Indução de indemonstrabilidades e independências em Complexidade Computacional

Claus Akira Matsushigue

TESE APRESENTADA AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA DA
UNIVERSIDADE DE SÃO PAULO PARA
OBTENÇÃO DO GRAU DE DOUTOR EM
MATEMÁTICA

Orientador: **Prof. Dr. Ricardo Bianconi**

Na elaboração deste trabalho, o autor obteve apoio financeiro da FAPESP.

— São Paulo, SP — Janeiro de 2003 —

Indução de indemonstrabilidades e independências em Complexidade Computacional

Claus Akira Matsushigue

Este exemplar corresponde à redação final da tese, devidamente corrigida, defendida por Claus Akira Matsushigue e aprovada pela comissão julgadora.

São Paulo, 13 de março de 2003.

Banca examinadora:

- Prof. Dr. Ricardo Bianconi (orientador) — MAT-IME-USP
- Prof. Dr. Francisco Miraglia Netto — MAT-IME-USP
- Prof. Dr. Ruy José Guerra Barretto de Queiroz — CIn-UFPE
- Prof. Dr. Walter Alexandre Carnielli — CLE-IFCH-UNICAMP
- Profª. Drª. Itala Maria Loffredo D'Ottaviano — CLE-IFCH-UNICAMP

Abstract

We obtain that each level (Σ_n^{P} or Π_n^{P}, with $n < \omega$) and the whole polynomial hierarchy ($\mathcal{PH}$) are Σ_3–complete. We also work with statements about time and space in non–deterministic Turing Machines and prove that some of them are Π_1– and Σ_2–complete.

These facts provide peculiar independence results in theories T which are only axiomatizable and which can codify this kind of statement. In particular, there exists a decision problem L belonging to Σ_n^{P}, or $\mathcal{PH}$, for which exactly this membership relation is independent of T. On the other hand, there exists a Turing Machine which has polynomial non–deterministic time, however the truth of this statement is also independent of T.

We even prove that we can obtain some of these independent elements in an effective/constructible way. For this, we have to demonstrate stronger versions of some theorems of Recursion Theory, such as the Universal Turing Machine, the Parameterization (or Smn, or Indexer) and the Fixed Point Theorems, taking into account bounds with respect to non–deterministic time and/or space.

Afterwards we present in a more objective way the study of the capacity of formalizing the Computational Complexity Theory problems into Formal Logic Systems.

In fact, given an axiomatizable theory T, we obtain, roughly speaking, that, for all decision problems L, there exists a way to interpret L into T for which the statement "L (in this interpretation) belongs to a level (Σ_n^{P} or Π_n^{P}, with $n < \omega$) —or the whole ($\mathcal{PH}$)— polynomial hierarchy" is independent of T.

On the other hand, for all decision problems L, there exist Turing Machines M and N (and interpretations of both in T) such that the decision problem accepted by M and N is L and the statements "M has polynomial non–deterministic time" and "N does not have polynomial non–deterministic time" are both unprovable in T.

Finally, some consequences involving the $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$ problem could be extracted from these results. We obtained unprovabilities concerning this problem from several points of view. In this way, we basically pass to discuss how small would be the capacity of Formal Logic Systems (viz., axiomatizable theories of Classical First Order Logic) to express correctly the Computational Complexity matters.

Resumo

Obtemos que cada nível (Σ_n^{P} ou Π_n^{P}, com $n < \omega$) e a hierarquia polinomial completa ($\mathcal{PH}$) são Σ_3–completas. Também trabalhamos com asserções sobre tempo e espaço de Máquinas de Turing não–determinísticas e provamos que algumas delas são Π_1– e Σ_2–completas.

Esses fatos fornecem peculiares resultados de independência em teorias T que são apenas axiomatizáveis e que conseguem codificar este tipo de asserção. Em particular, existe um problema de decisão L pertencente a Σ_n^{P}, ou $\mathcal{PH}$, para o qual exatamente essa relação de pertinência é independente de T. Por outro lado, existe uma Máquina de Turing que tem tempo não–determinístico polinomial, entretanto a veracidade dessa asserção também é independente de T.

Ainda provamos que podemos obter alguns desses elementos independentes de um modo efetivo/construtível. Para isso, necessitamos demonstrar versões fortes de alguns teoremas da Teoria da Recursão, tais como os Teoremas da Máquina de Turing Universal, da Parametrização (ou Smn, ou Indexador) e do Ponto Fixo, levando em conta limites de tempo e/ou espaço não–determinísticos.

Depois apresentamos de um modo mais objetivo o estudo da capacidade de se formalizar problemas da Teoria da Complexidade Computacional em Sistemas Lógicos Formais.

Com efeito, dada uma teoria axiomatizável T, obtemos, grosso modo, que, para todos os problemas de decisão L, existe um modo de interpretar L em T de tal modo que a asserção "L (nessa interpretação) pertence a um nível da hierarquia polinomial (Σ_n^{P} ou Π_n^{P}, com $n < \omega$) —ou à hierarquia inteira ($\mathcal{PH}$)" é independente de T.

Por outro lado, para todos os problemas de decisão L, existem Máquinas de Turing M e N (e interpretações de ambas em T) tais que o problema de decisão aceito por M e N é L e as duas asserções "M tem tempo não–determinístico polinomial" e "N não tem tempo não–determinístico polinomial" são indemonstráveis em T.

Finalmente, algumas conseqüências envolvendo o problema $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$ puderam ser extraídas desses resultados. Obtivemos indemonstrabilidades referentes a esse problema em diversos pontos de vista. Desse modo, basicamente passamos a discutir o quão pequena poderia ser a capacidade de Sistemas Lógicos Formais (viz., teorias axiomatizáveis da Lógica Clássica de Primeira Ordem) para expressar corretamente as questões da Complexidade Computacional.

À Fátima,

Ligbia

e

Karin

e

aoſ amigoſ do peito.

Quando se tem um problema,
e não se sabe (absolutamente) o quê fazer,
o que se faz? Matemática!

(Paulo Augusto Veloso, 2003)

Conteúdo

Capítulo I

Introdução

I.1 Contextualização

A Teoria da Complexidade Computacional começa a tomar corpo nos anos 70, impulsionada pela informatização crescente dos nossos tempos. Formulada assim nas inúmeras aplicações, a pergunta $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$ torna-se um dos principais focos dos estudos teóricos da Ciência da Computação.

O problema é decidir se a classe $\mathcal{P}$ —dos problemas computáveis deterministicamente em tempo polinomial— é igual à classe $\mathcal{NP}$ —dos problemas computáveis não-deterministicamente em tempo polinomial. Um problema correlato, mas também importante, é se $\mathcal{NP}$ é igual à classe co-$\mathcal{NP}$ —dos problemas cujos complementos estão em $\mathcal{NP}$—, o qual, se for respondido negativamente, implica que $\mathcal{P} \neq \mathcal{NP}$.

Perceba que, pela simplicidade na sua formulação, não se tinha, no início, noção que tal problema mostrar-se-ia de tão difícil solução pelos métodos da Análise de Algoritmos e da Combinatória até hoje utilizados.

Procurando dar uma visão mais global da área Complexidade Computacional Estrutural sabemos, sob provas triviais, que

$$\mathcal{L} \subseteq \mathcal{NL} = \text{co-}\mathcal{NL} \subseteq \mathcal{P} \subseteq (\mathcal{NP} \cap \text{co-}\mathcal{NP}) \subseteq \mathcal{NP} \subseteq \mathcal{PH} \subseteq \mathcal{PSPACE} = \mathcal{NPSPACE} \subseteq \mathcal{EXP} \subseteq \mathcal{NEXP}$$

(para maiores detalhes, vide Papadimitriou [Pap94]). Entretanto, também com provas simples, temos que $\mathcal{L} \neq \mathcal{PSPACE}$ e $\mathcal{P} \neq \mathcal{EXP}$, mas mais nada se sabe sobre quais dessas inclusões são próprias, apesar de a maioria delas serem amplamente conjecturadas na Teoria da Computação.

Em todos estes anos, tivemos muitos tipos diferentes de abordagens para tentar resolver esses problemas. As definições das classes de complexidade baseiam-se nas Máquinas de Turing e, assim sendo, as primeiras abordagens se davam por argumentos da Teoria da Recursão: como em

um critério de contagem, como nas manipulações sobre as propriedades da Máquina de Turing e, principalmente, procurando-se aplicar o Corte de Dedekind. Com a definição da classe de complexidade $\mathcal{NP}$–completa feita por Cook em 1971, que deu consistência ao problema $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$, cresceu o enfoque sobre problemas combinatórios.

Entretanto, essas abordagens diretas se mostram ineficazes. Procurando novas saídas, nos últimos anos caminhou-se na direção da complexidade dos circuitos booleanos. Não obstante, as dificuldades encontradas não foram muito diferentes das já existentes. Outro enfoque foi a expansão da capacidade computacional das Máquinas de Turing, como, por exemplo, utilizando-se de oráculos, ou permitindo-se a manipulação em uma fita de probabilidade, havendo ou não iteração entre computações.

Seguindo-se essa última linha, um dos mais fecundos resultados foi alcançado em [ALM+98] com a demonstração de $\mathcal{NP} = \mathcal{PCP}(\log n, 1)$. Isto significa que os problemas de decisão de $\mathcal{NP}$ são aqueles que, surpreendentemente, podem ser resolvidos, com erro probabilístico tão pequeno quanto se queira, por computações de tempo polinomial e que sabem verificar a correteza de testemunhas, necessitando para isso apenas de um número logarítmico no tamanho da entrada de bits aleatórios e ler um número constante de letras da testemunha. Estudamos, no mestrado [Mat97], tal teorema, detalhando toda a demonstração, que se mostrou de grande tecnicalidade e dificuldade. Mas a única conseqüência relevante até hoje obtida sobre esse resultado foi em Otimização Combinatória.

Mais recentemente, tem crescido a abordagem lógica. A Teoria da Recursão é um dos ramos da Lógica e assim pode-se dizer que está se voltando ao início, entretanto, por dois outros ramos: sobre modelos finitos e Aritmética de Peano.

O primeiro ramo é bem novo e é chamado de Teoria da Complexidade Descritiva. Mas, apesar de poder se tornar bem promissor, esse ramo ainda não possui um cabedal técnico grande. Por outro lado, os trabalhos com a Aritmética de Peano remontam mais de 80 anos e há pouco, sobre ela, surgiram profícuas (acreditamos) caracterizações de algumas classes de complexidade. Os fragmentos da Aritmética de Peano, por serem básicos, arrolam na Lógica sobre diversas vertentes, como nos Teoremas de Incompleteza de Gödel, nos modelos não–*standard*, na Teoria da Prova e na Teoria da Recursão.

Por outro lado, uma possibilidade plausível é que o problema $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$ seja independente de Teorias dos Conjuntos usuais. O objetivo dos nossos estudos é fornecer subsídios para a solução direta de tal problema —e correlatos—, mas também não podemos deixar de ter em mente essa realidade. Com isso em vista, evitamos caracterizações sobre Sistemas Lógicos Não–clássicos, ou, pelo menos, que não sejam interpretáveis em alguma Teoria dos Conjuntos.

Buscamos assim, no que for possível, aproximar a Complexidade Computacional dessas ferramentas lógicas, via Aritmética de Peano. Aqui surgem duas linhas gerais de enfoque do problema, sendo uma pela análise de demonstrações sintáticas (ou Teoria da Prova) e outra pela análise de modelos não–*standard* (Teoria dos Modelos).

A expressão *Aritmética de Peano*, PA refere-se ao Sistema Lógico Formal —na linguagem de primeira ordem contendo os símbolos $+, \cdot, '$ (sucessor), $\leq$ e 0—, com a axiomatização usual sobre os símbolos e com o esquema do princípio da indução finita para todas as fórmulas nessa linguagem. Já os *Fragmentos da Aritmética de Peano* são aqueles em que fazemos restrições ao esquema da

indução, acrescendo-se à linguagem, se necessário, novos símbolos e um número finito de axiomas. Como exemplo, se restringimos o esquema de indução às fórmulas que só tenham quantificadores limitados, o fragmento é chamado de *Aritmética de Peano Limitada*, $I\Delta_0$.

Começando pelo enfoque da Teoria da Prova, Wilkie [Wil80] prova que, se pudermos formalizar a demonstração da insolubilidade do Décimo Problema de Hilbert dentro da Aritmética de Peano Limitada, então vale a igualdade $\mathcal{NP} = \text{co-}\mathcal{NP}$. Jones e Matijasevič [JM91] fornecem uma prova mais moderna da negativa do Décimo Problema de Hilbert, ou seja, da indecidibilidade —via uma computação qualquer, tomando-se válida a *Tese de Church–Turing*— da existência de solução de equações diofantinas. Além disso, eles reconhecem que fazem uma tentativa incompleta de "embutir" tal prova na Aritmética de Peano Limitada, intencionando valerem-se do resultado de Wilkie.

Entretanto, talvez ainda esperançoso na resolução rápida dos problemas da Complexidade Computacional, Wilkie em [Wil80] propunha a sua aplicação no estudo da definibilidade na Aritmética de Peano Limitada e, assim sendo, poderia ter pretendido utilizar o resultado acima na contra–positiva. Como agora queremos o contrário, surgem então os seguintes problemas: o quanto da demonstração em [JM91] pode ser formalizada na Aritmética de Peano Limitada? Será que podemos enfraquecer a hipótese a ponto de, além da necessidade, tenhamos também a suficiência, ou seja, uma condição de provabilidade da insolubilidade do Décimo Problema de Hilbert em um Fragmento da Aritmética de Peano que seja equivalente ao problema $\mathcal{NP} \stackrel{?}{=} \text{co-}\mathcal{NP}$?

O resultado de Wilkie é uma decorrência simples de técnicas da Teoria da Prova e começa com a caracterização, por um resultado de Manders [Man80], da classe de complexidade $\mathcal{NP}$ pela classe dos Σ_1–conjuntos —i.e., os subconjuntos de $\omega^{<\omega}$ que são representáveis em $\mathbb{N}$ por fórmulas da Aritmética de Peano que só tenham quantificadores que sejam limitados ou que sejam existenciais, mas que apareçam positivamente na fórmula. Visto isso, é imediata a caracterização da classe co-$\mathcal{NP}$ pelos Π_1–conjuntos —com definição análoga aos Σ_1–conjuntos, trocando-se os quantificadores existenciais por universais e vice-versa.

Refinando tal técnica, Buss [Bus86] caracteriza, desta vez por Fragmentos da Aritmética de Peano, a Hierarquia Polinomial ($\mathcal{PH}$) e os seus níveis (Σ_n^p e Π_n^p), incluindo aí $\mathcal{P}$, $\mathcal{NP}$ e co-$\mathcal{NP}$. Também caracteriza as classes $\mathcal{PSPACE}$ —dos problemas computáveis deterministicamente em espaço polinomial— e $\mathcal{EXP}$ —em tempo exponencial—, mas agora na Aritmética de Peano em segunda ordem. As classes para baixo de $\mathcal{P}$, como $\mathcal{L}$ —em espaço logarítmico—, foram mais recentemente caracterizadas por Clote e Takeuti [CT95], com mais resultados em [Tak95].

Já Wilkie e Paris conseguem caracterizar $\mathcal{PH}$ pela Aritmética de Peano Limitada e fornecem um estudo mais aprofundado da indução limitada, $I\Delta_0$, em [WP87]. Assim, como na caracterização de Buss, também conseguiram estabelecer a equivalência entre o colapso na hierarquia de $\mathcal{PH}$ e serem finitamente axiomatizáveis os Fragmentos da Aritmética de Peano que a caracterizam. Agora [KPT91] e [Bus95] obtêm mais resultados relacionando o colapso nos diversos níveis da hierarquia polinomial com igualdades entre Fragmentos da Aritmética de Peano e ainda trazem abordagens sobre Fragmentos da Aritmética de Peano que não são finitamente axiomatizáveis.

Outro enfoque é via a Teoria dos Modelos da Aritmética, que foi iniciada e fortemente incentivada por Máté, em [Mát90]. Ele dá o exemplo do *forcing*, que foi introduzido por Cohen e em que, para se chegar aos seus vastos resultados, foi fundamental que se provasse que o "método" poderia ser totalmente traduzido sintaticamente. Mas não podemos imaginar o *forcing* como algo

diferente de uma extensão de modelos e portanto não tendo uma abordagem semântica, seja na sua concepção inicial, seja nas suas aplicações posteriores. Ou seja, trabalhar com modelos muitas vezes torna as técnicas mais visíveis e naturais, mesmo que pudéssemos obter os mesmos resultados sintaticamente.

Entretanto, nesse enfoque, não se têm mais caracterizadas as classes de complexidade, mas sim alguns dos seus problemas isoladamente. Por exemplo, Máté [Mát90] mostra que a existência de dois modelos não–*standard* específicos da Aritmética Verdadeira —i.e., de TA, a teoria completa de todas as fórmulas de primeira ordem que são satisfeitas pelo modelo *standard* $\mathbb{N}$— implicam que $\mathcal{NP} \neq$ co-$\mathcal{NP}$. Já Sureson [Sur95] completa a caracterização, mostrando a volta da implicação. Ainda Sureson [Sur96] retorna ao assunto, obtendo resultados mais fortes, desta vez sobre a existência de modelos não–*standard* de Fragmentos da Aritmética de Peano, que são bem menos restritos que a Aritmética Verdadeira TA.

Outra técnica lógica que tem se mostrando bastante promissora na obtenção de resultados de independência é o *forcing*, por exemplo Takeuti e Yasumoto [TY96, TY98]. Entretanto, ainda há muito que se fazer para se alcançar bons resultados nessa linha.

No início, propúnhamos caracterizar questões da Teoria da Complexidade Computacional que nos auxiliassem a, ou resolvê-las diretamente, ou —se possível e preferencialmente— mostrá-las independentes de Sistemas Lógicos Formais importantes. Desse modo, não só apresentaremos diretamente resultados de independência, como mostraremos a dificuldade de se caracterizar internamente aos Sistemas Lógicos certas noções da Complexidade Computacional.

I.2 Apresentação

A tese é baseada em dois artigos. Escrevemo-los durante o doutorado, sob supervisão do orientador, Ricardo Bianconi, e estão sendo submetidos conjuntamente. São eles:

a) *Independence Results via Arithmetical Complexity Hierarchy and Complexity Recursion Theory* [MB03a] e

b) *Imposing Fixed Unprovabilities and Independences of Complexity Theory Matters in a Formal Logic System* [MB03b].

No nosso trabalho, obtivemos diversos resultados de indemonstrabilidade e independência concernentes a questões da Teoria da Complexidade Computacional. Nosso objetivo é discutir a possibilidade (e também a impossibilidade) de se formalizar tais questões em Sistemas Lógicos (Formais). Ou seja, a capacidade de se "internalizar" questões da Complexidade Computacional em teorias da Lógica.

Podemos dividir o trabalho em três tópicos principais, que estão assim dispostos no Capítulo II *Descrição dos Resultados*:

(i) Seção II.1 *Hierarquia da Complexidade Aritmética;*

(ii) Seção II.4 *Teoria da Recursão com Complexidade;* e

(iii) os resultados de indemonstrabilidade e independência:

- Seção II.2 *Certificação e Imposição de uma Indemonstrabilidade/Independência,*
- Seção II.5 *Imposição Efetiva de uma Indemonstrabilidade* e
- Seção II.6 *Imposição de Indemonstrabilidades Fixadas.*

Os dois primeiros tópicos são totalmente independentes entre si, assim como do terceiro. Já o terceiro é conseqüência dos dois anteriores.

O primeiro tópico remete-se a provar novas classificações de questões da Complexidade Computacional em completeza/dificuldade na Hierarquia da Complexidade Aritmética. Tais questões são afirmações indexadas por uma variável que percorre os números naturais ω. Imediatamente dessas provas, podemos deduzir diversos resultados de indemonstrabilidade e independência (Seção II.2 *Certificação e Imposição de uma Indemonstrabilidade/Independência*).

Já no segundo tópico, necessitamos (re-)demonstrar os teoremas clássicos da Teoria da Recursão em um formato mais geral, limitando não–deterministicamente o tempo e o espaço das computações. Com isso, obtivemos de modo direto resultados de indemonstrabilidade —sem usar as propriedades de completeza/dificuldade do tópico anterior— e ainda fornecemos explicitamente (ou efetivamente) cada elemento indemonstrável (Seção II.5 *Imposição Efetiva de uma Indemonstrabilidade*).

No último tópico, analisamos diversos modos de se exprimir —i.e., *internalizar*— tais afirmações dentro de uma teoria. Depois fornecemos diversos critérios de indemonstrabilidade e independência, provando-os para algumas questões e deixando em aberto para outras.

Agora não mais como uma conseqüência tão imediata dos dois tópicos anteriores, mas sim com uma mescla deles —valendo-se das técnicas de classificação na Hierarquia da Complexidade Computacional, mas utilizando fortemente os resultados da Teoria da Recursão com Complexidade—, obtivemos resultados de indemonstrabilidade e independência bem mais interessantes (Seção II.6 *Imposição de Indemonstrabilidades Fixadas*).

I.2–1. A Hierarquia da Complexidade Aritmética é definida sobre a Álgebra Booleana de todos os subconjuntos dos números naturais ω ordenados pela relação de inclusão $\subseteq$. Ela separa asserções unárias de ω —i.e., relações unárias $R \subseteq \omega$— em níveis de conjuntos Σ_n– e Π_n–completos/difíceis.

Nessa hierarquia, os níveis mais baixos são dos conjuntos *recursivos* (os Δ_1–conjuntos) e *recursivamente enumeráveis* (os Σ_1–conjuntos). Logo os conjuntos Π_1–difíceis (ou Π_1–completos) *não* são recursivamente enumeráveis. Por outro lado, as asserções unárias de ω expressáveis em alguma teoria axiomatizável são conjuntos recursivamente enumeráveis. Isso nos mostra que as teorias axiomatizáveis não conseguem expressar conjuntos de níveis mais elevados (Π_1–difíceis) da Hierarquia da Complexidade Aritmética.

Nós provamos que cada nível Σ_n^{p} e Π_n^{p} da hierarquia polinomial é Σ_3–completa, além de a hierarquia completa $\mathcal{PH}$ e a classe do espaço não–determinístico logarítmico $\mathcal{NL}$ (Corolário II.1–2). Também mostramos que a computação de tempo não–determinístico polinomial de Máquinas de Turing é Σ_2–completa (Teorema II.1–3) e a computabilidade com limitação não–determinística de espaço por uma função fixada é Π_1–completo (Teorema II.1–4). Nossos resultados são extensões em direção a computações e classes de complexidade não–determinísticas, incluindo limites de espaço, dos resultados obtidos por P. Hájek [Háj77, Háj79], que, por sua vez, seguiram a idéia introduzida por J. Hartmanis e J. E. Hopcroft [HH76].

Hájek já havia provado que a questão de se saber se uma Máquina de Turing roda *deterministicamente* em tempo limitado por uma função fixa é Π_1–completa. Quando perguntamos sobre tempo *deterministicamente* limitado por algum (bom) conjunto "existencial" de funções (e.g., o conjunto de todos polinômios, ou funções exponenciais) em vez de uma única função, temos uma questão Σ_2–completa. Hájek também havia mostrado que classes de complexidade definidas sobre tempo *determinístico* limitado por esses conjuntos de funções são Σ_3–completas.

Desde que os resultados acima não levam em conta computações *não–determinísticas*, eles omitem abordagens referentes a questões muito importantes da Teoria da Complexidade Computacional, tais como $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$, $\mathcal{L} \stackrel{?}{=} \mathcal{NL}$ e o colapso da hierarquia polinomial.

Em vista disso, nós desejávamos tratar os casos não–determinísticos que aparecem nas questões acima, incluindo a hierarquia polinomial completa e todos os tipos de limites de espaço. Claramente esses fatos não poderiam ser surpresa, porque, se $\mathcal{NP}$ ou $\mathcal{PH}$ não fossem Σ_3–completos, teríamos $\mathcal{P} \neq \mathcal{NP}$; mas é um tanto quanto esperado que a classificação na Hierarquia da Complexidade Aritmética é algo mais simples do que o problema real $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$.

Ainda nas provas de Hájek, não há problema em se verificar que classes determinísticas com limite de espaço —incluindo as classes limitadas em espaço logaritmicamente $\mathcal{L}$ e polinomialmente $\mathcal{PSPACE}$— também são Σ_3–completas. Logo, nesse ponto, os casos não–determinísticos são os mais relevantes.[i] Na verdade, nós provamos mais do que isso. Existe um intervalo (um *gap*) tal que todas as classes de complexidade entre duas específicas —em que a primeira é uma classe de complexidade menor do que a classe de espaço logarítmico $\mathcal{L}$ e a segunda pode ser maior do que a classe de tempo exponencial $\mathcal{EXP}$— são Σ_3–difíceis.

I.2–2. Agora, como conseqüência dessas novas propriedades, nós podemos também obter diversos resultados de indemonstrabilidade e independência referentes a questões da Teoria da Complexidade Computacional. Expliquemos isso melhor.

Fixaremos uma enumeração de todos os problemas de decisão importantes $L_0, L_1, \ldots$ (os aceitáveis). Essa enumeração será assumida no decorrer de toda essa introdução e é feita tomando-se uma codificação das Máquinas de Turing, que será melhor descrita posteriormente.

Considere uma asserção, tal como "$L \in \mathcal{NP}$" (com L sendo uma variável percorrendo os problemas de decisão aceitáveis). Então desejamos que uma teoria T possa se referir, internamente a si própria, a essa asserção. Existem quatro modos básicos de se fazer isso: representação, expressividade, certificação, ou certificação fraca —os dois últimos são de nossa denominação. Grosso modo, precisamos ter uma fórmula $\ulcorner L \in \mathcal{NP} \urcorner_T := \varphi(L)$ na linguagem de T tal que:

[i] Embora a mesma simplicidade não ocorra com o limite de espaço nas questões de Π_1– e Σ_2–completeza.

- na representação:
$$\begin{array}{rcl} L \in \mathcal{NP} & \Rightarrow & T \vdash \ulcorner L \in \mathcal{NP}^{\urcorner T} \\ L \notin \mathcal{NP} & \Rightarrow & T \vdash \neg \ulcorner L \in \mathcal{NP}^{\urcorner T}; \end{array}$$

- na expressividade:
$$L \in \mathcal{NP} \quad \Leftrightarrow \quad T \vdash \ulcorner L \in \mathcal{NP}^{\urcorner T};$$

- na certificação:
$$\begin{array}{rcl} T \vdash \ulcorner L \in \mathcal{NP}^{\urcorner T} & \Rightarrow & L \in \mathcal{NP} \\ T \vdash \neg \ulcorner L \in \mathcal{NP}^{\urcorner T} & \Rightarrow & L \notin \mathcal{NP}; \end{array}$$

- na certificação fraca:
$$T \vdash \ulcorner L \in \mathcal{NP}^{\urcorner T} \quad \Rightarrow \quad L \in \mathcal{NP}.$$

A expressividade e a representação em T são propriedades que foram utilizadas por Gödel para formalizar parte da aritmética dentro da (Teoria da) Aritmética de Peano, PA. Nos seus famosos teoremas, foi necessário internalizar asserções recursivas —ou recursivamente enumeráveis— dentro de uma teoria axiomatizável.

Entretanto, quando T estende uma teoria fraca como a Aritmética de Robinson, $\mathrm{PA}^{-}(\subseteq \mathrm{PA})$, basicamente podemos *representar* somente conjuntos recursivos e *expressar* conjuntos recursivamente enumeráveis. Assim, esse não é o melhor modo de se imergir conjuntos de níveis mais elevados da Hierarquia da Complexidade Aritmética —que é o caso geral na Complexidade Computacional, como visto aqui— nessas teorias.

Portanto, baseado no ponto de vista formalista, buscamos o sentido oposto dessas implicações. A *certificação (fraca)* necessita garantir que, quando uma propriedade pode ser provada dentro de uma teoria, ela também vale fora desta teoria, i.e., olhamos a teoria como um espelho parcial do "mundo real".

Precisamos, mais ainda, levar em conta que a representação força um completeza da fórmula $\ulcorner L \in \mathcal{NP}^{\urcorner T}$ —isso é, para todo L, ou $T \vdash \ulcorner L \in \mathcal{NP}^{\urcorner T}$, ou $T \vdash \ulcorner L \notin \mathcal{NP}^{\urcorner T}$. Nesse caso, com o requisito óbvio de T ser consistente, temos que

$$\mathcal{NP} \;=\; \{L\colon T \vdash \ulcorner L \in \mathcal{NP}^{\urcorner T}\} \;=\; \{L\colon T \not\vdash \ulcorner L \notin \mathcal{NP}^{\urcorner T}\}.$$

E a expressividade nos diz que $\mathcal{NP} = \{L\colon T \vdash \ulcorner L \in \mathcal{NP}^{\urcorner T}\}$. Assim a *certificação fraca* relaxa essa noção e exige somente que $\{L\colon T \vdash \ulcorner L \in \mathcal{NP}^{\urcorner T}\} \subseteq \mathcal{NP}$. Já a *certificação* exige que

$$\begin{array}{rcl} \{L\colon T \vdash \ulcorner L \in \mathcal{NP}^{\urcorner T}\} & \subseteq & \mathcal{NP} \\ \{L\colon T \vdash \ulcorner L \notin \mathcal{NP}^{\urcorner T}\} & \subseteq & \mathcal{NP}^{\mathrm{c}}. \end{array} \qquad \text{(I.2:1)}$$

Agora observamos que, desse modo, a existência de uma certificação (ou de uma certificação fraca) de uma asserção implica a consistência da respectiva teoria. Por causa disso, todos os nossos resultados somente se referem às certificações (fracas), não confirmando se elas existem.

Por outro lado, como notado por P. Hájek [Háj79], para teorias axiomatizáveis T, $\{L\colon T \vdash \ulcorner L \in \mathcal{NP}^{\urcorner T}\}$ e $\{L\colon T \vdash \ulcorner L \notin \mathcal{NP}^{\urcorner T}\}$ são Σ_1–conjuntos, i.e., são conjuntos recursivamente enumeráveis. Entretanto, como $\mathcal{NP}$ é Σ_3–completo (Corolário II.1–2), as relações de inclusão na Equação (I.2:1) precisam ser estritas.

E é exatamente isso que queremos dizer por: a asserção "$L \in \mathcal{NP}$" *impõe uma independência em* T (Teorema II.2–1.(iii)). A saber, existe um problema de decisão L em $\mathcal{NP}$ para o qual a asserção

"$L \in \mathcal{NP}$" é independente de uma teoria axiomatizável fixa, em que a questão "pertencer à $\mathcal{NP}$" pode ser interpretada.

As mesmas propriedades de independência podem ser obtidas para todas as classes de complexidade no intervalo mencionado no Teorema II.1–1, tais como as classes $\mathcal{NL}$, $\mathcal{P}$, $\mathcal{PH}$, etc..

I.2–3. Posteriormente produzimos outras e diretas provas do Teorema II.2–1 —a parte da indemonstrabilidade dos Itens (i) e (ii)—, sem passar pela Π_1–completeza do Teorema II.1–4 e pela Σ_2–completeza do Teorema II.1–3. Além disso, obtemos a existência dos elementos indemonstráveis nesses itens de um *modo efetivo*.

Dada uma teoria axiomatizável T, provamos que podemos efetivamente achar uma Máquina de Turing M (ou seu índice de codificação) que tenha espaço não–determinístico logarítmico, mas a asserção "M tem espaço não–determinístico logarítmico" é indemonstrável em T (Teorema II.5–1). Também existe efetivamente uma Máquina de Turing M tal que ela roda em tempo não–determinístico f —em que f é uma função recursiva de ordem quadrática— e a asserção "M roda em tempo não–determinístico f" é indemonstrável em T (Teorema II.5–2). Em particular, a teoria T é consistente com a asserção "M *não* roda em tempo não–determinístico f".

Esse estudo foi iniciado por J. Hartmanis e J. E. Hopcroft [HH76]. As suas provas não usam o tema da completeza discutido acima e, na verdade, elas foram feitas anteriormente às de Hájek. Um resultado diz que, dada uma teoria axiomatizável T, podemos efetivamente achar uma Máquina de Turing M que roda em tempo determinístico n^2 e a asserção "M roda em tempo determinístico $< 2^n$" é independente de T.

Mas, nessa prova, a teoria T necessita provar internamente a ela uma verdade da Complexidade Computacional que é externa. Esse requisito para T parece não ser muito natural. Entretanto, os autores observaram que essa hipótese não é essencial, desde que provássemos os principais teoremas da Teoria da Recursão em uma versão mais forte. Nós fizemos isso de um modo mais específico ainda, o que nos permitiu completar os seus resultados de independência na direção de se levar em conta limitantes não–determinísticos de tempo e espaço.

Portanto precisamos prover o Teorema da Máquina de Turing Universal e da Parametrização —ou Smn, ou Indexador— (II.4–1) com convenientes limitações não–determinísticas de tempo e espaço. Intercalando-os, pudemos obter o Teorema do Ponto Fixo II.4–2. Dada uma Máquina de Turing M, podemos efetivamente tomar um número natural e tal que M, tendo e fixado na primeira fita de entrada, computa tal qual M_e —a e-ésima Máquina de Turing indexada, i.e., e é o índice de codificação de M_e. É isso que chamamos de Teoria da Recursão com Complexidade.

I.2–4. Lembremos da enumeração L_0, L_1, ... Havíamos definido quando uma asserção —tal como "$L_i \in \mathcal{NP}$" (com $i < \omega$)— impõe uma indemonstrabilidade (ou uma independência) em uma teoria. Agora estendemos essa definição dizendo quando uma asserção *impõe indemonstrabilidades (e independências) fixadas* em uma teoria.

Baseados em idéias de Grant [Gra80], mostramos que a asserção "$L_i \in \mathcal{P}$" satisfaz essa nova definição e generalizamos isso para muitas outras importantes classes da Complexidade Computacional, tais como $\mathcal{L}$, $\mathcal{NP}$, co-$\mathcal{NP}$, $\mathcal{PH}$, $\mathcal{PSPACE}$, $\mathcal{EXP}$ e $\mathcal{NEXP}$. Deixe-nos explicar a diferença dessas duas definições.

Sejam a asserção unária "$L_x \in \mathcal{NP}$" e uma teoria T. Se ${}^{\ulcorner}L_x \in \mathcal{NP}^{\neg T}$ é uma das suas certificações fracas em T (em que x é a única variável livre), defina

$$NP := \{i < \omega \parallel L_i \in \mathcal{NP}\}$$

e

$$I_{\mathcal{NP}} := \{i < \omega \parallel T \vdash {}^{\ulcorner}L_i \in \mathcal{NP}^{\neg T}\}.$$

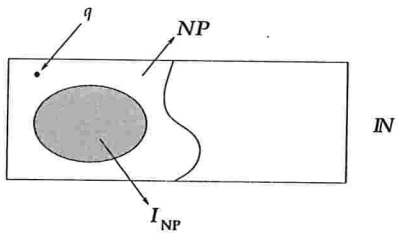
Portanto $I_{\mathcal{NP}} \subseteq NP$. Para uma certificação ${}^{\ulcorner}L_x \in \mathcal{NP}^{\neg T}$ em T, também defina

$$I_{\neg\mathcal{NP}} := \{i < \omega \parallel T \vdash {}^{\ulcorner}L_i \notin \mathcal{NP}^{\neg T}\},$$

assim $I_{\neg\mathcal{NP}} \subseteq NP^{c} := \mathbb{N} \setminus NP$. Então dizer que "$L_x \in \mathcal{NP}$" impõe uma indemonstrabilidade em T (Teorema II.2–1.(iii)) significa que, para toda certificação fraca ${}^{\ulcorner}L_x \in \mathcal{NP}^{\neg T}$ em T, existe

$$q \in NP \setminus I_{\mathcal{NP}} \neq \varnothing.$$

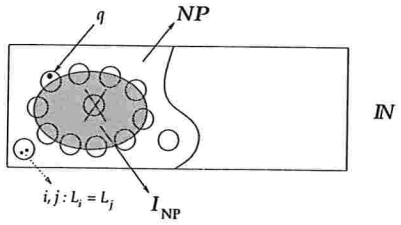
Veja a primeira figura.



Por outro lado, tome a partição do conjunto dos números naturais $\mathbb{N}$ em classes de equivalência de números que codificam o mesmo problema de decisão (i.e., $i \sim j$ sse[ii] $L_i = L_j$).

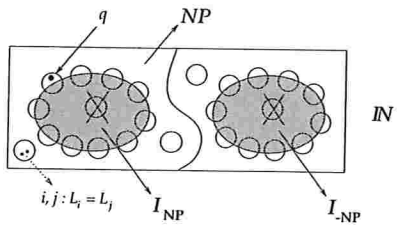
Logo dizer que "$L_x \in \mathcal{NP}$" impõe indemonstrabilidades fixadas em T (Corolário II.6–3) significa que, fixada uma certificação fraca ${}^{\ulcorner}L_x \in \mathcal{NP}^{\neg T}$ em T, existe ao menos um $q < \omega$ em cada classe de equivalência tal que ${}^{\ulcorner}L_q \in \mathcal{NP}^{\neg T}$ seja indemonstrável em T. Ou seja, para cada problema de decisão aceitável L —pode-se imaginar $L \in \mathcal{NP}$, e.g., o problema do caixeiro viajante,— existe $q < \omega$ tal que $L_q = L$ e ${}^{\ulcorner}L_q \in \mathcal{NP}^{\neg T}$ é indemonstrável em T.

As classes de equivalência também particionam o conjunto NP. Portanto $NP \setminus I_{\mathcal{NP}}$ não só não é um conjunto vazio, como tem intersecção com cada classe de equivalência em NP, assim como descrito na segunda figura.



Por sua vez, dizer que "$L_x \in \mathcal{NP}$" impõe independências fixadas em T (novamente Corolário II.6–3) significa que, fixada uma certificação ${}^{\ulcorner}L_x \in \mathcal{NP}^{\neg T}$ em T, existe ao menos um $q < \omega$ em cada classe de equivalência tal que ${}^{\ulcorner}L_q \in \mathcal{NP}^{\neg T}$ seja independente de T. Veja a última figura.

[ii] No decorrer de todo o texto, estaremos sempre utilizando a denotação "sse" para designar "se, e somente se,".



Ou seja, para essa afirmação, não só conseguimos um problema de decisão que a torne independente de T, como também podemos fixar um problema de decisão qualquer e, ainda assim, conseguimos que um dos seus índices torne essa mesma afirmação independente de T.

I.2–5. Ao final, utilizando os resultados referentes a impor uma independência (especificamente Teorema II.2–1.(iii)) e a impor independências fixadas (Corolário II.6–3), nós obtemos algumas conseqüências relativas aos Sistemas Lógicos Formais.

Esses sistemas devemos entender como sendo baseados em teorias axiomatizáveis da Lógica Clássica de Primeira Ordem. De modo geral, estamos pensando nas Teorias (Clássicas) dos Conjuntos —ou na Aritmética de Peano.

Mostramos que sempre temos uma maneira *honesta* de descrever a asserção $\mathcal{P} = \mathcal{NP}$ que a torna indemonstrável em um Sistema Lógico Formal. Tomando-se T uma teoria axiomatizável e $\overline{{}^{\ulcorner}\mathcal{P} = \mathcal{NP}^{\urcorner T}}$ uma (Π_4–)sentença da linguagem de T que expressa $\mathcal{P} = \mathcal{NP}$ —baseado-se nas asserções $L \in \mathcal{P}$ e $L \in \mathcal{NP}$—, podemos modificá-la um pouco para produzir uma outra sentença ${}^{\ulcorner}\mathcal{P} = \mathcal{NP}^{\urcorner T}$ que também descreve $\mathcal{P} = \mathcal{NP}$ tão bem quanto a anterior, mas ${}^{\ulcorner}\mathcal{P} = \mathcal{NP}^{\urcorner T}$ é indemonstrável em T. Note que não estamos afirmando se $\overline{{}^{\ulcorner}\mathcal{P} = \mathcal{NP}^{\urcorner T}}$ é, ou não, indemonstrável em T.

Também mostramos que, do mesmo modo, há maneiras honestas de modificar uma sentença que descreve a simples questão $\mathcal{P} \subseteq \mathcal{NP}$, tornado-a também indemonstrável em T. Entretanto isso parece bem estranho, porque $\mathcal{P} \subseteq \mathcal{NP}$ é facilmente demonstrável na metateoria. Novamente temos que não há um modo satisfatório de axiomatizar tal questão dentro da teoria T.

De outro lado, se de fato vale $\mathcal{P} \neq \mathcal{NP}$, também argumentamos que ${}^{\ulcorner}\mathcal{P} \neq \mathcal{NP}^{\urcorner T}$ é demonstrável em T sem utilizar um *passo não–construtivo* somente se existe $e < \omega$ tal que $T \vdash {}^{\ulcorner}L_e \in \mathcal{NP} \setminus \mathcal{P}^{\urcorner T}$. Entretanto essa última existência é bem improvável por causa da classificação Σ_3–completa de $\mathcal{P}$ e $\mathcal{NP}$ na Hierarquia da Complexidade Aritmética (Corolário II.1–2).

Em algum sentido, temos a intenção de discutir a capacidade de um Sistema Lógico Formal expressar sintaticamente —e dentro de si mesmo— questões da Complexidade tal como o faz semanticamente —i.e., fora de si, na metateoria.

Quem sabe Sistemas Lógicos Formais não sejam suficientes para manipular as definições da Teoria da Complexidade Computacional. Pense que esses sistemas podem ser incapazes de expressar as propriedades que distinguem as classes de complexidade $\mathcal{P}$ e $\mathcal{NP}$.

Capítulo II

Descrição dos Resultados

Neste texto, utilizaremos um alfabeto Γ —que é um conjunto finito não–vazio de letras— e, sobre ele, Γ^* indica o conjunto das palavras de Γ. Tomemos $L \subseteq \Gamma^*$ um problema de decisão —i.e., um conjunto qualquer de palavras do alfabeto Γ— e M uma Máquina de Turing sobre Γ. Se $r < \omega$, denotamos

$$\mathrm{L}^r(M) \;:=\; \{\, (\vec{x}) \in (\Gamma^*)^r \parallel M(\vec{x}) \text{ tem aceite (ou } M \text{ aceita a entrada } \vec{x}) \,\}$$

e $\mathrm{L}(M) := \mathrm{L}^1(M)$. Aqui, $\vec{x} := x_0, x_1, \ldots, x_{r-1}$ é uma r-upla de palavras do alfabeto Γ. Dizemos que M aceita L —ou que L é o problema de decisão aceito por M— sse (i.e., se, e somente se,) $L = \mathrm{L}(M)$.

Todas as Máquinas de Turing que estaremos considerando são determinísticas. Os aspectos não–determinísticos das suas computações serão obtidos através da segmentação da entrada em diversas fitas e da quantificação de parte delas. Iremos aprimorando a definição das Máquinas de Turing no decorrer do texto.

Fixemos uma (conveniente) enumeração $M_0, M_1, \ldots$ de todas as Máquinas de Turing sobre Γ. Para $r, k < \omega$, sejam $L_k^r := \mathrm{L}^r(M_k)$ e $L_k := L_k^1$. Assim temos a enumeração $L_0, L_1, \ldots$ dos problemas de decisão aceitos pelas Máquinas de Turing. Também seja a enumeração $W_0, W_1, \ldots$ dos problemas de decisão W_i que contêm as palavras para as quais a Máquina de Turing M_i pára, ou seja, $W_i := \{x \in \Gamma^* \parallel M_i(x) \text{ pára (a computação)}\}$.

Notemos que nem todo problema de decisão L é representado por algum $i < \omega$, tal que $L = L_i$. Antes de tudo, porque existem incontáveis problemas de decisão. Entretanto, todos os importantes problemas de decisão da Teoria da Complexidade Computacional estão na enumeração $(L_i)_{i<\omega}$ e chamá-los-emos de *aceitáveis* —de certo modo, eles fazem um paralelo com os conjuntos recursivamente enumeráveis na Teoria da Recursão.

II.1 Hierarquia da Complexidade Aritmética

Para maiores detalhes sobre a Hierarquia da Complexidade Aritmética, vide, por exemplo, Hinman [Hin78].

O Teorema de Rice [Pap94] nos diz que toda subclasse própria $\mathcal{C}$ dos problemas de decisão aceitáveis não é recursiva, o que significa que $\{ i < \omega \parallel L_i \in \mathcal{C} \}$ não é um Δ_1–conjunto. Mais do que isso, agora nós provamos que muitas dessas classes são Σ_3–difíceis e que as classes de complexidade usuais são Σ_3–completas.

Indicamos por $\mathcal{LIN}$ a classe de complexidade composta pela intersecção das classes de tempo linear e de espaço logarítmico. Lembremos que $\mathcal{EXP}$ é a classe de complexidade de tempo exponencial. Todas as mais importantes classes de complexidade estão entre $\mathcal{LIN}$ e $\mathcal{EXP}$. De qualquer forma, se necessário, podemos estender a classe $\mathcal{EXP}$ (para tempo duplamente exponencial, etc.) no teorema que segue.

Teorema II.1–1. *Toda classe de complexidade $\mathcal{C}$ entre $\mathcal{LIN}$ e $\mathcal{EXP}$ é Σ_3–difícil, i.e., o conjunto $\{ i < \omega \parallel L_i \in \mathcal{C} \}$ é Σ_3–difícil. Isso acontece independentemente de $\mathcal{C}$ ser, ou não, uma classe de complexidade definida sobre propriedades computacionais. Em particular, as classes de complexidade $\mathcal{LIN}$, $\mathcal{L}$, $\mathcal{P}$, $\mathcal{PSPACE}$, $\mathcal{EXP}$, etc. são Σ_3–completas.*

Mais ainda, os subconjuntos de números naturais C tais que

$$\{i < \omega \| L_i \in \mathcal{LIN}\} \subseteq C \subseteq \{i < \omega \| L_i \in \mathcal{EXP}\}$$

constituem um sub-reticulado de conjuntos Σ_3–difíceis.

Em direção às classes de complexidade não–determinísticas, temos

Corolário II.1–2. *Cada nível da hierarquia polinomial Π_n^{p} e Σ_n^{p} (incluindo as classes $\mathcal{NP}$ e co-$\mathcal{NP}$), toda a hierarquia polinomial $\mathcal{PH}$ e outras classes não–determinísticas (como $\mathcal{NL}$) são Σ_3–completas, i.e., para $n < \omega$, os conjuntos*

$$\{ i < \omega \parallel L_i \in \Sigma_n^{\mathrm{p}} \}, \quad \{ i < \omega \parallel L_i \in \Pi_n^{\mathrm{p}} \},$$

$$\{ i < \omega \parallel L_i \in \mathcal{PH} \} \quad e \quad \{ i < \omega \parallel L_i \in \mathcal{NL} \}$$

são Σ_3–completos.

Mais uma vez, não haveria nenhum problema em estender os limites de tempo exponencial e de espaço polinomial acima para limitantes maiores ainda.

Usualmente, para uma função $f : I \subseteq \mathbb{N}^k \to \mathbb{N}$ (com $k < \omega$),[i] $\mathcal{O}(f)$ é a classe de funções g limitadas linearmente por f[ii] e $\mathrm{Poly}(f)$ é constituído pelas funções g que são limitadas polinomialmente por f[iii] —todas essas g's têm o mesmo domínio e codomínio que f.

[i] Iremos diferenciar o conjunto ω da estrutura $\mathbb{N}$. O primeiro será interpretado como um número ordinal e, o segundo, como o modelo *standard* da Aritmética.

[ii] Isso é, existem $a, b \in \mathbb{N}$ tais que $g(\vec{m}) \leq a f(\vec{m}) + b$, para todos $\vec{m} \in \mathbb{N}$.

[iii] I.e., existem $a, b, c \in \mathbb{N}$ tais que $g(\vec{m}) \leq a f^c(\vec{m}) + b$, para todos $\vec{m} \in \mathbb{N}$.

Outra classe importante, denotada por $\mathcal{O}_{\mathrm{PLog}}(f)$, é a classe das funções g que têm ordem linear com fator polilogarítmico, com respeito a f. Isso é, $\mathcal{O}_{\mathrm{PLog}}(f)$ contêm as funções g que têm o mesmo domínio e codomínio que f e em que existem $a, b, c \in \mathbb{N}$ tais que $g(\vec{m}) \leq af(\vec{m}) \log^c(f(\vec{m}))+b$, para todos $\vec{m} \in \mathbb{N}$. É claro que $\mathcal{O}_{\mathrm{PLog}}(f) \subseteq \mathcal{O}(f^2) \subseteq \mathrm{Poly}(f)$. Também denotamos a função identidade por $\mathrm{Id}(m) := m$, em que $m \in \mathbb{N}$.

Teorema II.1–3. *Cada subconjunto de números naturais entre:*

ou

$$\{\, i < \omega \parallel M_i \text{ tem tempo não–determinístico linear com fator polilogarítmico} \,\},$$

ou

$$\{\, i < \omega \parallel M_i \text{ tem espaço não–determinístico logarítmico} \,\};$$

e

$$\{\, i < \omega \parallel M_i \text{ tem espaço não–determinístico exponencial} \,\}$$

é Σ_2–difícil.

Mais ainda, esses conjuntos que forem definidos por Máquinas de Turing limitadas por: tempo não–determinístico, ou linear com fator polilogarítmico, ou quadrático, ou polinomial, ou exponencial, ou $\mathcal{O}_{\mathrm{PLog}}(f)$, ou $\mathcal{O}(f^2)$, ou $\mathrm{Poly}(f)$ (com $f \geq \mathrm{Id}$,[iv] uma função recursiva); ou espaço não–determinístico, ou logarítmico, ou polinomial, ou exponencial, ou $\mathcal{O}(h)$, ou $\mathrm{Poly}(h)$ (com $h \geq \log$,[iv] uma função recursiva) são Σ_2–completos.

Teorema II.1–4. *Existem funções recursivas $f' \in \mathcal{O}_{\mathrm{PLog}}(\mathrm{Id}) \subseteq \mathcal{O}(\mathrm{Id}^2) \subseteq \mathrm{Poly}(\mathrm{Id})$*[v] *—que tem ordem linear com fator polilogarítmico (ou ordem quadrática, ou ordem polinomial)— e $h' \in \mathcal{O}(\log)$ —que tem ordem logarítmica— tais que, para todas funções recursivas $f \geq f'$ e $h \geq h'$, temos que os conjuntos*

$$\{\, i < \omega \parallel M_i \text{ roda em tempo não–determinístico } f \,\}$$

e

$$\{\, i < \omega \parallel M_i \text{ roda em espaço não–determinístico } h \,\}$$

são Π_1–completos.

II.2 Certificação e Imposição de uma Indemonstrabilidade/ Independência

Inicialmente apresentaremos algumas definições. As linguagens lógicas aqui consideradas sempre serão assumidas como contendo, para cada $i < \omega$, um termo livre de variáveis i, que precisa ser *obtido efetivamente*.

Sejam $R \subseteq \omega$ uma relação unária (ou uma asserção unária), T uma teoria e ${}^{\ulcorner}R(x)^{\urcorner_T} := \varphi(x)$ uma fórmula com apenas x sendo uma variável livre. Defina

$$I_R \;:=\; \{i < \omega \parallel T \vdash {}^{\ulcorner}R(i)^{\urcorner}\}$$

[iv] Assim, para todo $m \in \mathbb{N}$, $f(m) \geq \mathrm{Id}(m) := m$ e $h(m) \geq \log(m)$.

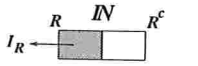
[v] Lembremos que Id é a função identidade.

e

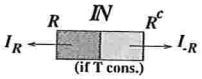
$$I_{\neg R} := \{i < \omega \parallel T \vdash \neg {}^{\ulcorner}R(\dot{i})^{\urcorner}\}.$$

Logo dizemos que:

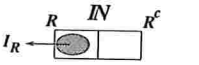
- A fórmula ${}^{\ulcorner}R(x)^{\urcorner_T}$ *expressa* R *em* T sse, para todo $i < \omega$, $R(i)$ vale sse $T \vdash {}^{\ulcorner}R(\dot{i})^{\urcorner_T}$.



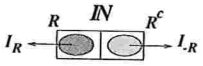
- A fórmula ${}^{\ulcorner}R(x)^{\urcorner_T}$ *representa* R *em* T sse, para todo $i < \omega$, se $R(i)$ vale, então $T \vdash {}^{\ulcorner}R(\dot{i})^{\urcorner_T}$; e, se $R(i)$ não vale, então $T \vdash \neg {}^{\ulcorner}R(\dot{i})^{\urcorner_T}$.



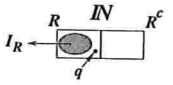
- A fórmula ${}^{\ulcorner}R(x)^{\urcorner_T}$ *certifica fracamente* R *em* T sse, para todo $i < \omega$, se $T \vdash {}^{\ulcorner}R(\dot{i})^{\urcorner_T}$, então $R(i)$ vale.



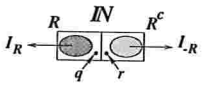
- A fórmula ${}^{\ulcorner}R(x)^{\urcorner_T}$ *certifica* R *em* T sse, para todo $i < \omega$, se $T \vdash {}^{\ulcorner}R(\dot{i})^{\urcorner_T}$, então $R(i)$ vale; e, se $T \vdash \neg {}^{\ulcorner}R(\dot{i})^{\urcorner_T}$, então $R(i)$ não vale.



- A relação R *impõe uma indemonstrabilidade em* T sse, para cada certificação fraca ${}^{\ulcorner}R(x)^{\urcorner_T}$ de $R(x)$ em T, existe $q < \omega$ tal que $R(q)$ vale e ${}^{\ulcorner}R(\dot{q})^{\urcorner_T}$ é indemonstrável em T (i.e., $T \nvdash {}^{\ulcorner}R(\dot{q})^{\urcorner_T}$).



- A relação R *impõe uma independência em* T sse, para cada certificação ${}^{\ulcorner}R(x)^{\urcorner_T}$ de $R(x)$ em T, existem $q, r < \omega$ tais que $R(q)$ vale, $R(r)$ não vale e ${}^{\ulcorner}R(\dot{q})^{\urcorner_T}$ e ${}^{\ulcorner}R(\dot{r})^{\urcorner_T}$ são independentes de T (i.e., $T \nvdash {}^{\ulcorner}R(\dot{q})^{\urcorner_T}$, $T \nvdash \neg {}^{\ulcorner}R(\dot{q})^{\urcorner_T}$ e o mesmo para ${}^{\ulcorner}R(\dot{r})^{\urcorner_T}$).



- A relação R *impõe uma indemonstrabilidade/independência em* T sse, ao mesmo tempo, R e R^c —a relação complementar de R— impõe uma indemonstrabilidade em T.

Façamos agora algumas observações sobre as definições acima. Quando T é uma teoria consistente, tanto uma representação, quanto uma expressão em T implicam uma certificação (fraca) em T. Portanto, se uma asserção unária R impõe uma indemonstrabilidade em T, então R não tem nem representação, nem expressão em T.

Nenhuma asserção R tem certificação em uma teoria inconsistente T. Supondo, ainda mais, que R é não–trivial (especificamente $R \neq \omega$), então R também não pode ter uma certificação fraca em T.

Também, se T é uma teoria completa, certificação em T implica representação em T. Cada asserção representável em uma teoria axiomatizável e consistente T é recursiva. Logo, uma teoria axiomatizável, consistente e completa T —como, por exemplo, a teoria dos corpos reais fechados— *não pode certificar* asserções não–recursivas —como são quase todas as asserções neste trabalho.

Observe que, em formulações gerais, não é tão simples obter-se uma certificação em uma

teoria. Antes de mais nada, T precisa ser consistente. Mais ainda, para se obter uma certificação de uma asserção R em T, *precisamos ser capazes de codificar*, em algum sentido, *as propriedades definidoras de R dentro de T*. Note que, se R não tem certificação em um teoria, então ela impõe trivialmente uma indemonstrabilidade/independência nessa teoria. Por isso, estaremos sempre supondo que as nossas teorias são extensões consistentes da Aritmética de Robinson, PA$^-$.

Entretanto, quando supomos que existe um modelo M da teoria T (i.e., $M \models T$), cada fórmula $\ulcorner R(x) \urcorner^M$ que *expressa* a asserção R sobre o modelo M —isso é, para todo $i < \omega$, $M \models \ulcorner R(\boldsymbol{i}) \urcorner^M$ sse $R(i)$ vale— é imediatamente uma certificação de R em T.

Portanto temos um modo natural de obter certificações, desde que possamos admitir a existência de um *modelo standard* M da teoria T e que a sua linguagem seja capaz de codificar as propriedades definidoras da asserção R em M. A idéia de "modelo *standard*" é exatamente que sua linguagem expressa a mesma coisa, tanto dentro do modelo, quanto fora —i.e., na metateoria.

No nosso caso, podemos codificar as questões da Complexidade Computacional —usando a linguagem da Aritmética de Peano— no conjunto dos números naturais $\mathbb{N}$. Lembre que $\mathbb{N}$ é o modelo *standard* da Aritmética de Peano, PA, e que ele pode ser interpretado —de modo absoluto— dentro de todos os modelos da Teoria dos Conjuntos de Zermelo–Frænkel, ZF. Assim existem certificações em PA e ZF de todas as asserções contidas neste trabalho. Entretanto, é de se notar que cada uma dessas asserções tem diversas certificações distintas e é impossível determinar uma delas como sendo "a canônica".

De outro lado, observe que uma asserção R que impõe uma indemonstrabilidade/independência em uma teoria T também impõe uma independência em T; mas, em situações atípicas, a recíproca pode não ser verdadeira. Já quando R impõe uma indemonstrabilidade em T e $\ulcorner R(x) \urcorner^T$ é uma das suas certificações em T, existe $q < \omega$ tal que $R(q)$ vale e também $\ulcorner R(\boldsymbol{q}) \urcorner^T$ é *independente de T*.

Agora, em vista dos três teoremas prévios e seguindo Hájek [Háj79], temos diversos resultados de independência.

Teorema II.2–1. *Seja T uma teoria axiomatizável. Logo:*

(i) *Tomando-se as funções recursivas f' linear com fator polilogarítmico e h' logarítmica do Teorema* II.1–4, *para funções recursivas $f \geq f'$ e $h \geq h'$, cada asserção*

$$\text{"}M_x \textit{ roda em tempo não–determinístico } f\text{"}$$

e

$$\text{"}M_x \textit{ roda em espaço não–determinístico } h\text{"}$$

—em que x é a única variável livre— impõe uma indemonstrabilidade em T.

(ii) *Para funções recursivas $f \geq \mathrm{Id}$ (a função identidade) e $h \geq \log$ (a função logarítmica), cada asserção*

$$\text{"}M_x \textit{ tem tempo não–determinístico } \mathcal{O}_{\mathrm{PLog}}(f)\text{"},^{[\mathrm{vi}]}$$

$$\text{"}M_x \textit{ tem espaço não–determinístico } \mathcal{O}(h)\text{"},^{[\mathrm{vii}]}$$

"M_x tem tempo não–determinístico Poly(f)*"*,[viii]

"M_x tem espaço não–determinístico Poly(h)*"*,

"M_x tem tempo não–determinístico linear com fator polilogarítmico",

"M_x tem tempo não–determinístico polinomial"

e

"M_x tem espaço não–determinístico logarítmico"

(e outras) impõe uma indemonstrabilidade/independência em T.

(iii) *Para cada classe de complexidade $\mathcal{C}$ em que $\mathcal{LIN} \subseteq \mathcal{C} \subseteq \mathcal{EXP}$, a asserção*

$$\text{"}L_x \in \mathcal{C}\text{"}$$

impõe uma indemonstrabilidade/independência em T.

(iv) *Se T é também uma teoria consistente, não existem, nem representação, nem expressão em T para as asserções acima.* ■

II.3 Aritmética de Peano e o Problema $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$

Com isso em mãos, podemos pensar sobre a Aritmética de Peano, PA (uma teoria axiomatizável), seu modelo canônico $\mathbb{N} \models$ PA e duas fórmulas tais que, para cada $i < \omega$,

$$\mathbb{N} \models \ulcorner L_i \in \mathcal{P} \urcorner \text{ sse } L_i \in \mathcal{P}$$

e

$$\mathbb{N} \models \ulcorner L_i \in \mathcal{NP} \urcorner \text{ sse } L_i \in \mathcal{NP}.$$

Tomando outra fórmula

$$\ulcorner \mathcal{P} = \mathcal{NP} \urcorner := \forall x \, (\ulcorner L_x \in \mathcal{NP} \urcorner \to \ulcorner L_x \in \mathcal{P} \urcorner),$$

temos que $\mathbb{N} \models \ulcorner \mathcal{P} = \mathcal{NP} \urcorner$ sse $\mathcal{P} = \mathcal{NP}$. Essas três fórmulas certificam as suas respectivas asserções em PA. Observe que todos esses argumentos também podem ser tomados para muitas outras teorias axiomatizáveis e seus modelos canônicos, assim como ZFC (se consistente).

Considere os seguintes subconjuntos dos números naturais:

$$P := \{i < \omega \parallel L_i \in \mathcal{P}\},$$

$$NP := \{i < \omega \parallel L_i \in \mathcal{NP}\},$$

$$I_{\mathcal{NP}} := \{i < \omega \parallel \text{PA} \vdash \ulcorner L_i \in \mathcal{NP} \urcorner\}$$

[vi] I.e., existem $a, b, c \in \mathbb{N}$ tais que "M_x roda em tempo não–determinístico $a \cdot \log^c(\cdot) + b$".
[vii] I.e., existem $a, b \in \mathbb{N}$ tais que "M_x roda em espaço não–determinístico $a \cdot + b$".
[viii] I.e., existem $a, b, c \in \mathbb{N}$ tais que "M_x roda em tempo não–determinístico $a f^c(\cdot) + b$".

e

$$I_{\neg\mathcal{P}} := \{i < \omega \parallel \mathrm{PA} \vdash \ulcorner L_i \notin \mathcal{P} \urcorner\}.$$

Sejam também

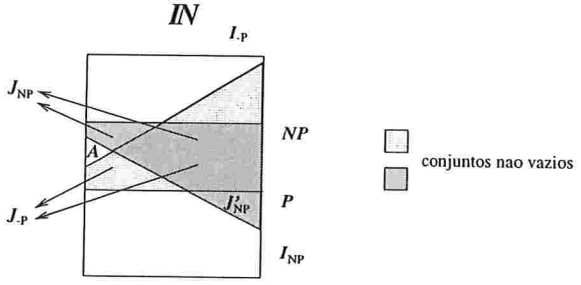
$$A := I_{\mathcal{NP}} \cap I_{\neg\mathcal{P}} := \{i < \omega \parallel \mathrm{PA} \vdash \ulcorner L_i \in \mathcal{NP} \urcorner \wedge \ulcorner L_i \notin \mathcal{P} \urcorner\},$$

$$J_{\mathcal{NP}} := NP \setminus (I_{\mathcal{NP}} \cup P) := \{i \in NP \setminus P \parallel \mathrm{PA} \nvdash \ulcorner L_i \in \mathcal{NP} \urcorner\},$$

$$J_{\neg\mathcal{P}} := NP \setminus (I_{\neg\mathcal{P}} \cup P) := \{i \in NP \setminus P \parallel \mathrm{PA} \nvdash \ulcorner L_i \notin \mathcal{P} \urcorner\}$$

e

$$J'_{\mathcal{NP}} := P \setminus I_{\mathcal{NP}} := \{i \in P \parallel \mathrm{PA} \nvdash \ulcorner L_i \in \mathcal{NP} \urcorner\}.$$



Assim o Teorema II.2–1.(iii) nos indica que $I_{\mathcal{NP}} \subsetneq \mathcal{NP}$ e $\mathcal{P} \subsetneq I^c_{\neg\mathcal{P}}$. Logo observe que:

(i) $$NP \setminus P = A \dot{\cup} (J_{\mathcal{NP}} \cup J_{\neg\mathcal{P}}),$$

(ii) $$NP \setminus I_{\mathcal{NP}} = J_{\mathcal{NP}} \dot{\cup} J'_{\mathcal{NP}} \neq \varnothing$$

e

(iii) $$J_{\neg\mathcal{P}} \subseteq (I_{\neg\mathcal{P}} \cup P)^c \neq \varnothing.$$

Portanto, pelo Item (ii), se $J'_{\mathcal{NP}} = \varnothing$, então $\mathcal{P} \neq \mathcal{NP}$. Também

$$\mathrm{PA} \vdash \ulcorner \mathcal{P} \neq \mathcal{NP} \urcorner \text{ sse } A \neq \varnothing.$$

Supondo $\mathcal{P} \neq \mathcal{NP}$, pelo Item (i), ou $A \neq \varnothing$, ou $J_{\mathcal{NP}} \cup J_{\neg\mathcal{P}} \neq \varnothing$. Mais ainda, os Itens (ii) e (iii) sugerem que $J_{\mathcal{NP}} \cup J_{\neg\mathcal{P}}$ não pode ser um conjunto vazio e que a chance de se ter $\mathrm{PA} \vdash \ulcorner \mathcal{P} \neq \mathcal{NP} \urcorner$ diminui, à medida que o conjunto $J_{\mathcal{NP}} \cup J_{\neg\mathcal{P}}$ seja maior.

II.4 Teoria da Recursão com Complexidade

Nas demonstrações dos teoremas das Seções II.2 *Certificação e Imposição de uma Indemonstrabilidade/Independência* e II.6 *Imposição de Indemonstrabilidades Fixadas*, utilizaremos o Teorema do Ponto

Fixo II.4–2. A sua formulação original na Teoria da Recursão só consegue certificar a existência de um ponto fixo que é, ao mesmo tempo, uma das entradas de uma dada Máquina de Turing e o índice de uma segunda Máquina de Turing que simula a computação da primeira. Aqui, simulação apenas significa que ambas têm a mesma saída, mas não há nenhum controle sobre limitantes de tempo e espaço.

Modificaremos a prova dos teoremas da Teoria da Recursão, buscando simulações que preservem, conjuntamente, tempo com eficiência de ordem linear com fator polilogarítmico e espaço com eficiência de ordem linear.

Se M e N são Máquinas de Turing e $\vec{x}$ e $\vec{y}$ são palavras do alfabeto Γ, então $t(M(\vec{x},\vec{y}))$ indica o tempo da respectiva computação e $s(M(\vec{x},\vec{y}))$, o espaço. Agora fixando as palavras $\vec{y}$, dizemos que $M(\vec{x},\vec{y})$ é *bem–simulado por* $N(\vec{y},\vec{x})$ *sobre* $\vec{x}$ ($\vec{x}$ variando sobre as palavras de Γ) e denotamo-lo por

$$M(\vec{y},\vec{x}) \preccurlyeq_{\vec{x}} N(\vec{y},\vec{x})$$

sse N tem *fitas de entrada* para alocar apenas entradas no formato $(\vec{x},\vec{y})$ e existem funções f linear com fator polilogarítmico e h linear tais que, para todos $\vec{x}$ (fixados os $\vec{y}$),

$$M(\vec{y},\vec{x}) = N(\vec{y},\vec{x}),$$

$$t(M(\vec{y},\vec{x})) \leq t(N(\vec{y},\vec{x})) \leq f(t(M(\vec{y},\vec{x})))$$

e

$$s(M(\vec{y},\vec{x})) \leq s(N(\vec{y},\vec{x})) \leq h(s(M(\vec{y},\vec{x}))).$$

Para os próximos resultados, precisamos restringir as nossas Máquinas de Turing — incluindo a enumeração $M_0, M_1, \ldots$— àquelas que operam com *somente duas fitas de trabalho* (ou um limitante maior para o um número de fitas de trabalho).

Também fixamos $0, 1, \ldots$ como uma enumeração recursiva canônica das palavra de Γ. Essa enumeração precisa ser feita por uma associação de duas vias: $i < \omega \mapsto \mathtt{i} \in \Gamma_*^*$ (também denotada por $\check{i}$, i.e., $\mathtt{i} :=: \check{i}$) e a sua inversa $x \in \Gamma_*^* \mapsto \widetilde{x} < \omega$. Aqui, Γ_*^* é o conjunto de palavras de Γ sem a *palavra vazia* ϵ. Para nós, um vetor $\vec{x}$ sempre representa uma seqüência de elementos $x_i, \ldots, x_j$ (com $i, j < \omega$).

Se f for uma função recursiva r-ária (com $r < \omega$), existe uma Máquina de Turing M_c tal que, para todo i no domínio de f, $f(i) := \widetilde{M_c(\mathtt{i})}$ e, para os demais i's, M_c não pára. Aqui, $\widetilde{M_c(\mathtt{i})}$ ($:=: \widetilde{M_c(\mathtt{i})}$) é o número natural que é codificado pela palavra de saída da respectiva computação e $\mathtt{i}$ ($:=: \check{i}$) denota a codificação do número natural i por palavras de Γ. O número c é chamado de *número índice de* f (é claro que ele não é univocamente determinado).

Teorema da Máquina de Turing Universal e da Parametrização II.4–1. *Dados $r, q, c, b < \omega$, pode-se obter efetivamente $u < \omega$ e uma função recursiva (total) $(c+b+2)$-ária v (i.e., o seu número índice) tais que, para todos $i, j, n_0, \ldots, n_{c-1}, m_0, \ldots, m_{b-1} < \omega$,*

(i) $$M_j(M_i(\vec{x}),\vec{y}) \preccurlyeq_{\vec{x},\vec{y}} M_u(\mathtt{i},\mathtt{j},\vec{x},\vec{y}),$$

(ii) $$M_j(M_i(\vec{n},\vec{x}),\vec{m},\vec{y}) \preccurlyeq_{\vec{x},\vec{y}} M_{v(i,j,\vec{n},\vec{m})}(\vec{x},\vec{y})$$

e, para todos $\vec{x} \in \Gamma^*$, *temos*

(iii) $$M_{\widetilde{M_i(\vec{n},\vec{x})}}(\vec{m},\vec{y}) \preccurlyeq_{\vec{y}} M_{v(i,u,\vec{n},\vec{m})}(\vec{x},\vec{y});$$

em que as palavras $x_0, \ldots, x_{r-1}, y_0, \ldots, y_{q-1}$ *estão variando em* Γ^*. *A função* v *é chamada de* `parâmetro indexador (total)` *e* M_u, *de* `Máquina de Turing Universal`.

Quando M tem as primeiras palavras da entrada $\vec{x}$ (sobre Γ) fixadas, denotá-la-emos por $M^{(\vec{x})}$. Além do mais, $M \preccurlyeq_r N$ indicará $M(\vec{x}) \preccurlyeq_{\vec{x}} N(\vec{x})$ (com as palavras $x_0, \ldots, x_{r-1}$ variando em Γ^*) ou apenas por $M \preccurlyeq N$, se a Máquina de Turing M usa no máximo r fitas de entrada.

Teorema do Ponto Fixo II.4–2. *Sejam* $r, c, b < \omega$ *e uma função recursiva parcial unária* f, *com domínio* $I \subseteq \omega$ *e tendo um dos seus números índice* $q < \omega$ *conhecido. Pode-se encontrar efetivamente funções recursivas parciais unárias* e *e* d, *que têm domínios* $I \times \omega^{c+b}$ *e* $I \times \omega^c$, *respectivamente, tais que, para todos* $i \in I$ *e* $n_0, \ldots, n_{c-1}, m_0, \ldots, m_{b-1} < \omega$,

(i) $$M^{(\vec{n})}_{\widetilde{M_{f(i)}(\vec{m},e(i,\vec{n},\vec{m}))}} \preccurlyeq_r M_{e(i,\vec{n},\vec{m})};$$

(ii) $$M^{(\vec{n},d(i,\vec{n}))}_{f(i)} \preccurlyeq_r M_{d(i,\vec{n})};$$

e,

(iii) *se a função* f *é constante ou* $M_{f(i)}$ *usa um número limitado de fitas de entrada (para todos* $i < \omega$*), então*
$$M^{(\vec{n},d(i,\vec{n}))}_{f(i)} \preccurlyeq M_{d(i,\vec{n})}.$$

Mais ainda, sejam g *uma função recursiva parcial unária,* M *uma Máquina de Turing —que computa usando somente duas fitas de trabalho— e* h *uma função recursiva parcial* $(c+1)$*-unária. Então, existem* $e, d, \ell < \omega$ *tais que*

$$M_{g(e)} \preccurlyeq_r M_e;$$
$$M^{(d)} \preccurlyeq M_d;$$

e, para cada $n_0, \ldots, n_{c-1} < \omega$, *temos*

$$h(\ell,\vec{n}) = \widetilde{M_\ell(\vec{n})}.$$

Nesses casos particulares, se as funções e a Máquina de Turing são "construtivamente obtidos" (i.e., conhecemos explicitamente os seus números índice), então podemos obter efetivamente os números $e, d, \ell < \omega$.

II.5 Imposição Efetiva de uma Indemonstrabilidade

Surpreendentemente, algumas vezes, para uma asserção $R(x)$ que impõe uma indemonstrabilidade em uma teoria T, a existência de $e < \omega$ —para cada certificação fraca $\ulcorner R(x) \urcorner_T$—, tal que

$R(e)$ vale e a sentença ${}^{\ulcorner}R(e)^{\urcorner_T}$ é indemonstrável em T, pode ser obtida *efetivamente*. Sempre que isso acontece,

- dizemos que R *impõe efetivamente uma indemonstrabilidade em* T.

Diremos que uma teoria T é *axiomatizável* sse conhecemos explicitamente uma Máquina de Turing M_h —isso é, o índice de codificação $h < \omega$— que aceita os teoremas de T.

No Lema IV.1–1, mostraremos como Hartmanis e Hopcroft [HH76] provaram uma forma forte da afirmação: para cada $q < \omega$, "M_x(q) não pára" impõe efetivamente uma indemonstrabilidade em uma teoria axiomatizável T.

Já havíamos comentado sobre outro resultado de Hartmanis e Hopcroft [HH76]. Ele nos fornece uma prova de que a asserção "M_x tem tempo determinístico polinomial" impõe *efetivamente* uma indemonstrabilidade em uma teoria T. Esta é uma das afirmações do Teorema II.2–1.(ii), entretanto não considerando mais o ponto de vista dos limites de tempo/espaço *não–determinísticos*, mas agora concebendo a existência como sendo *efetiva*. Contudo, nessa prova, como dissemos, a teoria T precisava ser mais do que simplesmente axiomatizável. Evitando essa restrição, temos

Teorema II.5–1. *Seja T uma teoria axiomatizável. Então cada asserção descrita no Teorema* II.2–1.(ii) *impõe* `efetivamente` *uma indemonstrabilidade na teoria T.*

Portanto agora sabemos que existe a Máquina de Turing N para a qual, dado $h < \omega$, N fornece $e < \omega$ tal que a Máquina de Turing M_e —em que o índice de codificação é e— tem espaço não–determinístico logarítmico, mas a asserção "M_e tem espaço não–determinístico logarítmico" é indemonstrável na teoria axiomatizável T_h —que tem h como seu índice de codificação. Aqui, T_h precisa apenas interpretas corretamente a asserção em questão "M_x tem espaço não–determinístico logarítmico", que tem apenas x como variável livre.

Também podemos completar esses resultados para as afirmações com uma função fixada, como segue.

Teorema II.5–2. *Supondo-se que T é uma teoria axiomatizável, cada asserção fornecida no Teorema* II.2–1.(i) *impõe* `efetivamente` *uma indemonstrabilidade em T.*

II.6 Imposição de Indemonstrabilidades Fixadas

Sejam uma asserção $R(x)$ —ou uma relação unária sobre ω— e uma teoria T. Definimos:

- Se, para cada certificação fraca ${}^{\ulcorner}R(x)^{\urcorner_T}$ de R em T e cada $k < \omega$, existe $q < \omega$ tal que $L_q = L_k$ e ${}^{\ulcorner}R(\boldsymbol{q})^{\urcorner_T}$ é indemonstrável em T (i.e., $T \not\vdash {}^{\ulcorner}R(\boldsymbol{q})^{\urcorner_T}$), então dizemos que R *impõe indemonstrabilidades fixadas em* T.

- Se, para cada certificação ${}^{\ulcorner}R(x)^{\urcorner_T}$ de R em T e cada $k < \omega$, existe $q < \omega$ tal que $L_q = L_k$ e ${}^{\ulcorner}R(\boldsymbol{q})^{\urcorner_T}$ é independente de T (i.e., $T \not\vdash {}^{\ulcorner}R(\boldsymbol{q})^{\urcorner_T}$ e $T \not\vdash \neg{}^{\ulcorner}R(\boldsymbol{q})^{\urcorner_T}$), então

dizemos que R *impõe independências fixadas em* T.

- Algumas vezes, conjuntamente R e R^c —a relação complementar de R— impõem indemonstrabilidades fixadas em T. Assim dizemos que R *impõe fortemente indemonstrabilidades fixadas em* T.

- Quando R impõe, ao mesmo tempo, fortemente indemonstrabilidades fixadas e independências fixadas em T, dizemos que R *impõe indemonstrabilidades/independências fixadas em* T.

Dizemos que uma asserção unária $Q \subseteq \omega$ *tem propriedade da diferença finita de problemas de decisão* sse, para cada $i, j < \omega$, se o problema de decisão L_i difere apenas finitamente de L_j, então $Q(i)$ vale sse $Q(j)$ vale. Tal asserção Q pode ser pensada como $Q(x) := "L_x \in \mathcal{P}"$, com x variando em ω, por exemplo.

Teorema II.6–1. *Sejam T uma teoria axiomatizável e $R, Q \subseteq \omega$ duas asserções unárias tais que $R \subseteq Q$ (i.e., para cada $i < \omega$, se $R(i)$ vale, $Q(i)$ também vale) e Q tenha propriedade da diferença finita de problemas de decisão. Logo:*

(i) *Se $\ulcorner R(x) \urcorner^T$ é uma certificação fraca —ou uma certificação— de $R(x)$ em T, temos que, para todos $k, \ell < \omega$, existe $q < \omega$ tal que, se, ou $L_k \subseteq L_\ell$, ou $L_\ell \subseteq L_k$,[ix] e $Q(\ell)$ não vale, então $L_q = L_k$ e $\ulcorner R(\boldsymbol{q}) \urcorner^T$ é indemonstrável em T (i.e., $T \nvdash \ulcorner R(\boldsymbol{q}) \urcorner^T$).*

(ii) *Supondo, mais ainda, que $R = Q$, $\ulcorner R(x) \urcorner^T$ é um certificação de $R(x)$ em T e $R(k)$ vale, então $\ulcorner R(\boldsymbol{q}) \urcorner^T$ é independente de T (i.e., também $T \nvdash \neg \ulcorner R(\boldsymbol{q}) \urcorner^T$) —e ainda $R(q)$ vale.*

O Item (ii) segue direto do Item (i). Se $R(k)$ vale e $L_q = L_k$, como R tem propriedade da diferença finita de problemas de decisão, $R(q)$ também vale. Pela certificação, $\neg \ulcorner R(\boldsymbol{q}) \urcorner^T$ é indemonstrável em T também. A prova do Item (i) será feita no próximo capítulo.

Notemos que, no caso geral, "R impõe fortemente indemonstrabilidades fixadas em T" não implica que "R impõe independências fixadas em T" e vice versa. Entretanto, a implicação de ida vale quando R tem propriedade da diferença finita de problemas de decisão e, assim, podemos concluir que "R impõe indemonstrabilidades/independências fixadas em T". Com efeito, se R tem propriedade da diferença finita de problemas de decisão, R^c também a tem. Para uma certificação $\ulcorner R(x) \urcorner^T$ de R em T e um $k < \omega$, use o Item (ii) em R, se $R(k)$ vale, e em R^c, se $R(k)$ não vale.

Observe também que, se $R(k)$ não vale e tomando-se $q := k$, pela certificação fraca, $\ulcorner R(\boldsymbol{q}) \urcorner^T$ é indemonstrável em T. Assim, usando o teorema acima, facilmente temos o que segue:

Teorema II.6–2. *Sejam T uma teoria axiomatizável e $R \subseteq \omega$ uma asserção unária. Para mostrar que R* `impõe indemonstrabilidades fixadas em` *T, é suficiente obter outra asserção unária $Q \subseteq \omega$ de tal modo que:*

a) $Q \supseteq R$;

[ix] Quando fixamos somente uma das possibilidades, ou $L_k \subseteq L_\ell$, ou $L_\ell \subseteq L_k$, a existência de q é *efetiva*.

b) *Q tenha propriedade da diferença finita de problemas de decisão; e*

c) *para todo $k < \omega$ tal que $R(k)$ valha, existe $\ell < \omega$ para o qual, ou $L_k \subseteq L_\ell$, ou $L_\ell \subseteq L_k$, e $Q(\ell)$ não vale.*

Agora suponha que:

a) *R tem propriedade da diferença finita de problemas de decisão; e*

b) *para todo $k < \omega$, existe $\ell < \omega$ para o qual, ou $L_k \subseteq L_\ell$, ou $L_\ell \subseteq L_k$, e: se $R(k)$ vale, então $R(\ell)$ não vale; e, se $R(k)$ não vale, então $R(\ell)$ vale.*

Logo temos que R `impõe indemonstrabilidades/independências fixadas em` *T.* ■

Vejamos algumas aplicações desse teorema.

Corolário II.6–3. *Para uma teoria axiomatizável T, a asserção "$L_x \in \mathcal{P}$" —com x sendo a única variável livre— impõe indemonstrabilidades/independências fixadas em T. O mesmo continua válido para as classes de complexidade usuais, tais como $\mathcal{L}$, $\mathcal{NP}$, co-$\mathcal{NP}$, $\mathcal{PH}$, $\mathcal{PSPACE}$, $\mathcal{EXP}$ e $\mathcal{NEXP}$, ao invés de $\mathcal{P}$.*

Note, mais uma vez, que é possível estender o resultado acima para uma classe de complexidade maior do que $\mathcal{NEXP}$.

Agora seja uma asserção unária $R \subseteq \omega$ para a qual a validade de $R(i)$ implique na validade de "$L_i \in \mathcal{P}$" (para todo $i < \omega$), como, e.g., a asserção $R(x) :=$ "M_x tem tempo (não–)determinístico polinomial". Logo uma certificação fraca de $R(x)$ também é uma certificação fraca da asserção "$L_x \in \mathcal{P}$". O Corolário II.6–3 nos mostra que esse R também impõe indemonstrabilidades fixadas em T. Mas esse resultado não é interessante porque, já na *metateoria*, para todo $k < \omega$ tal que M_k tem tempo não–determinístico polinomial, existe efetivamente $q < \omega$ tal que $L_q = L_k$ e M_q não tem tempo não–determinístico polinomial. Dessa maneira, daqui para frente, nós só preocuparemos impor indemonstrabilidades fixadas (em uma teoria) para R^c —a relação complementar de R.

Dizemos que uma asserção unária $Q \subseteq \omega$ *tem propriedade de co–limite de tempo* sse, para $i, j < \omega$, se M_j roda em tempo de ordem linear com fator polilogarítmico, com respeito à M_i, e $Q(j)$ vale, então $Q(i)$ também vale. Uma candidata para tal asserção é $Q(x) :=$ "M_x não tem tempo não–determinístico polinomial", em que x varia em ω.

Lembremos que M *é bem–simulada por* uma outra Máquina de Turing N, que denotamos por $M \preccurlyeq N$, quando N não usa mais fitas de entrada do que M e, com respeito às fitas de entradas utilizadas por N (suponha que elas são $r < \omega$), temos:

a) a mesma aceitação/rejeição de M e N —logo $L^r(M) = L^r(N)$;

b) M não gasta mais tempo do que N e N roda em tempo de ordem linear com fator polilogarítmico, com respeito à M; e

c) M não gasta mais espaço do que N e N roda em espaço de ordem linear, com respeito à M.

O teorema que segue melhora os resultados dos Teoremas II.6–1 e II.6–2 em direção aos limitantes de tempo não–determinísticos. Para duas Máquinas de Turing M e N, dizemos que M *é bem–tempo–simulada por* N (denotado por $M \overset{t}{\preccurlyeq} N$) quando os limitantes de espaço —i.e., Item *c)*— são negligenciados na definição de bem–simulada.

Teorema II.6–4. *Sejam T uma teoria axiomatizável, $R \subseteq Q \subseteq \omega$ duas asserções unárias tais que Q tenha propriedade de co–limite de tempo e $\ulcorner R(x) \urcorner^T$ seja uma certificação fraca —ou uma certificação— de $R(x)$ em T.*

Para $k, \ell < \omega$, seja $r < \omega$ o número de fitas de testemunha usadas por M_ℓ. Então existe $q < \omega$ para o qual, se $L_k^{r+1} \subseteq L_\ell^{r+1}$ e $Q(\ell)$ não vale, então $M_q \overset{t}{\succcurlyeq} M_k$ —portanto, $L_q^{r+1} = L_k^{r+1}$ e $L_q = L_k$— e $\ulcorner R(\boldsymbol{q}) \urcorner^T$ é indemonstrável em T (i.e., $T \nvdash \ulcorner R(\boldsymbol{q}) \urcorner^T$) —também M_q usa somente r fitas de testemunha.

Para mostrar que R `impõe indemonstrabilidades fixadas em` *T, é suficiente obter outra asserção unária $Q \subseteq \omega$ tal que:*

a) $Q \supseteq R$;

b) *Q tenha propriedade de co–limite de tempo; e*

c) *para todo $k < \omega$ tal que $R(k)$ valha, existe $\ell < \omega$ para o qual $L_k^{r+1} \subseteq L_\ell^{r+1}$ —em que $r < \omega$ é o número de fitas de testemunha usadas por M_ℓ— e $Q(\ell)$ não vale.*

Agora vamos à sua aplicação.

Corolário II.6–5. *Tomada uma teoria axiomatizável T, a asserção "M_x tem tempo não–determinístico polinomial" —em que x é a única variável livre— impõe fortemente indemonstrabilidades fixadas em T. O mesmo acontece nos casos tais como de tempo não–determinístico exponencial, e de espaço não–determinístico logarítmico e polinomial.*

II.7 Observações sobre os Resultados

Observe que, somente com respeito ao Teorema II.2–1.(i), como sabemos apenas que essas asserções R são Π_1–completas, *não* conseguimos dizer se a asserção complementar R^c também impõe uma indemonstrabilidade em uma teoria axiomatizável. Logo permanece a questão de verificar se R impõe uma *independência* nessa teoria.

Falta achar a *existência efetiva* para o Teorema II.2–1.(iii) —assim como feito pelos Teoremas II.5–2 e II.5–1 com respeito aos Teoremas II.2–1.(i) e (ii). Ainda está em aberto se também podemos estender os mesmos resultados de *efetividade* dos Teoremas II.5–2 e II.5–1 para a imposição de uma indemonstrabilidade/independência em uma teoria axiomatizável.

Possivelmente conseqüências sobre os conjuntos difíceis obtidos pelos Teoremas II.1–1,

II.1–3 e II.1–4 —e seus intervalos (i.e., *gaps*)— e outros resultados da Teoria da Recursão com Complexidade podem ser melhor investigados.

Com respeito ao Corolário II.6–5, continua em aberto se é possível provar que asserções tais como "M_x tem tempo não–determinístico polinomial" (do Teorema II.2–1.(ii)) impõem *independências fixadas* em teorias axiomatizáveis. Também permanece para se fazer esse tipo de extensão para asserções como "M_x roda em tempo não–determinístico f" (do Teorema II.2–1.(i)), em que f é uma função recursiva fixa.

Por outro lado, os Teoremas II.6–1, II.6–2 e II.6–4 (especialmente os dois primeiros) fornecem critérios para impor indemonstrabilidades e de independências fixadas em uma teoria axiomatizável para asserções muito mais gerais —não necessariamente envolvendo propriedades de classes de complexidade— daquelas apresentadas aqui.

Finalmente, também permanece para um estudo mais detalhado saber se os nossos resultados de indemonstrabilidade podem ser úteis na tentativa de se chegar à independência do problema $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$.

Nos capítulos que seguem, iremos provar os teoremas descritos até aqui.

II.8 Conseqüências ao Problema $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$

Neste momento, obtemos outros resultados de indemonstrabilidade e de independência, agora especificamente dizendo respeito ao problema $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$. Como conseqüência, temos interesse em chamar atenção sobre o que poderia ocorrer no caso de Sistemas Lógicos Formais não serem suficientemente fortes para resolver esse tipo de problema.

No restante desta seção, suporemos que T seja uma teoria axiomatizável em que, para todos $i, j < \omega$ distintos, ela consiga interpretar essa diferença, i.e., $T \vdash i \neq j$.

II.8.1 Estendendo uma Certificação

Tome uma fórmula $\overline{\ulcorner L_x \in \mathcal{NP} \urcorner^T}$ que seja uma certificação da asserção "$L_x \in \mathcal{NP}$" em T.

Suponha que para algum $e < \omega$, $T \nvdash \overline{\ulcorner L_e \in \mathcal{NP} \urcorner^T}$ e tenhamos a informação que $L_e \in \mathcal{NP}$. Observe que o Teorema II.2–1.(iii) —ou o Corolário II.6–3— fornece-nos este tipo de e. Assim podemos acrescentar essa nova informação à $\overline{\ulcorner L_x \in \mathcal{NP} \urcorner^T}$, tornando-a mais expressiva. Tome outra fórmula como sendo

$$\begin{aligned} \ulcorner L_x \in \mathcal{NP} \urcorner^T &:= \overline{\ulcorner L_x \in \mathcal{NP} \urcorner^T} \vee (x = e) \\ &\equiv \left((x \neq e) \rightarrow \overline{\ulcorner L_x \in \mathcal{NP} \urcorner^T} \right). \end{aligned}$$

Portanto $T \vdash \ulcorner L_e \in \mathcal{NP} \urcorner^T$ e vejamos que $\ulcorner L_x \in \mathcal{NP} \urcorner^T$ também é uma certificação da asserção "$L_x \in \mathcal{NP}$" em T.

Com efeito, como $L_e \in \mathcal{NP}$ e $T \vdash \ulcorner L_e \in \mathcal{NP} \urcorner^T$, levaremos em conta somente $i < \omega$ em que $i \neq e$. Se $T \vdash \ulcorner L_i \in \mathcal{NP} \urcorner^T$, então $T \vdash \overline{\ulcorner L_i \in \mathcal{NP} \urcorner^T}$ e, pela certificação em T, $L_i \in \mathcal{NP}$. Agora, se

$T \vdash {}^{\ulcorner}L_i \notin \mathcal{NP}^{\urcorner T}$, então $T \vdash \overline{{}^{\ulcorner}L_i \notin \mathcal{NP}^{\urcorner T}}$ e, novamente pela certificação em T, $L_i \notin \mathcal{NP}$. Note que essa propriedade continua funcionando quando trocamos a certificação em T pela certificação *fraca* em T.

Aplicando novamente o teorema supracitado, continua existindo $d < \omega$ para o qual $L_d \in \mathcal{NP}$ e $T \nvdash {}^{\ulcorner}L_d \in \mathcal{NP}^{\urcorner T}$. Podemos repetir esse passo um número finito de vezes.

Por outro lado, também podemos supor que para algum $e < \omega$, temos que $T \nvdash \overline{{}^{\ulcorner}L_e \notin \mathcal{NP}^{\urcorner T}}$ e $L_e \notin \mathcal{NP}$ —fornecido novamente pelo Teorema II.2–1.(iii) ou Corolário II.6–3. Então, dada a fórmula

$$ {}^{\ulcorner}L_x \in \mathcal{NP}^{\urcorner T} \;:=\; \overline{{}^{\ulcorner}L_x \in \mathcal{NP}^{\urcorner T}} \wedge (x \neq e), $$

temos que $T \vdash {}^{\ulcorner}L_e \notin \mathcal{NP}^{\urcorner T}$ e ${}^{\ulcorner}L_x \in \mathcal{NP}^{\urcorner T}$ permanece sendo uma certificação da asserção "$L_x \in \mathcal{NP}$" em T.

II.8.2 Indemonstrabilidade com respeito a $\mathcal{P} \supseteq \mathcal{NP}$

A questão $\mathcal{P} = \mathcal{NP}$ é uma asserção sem parâmetros indexando-a. Portanto a idéia de *certificação* na teoria T não se verifica. Internalizamos essa questão em T por uma única sentença — i.e., uma fórmula livre de variáveis. Diremos que uma sentença ${}^{\ulcorner}R^{\urcorner T} := \sigma$ *pseudo–certifica* a questão R —i.e., R é uma asserção não indexada— em T sse

$$ \begin{aligned} T \vdash {}^{\ulcorner}R^{\urcorner T} &\implies R \text{ vale}, \\ T \vdash \neg {}^{\ulcorner}R^{\urcorner T} &\implies R \text{ não vale}. \end{aligned} $$

Note que, se R vale, todas as sentenças prováveis são pseudo–certificações de R em T e, se R não vale, as sentenças refutáveis o são.

Antes de mais nada, denotaremos por $\forall'' x < \omega'' \varphi(x)$ —sendo $\varphi(x)$ uma fórmula— uma sentença em que supomos que

$$ T \;\vdash\; \forall'' x < \omega'' \varphi(x) $$

implique que, para todo $i < \omega$, $T \vdash \varphi(i)$. Podemos pensar a teoria T como sendo, ou a Aritmética de Peano, PA, com a quantificação normal $\forall x \varphi(x)$, ou a Teoria dos Conjuntos de Zermelo–Frænkel, ZF, com a quantificação $\forall x < \omega \varphi(x)$ —lembre que ω é absoluto em ZF.

Agora fixamos $\overline{{}^{\ulcorner}L_x \in \mathcal{P}^{\urcorner T}}$ e $\overline{{}^{\ulcorner}L_x \in \mathcal{NP}^{\urcorner T}}$ como sendo certificações em T, respectivamente, das asserções "$L_x \in \mathcal{P}$" e "$L_x \in \mathcal{NP}$". Note que, para isso, T precisa ser mais do que consistente.

O Teorema II.2–1.(iii) —ou o Corolário II.6–3— mostra-nos que existe $e < \omega$ tal que $L_e \in \mathcal{P} \subseteq \mathcal{NP}$ e $T \nvdash \overline{{}^{\ulcorner}L_e \in \mathcal{P}^{\urcorner T}}$. Dada novamente a certificação em T definida por

$$ {}^{\ulcorner}L_x \in \mathcal{NP}^{\urcorner T} \;:=\; \overline{{}^{\ulcorner}L_x \in \mathcal{NP}^{\urcorner T}} \vee (x = e), $$

podemos tomar a sentença

$$ {}^{\ulcorner}\mathcal{P} = \mathcal{NP}^{\urcorner T} \;:=:\; {}^{\ulcorner}\mathcal{P} \supseteq \mathcal{NP}^{\urcorner T} \;:=\; \forall'' x < \omega'' \left({}^{\ulcorner}L_x \in \mathcal{NP}^{\urcorner T} \longrightarrow \overline{{}^{\ulcorner}L_x \in \mathcal{P}^{\urcorner T}} \right). $$

Então a sentença ${}^{\ulcorner}\mathcal{P} \supseteq \mathcal{NP}^{\urcorner T}$ pseudo–certifica $\mathcal{P} \supseteq \mathcal{NP}$ em T. Mas observe que sabemos, na metateoria, que a questão $\mathcal{P} \subseteq \mathcal{NP}$ vale. Por isso, mesclando propriedades da metateoria, temos que a sentença ${}^{\ulcorner}\mathcal{P} \supseteq \mathcal{NP}^{\urcorner T}$ também pseudo–certifica $\mathcal{P} = \mathcal{NP}$ em T e assim iremos escrevê-la como sendo ${}^{\ulcorner}\mathcal{P} = \mathcal{NP}^{\urcorner T}$. Vejamos que ela é indemonstrável em T, i.e.,

$$T \not\vdash {}^{\ulcorner}\mathcal{P} = \mathcal{NP}^{\urcorner T}.$$

Com efeito, caso contrário, para todo $i < \omega$,

$$T \vdash \left({}^{\ulcorner}L_i \in \mathcal{NP}^{\urcorner T} \longrightarrow \overline{{}^{\ulcorner}L_i \in \mathcal{P}^{\urcorner T}}\right),$$

mas $T \vdash {}^{\ulcorner}L_e \in \mathcal{NP}^{\urcorner T}$ e $T \not\vdash \overline{{}^{\ulcorner}L_e \in \mathcal{P}^{\urcorner T}}$.

II.8.3 Observações

Não sabemos se a sentença original

$$\overline{{}^{\ulcorner}\mathcal{P} = \mathcal{NP}^{\urcorner T}} \;:=\; \forall''x < \omega'' \left(\overline{{}^{\ulcorner}L_x \in \mathcal{NP}^{\urcorner T}} \longrightarrow \overline{{}^{\ulcorner}L_x \in \mathcal{P}^{\urcorner T}}\right),$$

—mesmo contendo menos informação do que ${}^{\ulcorner}\mathcal{P} = \mathcal{NP}^{\urcorner T}$— é, ou não, indemonstrável em T.

Vejamos também que há diversos modos "não honestos" de se obter tal indemonstrabilidade. Seja σ uma sentença indemonstrável em T —assumindo T consistente— e suponha que exista $d < \omega$ tal que $T \vdash \overline{{}^{\ulcorner}L_{\boldsymbol{d}} \in \mathcal{NP}^{\urcorner T}}$. Defina

$${}^{\ulcorner}L_x \in \mathcal{P}^{\urcorner T} \;:=\; \overline{{}^{\ulcorner}L_x \in \mathcal{P}^{\urcorner T}} \wedge ((x = \boldsymbol{d}) \rightarrow \sigma).$$

Novamente podemos verificar de modo corriqueiro que essa fórmula também é uma certificação (fraca) da asserção "$L_x \in \mathcal{P}$" em T. Mas agora ${}^{\ulcorner}L_{\boldsymbol{d}} \in \mathcal{P}^{\urcorner T}$ é indemonstrável em T. Com efeito, se $T \vdash {}^{\ulcorner}L_{\boldsymbol{d}} \in \mathcal{P}^{\urcorner T}$, então $T \vdash \sigma$, contra a hipótese. Então a sentença

$$\forall''x < \omega'' \left(\overline{{}^{\ulcorner}L_x \in \mathcal{NP}^{\urcorner T}} \longrightarrow {}^{\ulcorner}L_x \in \mathcal{P}^{\urcorner T}\right)$$

também é indemonstrável em T. Não foi isso que fizemos.

As sentenças ${}^{\ulcorner}L_x \in \mathcal{P}^{\urcorner T}$ e $\overline{{}^{\ulcorner}L_x \in \mathcal{P}^{\urcorner T}}$ são certificações da asserção "$L_x \in \mathcal{P}$" em T e não existe nenhuma razão para preferir uma no lugar da outra. Em algum sentido, a última é até mais forte —porque consegue descrever mais informações— do que a primeira. Assim dizemos que essa extensão foi feita de um modo *honesto*.

Lembremos que sempre existem diversas maneiras distintas —mas, na metateoria, todas equivalentes entre si— de descrever formalmente uma mesma questão da Complexidade Computacional. Mais ainda, não podemos escolher uma delas e dizer que ela é a melhor.

II.8.4 Adicionando Axiomas

Suponha que a teoria T tenha um modelo $M \models T$ —pensemos ele como sendo um modelo *standard*. Sejam $\overline{\ulcorner L_x \in \mathcal{P}\urcorner^T}$ e $\overline{\ulcorner L_x \in \mathcal{NP}\urcorner^T}$ *expressando* as respectivas asserções sobre M; isso é, para todo $i < \omega$,

$$M \models \overline{\ulcorner L_i \in \mathcal{P}\urcorner^T} \iff L_i \in \mathcal{P},$$
$$M \models \overline{\ulcorner L_i \in \mathcal{NP}\urcorner^T} \iff L_i \in \mathcal{NP}.$$

Então ambas fórmulas $\overline{\ulcorner L_x \in \mathcal{P}\urcorner^T}$ e $\overline{\ulcorner L_x \in \mathcal{NP}\urcorner^T}$ são automaticamente certificações das respectivas asserções em T. Tome $e < \omega$ e as fórmulas $\ulcorner L_x \in \mathcal{NP}\urcorner^T$, $\overline{\ulcorner \mathcal{P} = \mathcal{NP}\urcorner^T}$ e $\ulcorner \mathcal{P} = \mathcal{NP}\urcorner^T$ definidos como acima. Assim

$$M \models \left(\ulcorner \mathcal{P} = \mathcal{NP}\urcorner^T \leftrightarrow \overline{\ulcorner \mathcal{P} = \mathcal{NP}\urcorner^T}\right)$$

e $T \not\vdash \ulcorner \mathcal{P} = \mathcal{NP}\urcorner^T$.

Quando $T \not\vdash \overline{\ulcorner \mathcal{P} = \mathcal{NP}\urcorner^T}$, podemos fazer a extensão

$$T' := T + \overline{\ulcorner \mathcal{P} = \mathcal{NP}\urcorner^T}.$$

Se acrescentamos de modo indiscriminado o axioma $\overline{\ulcorner \mathcal{P} = \mathcal{NP}\urcorner^T}$ —ou sua negação— em T, podemos perder as certificações de $\overline{\ulcorner L_x \in \mathcal{P}\urcorner^T}$ e $\overline{\ulcorner L_x \in \mathcal{NP}\urcorner^T}$ na extensão T'. Então, para garantir que elas permanecem sendo certificações em T', suporemos também que $M \models \overline{\ulcorner \mathcal{P} = \mathcal{NP}\urcorner^T}$, assim $M \models T'$ (e $\mathcal{P} = \mathcal{NP}$ vale na metateoria).

Portanto temos que $T' \vdash \overline{\ulcorner \mathcal{P} = \mathcal{NP}\urcorner^T}$ e $T' \not\vdash \ulcorner \mathcal{P} = \mathcal{NP}\urcorner^T$ (e $M \models \ulcorner \mathcal{P} = \mathcal{NP}\urcorner^T$).

II.8.5 Indemonstrabilidade com respeito a $\mathcal{P} \subseteq \mathcal{NP}$

Por outro lado, algumas coisas podem ser mais complicadas ainda. Tome um problema de decisão $L \in \mathcal{P}$. Pelo Corolário II.6–3, existe $e < \omega$ tal que $L_e = L$ ($\in \mathcal{P} \subseteq \mathcal{NP}$) e $T \not\vdash \overline{\ulcorner L_e \in \mathcal{NP}\urcorner^T}$. Seja a certificação em T definida por

$$\ulcorner L_x \in \mathcal{P}\urcorner^T := \overline{\ulcorner L_x \in \mathcal{P}\urcorner^T} \vee (x = e).$$

Portanto $T \vdash \ulcorner L_e \in \mathcal{P}\urcorner^T$ e $T \not\vdash \overline{\ulcorner L_e \in \mathcal{NP}\urcorner^T}$. Tomando a pseudo–certificação

$$\ulcorner \mathcal{P} \subseteq \mathcal{NP}\urcorner^T := \forall'' x < \omega'' \left(\ulcorner L_x \in \mathcal{P}\urcorner^T \longrightarrow \overline{\ulcorner L_x \in \mathcal{NP}\urcorner^T}\right)$$

em T, ela é indemonstrável, isto é,

$$T \not\vdash \ulcorner \mathcal{P} \subseteq \mathcal{NP}\urcorner^T.$$

Entretanto, se tudo estiver ocorrendo bem, precisamos ter que $T \vdash \overline{\ulcorner \mathcal{P} \subseteq \mathcal{NP}\urcorner^T}$, sendo

$$\overline{\ulcorner \mathcal{P} \subseteq \mathcal{NP}\urcorner^T} := \forall'' x < \omega'' \left(\overline{\ulcorner L_x \in \mathcal{P}\urcorner^T} \longrightarrow \overline{\ulcorner L_x \in \mathcal{NP}\urcorner^T}\right).$$

Isso porque a asserção $L_x \in \mathcal{P}$ pode ser somente uma restrição sintática da asserção $L_x \in \mathcal{NP}$ —em que não utiliza a assistência das testemunhas, assim como definimos anteriormente.

II.8.6 Não–construtibilidade

Agora desejamos estudar o outro lado. Sejam novamente ${}^{\ulcorner}L_x \in \mathcal{P}^{\urcorner_T}$ e ${}^{\ulcorner}L_x \in \mathcal{NP}^{\urcorner_T}$ certificações das respectivas asserções em T. Observe que $\{i < \omega \| T \vdash {}^{\ulcorner}L_i \in \mathcal{NP}^{\urcorner_T}\}$ é somente um *minúsculo subconjunto* de $\{i < \omega \| L_i \in \mathcal{NP}\}$, pois o último é um conjunto Σ_3–completo e o primeiro é somente um Σ_1–conjunto (veja Corolário II.1–2). O mesmo acontece com o conjunto $\{i < \omega \| T \vdash {}^{\ulcorner}L_i \notin \mathcal{P}^{\urcorner_T}\}$ em comparação com $\{i < \omega \| L_i \notin \mathcal{P}\}$.

Desse modo, sendo

$$ {}^{\ulcorner}L_x \in \mathcal{NP} \setminus \mathcal{P}^{\urcorner_T} \;:=\; {}^{\ulcorner}L_x \in \mathcal{NP}^{\urcorner_T} \wedge {}^{\ulcorner}L_x \notin \mathcal{P}^{\urcorner_T}, $$

a chance de existir um $e < \omega$ para o qual

$$ T \vdash {}^{\ulcorner}L_e \in \mathcal{NP} \setminus \mathcal{P}^{\urcorner_T} $$

é muito pequena. Talvez ele ainda nem exista, mesmo ocorrendo $\mathcal{P} \neq \mathcal{NP}$. E qual significado podemos tirar se, na verdade, *não existir* tal e acima?

Seja a sentença

$$ {}^{\ulcorner}\mathcal{P} \neq \mathcal{NP}^{\urcorner_T} \;:=\; \exists'' x < \omega'' \; {}^{\ulcorner}L_x \in \mathcal{NP} \setminus \mathcal{P}^{\urcorner_T}, $$

que precisa ser pensada como descrevendo que existe um $i < \omega$ para o qual $L_i \in \mathcal{NP} \setminus \mathcal{P}$ (ou $M \models {}^{\ulcorner}L_i \in \mathcal{NP} \setminus \mathcal{P}^{\urcorner_T}$, em que M é um modelo *standard* de T). Agora suponha que

$$ T \vdash {}^{\ulcorner}\mathcal{P} \neq \mathcal{NP}^{\urcorner_T}. $$

Esse fato pode ser suposto independentemente da sentença ${}^{\ulcorner}\mathcal{P} \neq \mathcal{NP}^{\urcorner_T}$ ser, ou não, agrupada à T por meio de um axioma.

Entretanto, em uma demonstração construtiva de

$$ T \vdash \exists'' x < \omega'' \; {}^{\ulcorner}L_x \in \mathcal{NP} \setminus \mathcal{P}^{\urcorner_T}, $$

podemos —fora de T, na metateoria— achar efetivamente $e < \omega$ tal que $T \vdash {}^{\ulcorner}L_e \in \mathcal{NP} \setminus \mathcal{P}^{\urcorner_T}$. Isso nos informa que a teoria T precisaria utilizar algum *processo não–construtivo* na demonstração da sentença ${}^{\ulcorner}\mathcal{P} \neq \mathcal{NP}^{\urcorner_T}$.

II.8.7 Considerações Complementares

Nossos resultados anteriores, especialmente os primeiros itens do Teorema II.2–1, mostram que muitas asserções da Teoria da Complexidade Computacional —tais como "$L \in \mathcal{NP}$"— não podem ser descritas em Sistema Lógicos Formais, talvez requerendo uma Lógica de Segunda Ordem. Essa situação é definitiva porque essas asserções foram classificadas como sendo Π_1–difíceis na Hierarquia da Complexidade Aritmética (Corolário II.1–2 e Teoremas II.1–3 e II.1–4).

Entretanto isto pode ser somente uma contingência. Quem sabe, a despeito de uma teoria axiomatizável —em Lógica Clássica de Primeira Ordem— não poder reconhecer por inteiro a relação

de pertinência em $\mathcal{NP}$, alguma Teoria dos Conjuntos pode ser capaz de provar todas as questões importantes envolvendo essa relação. Assim pretendemos discutir a possibilidade de uma teoria provar, ou refutar, a questão $\mathcal{P} = \mathcal{NP}$.

Se uma sentença ${}^{\ulcorner}\mathcal{P} = \mathcal{NP}^{\urcorner_T}$ que pseudo-certifica a questão $\mathcal{P} = \mathcal{NP}$ é independente de uma teoria axiomatizável T, podemos construir uma teoria T' mais forte, adicionando-a —ou sua negação— a T. Esse procedimento é muito comum e, de fato, verdadeiramente estabelecido na Matemática.

Mas, como conseqüência da Subseção II.8.4, se $\mathcal{P} = \mathcal{NP}$ vale, por mais que se adicione axiomas, nunca conseguimos chegar a uma teoria axiomatizável que prove a questão $\mathcal{P} = \mathcal{NP}$ de todos os modos (honestos) em que ela pode ser descrita. Isso é, em um Sistema Lógico Formal, sempre existe uma maneira de descrever $\mathcal{P} = \mathcal{NP}$ que a torna indemonstrável.

Diferentemente de outras questões da Matemática, essa questão parece ser melhor interpretada na metateoria —de um modo semântico— do que em uma teoria formal —de um modo sintático. Isso pode ser uma conseqüência das referidas classificações em níveis superiores na Hierarquia da Complexidade Aritmética.

As sentenças ${}^{\ulcorner}L_x \in \mathcal{P}^{\urcorner_T}$ e $\overline{{}^{\ulcorner}L_x \in \mathcal{P}^{\urcorner_T}}$ na Subseção II.8.2 —escritas em uma linguagem Lógica— são *descrições sintáticas* da mesma asserção. Elas são equivalentes em um modelo *standard*, mas podem ter uma condição de demonstrabilidade muito diferente em uma teoria.

Uma descrição sintática da questão $\mathcal{P} = \mathcal{NP}$ depende de: ***a)*** a definição de Máquina de Turing, de problema de decisão e das classes de complexidade $\mathcal{P}$ e $\mathcal{NP}$ —em que existem incontáveis definições equivalentes na metateoria; ***b)*** uma codificação aritmética dessas definições —em que também existem muitos modos de ser feita; e ***c)*** a interpretação dessa codificação na linguagem da teoria. Portanto a questão $\mathcal{P} = \mathcal{NP}$ pode ser provável em T somente para uma descrição sintática específica, mas não para as demais.

Mais ainda, no caso de $\mathcal{P} = \mathcal{NP}$ ser demonstrável em T, a Subseção II.8.5 nos mostra que o modo como se faz a descrição sintática dessa questão terá um papel importante na sua demonstração. Isso não é, em geral, o que acontece na Teoria da Complexidade Computacional. Regularmente trabalhamos somente com definições genéricas, tratando-as por *designações semânticas* e não levando em conta a sintaxe.

Por outro lado, argumentamos na Subseção II.8.6 por que há a possibilidade de que, para se obter $\mathcal{P} \neq \mathcal{NP}$, seja necessário uma *demonstração não-construtiva*. Em outras palavras, a prova de uma sentença ${}^{\ulcorner}\mathcal{P} \neq \mathcal{NP}^{\urcorner_T}$ dentro de uma teoria axiomatizável talvez requeira processos lógicos não-construtivos.

Lembramos que, no caso contrário, sendo $\mathcal{P} = \mathcal{NP}$, é essencial que a demonstração de tal questão seja construtiva. Isso porque não faz sentido algum existir uma Máquina de Turing determinística de tempo polinomial que possa resolver um problema de decisão $\mathcal{NP}$-completo, mas não exista um modo efetivo de se obter tal máquina.

Concluindo, o que aconteceria no caso de algumas afirmações necessitarem de uma Lógica de ordem superior para serem resolvidas? Não poderia ocorrer de as propriedades tratadas em cada uma dessas afirmações e que sejam inexpressáveis em uma Lógica Clássica de Primeira Ordem serem indispensáveis na sua demonstração?

Talvez não somente o problema $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$ seja independente das Teorias dos Conjuntos usuais, mas também a Matemática necessite construir uma Teoria dos Conjuntos de Segunda Ordem que seja capaz de resolver as questões em aberto da Teoria da Complexidade Computacional.

Capítulo III

Hierarquia da Complexidade Aritmética

III.1 Computações Não–determinísticas

Usamos $\natural$ como o *símbolo de separação* e $\not\flat$ como o *espaço em branco*, assumindo que essas duas letras não pertencem ao alfabeto Γ. Uma fita de entrada sempre começa a computação com somente uma palavra de Γ no seu início, tendo todas as demais posições à direita dessa palavra preenchidas com espaços em branco $\not\flat$. As *fita de trabalhos* são vazias no início das computações, i.e., com a totalidade das suas posições ocupadas por espaços em branco.

A primeira fita de entrada é a *fita de entrada–principal* e as demais são as *fitas de testemunha*. Se x e $\vec{y} := y_0, \ldots, y_{k-1}$ (com $k < \omega$) são palavras de Γ e M é uma Máquina de Turing, $M(x, \vec{y})$ denota a computação de M com a entrada x e as testemunhas $y_0, \ldots, y_{k-1}$. A entrada x é escrita na fita de entrada–principal e as testemunhas $\vec{y}$, uma em cada fita de testemunha. Se M tem mais do que k fitas de testemunha, o restante das fitas de testemunha são vazias. Por outro lado, M considerará somente as testemunhas que tenham fita de testemunha para tomá-las como entrada, negligenciando as demais "falsas testemunhas".

Uma Máquina de Turing M roda em tempo não–determinístico f sse, para todas entradas no formato $(x, \vec{y})$ (com $k < \omega$ variando também), $M(x, \vec{y})$ pára em no máximo $f(\|x\|)$ passos. Aqui $\|x\|$ é o comprimento da palavra x. Ainda, M roda em espaço não–determinístico h sse todas as cabeças de leitura de M alcançam no máximo $h(\|x\|)$ posições em cada fita de trabalho e de testemunha durante a computação de uma entrada $(x, \vec{y})$. Daí M tem tempo (ou espaço) não–determinístico polinomial, quando M roda em tempo (ou espaço) não–determinístico para um polinômio qualquer.

Uma computação determinística, i.e. $M(x)$, é um modo *restrito* da computação não–determinística, em que não se faz uso das testemunhas $\vec{y}$. Notemos que todos os nossos resultados referem-se a computações não–determinísticas, entretanto nossas provas não se preocuparão com as testemunhas. Logo os mesmos resultados também são válidos para *computações determinísticas*. Por

exemplo, no Teorema II.1–4, para as mesmas funções recursivas f e g, os conjuntos

$$\{\, i < \omega \parallel M_i \text{ roda em tempo determinístico } f \,\}$$

e

$$\{\, i < \omega \parallel M_i \text{ roda em espaço determinístico } h \,\}$$

são Π_1–completos.

***Relembrando o Teorema* II.1–3** *Cada subconjunto de números naturais entre:*

ou

$$\{\, i < \omega \parallel M_i \textit{ tem tempo não–determinístico linear com fator polilogarítmico} \,\},$$

ou

$$\{\, i < \omega \parallel M_i \textit{ tem espaço não–determinístico logarítmico} \,\};$$

e

$$\{\, i < \omega \parallel M_i \textit{ tem espaço não–determinístico exponencial} \,\}$$

é Σ_2–difícil.

Mais ainda, esses conjuntos que forem definidos por Máquinas de Turing limitadas por: tempo não–determinístico, ou linear com fator polilogarítmico, ou quadrático, ou polinomial, ou exponencial, ou $\mathcal{O}_{\mathrm{PLog}}(f)$, ou $\mathcal{O}(f^2)$, ou Poly(f) *(com $f \geq$ Id,[i] uma função recursiva); ou espaço não–determinístico, ou logarítmico, ou polinomial, ou exponencial, ou $\mathcal{O}(h)$, ou* Poly(h) *(com $h \geq$ log,[i] uma função recursiva) são Σ_2–completos.*

Prova. Separamos a prova em duas partes: uma relacionada com tempo e outra com espaço. Por simplicidade, obteremos somente que as asserções ***A)*** "M_i tem tempo não–determinístico polinomial" e ***B)*** "M_i tem espaço não–determinístico logarítmico" são Σ_2–completas. Contudo, não haverá problema em completar esse resultado para que todos os conjuntos entre

$$\{\, i < \omega \parallel M_i \text{ tem tempo não–determinístico linear com fator polilogarítmico} \,\}$$

e, ou

$$\{\, i < \omega \parallel M_i \text{ tem tempo não–determinístico exponencial} \,\},$$

ou

$$\{\, i < \omega \parallel M_i \text{ tem tempo não–determinístico Poly}(f) \,\}$$

são Σ_2–difíceis, em que $f \geq$ Id é uma função recursiva. O mesmo acontece com os conjuntos entre

$$\{\, i < \omega \parallel M_i \text{ tem espaço não–determinístico logarítmico} \,\}$$

e, ou

$$\{\, i < \omega \parallel M_i \text{ tem espaço não–determinístico exponencial} \,\}$$

ou, com $h \geq$ log uma função recursiva,

$$\{\, i < \omega \parallel M_i \text{ tem espaço não–determinístico Poly}(h) \,\}.$$

[i] Assim, para todo $m \in \mathbb{N}$, $f(m) \geq \mathrm{Id}(m) := m$ e $h(m) \geq \log(m)$.

A) Esta parte, com respeito aos limites de tempo, é baseada na prova de Hájek [Háj79] e foi modificada somente no sentido de se levar em conta as testemunhas. Não é difícil escrever

$$\ulcorner M_x \text{ tem tempo não–determinístico polinomial} \urcorner$$

como uma Σ_2–fórmula. Então focaremos na prova de que o conjunto

$$\{\, i < \omega \parallel M_i \text{ tem tempo não–determinístico polinomial} \,\}$$

é Σ_2–difícil. Para fazer isso, apenas usaremos o fato de que o conjunto $\{i < \omega \parallel W_i \text{ é finito}\}$ é Σ_2–completo e procuraremos por uma redução dele.

Tome $b(m) := 2^m$, em que $m \in \mathbb{N}$, —ou outra função recursiva que domine todos os polinômios. Para $i < \omega$, seja $M_{\sigma(i)}$ uma Máquina de Turing com uma entrada $(zx, \vec{y})$. Aqui, z, x e $\vec{y} := y_0, \ldots, y_{k-1}$ (com $k < \omega$) são palavras sobre o alfabeto $\Gamma \dot{\cup} \{\natural\}$ tais que a palavra z é formada somente pela letra $\natural$ (i.e., $z := \natural^{\|z\|}$) e x é a palavra vazia ou tem a sua primeira letra no alfabeto Γ.

A computação de $M_{\sigma(i)}(zx, \vec{y})$ executa duas tarefas ao mesmo tempo: conta o passo–número da sua computação; e faz a simulação da computação de $M_i(x)$ nos seus $\|z\|$ primeiros passos. Ela determina o tempo–limite (i.e., $\|z\|$ passos) da simulação de $M_i(x)$ através de um dispositivo–de–tempo (i.e., *a clock device*). Antes da simulação de $M_i(x)$, algumas "preparações para a simulação" precisam ser feitas: copiar z para uma *nova* fita de trabalho; procurar em x se não existe uma letra $\natural$, parando imediatamente se existir; e colocar a cabeça de leitura no começo de x. Logo a simulação de $M_i(x)$ pode ser feita sobre x escrita na fita de entrada, mas agora tomando cuidado para tratar corretamente o início de x —não permitindo qualquer ultrapassagem da cabeça de leitura por sobre a palavra z. Mais ainda,

- ***a)*** Se $M_i(x)$ pára exatamente no seu $\|z\|$-ésimo passo–número de simulação, então forçamos $M_{\sigma(i)}(zx, \vec{y})$ a parar em, no mínimo, seu $b(\|zx\|)$-ésimo passo.
- ***b)*** Caso contrário, $M_{\sigma(i)}(zx, \vec{y})$ deve parar imediatamente depois da simulação.

Durante toda a computação, as testemunhas $\vec{y}$ não são levadas em conta. Concluímos que a computação de $M_{\sigma(i)}(zx, \vec{y})$ gasta no máximo da ordem linear de $\|zx\|$ passos para se preparar para a simulação de $M_i(x)$ e gasta mais no máximo da ordem linear de $\|z\|$ passos na simulação em si![ii]

Note que, na verdade, $M_{\sigma(i)}$ trabalha no alfabeto $\Gamma \dot{\cup} \{\natural\}$, mas, para colocar isso de acordo com nossa definição de computação de Máquinas de Turing, não há dificuldade em computar $M_{\sigma(i)}$ somente no alfabeto Γ com perda de tempo de ordem linear.

Também, para possibilitar que essa computação execute a simulação e cheque o dispositivo–de–tempo ao mesmo tempo, fizemos uso de uma *nova* fita de trabalho. Na nossa definição geral de Máquina de Turing, isso é permitido. Entretanto, na nossa definição *restrita* de Máquina de Turing —empregada no Teorema da Máquina de Turing Universal e da Parametrização

[ii] No nosso modelo computacional, por exemplo, podemos fazer a simulação de $\|z\|$ passos de $M_i(x)$ basicamente colocando dois comandos —ver se o tempo–limite $\|z\|$ no dispositivo–de–tempo não se expirou e pular uma posição nesse dispositivo–de–tempo— antes de cada comando de M_i. Precisamos apenas cuidar do comando de parada de M_i e ler corretamente x na fita de entrada —não considerando também z como uma entrada. Desse modo, gastamos $3\|z\| + 2$ passos para fazer essa simulação. Entretanto, as constantes exatas dependem do modelo computacional adotado.

II.4–1—, uma computação só pode usar duas fitas de trabalho. Assim temos uma perda de tempo de ordem linear com fator polilogarítmico —ou de ordem quadrática, ou de ordem polinomial— para simular todas essas fitas de trabalho em somente duas delas (veja [HU79, Pap94]). Como conseqüência, infelizmente não pudemos tratar da prova da Σ_2–completeza da asserção "M_i tem tempo não–determinístico linear" —que, na verdade, foi provado para a asserção "M_i tem tempo não–determinístico linear com fator polilogarítmico"– nesse nosso modelo computacional *restrito*.

Portanto, quando o Item *b)* acontece, $M_{\sigma(i)}(zx, \vec{y})$ roda em no máximo $g(\|zx\|)$ passos, com $g \in \mathcal{O}_{\mathrm{PLog}}(\mathrm{Id})$ —uma função linear com fator polilogarítmico![iii] Note também que a função g depende apenas do modelo computacional tomado por nós.

Mostremos a redução. Supondo $x \in W_i$, sejam $z_x \in \{\natural\}^*$ tal que $\|z_x\|$ seja o passo–número para o qual $M_i(x)$ pára e $s_x \in \mathbb{N}$ o passo–número tal que $M_{\sigma(i)}(z_x x, \vec{y})$ pare. O Item *a)* ocorrerá para essa entrada, assim $s_x \geq b(\|z_x x\|)$. Agora, se W_i é infinito, existem infinitos x's em W_i, logo que $s_x \geq b(\|z_x x\|)$ $(:= 2^{\|z_x x\|})$. Portanto o tempo de $M_{\sigma(i)}$ não pode ser limitado polinomialmente. Por outro lado, quando $x \notin W_i$, o Item *a)* nunca acontece. Então, se W_i é finito, $M_{\sigma(i)}(zx, \vec{y})$ pára em não mais do que $g(\|zx\|) + \max_{v \in W_i} s_v$ passos. Daí, $M_{\sigma(i)}$ tem tempo não–determinístico linear com fator polilogarítmico (portanto polinomial).

Podemos construir efetivamente a máquina $M_{\sigma(i)}$ baseada na máquina M_i —ou, melhor, no número $i < \omega$. Então, podemos tomar a função σ como sendo uma função recursiva e, pela redução do conjunto $\{i < \omega \parallel W_i \text{ é finito}\}$ para o conjunto $\{i < \omega \parallel M_i \text{ tem tempo não–determinístico polinomial}\}$, provamos que o último conjunto é Σ_2–completo.

B) Agora provaremos os limites de espaço, em que apenas levaremos em conta o caso do espaço logarítmico, pela analogia com os demais. Tome $\#$ um segundo (e distinto) símbolo de separação e a entrada será considerada no formato $(zvx, \vec{y})$, em que novamente z é formada apenas por letras $\natural$, v é formada apenas por letras $\natural$ e $\#$, começando por $\#$, x tem a primeira letra no alfabeto Γ e $\vec{y}$ são $k < \omega$ testemunhas de Γ. A palavra v serve somente para forçar que a simulação possa parar —de fato, isso não é necessário.

Nessa parte, a computação de $M_{\sigma(i)}(zvx, \vec{y})$ primeiro procura pelas letras $\natural$ e $\#$ em x, e pára se achar uma delas. Depois disso, ela escreve o número $\|z\|$, mais uma vez, em uma *nova* fita de trabalho (em espaço $\log(\|z\|)$); marca a $\log(\|zvx\|)$-ésima posição em cada fita de trabalho de M_i com a nova letra $\natural$; e simula a computação de $M_i(x)$ nos seus primeiros $\|z\|$ passos. Para saber quando o tempo de simulação terminou, usa-se o contador $\|z\|$ —escrita nessa segunda nova fita de trabalho— como um dispositivo–de–tempo. Completando a descrição:

- ***a)*** Em cada passo da simulação de $M_i(x)$, $M_{\sigma(i)}(zvx, \vec{y})$ faz uso da letra–marca $\natural$ para saber se alguma fita de trabalho intenciona ocupar mais do que $\log(\|zvx\|)$ posições e, se isso acontece, pára imediatamente.

- ***b)*** Se $M_i(x)$ parou exatamente no seu $\|z\|$-ésimo passo–número de simulação, forçamos $M_{\sigma(i)}(zvx, \vec{y})$ a usar pelo menos $b(\|zvx\|)$ posições de alguma fita de trabalho e a parar logo depois.

[iii] Isso é, $g(m) := am \log^c(m) + b$, para cada $a, b, c \in \mathbb{N}$.

c) Nos outros casos em que a simulação de $M_i(x)$ pára depois de $\|z\|$ passos, $M_{\sigma(i)}(zvx, \vec{y})$ irá apenas parar.

Note que $M_{\sigma(i)}(zvx, \vec{y})$ sempre pára e denote o espaço utilizado por $s_{zvx} \in \mathbb{N}$. Se $x \notin W_i$, então os Itens *b)* e *c)* nunca ocorrem e $s_{zvx} \leq \log(\|zvx\|)$, pelo Item *a)*. Supondo $x \in W_i$, sejam $z_x \in \{\natural\}^*$ e $v_x \in \{\#\}^*$ tais que $\|z_x\|$ é o passo–número para o qual $M_i(x)$ pára e $\|v_x\|$ é a exponencial do espaço gasto por $M_i(x)$. Então $M_{\sigma(i)}(z_x v_x x, \vec{y})$ gasta espaço $\log(\|z_x v_x x\|)$ na computação inicial e espaço $\log(\|v_x\|) \leq \log(\|z_x v_x x\|)$ durante a simulação de $M_i(x)$. Daí, o Item *a)* nunca ocorrerá, parando a computação pelo Item *b)*, assim $s_{z_x v_x x} \geq b(\|z_x v_x x\|)$ $(:= 2^{\|z_x v_x x\|})$. Portanto, a redução, focando-se no caso em que W_i é ou não finito, é análoga à anterior.

Do mesmo modo, podemos simular o alfabeto $\Gamma \cup \{\natural, \#\}$ no alfabeto Γ com perda de espaço de ordem linear. Entretanto, diferentemente dos limites de tempo, também temos apenas uma perda de espaço de ordem linear para simular todas as fitas de trabalho em apenas duas (veja [HU79, Pap94]). Portanto esse resultado não depende da escolha da nossas definições *restrita* ou *genérica* de Máquina de Turing. [II.1–3] ■

Com essa prova em mãos, a próxima se torna fácil.

***Relembrando o Teorema* II.1–4** *Existem funções recursivas* $f' \in \mathcal{O}_{\mathrm{PLog}}(\mathrm{Id}) \subseteq \mathcal{O}(\mathrm{Id}^2) \subseteq \mathrm{Poly}(\mathrm{Id})$[iv] *—que tem ordem linear com fator polilogarítmico (ou ordem quadrática, ou ordem polinomial)— e* $h' \in \mathcal{O}(\log)$ *—que tem ordem logarítmica— tais que, para todas funções recursivas* $f \geq f'$ *e* $h \geq h'$*, temos que os conjuntos*

$$\{\, i < \omega \parallel M_i \textit{ roda em tempo não–determinístico } f \,\}$$

e

$$\{\, i < \omega \parallel M_i \textit{ roda em espaço não–determinístico } h \,\}$$

são Π_1*–completos.*

Prova. Fixando as palavras x e $\vec{y}$ de Γ, a asserção "$M_i(x, \vec{y})$ pára em no máximo $f(\|x\|)$ passos" é recursiva, portanto define um Δ_1–conjunto. Para se ver que a asserção "M_i roda em tempo não–determinístico f" define um Π_1–conjunto, necessitamos apenas quantificar universalmente as entradas x e $\vec{y}$. Por outro lado, na asserção "$M_i(x, \vec{y})$ usa no máximo $h(\|x\|)$ posições", $M_i(x, \vec{y})$ pode nunca parar, logo ela poderia não ser recursiva. Mas sabemos que existe uma função recursiva g para a qual $M_i(x, \vec{y})$ entrará em *loop* —apenas repetindo passos anteriores—, se ela rodar em espaço limitado por $h(\|x\|)$ e não parar antes de $g(\|x\|)$ passos. Então a asserção "$M_i(x, \vec{y})$ usa no máximo $h(\|x\|)$ posições" também é recursiva.

O conjunto $\{\, i < \omega \parallel \epsilon \notin W_i \,\}$ é Π_1–completo, em que ϵ é a *palavra vazia*. De um modo similar à prova acima, podemos demonstrar que existem reduções desse conjunto aos conjuntos

$$\{\, i < \omega \parallel M_i \text{ roda em tempo não–determinístico } f \,\}$$

e

$$\{\, i < \omega \parallel M_i \text{ roda em espaço não–determinístico } h \,\},$$

[iv] Lembremos que Id é a função identidade.

daí chegamos à Π_1–dificuldade deles.

Seja $M_{\sigma(i)}(x, \vec{y})$ a Máquina de Turing que simula a computação de $M_i(\epsilon)$ nos seus primeiros $\|x\|$ passos. Como visto na prova anterior, existe uma função recursiva linear com fator polilogarítmico f' —que depende somente do modelo computacional adotado— tal que $M_{\sigma(i)}(x, \vec{y})$ gasta no máximo $f'(\|x\|) - 1$ passos para fazer essa simulação. Então: ***a)***, se a simulação termina naquele ponto, $M_{\sigma(i)}(x, \vec{y})$ gastará mais $f(\|x\|) + 1$ passos; e ***b)***, caso contrário, ela parará imediatamente depois.

Se $\epsilon \notin W_i$, o Item *a)* não acontece e $M_{\sigma(i)}$ roda em tempo não–determinístico $f' \leq f$. Agora, se $\epsilon \in W_i$, seja uma palavra x de Γ tal que $\|x\|$ é o passo–número para o qual $M_i(\epsilon)$ pára. Então $M_{\sigma(i)}(x, \vec{y})$ não pára em $f(\|x\|) < f(\|x\|) + 1$ passos.

No caso do limitante de espaço não–determinístico, novamente a prova acima fornece uma função recursiva logarítmica h' que permite $M_{\sigma(i)}(x, \vec{y})$ simular $M_i(\epsilon)$ nos seus primeiros $\|x\|$ passos, não usando mais do que $h'(\|x\|)$ posições. Para isso, $M_{\sigma(i)}$ marcará a $\log(\|x\|)$-ésima posição de cada fita de trabalho de M_i com a nova letra $\natural$. Para obter h', simulamos cada letra em $\Gamma \cup \{\natural\}$ com duas letras em Γ —que é nosso alfabeto "oficial"—, daí $h' := 2h$. Portanto: ***a)***, se a simulação alcança a letra–marca $\natural$ em alguma fita de trabalho, $M_{\sigma(i)}(x, \vec{y})$ somente pára; ***b)***, se a simulação pára no seu $\|x\|$-ésimo passo, ela ocupa pelo menos $h(\|x\|) + 1$ posições em uma fita de trabalho; e ***c)***, caso contrário, ela parará imediatamente depois.

Novamente a prova se completa com alguns comentários. Se $\epsilon \notin W_i$, o Item *b)* nunca acontece e $M_{\sigma(i)}$ roda em espaço não–determinístico $h' \leq h$. Se $\epsilon \in W_i$, existe uma palavra x que fará $M_{\sigma(i)}(x, \vec{y})$ ocupar mais do que $h(\|x\|)$ posições em uma fita de trabalho. Então provamos que $\epsilon \notin W_i$ sse $M_{\sigma(i)}$ roda em espaço não–determinístico h.

Note que, quando estamos baseados no modelo computacional da nossa definição de Máquinas de Turing mais *restrita* —que apenas nos permite usar duas fitas de trabalho—, no caso do limitante de tempo, f' precisa, na verdade, ser linear com um fator polilogarítmico. Todavia, isso não é problema no caso do limitante de espaço, em que h' pode continuar sendo logarítmico.

[II.1–4] ■

III.2 Classes de Complexidade Não–determinísticas

Duas Máquinas de Turing são de aceite equivalente sse elas sempre têm a mesma resposta, para as mesmas entradas. Através do uso de um dispositivo–de–tempo (i.e., *a clock device*), podemos supor que existe uma enumeração recursiva $E_0, E_1, \ldots$ de todas as maquinas de tempo não–determinístico exponencial (módulo aceite equivalente). Então a classe de tempo exponencial pode ser escrita como $\mathcal{EXP} := \{\mathrm{L}(E_t) \parallel t < \omega\}$.

Podemos tomar o mesmo tipo de enumeração sobre máquinas com tempo não–determinístico linear, ou com espaço não–determinístico logarítmico, etc.. Em particular, fixamos a enumeração recursiva $P_0, P_1, \ldots$ de todas as Máquinas de Turing de tempo não–determinístico polinomial (novamente módulo aceite equivalente). Também é possível encontrar recursivamente uma

função polinomial p_t (com $t < \omega$) que limita não–deterministicamente o tempo da máquina P_t.

Agora podemos obter o resultado do Teorema II.1–1 com uma prova inspirada na de Hájek. O formato da prova aqui apresentado nos possibilitará modificá-la um pouco, no sentido de se levar em conta a computação não–determinística, o que permitiria fazer uma prova direta para, e.g., a hierarquia polinomial do Corolário II.1–2. Entretanto ela será demonstrada indiretamente logo abaixo, valendo-se do intervalo do teorema a seguir.

***Relembrando o Teorema* II.1–1** *Toda classe de complexidade $\mathcal{C}$ entre $\mathcal{LIN}$ e $\mathcal{EXP}$ é Σ_3–difícil, i.e., o conjunto $\{ i < \omega \parallel L_i \in \mathcal{C} \}$ é Σ_3–difícil. Isso acontece independentemente de $\mathcal{C}$ ser, ou não, uma classe de complexidade definida sobre propriedades computacionais. Em particular, as classes de complexidade $\mathcal{LIN}$, $\mathcal{L}$, $\mathcal{P}$, $\mathcal{PSPACE}$, $\mathcal{EXP}$, etc. são Σ_3–completas.*

Mais ainda, os subconjuntos de números naturais C tais que

$$\{i < \omega \| L_i \in \mathcal{LIN}\} \subseteq C \subseteq \{i < \omega \| L_i \in \mathcal{EXP}\}$$

constituem um sub-reticulado de conjuntos Σ_3–difíceis.

Prova. Com essas definições de classe de complexidade, podemos facilmente verificar que $\mathcal{LIN}$, $\mathcal{L}$, $\mathcal{P}$, $\mathcal{PSPACE}$, $\mathcal{EXP}$, etc. são Σ_3–conjuntos. Logo resta provar que uma classe de complexidade $\mathcal{C}$ entre $\mathcal{LIN}$ e $\mathcal{EXP}$ é Σ_3–difícil. Para fazer isso, procuraremos por uma redução do conjunto Σ_3–completo $\{i < \omega \parallel W_i \text{ é co–finito}\}$ para $\{i < \omega \parallel L_i \in \mathcal{C}\}$.

Tome $i < \omega$. Construiremos efetivamente um problema de decisão $C_i \subseteq \Gamma^*$ tal que seja recursivamente enumerável e possamos provar que, se W_i é co–finito, então $C_i \in \mathcal{LIN} \subseteq \mathcal{C}$; e, se $C_i \in \mathcal{EXP} \supseteq \mathcal{C}$, então W_i é co–finito. Isso significa que tomamos mecanicamente (com respeito a i) uma Máquina de Turing N_i que enumera os elementos de C_i. Com isso às mãos, construímos efetivamente outra máquina $M_{\sigma(i)}$ que aceita o problema de decisão C_i, i.e., $C_i = L_{\sigma(i)} := \mathrm{L}(M_{\sigma(i)})$. Portanto temos uma função recursiva σ que faz a redução que estamos procurando.

Construiremos C_i em estágios, tomando crescentes n's em ω:

- ***a)*** Para cada $t \leq n$, inclua o par $(\mathtt{n}, \mathtt{t})$ (ou a sua codificação) em C_i sse E_t não aceitar $(\mathtt{n}, \mathtt{t})$.
- ***b)*** Para cada $q \leq n$, inclua todos os pares $(\mathtt{q}, 0)$, $(\mathtt{q}, 1), \ldots,$ $(\mathtt{q}, \mathtt{q})$ em C_i sse $M_i(\mathtt{q})$ parar em no máximo n passos.

Defina $D := \{ (\mathtt{n}, \mathtt{q}) \parallel q \leq n < \omega \}$. Observe que $C_i \subseteq D$, $D \in \mathcal{LIN}$, e, se $\mathtt{q} \in W_i$, pelo Item *b)*, temos que $(\mathtt{q}, 0), (\mathtt{q}, 1), \ldots, (\mathtt{q}, \mathtt{q}) \in C_i$. Daí, se W_i é co–finito, então $D \setminus C_i$ é finito e $C_i \in \mathcal{LIN}$. Por outro lado, se W_i é co–infinito, mostraremos que $C_i \notin \mathcal{EXP}$. Na verdade, tome $t, q < \omega$ tais que $t \leq q$ e $\mathtt{q} \notin W_i$. Logo o Item *b)* nunca incluirá o par $(\mathtt{q}, \mathtt{t})$ em C_i, apesar de todo o tempo que possa ser dado (i.e., não importando quão grande $n < \omega$ é tomado) para a simulação de $M_i(\mathtt{q})$. Pelo Item *a)*, temos que $(\mathtt{q}, \mathtt{t}) \in C_i$ sse E_t não aceitar $(\mathtt{q}, \mathtt{t})$, portanto $C_i \neq \mathrm{L}(E_t)$, como queríamos. [II.1–1] ■

Agora estaremos trabalhando com as classes de complexidade não–determinísticas. Sabemos que cada nível da hierarquia polinomial Σ_n^{p} e Π_n^{p} (com $n < \omega$) e a hierarquia polinomial como

um todo $\mathcal{PH}$ são Σ_3–difíceis pelo Teorema II.1–1. Portanto, para mostrar que eles são Σ_3–completos, apenas resta saber se eles são Σ_3–conjuntos.

Definição III.2–1. *Quando $n < \omega$, $x \in \Gamma^*$ é uma palavra e M é uma Máquina de Turing, dizemos que M tem um Σ_n–esquema de testemunha para x sse tivermos que*

$$\exists y_0 \in \Gamma^* \quad \forall y_1 \in \Gamma^* \quad \ldots \quad Q y_{n-1} \in \Gamma^*$$

tal que $M(x, y_0, \ldots, y_{n-1})$ tem aceite —ou $(x, y_0, \ldots, y_{n-1}) \in \mathrm{L}(M)$. Aqui, como é usual, Q representa um dos dois quantificadores $\exists$ e $\forall$, dependendo da paridade de n. Para um Π_n–esquema de testemunha para x, precisamos ter

$$\forall y_0 \in \Gamma^* \quad \exists y_1 \in \Gamma^* \quad \ldots \quad Q y_{n-1} \in \Gamma^*,$$

com aceite de $M(x, y_0, \ldots, y_{n-1})$. Denotamos o problema de decisão Σ_n–aceitável de M por

$$\mathrm{L}\Sigma_n(M) \;:=\; \{\, x \in \Gamma^* \parallel M \text{ tem o } \Sigma_n\text{–esquema de testemunha para } x \,\}.$$

Analogamente, temos o problema de decisão Π_n–aceitável de M. Portanto podemos definir

$$\Sigma_n^{\mathrm{p}} \;:=\; \{\, \mathrm{L}\Sigma_n(P_t) \parallel t < \omega \,\},$$

$$\Pi_n^{\mathrm{p}} \;:=\; \{\, \mathrm{L}\Pi_n(P_t) \parallel t < \omega \,\}$$

e

$$\mathcal{PH} \;:=\; \bigcup_{n<\omega} \Sigma_n^{\mathrm{p}} \;=\; \bigcup_{n<\omega} \Pi_n^{\mathrm{p}}.$$

***Relembrando o Corolário* II.1–2** *Cada nível da hierarquia polinomial Π_n^{p} e Σ_n^{p} (incluindo as classes $\mathcal{NP}$ e co–$\mathcal{NP}$), toda a hierarquia polinomial $\mathcal{PH}$ e outras classes não–determinísticas (como $\mathcal{NL}$) são Σ_3–completas, i.e., para $n < \omega$, os conjuntos*

$$\{\, i < \omega \parallel L_i \in \Sigma_n^{\mathrm{p}} \,\}, \quad \{\, i < \omega \parallel L_i \in \Pi_n^{\mathrm{p}} \,\},$$

$$\{\, i < \omega \parallel L_i \in \mathcal{PH} \,\} \quad e \quad \{\, i < \omega \parallel L_i \in \mathcal{NL} \,\}$$

são Σ_3–completos.

Prova. Sabemos que cada Máquina de Turing P_t (com $t < \omega$) tem tempo não–determinístico limitado por uma função polinomial p_t. Então, para cada $x \in \Gamma^*$, $x \in \mathrm{L}\Sigma_n(P_t)$ sse

$$\exists y_0 \in \Gamma^{p_t(\|x\|)} \quad \forall y_1 \in \Gamma^{p_t(\|x\|)} \quad \ldots \quad Q y_{n-1} \in \Gamma^{p_t(\|x\|)}$$

para o qual $P_t(x, y_0, \ldots, y_{n-1})$ tenha aceite. O mesmo vale para o caso $\mathrm{L}\Pi_n(P_t)$. Logo eles podem ser checados recursivamente —i.e., são uma Δ_1–asserção. Daí conseguimos verificar que "$L_i \in \mathcal{PH}$" é uma Σ_3–asserção, por exemplo, tomando a fórmula, grosso modo,

$$\exists n, t \quad \forall x \;\; (x \in L_i \;\leftrightarrow\; x \in \mathrm{L}\Sigma_n(P_t)).$$

Lembre-se que "$x \in L_i$" é recursivamente enumerável, logo uma Σ_1–asserção. Assim esse corolário é uma conseqüência do Teorema II.1–1. [II.1–2] ■

Capítulo IV

Indemonstrabilidade e Independência

IV.1 Imposição Efetiva de uma Indemonstrabilidade

Derivaremos novamente alguns dos resultados, que se referem às asserções que impõem uma indemonstrabilidade em teorias axiomatizáveis, previamente obtidos. Entretanto, aqui, não faremos uso de questões de completeza e —o que é importante— agora mostraremos que eles imporão *efetivamente* uma indemonstrabilidade nessas teorias.

Começamos seguindo Hartmanis e Hopcroft [HH76]. As suas provas são baseadas somente no formato clássico do Teorema do Ponto Fixo —e não na nossa versão expandida (II.4–2), que será necessária nas demais provas.

Lema IV.1–1 (Hartmanis e Hopcroft [HH76]). *Seja T uma teoria axiomatizável. Para todo $q < \omega$, a asserção "$M_x(\mathrm{q})$ não pára" —em que esse x é a única variável livre— impõe efetivamente uma indemonstrabilidade em T.*

Portanto, se T é uma teoria consistente, para cada $q < \omega$, a asserção "$M_x(\mathrm{q})$ pára" não tem, nem uma representação, nem uma expressão em T.

Além disso, podemos efetivamente encontrar $e < \omega$ tal que, para cada $q < \omega$, $M_e(\mathrm{q})$ não pára e uma fórmula fixada $\ulcorner M_e(\mathbf{q}) \text{ não pára}\urcorner^T$ —que certifica fracamente a asserção correspondente em T— é indemonstrável em T.

Se T é uma extensão consistente da Aritmética de Robinson, PA^-, [Kay91, HP93] e $R(x)$ é uma asserção recursiva, existe uma Δ_1–representação $\ulcorner R(x)\urcorner^T$ em T. Claramente a asserção "$M_x(\mathrm{q})$ pára" é conhecida como o *Problema da parada*, que não é recursiva.

Prova. Seja $\ulcorner M_x(y) \text{ não pára}\urcorner^T$ (com variáveis livres x e y) uma certificação fraca em T da

respectiva asserção. Tome a Máquina de Turing definida, para $i, q < \omega$, como segue

$$M(\mathtt{i},\mathtt{q}) := \begin{cases} \text{pára}, & \text{se } T \vdash\!\!\!\!- \ulcorner M_i(\mathfrak{q}) \text{ não pára}\urcorner^T; \text{ e} \\ \text{não pára}, & \text{caso contrário.} \end{cases}$$

Na realidade, $M(\mathtt{i},\mathtt{q})$ efetivamente procura pela prova de $T \vdash\!\!\!\!- \ulcorner M_i(\mathfrak{q})$ não pára$\urcorner^T$ e isso é possível porque T é uma teoria axiomatizável. É importante notar que M pode ser efetivamente construída e então, pelo Teorema do Ponto Fixo II.4–2 na versão usual, efetivamente existe $e < \omega$ para o qual $M(\mathtt{e}, x) = M_e(x)$, para cada palavra x de Γ.

Fixe $q < \omega$ e temos que $M_e(\mathtt{q})$ pára sse $T \vdash\!\!\!\!- \ulcorner M_e(\mathfrak{q})$ não pára$\urcorner^T$. Em vista da definição de certificação fraca em T, $M_e(\mathtt{q})$ não pára. Então, $T \not\vdash\!\!\!\!- \ulcorner M_e(\mathfrak{q})$ não pára$\urcorner^T$. [IV.1–1] ■

Agora, as próximas provas requererão o Teorema do Ponto Fixo II.4–2 no seu formato integral.

***Relembrando o Teorema* II.5–1** *Seja T uma teoria axiomatizável. Então cada asserção descrita no Teorema* II.2–1.(ii) *impõe* `efetivamente` *uma indemonstrabilidade na teoria T.*

Prova. Primeiro, somente consideraremos o caso de tempo não–determinístico polinomial. Seja a função recursiva $b(m) := 2^m$, em que $m \in \mathbb{N}$. Tome uma certificação fraca $\ulcorner M_x$ tem tempo não–determinístico polinomial$\urcorner^T$ (com x sendo a única variável livre) e uma Máquina de Turing $N(\mathtt{i})$ "efetivamente construída" que verifica se $T \vdash\!\!\!\!- \ulcorner M_i$ tem tempo não–determinístico polinomial$\urcorner^T$ (com $i < \omega$).

Então podemos efetivamente definir a máquina $M(\mathtt{i}, x)$ —com somente duas fitas de entrada— que, com $i < \omega$ e uma palavra x de Γ, simula $N(\mathtt{i})$ nos seus primeiros $\|x\|$ passos. Para fazer isso, $M(\mathtt{i}, x)$ gasta da ordem linear de $\|x\|$ passos. Depois disso,

$$M(\mathtt{i}, x) := \begin{cases} \text{pára em no mínimo } b(\|x\|) \text{ passos}, & \text{se foi possível alcançar o aceite na simulação de } N(\mathtt{i}); \\ \text{apenas pára}, & \text{caso contrário.} \end{cases}$$

Note que, pela definição, $M^{(\mathtt{i})}$ tem tempo não–determinístico polinomial[i] —sse $M^{(\mathtt{i})}$ tem tempo não–determinístico linear— sse $T \not\vdash\!\!\!\!- \ulcorner M_i$ tem tempo não–determinístico polinomial$\urcorner^T$. Agora, usando o Teorema do Ponto Fixo II.4–2, podemos efetivamente encontrar $d < \omega$ para o qual $M^{(\mathtt{d})} \preccurlyeq M_d$ — assim M_d tem somente uma fita de entrada e não usa testemunhas. Daí as Máquinas de Turing $M^{(\mathtt{d})}$ e M_d, para entradas x variando nas palavras de Γ, tem a mesma computação e a segunda tem perda de tempo de ordem linear com fator polilogarítmico, com respeito à primeira.

Suponha que M_d não tenha tempo não–determinístico polinomial. Logo $M^{(\mathtt{d})}$ também não tem, assim $T \vdash\!\!\!\!- \ulcorner M_d$ tem tempo não–determinístico polinomial$\urcorner^T$; um absurdo, por causa da definição de certificação fraca. Então M_d tem tempo não–determinístico polinomial, daí $M^{(\mathtt{d})}$ também tem. Como comentado acima, $T \not\vdash\!\!\!\!- \ulcorner M_d$ tem tempo não–determinístico polinomial$\urcorner^T$.

[i] Aqui, $M^{(\mathtt{i})}$ indica que a Máquina de Turing M computa assumindo que a palavra $\mathtt{i}$ está fixada, ou seja, o tempo não–determinístico não considera a fita em que ela está escrita como sendo uma fita de entrada.

No caso de espaço não–determinístico logarítmico, novamente $M(\mathtt{i}, x)$ simula $N(\mathtt{i})$ —que agora verifica se $T \vdash \ulcorner M_i \urcorner$ tem espaço não–determinístico logarítmico$^{\urcorner T}$—, mas, primeiro, marca cada uma das suas fitas de trabalho na $\log(\|x\|)$-ésima posição. Portanto: ***a)*** $M(\mathtt{i}, x)$ parará imediatamente se a simulação tentar usar mais do que espaço $\log(\|x\|)$; ***b),*** se $N(\mathtt{i})$ pára com uma resposta positiva, então $M(\mathtt{i}, x)$ usará no mínimo $b(\|x\|)$ posições em uma fita de trabalho; e ***c),*** se $N(\mathtt{i})$ tem rejeição, então $M(\mathtt{i}, x)$ apenas pára. O restante é similar. [II.5–1] ■

***Relembrando o Teorema* II.5–2** *Supondo-se que T é uma teoria axiomatizável, cada asserção fornecida no Teorema* II.2–1.(i) *impõe* `efetivamente` *uma indemonstrabilidade em T.*

Prova. Tome $j < \omega \mapsto \varphi_j(x)$ uma codificação recursiva de todas as fórmulas na linguagem de T tendo somente x como uma variável livre. Sejam duas Máquinas de Turing N e M "efetivamente construídas". Para $j, i < \omega$ e uma palavra x de Γ, $N(\mathtt{j}, \mathtt{i})$ verifica se $T \vdash \varphi_j(\boldsymbol{i})$ e $M(\mathtt{j}, \mathtt{i}, x)$ —com somente três fitas de entrada— gasta no máximo $f''(\|x\|)$ passos para simular $N(\mathtt{j}, \mathtt{i})$ nos seus primeiros $\|x\|$ passos, em que $f'' \in \mathcal{O}(\mathrm{Id})$ é linear. Se $M(\mathtt{j}, \mathtt{i}, x)$ alcança o aceite na simulação de $N(\mathtt{j}, \mathtt{i})$, então ***a)*** ela aceita. Caso contrário, ***b)*** $M(\mathtt{j}, \mathtt{i}, x)$ rejeita.

Pelo Teorema do Ponto Fixo II.4–2, podemos efetivamente encontrar $d'(j) < \omega$ para o qual $M^{(\mathtt{j}, \mathtt{d}'(j))} \preccurlyeq M_{d'(j)}$ —logo $M_{d'(j)}$ não se preocupa com as testemunhas. Mais ainda, existe $f' \in \mathcal{O}_{\mathrm{PLog}}(f'') = \mathcal{O}_{\mathrm{PLog}}(\mathrm{Id})$ tal que, para cada $j < \omega$, $M_{d'(j)}(x)$ sempre pára em no máximo $f'(\|x\|)$ passos. Sendo $f \geq f'$ uma função recursiva, podemos efetivamente encontrar $d(j) < \omega$ tal que $M_{d(j)}(x)$ roda exatamente como $M_{d'(j)}(x)$, exceto que ela gasta mais $f(\|x\|) + 1$ passos depois de $M_{d'(j)}(x)$ ter um aceite.

Daí $M_{d(j)}$ roda em tempo não–determinístico f sse o Item *a)* nunca acontece —assim $T \nvdash \varphi_j(\boldsymbol{d}(j))$. Agora suponha $j < \omega$ para o qual a fórmula $\varphi_j(x)$ é uma certificação fraca da asserção "M_x roda em tempo não–determinístico f". Logo, afora contradições, a asserção "$M_{d(j)}$ roda em tempo não–determinístico f" vale e a fórmula $\varphi_j(\boldsymbol{d}(j))$ é indemonstrável em T.

Repita essa idéia no caso de espaço não–determinístico. [II.5–2] ■

IV.2 Imposição de Indemonstrabilidades Fixadas

Agora provaremos os resultados da Seção II.6 *Imposição de Indemonstrabilidades Fixadas.*

***Relembrando o Teorema* II.6–1** *Sejam T uma teoria axiomatizável e $R, Q \subseteq \omega$ duas asserções unárias tais que $R \subseteq Q$ (i.e., para cada $i < \omega$, se $R(i)$ vale, $Q(i)$ também vale) e Q tenha propriedade da diferença finita de problemas de decisão. Logo:*

(i) *Se $\ulcorner R(x) \urcorner^{\urcorner T}$ é uma certificação fraca —ou uma certificação— de $R(x)$ em T, temos que, para todos $k, \ell < \omega$, existe $q < \omega$ tal que, se, ou $L_k \subseteq L_\ell$, ou $L_\ell \subseteq L_k$,*[ii] *e $Q(\ell)$ não vale, então $L_q = L_k$ e $\ulcorner R(\boldsymbol{q}) \urcorner^{\urcorner T}$ é indemonstrável em T (i.e., $T \nvdash \ulcorner R(\boldsymbol{q}) \urcorner^{\urcorner T}$).*

[ii] Quando fixamos somente uma das possibilidades, ou $L_k \subseteq L_\ell$, ou $L_\ell \subseteq L_k$, a existência de q é *efetiva*.

(ii) *Supondo, mais ainda, que* $R = Q$, $^{\ulcorner}R(x)^{\urcorner_T}$ *é um certificação de* $R(x)$ *em* T *e* $R(k)$ *vale, então* $^{\ulcorner}R(\boldsymbol{q})^{\urcorner_T}$ *é independente de* T *(i.e., também* $T \not\vdash \neg\, ^{\ulcorner}R(\boldsymbol{q})^{\urcorner_T}$*) —e ainda* $R(q)$ *vale.*

Prova. Tome $k, \ell < \omega$ tais que $Q(\ell)$ não vale e, primeiro, suponha que $L_k \subseteq L_\ell$. Para uma palavra x de Γ e $i < \omega$, definimos a Máquina de Turing que segue (que pode ser efetivamente obtida):

$$M(\mathtt{i}, x) := \begin{cases} \text{aceita}, & \text{se } x \in L_k \text{ ou } [x \in L_\ell \text{ e } T \text{ prova } ^{\ulcorner}R(\boldsymbol{i})^{\urcorner_T} \text{ em no} \\ & \text{máximo } \|x\| \text{ passos}];^{[\text{iii}]} \\ \text{rejeita}, & \text{caso contrário.} \end{cases}$$

Usando o Teorema do Ponto Fixo II.4–2, podemos efetivamente encontrar $q < \omega$ para o qual $M^{(\mathtt{q})} \preccurlyeq M_q$.[iv] Suponha que $T \vdash {}^{\ulcorner}R(\boldsymbol{q})^{\urcorner_T}$. Logo L_q difere apenas finitamente de L_ℓ, assim $Q(q)$ não vale e nem $R(q)$. Pela certificação fraca, $T \not\vdash {}^{\ulcorner}R(\boldsymbol{q})^{\urcorner_T}$, uma contradição, portanto $T \not\vdash {}^{\ulcorner}R(\boldsymbol{q})^{\urcorner_T}$. Na verdade, $L_k \subseteq L_q$, assim seja $x \in L_q \smallsetminus L_k$. Então $M(\mathtt{q}, x) = M_q(x)$ tem aceite, daí $T \vdash {}^{\ulcorner}R(\boldsymbol{q})^{\urcorner_T}$, mais uma vez uma contradição. Logo $L_q = L_k$.

No caso em que $L_\ell \subseteq L_k$, tome a Máquina de Turing (que também pode ser efetivamente obtida):

$$M(\mathtt{i}, x) := \begin{cases} \text{aceita}, & \text{se } x \in L_k \text{ e } [x \in L_\ell \text{ ou } T \text{ não prova } ^{\ulcorner}R(\boldsymbol{i})^{\urcorner_T} \\ & \text{em no máximo } \|x\| \text{ passos}];^{[\text{iv}]} \\ \text{rejeita}, & \text{caso contrário} \end{cases}$$

e novamente $M^{(\mathtt{q})} \preccurlyeq M_q$. Se $T \vdash {}^{\ulcorner}R(\boldsymbol{q})^{\urcorner_T}$, então L_q difere apenas finitamente de L_ℓ. De novo, $T \not\vdash {}^{\ulcorner}R(\boldsymbol{q})^{\urcorner_T}$, uma contradição. Logo $T \not\vdash {}^{\ulcorner}R(\boldsymbol{q})^{\urcorner_T}$. Agora $L_q \subseteq L_k$ e, se $x \in L_k \smallsetminus L_q$, então $M(\mathtt{q}, x) = M_q(x)$ não tem aceite, assim $T \vdash {}^{\ulcorner}R(\boldsymbol{q})^{\urcorner_T}$, uma contradição. Uma vez mais, $L_q = L_k$.

[II.6–1] ■

***Relembrando o Corolário* II.6–3** *Para uma teoria axiomatizável* T*, a asserção "*$L_x \in \mathcal{P}$*" —com* x *sendo a única variável livre— impõe indemonstrabilidades/independências fixadas em* T*. O mesmo continua válido para as classes de complexidade usuais, tais como* $\mathcal{L}$*,* $\mathcal{NP}$*,* co-$\mathcal{NP}$*,* $\mathcal{PH}$*,* $\mathcal{PSPACE}$*,* $\mathcal{EXP}$ *e* $\mathcal{NEXP}$*, ao invés de* $\mathcal{P}$*.*

Prova. Considere a asserção $R(x) :=$ "$L_x \in \mathcal{P}$" —que tem propriedade da diferença finita de problemas de decisão. Faremos uso do Teorema II.6–1. Claramente o problema de decisão de todas as palavras de $\Gamma^* =: L_\ell$ pertence a $(\mathcal{L} \subseteq)\ \mathcal{P}$ e $L_k \subseteq L_\ell$, para todo $k < \omega$. Logo precisamos apenas olhar para o caso oposto. Mostraremos abaixo que existe $\ell < \omega$, para cada $k < \omega$ tomado, tal que $L_\ell \notin (\mathcal{EXP} \supseteq)\ \mathcal{P}$ e, ou $L_\ell \subseteq L_k$, ou $L_k \subseteq L_\ell$.

Tome algum $k < \omega$ e $\ell < \omega$ para os quais M_ℓ tenha tempo determinístico duplamente exponencial e $L_\ell \notin \mathcal{EXP}$. Pretendemos obter que, ou $L_k \cup L_\ell \notin \mathcal{EXP}$, ou co–$L_k \cup L_\ell \notin \mathcal{EXP}$.

[iii] Note que, em ambos casos, a validade de $x \in L_k$ e de $x \in L_\ell$ precisa ser garantida pela simulação simultânea de $M_k(x)$ e de $M_\ell(x)$. Desse modo, muitas vezes podemos decidir positivamente a declaração inteira, mesmo quando uma das computações $M_k(x)$ e $M_\ell(x)$ nunca parar.

[iv] Lembre-se que $M^{(\mathtt{q})}$ significa que a primeira fita de entrada é fixada com a palavra $\mathtt{q}$ e negligencia-se-a como fita de entrada.

Suponha que $L_k \cup L_\ell$, co–$L_k \cup L_\ell \in \mathcal{EXP}$. Então simulamos —conjuntamente e também em tempo determinístico exponencial— a computação de $L_k \cup L_\ell$ e co–$L_k \cup L_\ell$ para uma entrada $x \in \Gamma^*$. Aceita-se x sse os dois casos têm aceite. Desse modo, decidimos L_ℓ em tempo exponencial, um absurdo.

Portanto, ou $L_k \cup L_\ell \supseteq L_k$, ou co–(co–$L_k \cup L_\ell$) $= L_k \cap$ co–$L_\ell \subseteq L_k$ não pertence a $\mathcal{EXP} \supseteq \mathcal{P}$. Como M_ℓ sempre pára (em tempo determinístico duplamente exponencial), não existe problema em achar $\ell', \ell'' < \omega$ tais que $L_{\ell'} := L_k \cup L_\ell$ e $L_{\ell''} := L_k \cap$ co–L_ℓ. [II.6–3] ■

***Relembrando o Teorema* II.6–4** *Sejam T uma teoria axiomatizável, $R \subseteq Q \subseteq \omega$ duas asserções unárias tais que Q tenha propriedade de co–limite de tempo e $^{\ulcorner}R(x)^{\urcorner_T}$ seja uma certificação fraca —ou uma certificação— de $R(x)$ em T.*

Para $k, \ell < \omega$, seja $r < \omega$ o número de fitas de testemunha usadas por M_ℓ. Então existe $q < \omega$ para o qual, se $L_k^{r+1} \subseteq L_\ell^{r+1}$ e $Q(\ell)$ não vale, então $M_q \overset{t}{\nleq} M_k$ —portanto, $L_q^{r+1} = L_k^{r+1}$ e $L_q = L_k$— e $^{\ulcorner}R(\boldsymbol{q})^{\urcorner_T}$ é indemonstrável em T (i.e., $T \nvdash {}^{\ulcorner}R(\boldsymbol{q})^{\urcorner_T}$) —também M_q usa somente r fitas de testemunha.

Para mostrar que R `impõe indemonstrabilidades fixadas em` T, é suficiente obter outra asserção unária $Q \subseteq \omega$ tal que:

- ***a)*** *$Q \supseteq R$;*
- ***b)*** *Q tenha propriedade de co–limite de tempo; e*
- ***c)*** *para todo $k < \omega$ tal que $R(k)$ valha, existe $\ell < \omega$ para o qual $L_k^{r+1} \subseteq L_\ell^{r+1}$ —em que $r < \omega$ é o número de fitas de testemunha usadas por M_ℓ— e $Q(\ell)$ não vale.*

Prova. Sejam $k, \ell < \omega$ tais que $r < \omega$ é o número de fitas de testemunha usadas por M_ℓ,[v] $L_k^{r+1} \subseteq L_\ell^{r+1}$ e $Q(\ell)$ não vale.

Tome a Máquina de Turing $M(\mathtt{i}, x, \vec{y})$ —em que x e $\vec{y} := y_0, \ldots, y_{r-1}$ são as palavras de Γ e $i < \omega$— que computa, ao mesmo tempo: ***a)*** $M_k(x, \vec{y})$; e ***b)***, primeiro, $T \vdash {}^{\ulcorner}R(\boldsymbol{i})^{\urcorner_T}$ e, depois, $M_\ell(x, \vec{y})$. Aceita-se somente se, ou $M_k(x, \vec{y})$ tem aceite, ou [$M_\ell(x, \vec{y})$ tem aceite e $T \vdash {}^{\ulcorner}R(\boldsymbol{i})^{\urcorner_T}$]. Pelo Teorema do Ponto Fixo II.4–2, existe $q < \omega$ para o qual $M^{(\mathtt{q})} \preccurlyeq M_q$. Note que M_q usa no máximo r fitas de testemunha, como $M^{(\mathtt{q})}$ (e M_ℓ). Pretendemos mostrar que $T \nvdash {}^{\ulcorner}R(\boldsymbol{q})^{\urcorner_T}$ e, desse modo, iremos ter que $M_k \overset{t}{\preccurlyeq} M_q$, o que completa a prova.

Se $T \vdash {}^{\ulcorner}R(\boldsymbol{q})^{\urcorner_T}$, desde que $L_k^{r+1} \subseteq L_\ell^{r+1}$, $M^{(\mathtt{q})}$ roda com perda de tempo de ordem linear, com respeito à M_ℓ. Daí M_q roda com perda de tempo de ordem linear com fator polilogarítmico, com respeito à M_ℓ. Como $Q(\ell)$ não vale por hipótese, nem $Q(q)$ —porque Q tem propriedade de co–limite de tempo—, nem $R(q)$ valem. Pela certificação fraca, temos que $T \nvdash {}^{\ulcorner}R(\boldsymbol{q})^{\urcorner_T}$, um absurdo.

[II.6–4] ■

***Relembrando o Corolário* II.6–5** *Tomada uma teoria axiomatizável T, a asserção "M_x tem tempo não–determinístico polinomial" —em que x é a única variável livre— impõe fortemente indemonstra-*

[v] Isso será necessário apenas no sentido de se obter o co–limite *não*–determinístico e, com isso, perderemos a *efetividade* na existência de q.

bilidades fixadas em T. O mesmo acontece nos casos tais como de tempo não–determinístico exponencial, e de espaço não–determinístico logarítmico e polinomial.

Prova. Seja $R(x)$ a asserção "M_x tem tempo não–determinístico polinomial". Assim R impõe indemonstrabilidades fixadas em T, como comentado um pouco antes do Teorema II.6–4.

Agora suponha que $\ell < \omega$ seja tal que a Máquina de Turing M_ℓ aceita tudo —não usa fitas de testemunha e roda em tempo constante igual a 1. Logo, para todo $k < \omega$, temos que $L_k \subseteq L_\ell := \Gamma^*$ —o problema de decisão de todas as palavras— e $R(\ell)$ vale. Assim, aplicando-se o Teorema II.6–4, desde que a asserção R^c tem propriedade de co–limite de tempo, obtemos que R^c também impõe indemonstrabilidades fixadas em T. [II.6–5] ■

Capítulo V

Teoria da Recursão com Complexidade

Finalmente, resta provar versões fortes dos seguintes resultados clássicos da Teoria da Recursão: Teorema da Máquina de Turing Universal e da Parametrização II.4–1 e Teorema do Ponto Fixo II.4–2, vide [End77, Cut80, Smu93].

Começamos esse capítulo definindo nossas Máquinas de Turing e suas computações. Depois interpretamos as simulações das computações e provamos o primeiro teorema. No final, usamos esse teorema para provar o segundo.

A Máquina de Turing

Fixamos um alfabeto $\Gamma := \{\sigma_1, \ldots, \sigma_\ell\}$, isso é, Γ é um conjunto finito (de letras) com mais do que um elemento e que não contém a letra (símbolo) *espaço em branco* $\not\flat := \sigma_0$. Também, seja $\Lambda := \{0, 1, \natural\}$ o *alfabeto universal*, em que $\natural$ é a letra *símbolo de separação*.

Tome $\bar{\Lambda} := \Lambda \setminus \{\natural\}$ e a associação injetiva $i < \omega \mapsto \bar{i} \in \bar{\Lambda}^*$, definida pela representação binária tal que o número 0 é associado com a *palavra vazia* ϵ; o dígito mais significativo é escrito à direita; e a letra mais à direita nunca é 0. Essa associação tem a inversa $\bar{\Lambda}^* \rightsquigarrow \omega$, que sempre descartará os 0's à direita.

Também tomamos outra associação injetiva $\tau \in \Gamma \cup \{\not\flat\} \mapsto \bar{\tau} \in \bar{\Lambda}_*^{\lceil \log_2(\ell+1) \rceil}$ —i.e., do espaço em branco $\not\flat$ e das letras de Γ nas palavras de Λ de comprimento $\lceil \log_2(\ell+1) \rceil$, mas sem a palavra vazia ϵ. Dessa maneira, podemos estendê-la para outra associação injetiva $x \in \Gamma^* \mapsto \bar{x} \in \bar{\Lambda}^*$, com uma inversa compatível.

Por conveniência, chamamos as palavras em Λ de *Máquinas de Turing*, ou, melhor, toda Máquina de Turing é uma palavra $M \in \Lambda^*$ e vice–versa. Entretanto, para cada alfabeto Γ, temos uma noção diferente do que é a *computação* sobre esse alfabeto. Isso é, agora iremos dar o significado da computação $M(\vec{x})$, quando $\vec{x}$ é um número fixo de palavras variando em um alfabeto Γ específico.

Abaixo, apresentamos certos tipos de palavras de Λ, que chamamos de *palavras comando*. Lembre-se que temos $\Gamma \cup \{\not b\} := \{\sigma_0\ (:= \not b), \sigma_1, \ldots, \sigma_\ell\}$. Então, dados j, c, $n_0, \ldots, n_\ell < \omega$ tal que $c \geq 4$, associamos:

Palavra comando	Sinal comando
$\bar{0}$	[pára]
$\bar{1}\natural\bar{j}$	[esquerda em j]
$\bar{2}\natural\bar{j}$	[direita em j]
$\bar{3}\natural\bar{j}\natural\bar{\tau}$	[escreve τ em j]
$\bar{c}\natural\bar{j}\natural\bar{n}_0\natural\ldots\natural\bar{n}_\ell$	[caso em j, vai para $n_0, \ldots, n_\ell$]

Iremos descrever a ação de todos esses comandos adiante, ao definirmos a computação de uma Máquinas de Turing.

Cada concatenação de palavras comando, quando separadas pelo símbolo de separação $\natural$, é também uma palavra de Λ. Além disso, todas as palavras de Λ têm um único formato de concatenação. Ou seja, se $M \in \Lambda^*$, existem únicas palavras comando $z_0, \ldots, z_n$ (e $n < \omega$) de Λ tais que $M = z_0\natural\cdots\natural z_n$.

Sendo assim, a última palavra comando z_n pode ser incompleta e, nesse caso, iremos supor os parâmetros inexistentes como sendo a palavra vazia ϵ. Por exemplo, se $z_n := \bar{1}$, interpretamos o parâmetro de fita que falta $\bar{j}$ como $j := 0$, isso é, associamos z_n com a palavra comando $\bar{1}\natural\epsilon$.

Em adição à associação injetiva inicial $\omega \rightsquigarrow \bar{\Lambda}^*$, obtemos outra, em que agora o 0 não será mais associado com a palavra vazia ϵ. Pela ordem lexicográfica, tomamos a enumeração $i < \omega \mapsto M_i \in \Lambda_*^*$ de todas as Máquina de Turing. Aqui, Λ_*^* é o conjunto das palavras de Λ sem a palavra vazia ϵ. Note que, tanto essa associação, quanto a sua inversa, podem ser efetivamente construídas.

Como a definimos, uma Máquina de Turing é um objeto concreto —uma palavra de Λ—, entretanto pensamos na sua computação como um "elemento ideal". A computação de uma Máquina de Turing é determinada pelas suas palavras comando, mas também depende do alfabeto Γ fixado. Um comando expressa um passo da computação. Ele é definido pela palavra comando da Máquina de Turing que é corrente na computação, mas tomada sobre Γ.

A Configuração Inicial de uma Computação

Uma *fita* tem contáveis *posições*. Em cada posição, está escrita uma letra de $\Gamma \cup \{\not b\}$, que será modificada de acordo com os passos da computação. Dizemos que a $(i+1)$-ésima posição está à direita de (ou sucede) a i-ésima posição e que a i-ésima posição está à esquerda de (ou precede) a $(i+1)$-ésima posição (com $i < \omega$). Todas as fitas têm uma, e somente uma, *cabeça de leitura*, que marca uma posição específica na fita, a qual vai se alterando no decorrer da computação.

Quando a computação começa, a cabeça de leitura deve estar sempre na 1-ésima posição da sua fita —que, na realidade, é a segunda posição da fita, pois ela começa na 0-ésima posição— e cada fita deve ter uma palavra x de Γ (talvez a palavra vazia ϵ) escrita nela. A primeira letra de x é escrita na 1-ésima posição da fita e assim por diante, em direção à direita. O espaço em branco $\not b$ é escrito

na 0-ésima posição e também em todas as demais posições à direita de x —i.e., se $x := \tau_0\tau_1\cdots\tau_t$, com $\tau_i \in \Gamma$, então essa fita terá $\not\flat\tau_0\tau_1\cdots\tau_t\not\flat\not\flat\cdots$ no começo da computação.

Uma Máquina de Turing tem contáveis fitas: 0-fita, 1-fita, e assim por diante. Dada uma Máquina de Turing M e palavras $x_0,\ldots,x_{r-1}$ de Γ (em que $r < \omega$), denotamos por $M(\vec{x})$ a computação da máquina M com a entrada $\vec{x}$. Nesse caso, a 1-fita até a r-fita são chamadas *fitas de entrada* e elas são de somente–leitura. A 0-fita é a *fita de saída*, que é de somente–escrita. As restantes fitas podem ser usadas como *fitas de trabalho*, que são de leitura–escrita.

No começo da computação, cada fita de entrada tem escrita nela uma palavra de $\vec{x}$ e todas as outras fitas contêm somente a palavra vazia —i.e., elas são completamente preenchidas com o espaço em branco $\not\flat$. Uma computação é habilitada a usar somente as fitas de entrada, de saída e de trabalho, e nada mais.

Tudo isso faz parte da nossa `definição genérica` de Máquina de Turing. Entretanto, somente para as demonstrações dos próximos teoremas, começando pelo argumento da Máquina de Turing Universal, precisamos apresentar uma `definição restrita` de Máquina de Turing. Nela, permitimos que as Máquinas de Turing usem apenas *duas fitas de trabalho*, as $(r+1)$- e $(r+2)$-fitas.

Mas, todo tempo, faremos uso do seguinte resultado: toda Máquina de Turing na definição genérica pode ser simulada por uma Máquina de Turing na definição restrita (efetivamente obtida), tendo para isso uma perda de tempo de ordem linear com fator polilogarítmico e de espaço de ordem linear (veja [HU79, Pap94]).

Quando a computação pára, podemos escolher duas espécies de saídas, dependendo do tipo da computação: ou uma computação de aceite, ou uma computação funcional. Na computação de aceite, a resposta é de aceite sse a cabeça de leitura da fita de saída pára sobre o espaço em branco $\not\flat$ no final da computação. Na computação funcional, a saída é considerada como sendo a primeira palavra de Γ à direita da cabeça de leitura da fita de saída.

A Computação

A computação $M(\vec{x})$ começa com a primeira palavra comando de M. Em cada *comando (passo)*, M pode executar uma das seguintes instruções:

Sinal comando	Descrição da instrução
[pára]	Pára a computação.
[esquerda em j]	Coloca a cabeça de leitura da j-fita uma posição à esquerda e vai para a próxima palavra comando de M.
[direita em j]	Coloca a cabeça de leitura da j-fita uma posição à direita e vai para a próxima palavra comando de M.
[escreve τ em j]	Escreve a letra $\tau \in \Gamma \cup \{\not b\}$ na posição da cabeça de leitura da j-fita e vai para a próxima palavra comando de M.
[caso em j, vai para $n_0, n_1, \ldots, n_\ell$]	Dependendo da letra $\sigma_t \in \Gamma \cup \{\not b\} := \{\sigma_0 (:= \not b), \sigma_1, \ldots, \sigma_\ell\}$ escrita na posição apontada pela cabeça de leitura da j-fita, vai para a n_t-ésima palavra comando de M.

Uma computação é uma sucessão desses passos. Podem ocorrer duas possibilidades distintas durante ela: ou passa pelo comando [pára]; ou nunca pára —i.e., entrando em *loop*.

Em cada computação, existem passos que *negligenciarão* as instruções acima, pulando-os e indo para a próxima palavra comando. Todas as instruções sobre fitas que não são, nem de entrada, nem de saída e nem de trabalho, são negligenciadas. Quando uma fita é de somente–leitura, todos os comandos [escreve] sobre ela são negligenciados. Do mesmo modo, se uma fita é de somente–escrita, todos os comandos [caso] sobre ela também são negligenciados. Quando a cabeça de leitura está sobre o espaço em branco $\not b$, os comandos [esquerda] são negligenciados de dois modos: se na sua posição à direita existe uma letra diferente do espaço em branco; e se na sua posição à esquerda (e também na sua posição à direita) tem o espaço em branco. Isso serve para a computação nunca extrapolar em direção ao lado esquerdo de uma fita.

Por convenção, o último comando de M, e somente ele, é sempre o comando [pára]. Mais ainda, não deve haver comando [caso] cujo número vai–para é maior que o número do comando [pára] em M. As máquinas que não respeitam essas convenções são chamadas de *Máquinas de Turing ilegais* e, no aspecto computacional, elas devem ser pensadas como sempre entrando em *loop*.

A Máquina de Turing Universal

Primeiro notamos que, como Λ é um alfabeto, podemos pensar em uma Máquina de Turing U' trabalhando com esse alfabeto. Portanto, se olhamos para uma palavra de Λ, ela é uma outra Máquina de Turing, ou seja, as entradas de U' são Máquinas de Turing M_i (com $i < \omega$).

Supondo-se que M_i computa sobre o alfabeto Λ, podemos obter uma Máquina de Turing U' sobre Λ para a qual $U'(M_i, \vec{x})$ tem praticamente a mesma computação de $M_i(\vec{x})$, em que $x_0, \ldots, x_{r-1}$ são palavras em Λ. Mas nós temos a intenção de computar sobre o alfabeto fixo Γ.

Tome uma injeção $\tau \in \Lambda \mapsto \bar{\bar{\tau}} \in \Gamma_*^{\lceil \log_\ell(3) \rceil}$ —i.e., das letras de Λ para as palavras de Γ de comprimento $\lceil \log_\ell(3) \rceil$, sem a palavra vazia ϵ— e também acrescente $\not b \mapsto \bar{\bar{\not b}} := \not b$. Então expanda

para a associação injetiva $x \in \Lambda^* \mapsto \bar{\bar{x}} \in \Gamma_*^*$. Juntando-se com a associação anterior $\omega \rightsquigarrow \Lambda_*^*$, temos

$$\omega \quad > \quad \begin{matrix} i & \longmapsto & \mathtt{i} =: \check{i} := \bar{\bar{M}}_i \\ \widetilde{x} & \longleftarrow & x \end{matrix} \quad \in \quad \Gamma_*^*.$$

Isso também pode ser efetivamente construído. Daqui para frente, utilizaremos diversas vezes essa associação e sua inversa. Sobre ela, conseguimos fazer a clássica relação entre Máquinas de Turing e funções recursivas parciais.

Retornando à simulação, para todo $i < \omega$, a palavra $\mathtt{i}$ (ou $\check{i}$) agora corresponde à Máquina de Turing M_i codificada no alfabeto Γ. Então podemos perguntar se existe uma Máquina de Turing U sobre Γ para a qual $U(\mathtt{i}, \vec{x})$ simula a computação de $M_i(\vec{x})$. Vejamos como fazer isso.

A entrada $(\mathtt{i}, \vec{x})$ é escrita sobre $r+1$ fitas de entrada. Usaremos as duas primeiras fitas de trabalho para simular as fitas de trabalho de M_i e reservamos as próximas $h < \omega$ (digamos) fitas como *novas* fitas de trabalho, somente para servirem de contadores da simulação. Na nossa definição, todas as Máquinas de Turing podem usar somente duas fitas de trabalho. Para colocar essa computação em acordo com nossa definição *restrita*, posteriormente também simularemos essas $h+2$ fitas de trabalho em somente duas delas, tendo para isso uma perda de tempo de ordem linear com fator polilogarítmico e de espaço de ordem linear.

Seja $g(0) := 0$ —representando a fita de saída—; e $g(j) := j+1$ —representando as fitas de entrada (para $1 \leq j \leq r$) e as fitas de trabalho (para $j = r+1$ e $r+2$)—, se $j \geq 1$. A simulação segue as instruções:

- Verifica se $\mathtt{i}$ não codifica (ou M_i não é) uma Máquina de Turing ilegal. Nesse caso, a simulação entra em *loop*. Note que essa verificação pode ser feita independentemente da entrada $\vec{x}$.

- Sempre que passa por uma palavra comando de M_i —codificada pela palavra de entrada $\mathtt{i}$—, a simulação executa o mesmo comando, mas:

  - Se o comando de M_i envolve a fita j, a simulação executa-o na fita $g(j)$.

  - Negligencia-o e vai para a próxima palavra comando de M_i, se o comando de M_i precisa ser negligenciado.

Em todos esses comandos, a simulação $U(\mathtt{i}, \vec{x})$ gasta $\mathcal{O}(1)$ passos, exceto para o movimento à próxima palavra comando de M_i e "ir para a n_t-ésima palavra comando de M_i", que gastam $\mathcal{O}(\|\mathtt{i}\|) = \mathcal{O}(\log(i))$ passos. Portanto a comutação completa gasta tempo de ordem linear em $\log(i)$ vezes o número de passos na computação de $M_i(\vec{x})$. Seja t a função do tempo gasto na computação, então

$$t(U(\mathtt{i}, \vec{x})) \quad \in \quad \mathcal{O}_{\vec{x}}(\, \log(i) t(M_i(\vec{x}))\,).$$

Aqui, $\mathcal{O}_{\vec{x}}$ indica um limitante de ordem linear, com $\vec{x}$ variando.

Em termos de espaço, o que é necessário ser feito é utilizar um contador para se ir à n_t-ésima palavra comando de M_i e outro para se saber em que palavra comando de M_i a simulação situa-se em cada passo. Assim o espaço requerido é $\mathcal{O}(\log(\|\mathtt{i}\|)) = \mathcal{O}(\log\log(i))$ mais uma ordem linear do espaço necessário na computação de $M_i(\vec{x})$. Seja s a função do espaço ocupado na

computação, então

$$s(U(\mathtt{i},\vec{x})) \;\in\; \mathcal{O}_{\vec{x}}(\; s(M_i(\vec{x})) + \log\log(i)\;).$$

Entretanto temos que fazer a computação de $U(\mathtt{i},\vec{x})$ adequar-se à nossa definição *restrita* de Máquina de Turing, que permite o uso de somente duas fitas de trabalho, ao invés das $h+2$ fitas utilizadas —repare que h é um número fixo. Para isso, fazemos outra simulação sobre essa primeira, em que a computação com todas essas fitas de trabalho é feita em somente duas delas. Realizamos isso com gasto de tempo de ordem linear com fator polilogarítmico e com gasto de espaço de ordem linear (vide [HU79, Pap94]). Nós ainda continuamos a descrevê-la por $U(\mathtt{i},\vec{x})$.

Fixe $i<\omega$, então $M_i(\vec{x}) = U(\mathtt{i},\vec{x})$ e a computação de $U(\mathtt{i},\vec{x})$ gasta mais tempo e espaço do que $M_i(\vec{x})$, para todas palavras $x_0,\ldots,x_{r-1}$ de Γ. Mais ainda, quando comparada com a computação de $M_i(\vec{x})$ —com entradas $\vec{x}$—, $U(\mathtt{i},\vec{x})$ tem perda de tempo de ordem linear com fator polilogarítmico e de espaço de ordem linear. Por causa disso, dizemos que $M_i(\vec{x})$ é bem–simulado por $U(\mathtt{i},\vec{x})$ sobre $\vec{x}$ e denotâmo-lo por

$$M_i(\vec{x}) \;\preccurlyeq_{\vec{x}}\; U(\mathtt{i},\vec{x}).$$

Os Teoremas

Suponha dada uma outra Máquina de Turing M_j (com $j<\omega$) e palavras $y_0,\ldots,y_{q-1}$ de Γ (com $q<\omega$ também fixo). Considere M_i como uma computação funcional e queremos simular a computação de $M_j(M_i(\vec{x}),\vec{y})$ por $U(\mathtt{i},\mathtt{j},\vec{x},\vec{y})$. Aqui, se $M_i(\vec{x})$ entra em *loop*, assumimos que $M_j(M_i(\vec{x}),\vec{y})$ também entra.

Simulamos $M_i(\vec{x})$ como acima, mas usando uma nova fita de trabalho ao invés da fita de saída e pulando as fitas de entrada com as palavras $\mathtt{j}$ e $\vec{y}$. Depois disso, o seguinte é executado: a cabeça de leitura vai uma posição à esquerda nessa nova fita de trabalho; escreve o espaço em branco $\not b$; e retorna à direita. Isso é feito para certificar que, no começo da próxima simulação, essa fita tem a palavra de saída $z := M_i(\vec{x})$ —e a sua cabeça de leitura está na primeira letra de z. A seguir, inicializamos a simulação de $M_j(z,\vec{y})$. Para fazer isso, precisamos apenas escolher corretamente as fitas de entrada e usar novas fitas de trabalho. Então $U(\mathtt{i},\mathtt{j},\vec{x},\vec{y})$ perde novamente tempo de ordem linear com fator polilogarítmico e espaço de ordem linear para simular todas essas fitas de trabalho em somente duas.

Note que podemos efetivamente encontrar a Máquina de Turing (uma palavra de Λ) que faz a simulação $U(\mathtt{i},\mathtt{j},\vec{x},\vec{y})$. Além disso, também existe efetivamente $u<\omega$ tal que $U = M_u$. Então $M_u(\mathtt{i},\mathtt{j},\vec{x},\vec{y})$ simula a computação de $M_j(M_i(\vec{x}),\vec{y})$, para toda palavra $x_0,\ldots,x_{r-1}$ e $y_0,\ldots,y_{q-1}$ de Γ e $i,j<\omega$.

Analisando o tempo e o espaço usados nessa simulação, temos

$$\begin{aligned} t(M_u(\mathtt{i},\mathtt{j},\vec{x},\vec{y})) \;&\in\; \mathcal{O}_{\mathrm{PLog}_{z,\vec{y}}}(\;\log(j)t(M_j(z,\vec{y}))\;) \\ &\quad + \mathcal{O}_{\mathrm{PLog}_{\vec{x}}}(\;\log(i)t(M_i(\vec{x}))\;) \\ &\subseteq\; \mathcal{O}_{\mathrm{PLog}_{\vec{x},\vec{y}}}(\;\log(\max\{i,j\})\;t(M_j(M_i(\vec{x}),\vec{y}))\;) \end{aligned}$$

e

$$\begin{aligned} s(M_u(\mathtt{i},\mathtt{j},\vec{x},\vec{y})) \;&\in\; \mathcal{O}_{z,\vec{y}}(\, s(M_j(z,\vec{y})) + \log\log(j)\,) \\ &\quad + \mathcal{O}_{\vec{x}}(\, s(M_i(\vec{x})) + \log\log(i)\,) \\ &\subseteq\; \mathcal{O}_{\vec{x},\vec{y}}(\, s(M_j(M_i(\vec{x}),\vec{y})) + \log\log(\max\{i,j\})\,). \end{aligned}$$

Já aqui, $\mathcal{O}_{\mathrm{PLog}_{\vec{x}}}$ indica limitação de ordem linear com fator polilogarítmico, em que $\vec{x}$ variam. Portanto,

$$M_j(M_i(\vec{x}),\vec{y}) \;\preccurlyeq_{\vec{x},\vec{y}}\; M_u(\mathtt{i},\mathtt{j},\vec{x},\vec{y}).$$

Imagine uma simulação que escreve em $c+b+2$ (com $c,b<\omega$ fixados) diferentes "fitas de entrada" a codificação dos números naturais i, j, $n_0,\ldots,n_{c-1}$ e $m_0,\ldots,m_{b-1}$ e, depois, simula a computação de $M_j(M_i(\vec{\mathtt{n}},\vec{x}),\mathtt{m},\vec{y})$ como acima. Novamente gastamos tempo de ordem linear com fator polilogarítmico e espaço de ordem linear para colocar o formato da computação em acordo com nossa definição *restrita*. Note que essa simulação $M_{v(i,j,\vec{n},\vec{m})}\,(\vec{x},\vec{y})$ tem somente $\vec{x}$ e $\vec{y}$ como entradas reais, em que $v(i,j,\vec{n},\vec{m})$ é um número natural.

Embora $M_{v(i,j,\vec{n},\vec{m})}$ seja uma Máquina de Turing, ela também é uma palavra de Λ. Aplicando-se o operador $(\bar{\bar{\cdot}})$ sobre ela, temos uma palavra de Γ. Claramente podemos encontrar uma Máquina de Turing M_w sobre Γ —ou o seu índice w—, para a qual, tendo como entrada $(\mathtt{i},\mathtt{j},\vec{\mathtt{n}},\vec{\mathtt{m}})$, a sua computação tem como saída a referida palavra de Γ, i.e.:

$$M_w(\mathtt{i},\mathtt{j},\vec{\mathtt{n}},\vec{\mathtt{m}}) \;=\; \overline{\overline{M_{v(i,j,\vec{n},\vec{m})}}} \;=:\; \check{v}(i,j,\vec{n},\vec{m})).$$

Aqui pretendemos dizer que o operador $(\check{\cdot})$ é aplicado sobre o número natural $v(i,j,\vec{n},\vec{m})$. Ou seja,

$$v(i,j,\vec{n},\vec{m}) \;=\; \widetilde{M_w}(\mathtt{i},\mathtt{j},\vec{\mathtt{n}},\vec{\mathtt{m}}),$$

em que, novamente, o operador $(\tilde{\cdot})$ é aplicado sobre a saída da computação $M_w(\mathtt{i},\mathtt{j},\vec{\mathtt{n}},\vec{\mathtt{m}})$. Mais uma vez, podemos fazer tal procura por $w<\omega$ efetivamente, lembrando que $r,q,c,b<\omega$ estão fixados.

Nessa simulação, mais do que na anterior, apenas necessitamos escrever os números naturais i, j, $\vec{n}$ e $\vec{m}$, o que gasta tempo $\mathcal{O}\big(i+j+\sum\vec{n}+\sum\vec{m}\big)$ e espaço $\mathcal{O}\big(\log(i)+\log(j)+\sum\log(\vec{n})+\sum\log(\vec{m})\big)$. Então ainda temos

$$M_j(M_i(\vec{\mathtt{n}},\vec{x}),\vec{\mathtt{m}},\vec{y}) \;\preccurlyeq_{\vec{x},\vec{y}}\; M_{v(i,j,\vec{n},\vec{m})}(\vec{x},\vec{y}).$$

Utilizando os dois resultados acima, temos

$$M_{\widetilde{M_i(\vec{\mathtt{n}},\vec{x})}}(\vec{\mathtt{m}},\vec{y}) \;\preccurlyeq_{\vec{\mathtt{m}},\vec{y}}\; M_u(M_i(\vec{\mathtt{n}},\vec{x}),\vec{\mathtt{m}},\vec{y}) \;\preccurlyeq_{\vec{x},\vec{y}}\; M_{v(i,u,\vec{n},\vec{m})}(\vec{x},\vec{y}).$$

O primeiro elemento precisa ser pensado como uma computação que nunca pára, sempre que $M_i(\vec{\mathtt{n}},\vec{x})$ diverge.

Com isso, nós concluímos o teorema que segue:

***Relembrando o Teorema da Máquina de Turing Universal e da Parametrização* II.4–1** *Dados $r,q,c,b<\omega$, pode-se obter efetivamente $u<\omega$ e uma função recursiva (total) $(c+b+2)$-ária v (i.e., o seu número índice) tais que, para todos i, j, $n_0,\ldots,n_{c-1}$, $m_0,\ldots,m_{b-1}<\omega$,*

(i) $$M_j(M_i(\vec{x}),\vec{y}) \preccurlyeq_{\vec{x},\vec{y}} M_u(\mathtt{i},\mathtt{j},\vec{x},\vec{y}),$$

(ii) $$M_j(M_i(\vec{\mathtt{n}},\vec{x}),\vec{\mathtt{m}},\vec{y}) \preccurlyeq_{\vec{x},\vec{y}} M_{v(i,j,\vec{n},\vec{m})}(\vec{x},\vec{y})$$

e, para todos $\vec{x} \in \Gamma^*$, *temos*

(iii) $$M_{\widetilde{M_i(\vec{\mathtt{n}},\vec{x})}}(\vec{\mathtt{m}},\vec{y}) \preccurlyeq_{\vec{y}} M_{v(i,u,\vec{n},\vec{m})}(\vec{x},\vec{y});$$

em que as palavras $x_0,\ldots,x_{r-1}$, $y_0,\ldots,y_{q-1}$ *estão variando em* Γ^*. *A função* v *é chamada de* `parâmetro indexador (total)` *e* M_u, *de* `Máquina de Turing Universal`. ■

Agora provemos o último teorema.

***Relembrando o Teorema do Ponto Fixo* II.4–2** *Sejam* $r,c,b < \omega$ *e uma função recursiva parcial unária* f, *com domínio* $I \subseteq \omega$ *e tendo um dos seus números índice* $q < \omega$ *conhecido. Pode-se encontrar efetivamente funções recursivas parciais unárias e e d, que têm domínios* $I \times \omega^{c+b}$ *e* $I \times \omega^c$, *respectivamente, tais que, para todos* $i \in I$ *e* $n_0,\ldots,n_{c-1}, m_0,\ldots,m_{b-1} < \omega$,

(i) $$M^{(\vec{\mathtt{n}})}_{\widetilde{M_{f(i)}}(\vec{\mathtt{m}},\mathtt{e}(i,\vec{n},\vec{m}))} \preccurlyeq_r M_{e(i,\vec{n},\vec{m})};$$

(ii) $$M^{(\vec{\mathtt{n}},\mathtt{d}(i,\vec{n}))}_{f(i)} \preccurlyeq_r M_{d(i,\vec{n})};$$

e,

(iii) *se a função* f *é constante ou* $M_{f(i)}$ *usa um número limitado de fitas de entrada (para todos* $i < \omega$*), então*

$$M^{(\vec{\mathtt{n}},\mathtt{d}(i,\vec{n}))}_{f(i)} \preccurlyeq M_{d(i,\vec{n})}.$$

Mais ainda, sejam g *uma função recursiva parcial unária,* M *uma Máquina de Turing —que computa usando somente duas fitas de trabalho— e* h *uma função recursiva parcial* $(c+1)$*-unária. Então, existem* $e,d,\ell < \omega$ *tais que*

$$M_{g(e)} \preccurlyeq_r M_e;$$
$$M^{(\mathtt{d})} \preccurlyeq M_d;$$

e, para cada $n_0,\ldots,n_{c-1} < \omega$, *temos*

$$h(\ell,\vec{n}) = \widetilde{M_\ell(\vec{\mathtt{n}})}.$$

Nesses casos particulares, se as funções e a Máquina de Turing são "construtivamente obtidos" (i.e., conhecemos explicitamente os seus números índice), então podemos obter efetivamente os números $e,d,\ell < \omega$.

Prova. Suponha $r < \omega$ e seja I o domínio da função recursiva parcial f. Podemos descrever f por um dos seus números índice $q < \omega$, i.e., $f(i) := \widetilde{M_q}(\mathtt{i})$ (com $i \in I$).

Usaremos o Teorema da Máquina de Turing Universal e da Parametrização II.4–1 várias vezes. Antes de mais nada, notemos que, se temos algumas funções recursivas parciais (na verdade,

os seus números índice), usando o Item II.4–1.(i), podemos efetivamente obter (o código de) as suas composições.

(i) Pelo Item II.4–1.(ii), efetivamente existe o parâmetro indexador (total) —uma função recursiva (total)— $v(j,\vec{m})$ tal que, para todos j, $m_0,\ldots,m_{b-1} < \omega$ (com $b < \omega$),

$$M_j^{(\mathtt{j},\vec{\mathtt{m}})} \preccurlyeq_r M_{v(j,\vec{m})}.$$

Isso significa que

$$M_j(\mathtt{j},\vec{\mathtt{m}},\vec{x}) \preccurlyeq_{\vec{x}} M_{v(j,\vec{m})}(\vec{x}),$$

com as palavras $x_0,\ldots,x_{r-1}$ variando em Γ^*. Esse parâmetro indexador (total) é descrito pelo seu número índice $w < \omega$, assim $v(j,\vec{m}) := \widetilde{M}_w(\mathtt{j},\vec{m})$.

Primeiro, usando o Item II.4–1.(ii) e depois pela observação acima, podemos efetivamente encontrar o parâmetro indexador parcial $g(i)$, com domínio I, para o qual

$$M_{f(i)}(\mathtt{v}(j,\vec{m}),\vec{\mathtt{m}}) \quad (:= M_{\widetilde{M_q(\mathtt{i})}}(M_w(\mathtt{j},\vec{\mathtt{m}}),\vec{\mathtt{m}})) \quad = \quad M_{g(i)}(\mathtt{j},\vec{\mathtt{m}}),$$

para todos $j,\vec{m} < \omega$ e $i \in I$. Aqui, $\mathtt{v}(j,\vec{m}) :=: \check{v}(j,\vec{m})$ denota a codificação do número natural $v(j,\vec{m})$ por uma palavra de Γ.

Agora, pelo Item II.4–1.(iii) e novamente pela observação da composição das funções recursivas parciais acima, podemos efetivamente obter o parâmetro indexador parcial $d(i,\vec{n})$, com domínio $I \times \omega^c$, tal que, para cada $i \in I$ e j, $\vec{m}$, $n_0,\ldots,n_{c-1} < \omega$ (com $c < \omega$),

$$M^{(\vec{\mathtt{n}})}_{\widetilde{M}_{f(i)}(\mathtt{v}(j,\vec{m}),\vec{\mathtt{m}})} \quad = \quad M^{(\vec{\mathtt{n}})}_{\widetilde{M}_{g(i)}(\mathtt{j},\vec{\mathtt{m}})} \quad \preccurlyeq_r \quad M^{(\mathtt{j},\vec{\mathtt{m}})}_{d(i,\vec{n})}.$$

Como a função recursiva $v(j,\vec{m})$ é total, tomamos efetivamente a função recursiva parcial $e(i,\vec{n},\vec{m}) := v(d(i,\vec{n}),\vec{m})$, com domínio $I \times \omega^{c+b}$? Logo, para todos $i \in I$ e $\vec{n},\vec{m} < \omega$,

$$\begin{aligned} M^{(\vec{\mathtt{n}})}_{\widetilde{M}_{f(i)}(\mathtt{e}(i,\vec{n},\vec{m}),\vec{\mathtt{m}})} \quad &:= \quad M^{(\vec{\mathtt{n}})}_{\widetilde{M}_{f(i)}(\mathtt{v}(d(i,\vec{n}),\vec{m}),\vec{\mathtt{m}})} \\ &\preccurlyeq_r \quad M^{(\mathtt{d}(i,\vec{n}),\vec{\mathtt{m}})}_{d(i,\vec{n})} \\ &\preccurlyeq_r \quad M_{v(d(i,\vec{n}),\vec{m})} \quad =: \quad M_{e(i,\vec{n},\vec{m})}. \end{aligned}$$

(ii) De novo pelo Item II.4–1.(ii), efetivamente existe o parâmetro indexador $v(i,j)$, com a primeira variável percorrendo o domínio I —e a segunda variável sendo total—, para o qual

$$M^{(\vec{\mathtt{n}},\mathtt{j})}_{f(i)} \quad \preccurlyeq_r \quad M^{(\vec{\mathtt{n}})}_{v(i,j)},$$

para cada $i \in I$ e j, $\vec{n} < \omega$. Logo, pelo Item (i) acima, podemos efetivamente obter uma função recursiva parcial $d(i,\vec{n})$, com domínio $I \times \omega^c$, tal que, para todos $i \in I$ e $\vec{n} < \omega$,

$$M^{(\vec{\mathtt{n}},\mathtt{d}(i,\vec{n}))}_{f(i)} \quad \preccurlyeq_r \quad M^{(\vec{\mathtt{n}})}_{v(i,d(i,\vec{n}))} \quad \preccurlyeq_r \quad M_{d(i,\vec{n})}.$$

[II.4–2] ■

Referências Bibliográficas

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[Bus86] Samuel R. Buss. *Bounded arithmetic*, volume 3 of *Studies in Proof Theory. Lecture Notes*. Bibliopolis, Naples, 1986.

[Bus95] Samuel R. Buss. Relating the bounded arithmetic and polynomial time hierarchies. *Ann. Pure Appl. Logic*, 75(1-2):67–77, 1995. Proof theory, provability logic, and computation (Berne, 1994).

[CT95] Peter Clote and Gaisi Takeuti. First order bounded arithmetic and small Boolean circuit complexity classes. In *Feasible mathematics, II (Ithaca, NY, 1992)*, volume 13 of *Progr. Comput. Sci. Appl. Logic*, pages 154–218. Birkhäuser Boston, Boston, MA, 1995.

[Cut80] Nigel Cutland. *Computability*. Cambridge University Press, Cambridge, 1980. An introduction to recursive function theory.

[End77] Herbert B. Enderton. Elements of recursion theory. pages 527–566. Studies in Logic and the Foundations of Math., Chapter C.1, Vol. 90, 1977. Handbook of mathematical logic, Edited by Jon Barwise.

[Gra80] Philip W. Grant. Some more independence results in complexity theory. *Theoret. Comput. Sci.*, 12(2):119–126, 1980.

[Háj77] Petr Hájek. Arithmetical complexity of some problems in computer science. pages 282–287. Lecture Notes in Comput. Sci., Vol. 53, 1977.

[Háj79] Petr Hájek. Arithmetical hierarchy and complexity of computation. *Theoret. Comput. Sci.*, 8(2):227–237, 1979.

[HH76] Juris Hartmanis and John E. Hopcroft. Independence results in computer science. *SIGACT News*, 8(4):13–24, Oct.–Dec. 1976.

[Hin78] Peter G. Hinman. *Recursion-theoretic hierarchies*. Springer-Verlag, Berlin, 1978. Perspectives in Mathematical Logic.

[HP93] Petr Hájek and Pavel Pudlák. *Metamathematics of first-order arithmetic*. Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1993.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley Publishing Co., Reading, Mass., 1979. Addison-Wesley Series in Computer Science.

[JM91] James P. Jones and Yuri V. Matijasevič. Proof of recursive unsolvability of Hilbert's tenth problem. *Amer. Math. Monthly*, 98(8):689–709, 1991.

[Kay91] Richard Kaye. *Models of Peano arithmetic*, volume 15 of *Oxford Logic Guides*. The Clarendon Press Oxford University Press, New York, 1991. Oxford Science Publications.

[KPT91] Jan Krajíček, Pavel Pudlák, and Gaisi Takeuti. Bounded arithmetic and the polynomial hierarchy. *Ann. Pure Appl. Logic*, 52(1-2):143–153, 1991. International Symposium on Mathematical Logic and its Applications (Nagoya, 1988).

[Man80] Kenneth L. Manders. Computational complexity of decision problems. In *Model theory of algebra and arithmetic (Proc. Conf., Karpacz, 1979)*, volume 834 of *Lecture Notes in Math.*, pages 211–227. Springer, Berlin, 1980.

[Mát90] Attila Máté. Nondeterministic polynomial-time computations and models of arithmetic. *J. Assoc. Comput. Mach.*, 37(1):175–193, 1990.

[Mat97] Claus Akira Matsushigue. Uma introdução técnica relativa às provas robustas checáveis probabilisticamente. Master's thesis, IME-USP, São Paulo, Brasil, 1997. Supervised by PhD Ricardo Bianconi.

[MB03a] Claus Akira Matsushigue and Ricardo Bianconi. Independence results via arithmetical complexity hierarchy and complexity recursion theory. *Submitted*, 2003.

[MB03b] Claus Akira Matsushigue and Ricardo Bianconi. Inducing equalized indemonstrabilities and independences of complexity theory issues in a formal logic system. *Submitted*, 2003.

[Pap94] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley Publishing Company, Reading, MA, 1994.

[Smu93] Raymond M. Smullyan. *Recursion theory for metamathematics*. The Clarendon Press Oxford University Press, New York, 1993.

[Sur95] Claude Sureson. NP $\neq$ co-NP and models of arithmetic. *Theoret. Comput. Sci.*, 147(1-2):55–67, 1995.

[Sur96] Claude Sureson. P, NP, co-NP and weak systems of arithmetic. *Theoret. Comput. Sci.*, 154(2):145–163, 1996.

[Tak95] Gaisi Takeuti. Separations of theories in weak bounded arithmetic. *Ann. Pure Appl. Logic*, 71(1):47–67, 1995.

[TY96] Gaisi Takeuti and Masahiro Yasumoto. Forcing on bounded arithmetic. In *Gödel '96 (Brno, 1996)*, pages 120–138. Springer, Berlin, 1996.

[TY98] Gaisi Takeuti and Masahiro Yasumoto. Forcing on bounded arithmetic. II. *J. Symbolic Logic*, 63(3):860–868, 1998.

[Wil80] Alex J. Wilkie. Applications of complexity theory to Σ_0-definability problems in arithmetic. In *Model theory of algebra and arithmetic (Proc. Conf., Karpacz, 1979)*, volume 834 of *Lecture Notes in Math.*, pages 363–369. Springer, Berlin, 1980.

[WP87] Alex J. Wilkie and Jeffrey B. Paris. On the scheme of induction for bounded arithmetic formulas. *Ann. Pure Appl. Logic*, 35(3):261–302, 1987.