal Verification of a Realistic Compiler. *Communications of the ACM*, 52(7):107, 2009.

[McC10] W. McCune. Prover9 and mace4. http://www.cs.unm.edu/~mccune/prover9/, 2005/2010.

[NNdMA10] R. B. Nogueira, A. C. A. Nascimento, F. L. C. de Moura, and M. Ayala-Rincón. Formalization of Security Proofs Using PVS in the Dolev-Yao Model. In *Booklet Proc. Computability in Europe - CiE*, 2010.

[NPW02] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lncs*. Springer, 2002.

[ORS92] S. Owre, J. M. Rushby, and N. Shankar. PVS: A Prototype Verification System. In D. Kapur, editor, *CADE*, volume 607 of *Lnai*, pages 748–752. sv, 1992.

[Pau14] C. Paulin-Mohring. Introduction to the Calculus of Inductive Constructions. 2014.

[Pau15] Lawrence C. Paulson. A Mechanised Proof of Gödel's Incompleteness Theorems Using Nominal Isabelle. *J Autom Reasoning*, 55(1):1–37, 2015.

[PCG+14] Benjamin C. Pierce, Chris Casinghino, Marco Gaboardi, Michael Greenberg, Catvalin Hriatcu, Vilhelm Sjoberg, and Brent Yorgey. *Software Foundations*. Electronic textbook, 2014.

[RV02] Alexandre Riazanov and Andrei Voronkov. The design and implementation of VAMPIRE. *AI Commun.*, 15(2-3):91–110, 2002.

[Smu09] Raymond Smullyan. *Logical Labyrinths*. AK Peters, 2009.

[Tea21] The Coq Development Team. The Coq Proof Assistant. Zenodo, October 2021.

[vD08] D. van Dalen. *Logic and Structure (4. Ed.)*. Universitext. Springer, 2008.

[vD13] D. van Dalen. *Logic and Structure*. Universitext. Springer London, 2013.