Uma Introdução Técnica Relativa às Provas Robustas Checáveis Probabilisticamente

Claus Akira Matsushigue

DISSERTAÇÃO APRESENTADA AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA DA
UNIVERSIDADE DE SÃO PAULO PARA
OBTENÇÃO DO GRAU DE MESTRE EM
MATEMÁTICA APLICADA

Área de Concentração: **Ciência da Computação**
Orientador: **Prof. Dr. Ricardo Bianconi**

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

— São Paulo, SP — Junho de 1997 —

Uma Introdução Técnica Relativa às Provas Robustas Checáveis Probabilisticamente

Claus Akira Matsushigue

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

São Paulo, 10 de julho de 1997.

Banca examinadora:

- Prof. Dr. Ricardo Bianconi (orientador) — MAT-IME-USP
- Prof. Dr. Yoshiharu Kohayakawa — MAC-IME-USP
- Profª. Drª. Katia Silva Guimarães — DI-UFPE

Abstract

Various types of *systems of probabilistic proofs* have played a decisive role in the development of Computer Science Theory in the last decade. This can be verified through the great number of studies about interactive proofs, zero–knowledge proofs, and transparent (or holographic) proofs. These topics are guided by the robustness of the codifications and by the computational capacity of checking them. In this text, we aim at presenting *a tecnical introduction relative to the probabilistically checkable robust proofs*. Within this approach, the new characterization of the non–deterministic polynomial–time class through the **Probabilistically Checkable Proofs** class formulated by Arora, Lund, Motwani, Sudan e Szegedy in [ALM$^+$92], $\mathcal{NP} = \mathcal{PCP}(\log n, 1)$, is of central importance. We intend to prove this characterization, because it encompasses the principal points of the subject and, furthermore, covers subjacently a wide set of computational, algebraic, and probabilistic tools, which are fundamental in this topic.

The theorem $\mathcal{NP} = \mathcal{PCP}(\log n, 1)$ contains the surprising property that problems of decision in $\mathcal{NP}$ may be solved by Turing Machines with polynomial–time in the input length, if they are helped by a robust proof (therefore, a non–deterministic machine) and can toss "honest" coins (a probabilistic machine). However, these machines are restricted to read only a constant number in the input length of letters of the robust proofs, drawn like a logarithmic number of random bits, and to have a constant, and as low as wanted, probability of accepting "wrongly" an input.

Taking as basis the referred theorem, or using its techniques, several results of the *inapproximability of optimal solutions* may be obtained. Thus, the text also presents a simple example of the use of the probabilistically checkable robust proofs. With it, in Combinatorial Optimization it is now known that problems $\mathcal{MAX}$–$\mathcal{SNP}$–hard are not in $\mathcal{PTAS}$ or, in other words, it is impossible to obtain the optimal solution, within a whatsoever constant fraction, in polynomial–time algorithms, supposing $\mathcal{P} \neq \mathcal{NP}$.

Resumo

Na última década, vários tipos de sistemas de provas probabilísticas mostraram ter papel decisivo no desenvolvimento da Teoria da Ciência da Computação. Isto pode ser verificado pelo grande número de estudos acerca das provas interativas, provas com conhecimento–zero e provas transparentes (ou holográficas). Estes tópicos se norteiam pela robustez das codificações e pela capacidade computacional em checá-las. Buscamos, no texto, trazer uma introdução técnica relativa às provas robustas checáveis probabilisticamente. Neste enfoque, a nova caracterização da classe não–determinística de tempo polinomial pela classe das ***Provas Checáveis Probabilisticamente*** *formulada por Arora, Lund, Motwani, Sudan e Szegedy em [ALM$^+$92], $\mathcal{NP} = \mathcal{PCP}(\log n, 1)$, é de importância central. Pretendemos provar tal caracterização, visto que ela perpassa os principais pontos do assunto e, ainda, abrange subjacentemente um grande ferramental computacional, algébrico e probabilístico, que é fundamental neste tópico.*

O teorema $\mathcal{NP} = \mathcal{PCP}(\log n, 1)$ tem no seu bojo a surpreendente propriedade que problemas de decisão em $\mathcal{NP}$ podem ser resolvidos por Máquinas de Turing de tempo polinomial no comprimento da entrada, se obtiverem ajuda de uma prova robusta (portanto, uma máquina não–determinística) e se puderem lançar moedas "justas" (uma máquina probabilística). Entretanto, estas máquinas são restritas a só poderem ler um número constante no comprimento da entrada de letras da prova robusta, sorteadas por um número logarítmico de bits aleatórios, e têm uma probabilidade constante, e tão pequena quanto se queira, de aceitarem "erradamente" uma entrada.

Tendo como base o referido teorema, ou utilizando-se as suas técnicas, pode-se obter diversos resultados de inaproximabilidade de soluções ótimas. Assim, o texto também fornece um exemplo simples do uso das provas robustas checáveis probabilisticamente. Com ele, na Otimização Combinatória sabe-se hoje que os problemas $\mathcal{MAX}$–$\mathcal{SNP}$–difíceis não estão em $\mathcal{PTAS}$ ou, em outras palavras, não se pode obter a solução ótima, a menos de uma fração constante qualquer, com algoritmos de tempo polinomial, supondo-se $\mathcal{P} \neq \mathcal{NP}$.

Conteúdo

Olhando para
o céu, vejo
Leonardo Pareja
muito mais parecendo
uma luz. Homenageio-o
pela sua sensatez,
agradecendo-lhe por ter
me iluminado o caminho,
e dedico, humildemente,
anonimamente, todo
o meu suor
a você!!!

É parte intrínseca da minha vida a presença carinhosa e constante da Fátima. Dia após dia, ela tem me sustentado afetivamente e, nos últimos tempos, financeiramente também. Percebo que quanto mais eu trabalho, mais pobre estou ficando... Agradeço-lhe então pela enorme compreensão a mim dispensada, pois a vida ao meu lado acaba requisitando-a em muito. Possuo também um forte e seguro ancoradouro nas pessoas da minha família: meus pais Tossio e Lighia, minhas irmãs Cynthia e Karin, meus tios, primos, sogros e cunhados e tenho uma especial lembrança dos meu avós. Quero ressaltar a minha irmã mais nova, que tem passado tantos anos ao meu lado, e a minha mãe, que, em tudo que pode e em que não pode, tanto me incentivou e apoiou no meu sofrido caminho pessoal e profissional.

Agradeço de coração à frutífera (para mim) orientação do Ricardo Bianconi, com quem aprendi muito, e posso dizer hoje, depois deste tempo conjunto: sou uma outra pessoa! Também expresso a minha enorme gratidão, pela minha introdução ao mundo da Ciência da Computação Teórica, ao José Augusto Ramos Soares e ao Yoshiharu Kohayakawa. Por outro lado, fui formado pelas diversas instituições por onde passei e quero homenagear o mestre dos mestres Jacob Zimbarg Sobrinho da USP - São Paulo e os professores Henrique Lazari e Irineu Bicudo da UNESP - Rio Claro e José Carlos da Escola Mutirão. Agradeço, em particular, à grande contribuição do colega Armando Ramos Gouveia pelas suas observações e dicas ao texto.

Entretanto, passamos o dia inteiro com os inesquecíveis amigos, que acabam sendo, para nós, o grande referencial da vida. Eles não poderiam ficar de fora dos agradecimentos. Lembro alguns: Fátima, Peter, Andrea, Astolfo, Sanae, Birgit, Sandra, Luciano, Walquiria, Luiz, Dirce, Marília, Clarice, Thereza, Emico, Moira, Suzana, Rosa, Erika, Sheila, Olga, Sergio... Jair, Leonardo, Francisco, Fábio, Edna, Marco, Eliany, Marcelo, Renata, Ayumi, Renata, Edson, Orlando, Fabiana, Flavio, Eloi, Alfredo, Roberto, Cássio, Marcelo, Antônio, Gaspar, Lúcia, Juací, Márcio, Carlos, Jorge, Cecília, Irene, Fernando, Daniela, Marcela, Nelson, Marcelo, Carolina, Cristina, Péricles, Fábio, Ana Lúcia, Carlos, Marko, Victor, Alancardek, Samuel, Raul... Adriana, Cláudia, Maurício, Marco, Patrícia, Sílvia, Ana Cláudia, Marcelo, Nelson, Cássio, Andréia, Renata, Cozin, Nadir, Flávia, Eliana, Alexandre, Cátia, Luciana, Silvana, Carlos, Carla, Capivara, Roberto, Elisabeth, Cláudia, Silvano... E obrigado por tudo mais...

Claus

Capítulo I

Esquema de Provas Robustas

Neste texto tentaremos introduzir o leitor ao fascinante mundo das **provas robustas checáveis probabilisticamente**. Na *Teoria da Complexidade Computacional*, elas originaram a classe de complexidade $\mathcal{PCP}(R(n), Q(n))$, que têm as suas iniciais correspondendo a *Probabilistically Checkable Proofs*.

Temos como meta apresentar, apesar de um modo introdutório e direto, um intrincado arcabouço técnico no qual está imersa esta área. Por isto não nos preocuparemos muito com relação à motivação ao tema, nem daremos um apanhado mais geral sobre o mesmo. Pode-se, também, sentir falta de uma introdução histórica. Portanto, para quem é leigo no assunto, recomendamos fortemente se inicializar por outros textos, com por exemplo Goldreich [Gol94]. Nele temos um resumo sobre os *sistemas de provas probabilísticas*. Entretanto, talvez o maior motivador às *classes de complexidade probabilísticas* seja Babai em [Bab90]. No Brasil, recomendamos a leitura de Kohayakawa e Soares [KS95].

Neste sentido, o nosso texto procurará se centrar no arsenal de **técnicas** que esta área exige. Elas se baseiam: na parte computacional, num melhor tratamento dos *esquemas de provas* (ou seja, dos *testes sobre testemunhas*), que embutem aí as *provas interativas*, sendo que as mesmas estavam tão em "voga" anteriormente; e na parte algébrico–probabilística, num aprofundamento no acoplamento entre estas duas teorias, permitindo assim um grande avanço em relação aos algoritmos probabilísticos. Estas técnicas surgem da aproximação da Área de Complexidade Computacional com áreas como Criptografia e Teoria dos Códigos. Passam, deste modo, a Álgebra e a Probabilidade, a serem o "carro chefe" dos ferramentais teóricos e contribuem sobremaneira na definição do **esquema de provas robustas**. As *provas robustas*, também chamadas de *provas transparentes* ou *holográficas*, permitem uma "fácil" verificação da corretude das suas testemunhas.

Até agora, talvez, o ápice da utilização destas novas técnicas seja o intrigante Teorema do $\mathcal{PCP}$ I.4–4 (*Probabilistically Checkable Proofs Class*), isto é, $\mathcal{NP} = \mathcal{PCP}(\log n, 1)$, de Arora, Lund, Motwani, Sudan e Szegedy [ALM$^+$92], que se baseou em Arora e Safra [AS92]. Nele, todos os problemas em $\mathcal{NP}$ (*Non–deterministic Polynomial Time Class*) têm um *esquema de provas robustas* que permite checar, em tempo polinomial e com probabilidade de erro tão pequena quanto se queira, se uma dada *testemunha* é uma *correta prova*. Para isto, podem-se utilizar *bits* aleatórios em número apenas logarítmico no comprimento da entrada e ler no máximo um número constante de letras da testemunha.

O maior problema da Teoria da Complexidade Computacional é saber se realmente $\mathcal{P} \neq \mathcal{NP}$ (em

que $\mathcal{P}$ advém de *Deterministic Polynomial Time Class*) e, muito a grosso modo, isto significa saber, com relação às computações de tempo polinomial, se as testemunhas (e/ou os *oráculos*) são "efetivamente úteis". Ou seja, se temos um problema de decisão que sabemos resolver em tempo polinomial com a ajuda das testemunhas, também podemos resolvê-lo em tempo polinomial dispensando-se as testemunhas? Agora o Teorema do $\mathcal{PCP}$ I.4–4, isto é $\mathcal{NP} = \mathcal{PCP}(\log n, 1)$, nos afirma que se temos uma computação que pode contar, além da testemunha, com lances "justos" de dados e uma permissão de "erro" na resposta (mesmo que se exija um erro muito pequeno e constante), então toda computação de tempo polinomial com a ajuda da testemunha também pode ser feita lendo-se apenas um número constante de letras da testemunha e lançando-se apenas um número logarítmico de vezes o dado. Isto é realmente incrível, tendo-se em vista a nossa convicção que $\mathcal{P} \neq \mathcal{NP}$!

No final dos anos 80 emergiu a utilização prática dos *algoritmos probabilísticos* para diversos fins. No seu rastro, fortaleceram-se muito dois tópicos distintos da Teoria da Complexidade Computacional. O primeiro inaugurou a ascensão dos testes computacionais que buscam *checar codificações* e *auto–corrigi*-las. Sobre isto, pode-se ler Blum e Kannan [BK89], Friedl, Hátsági e Shen [FHS94], Gemmell, Lipton, Rubinfeld, Sudan e Wigderson [GLR$^+$91], Rubinfeld e Sudan [RS92] e Blum, Luby e Rubinfeld [BLR93]. Por outro lado, temos o segundo tópico centrado nas classes de complexidade com *provas interativas* e nas suas respectivas hierarquias. O seu percurso pode ser acompanhado em Goldwasser e Sipser [GS86], Babai [Bab88], Babai e Moran [BM89], Lund, Fortnow, Karloff e Nisan [LFKN92], Shamir [Sha92] e Shen [She92]. Note que, pela proximidade, visto que ambos são igualmente baseados nas *características probabilísticas sobre propriedades algébricas*, estes dois tópicos acabam se fundindo. Fruto disto, definiu-se a classe $\mathcal{PCP}$, que é uma generalização das classes interativas, e obtiveram-se surpreendentes resultados, como o Teorema do $\mathcal{PCP}$ I.4–4.

O interessante é que, apesar de intrigante, o Teorema do $\mathcal{PCP}$ I.4–4 poderia ter apenas uma conseqüência abstrata. Mas, não! Os seus principais resultados surgiram na Área de Otimização Combinatória e se referem a **inaproximabilidade** às soluções ótimas por computações de tempo polinomial de diversos e importantes problemas de otimização compatíveis aos $\mathcal{NP}$–difíceis. No começo dos anos 90, pesquisadores, ao se darem conta do quão fértil este assunto é para a Otimização Combinatória, acabaram tomando esta última, que seria apenas uma conseqüência, como o grande impulsor do estudo sobre a Classe de Complexidade $\mathcal{PCP}$.

Com isto, buscaremos somente provar o Teorema do $\mathcal{PCP}$ I.4–4, visto que ele abrange um contingente grande de detalhes, podendo assim servir como *uma introdução técnica relativa às provas robustas checáveis probabilisticamente*. Apenas como ilustração, daremos no final do primeiro capítulo uma rápida prova da inaproximabilidade em tempo polinomial para a Classe de Complexidade $\mathcal{MAX}-\mathcal{SNP}$. Deste modo, dedicamos o Capítulo I *Esquema de Provas Robustas* às concepções mais características da Teoria da Complexidade Computacional propriamente dita. A partir do segundo capítulo iniciamos a demonstração do Teorema do $\mathcal{PCP}$ I.4–4 e de imediato começaremos com a Seção II.1 *Estrutura Global da Demonstração do Teorema do $\mathcal{PCP}$* que procurará fornecer uma visão geral da referida demonstração. No restante do Capítulo II *Codificações Algébricas* e por todo o Capítulo III *Testes Probabilísticos* continuamos fornecendo os subsídios teóricos preliminares necessários aos *esquemas de provas robustas*. Fica, assim, para o Capítulo IV *Classe de Complexidade $\mathcal{PCP}$* o fechamento da demonstração do Teorema do $\mathcal{PCP}$ I.4–4, utilizando-se da aritmetização de fórmulas booleanas e do importante Teorema da Composição de Testes IV.4–2. Já o Capítulo V *Polinômios de Grau Baixo* é mais técnico e trata do ferramental essencial, envolvendo Álgebra e Probabilidade. Tal técnica nos permite asseverar sobre a proximidade de uma função qualquer a um polinômio de grau baixo e é fundamental na prova da corretude de alguns testes do terceiro capítulo.

As referências básicas do texto são Hougardy, Prömel e Steger [HPS94], Sudan [Sud92] e, principalmente, Arora [Aro94]. Começaremos, agora, fixando um pouco de notação.

I.1 Introdução

I.1–1. Como de costume, $[a, b)$, $(a, b]$, etc. serão intervalos reais (i.e., de $\mathbb{R}$), fechados para "[" e "]", e abertos para "(" e ")". Já para um conjunto A, denotaremos $\boldsymbol{A_*} := A \setminus \{0\}$, ou seja, retira-se o elemento zero de A e $\boldsymbol{A_+} := \{x \in A \cap \mathbb{R} \parallel x \geq 0\}$. O símbolo "**:=**" (ou "**=:**") será utilizado significando *definição*. Por outro lado, $\boldsymbol{\{i \ldots j\}}$ é o conjunto de inteiros (i.e., em $\mathbb{Z}$) de i a j. O conjunto $\mathbb{Z}_{\boldsymbol{k}} := \{0 \ldots k-1\}$ é um grupo aditivo de k elementos. Mais ainda, para um primo p, $\mathbb{Z}_p$ é um corpo e $\mathbb{Z}_{p^n}$ pode ser visto como o único (a menos de isomorfismo) $\mathbb{Z}_p$–espaço vetorial de dimensão n. Se o produto por escalar deste espaço vetorial for tomado convenientemente, sempre podemos estendê-lo para todo o $\mathbb{Z}_{p^n}$, transformando-o também em um corpo. Se $x \in \mathbb{R}$ é um número real, então $\boldsymbol{\lceil x \rceil}$ será chamado de `teto de` $\boldsymbol{x}$ e é o menor número natural $\mathbb{N}$ que é pelo menos x.

Para conjuntos não vazios A, X, denotaremos por $\boldsymbol{^A X}$ ao conjunto de todas as funções $\boldsymbol{f\colon A \to X}$. Continuando, as funções $\boldsymbol{f\colon A \hookrightarrow X}$, $\boldsymbol{f\colon A \twoheadrightarrow X}$ e $\boldsymbol{f\colon A \hookrightarrow\!\!\!\!\twoheadrightarrow X}$ são respectivamente, injetora, sobrejetora e bijetora. Se $C \subseteq A$ então $\boldsymbol{f[\![C]\!]}$ é a imagem de C por f e $\boldsymbol{f\restriction_C}$ é f com o domínio restrito a C. Já $\boldsymbol{\#A}$ será a cardinalidade de A.

Chamaremos de `função natural` às funções com domínio natural positivo ($\mathbb{N}_*$), contra–domínio real ($\mathbb{R}$) e que possam ser computadas em tempo polinomial.[i] Seja uma família de funções naturais $\mathcal{F}(n)$, denotaremos por $\boldsymbol{\mathbf{Poli}(\mathcal{F}(n))}$ a também família de funções naturais que tenham crescimento `polinomial sobre` $\boldsymbol{\mathcal{F}(n)}$. Ou seja, para todas as funções $f_1(n), \ldots, f_{k(n)}(n) \in \mathcal{F}(n)$, $\mathrm{Poli}(\mathcal{F}(n))$ é a família das funções dadas por $P(f_1(n), \ldots, f_{k(n)}(n))$, em que P é um polinômio de k variáveis[ii] com coeficientes em $\mathbb{N}$. A família das funções `de no máximo da ordem de` $\boldsymbol{\mathcal{F}(n)}$, ou simplesmente `da ordem de` $\boldsymbol{\mathcal{F}(n)}$, será denotada por $\boldsymbol{\mathcal{O}(\mathcal{F}(n))}$ e representa as funções majoradas por funções de crescimento linear sobre $\mathcal{F}(n)$, isto é, $h(n) \in \mathcal{O}(\mathcal{F}(n))$ sse[iii] existem $a, b \in \mathbb{N}$ e $f(n) \in \mathcal{F}(n)$, tais que $h(n) \leq af(n) + b$, para todo $n \in \mathbb{N}$. Por outro lado, $\boldsymbol{\Omega(\mathcal{F}(n))}$ são as funções `de no mínimo da ordem de` $\boldsymbol{\mathcal{F}(n)}$. Logo são as funções $h(n)$, tais que existem $a, b \in \mathbb{N}$ e $f(n) \in \mathcal{F}(n)$, tais que $af(n) - b \leq h(n)$, para todo $n \in \mathbb{N}$. É evidente que, se a família $\mathcal{F}(n) := \{f(n)\}$ tiver uma só função, denotá-la-emos por $\mathrm{Poli}(f(n))$, $\mathcal{O}(f(n))$ e $\Omega(f(n))$ no lugar de $\mathrm{Poli}(\{f(n)\})$, $\mathcal{O}(\{f(n)\})$ e $\Omega(\{f(n)\})$. Por abuso, tomaremos somente a função nula como de ordem zero (i.e., $\boldsymbol{\mathcal{O}(0) := \Omega(0) := \{0\}}$).

Se φ for uma fórmula lógica, então a `função característica` $\boldsymbol{[\varphi]}$ é 1 se φ é verdadeiro e 0 se φ é falso. Tomando-se x como uma variável que percorre o conjunto X e, se $\varphi(x)$ estiver em função dela, denotaremos por $\boldsymbol{\{\varphi(x)\}} := \{\, x \in X \parallel \varphi(x) \,\}$ o conjunto que a satisfaz. Uma `probabilidade` $\boldsymbol{\mathcal{P}r_X\,(\,A\,)}$ (de A em X) é uma medida sobre uma σ–álgebra, (a qual pressupomos englobar todos os subconjuntos A de X), em que vale $\mathcal{P}r_X\,(\,X\,) = 1$. Seguindo a convenção anterior, escrevemos $\boldsymbol{\mathcal{P}r_{x \in X}\{\,\varphi(x)\,\}}$ para designar a probabilidade de ser satisfeita $\varphi(x)$, com x em X.

I.2 Classes de Complexidade sobre Problemas de Decisão

Introduziremos, no que for necessário, a formalidade da *Teoria da Complexidade sobre Problemas de Decisão*. Não temos a pretensão de ensinar a ninguém que já não tenha tido contacto com o assunto, somente fixaremos a nomenclatura. Para uma melhor familiaridade com os termos aqui utilizados, recomendamos a

[i] Seria necessário precisar esta definição e declarar mais restrições computacionais às *funções naturais*, entretanto, não entraremos nestes detalhes por ora.

[ii] Aqui k varia em $\mathbb{N}_*$.

[iii] No decorrer do texto, utilizaremos, como é de praxe, *"sse"* como sendo *"se e somente se"*.

leitura de Papadimitriou [Pap94] e de Lovász [Lov94].

I.2–1. Um `alfabeto` Σ é um conjunto finito e não vazio, cujos elementos chamaremos de `letras`. O `símbolo separador` será uma letra especial, que denotaremos por $\not\mid$. Assim, $\mathbf{\Sigma_*} := \Sigma \setminus \{\not\mid\}$ é o alfabeto Σ sem o símbolo separador. Letras distintas do símbolo separador serão chamadas de `letras relevantes`.[iv] Para $n \in \mathbb{N}_*$, destacaremos o alfabeto especial $\mathbf{\Psi_{(n)}}$, o qual contém o símbolo separador e as suas letras relevantes serão os elementos de $\mathbb{Z}_n$. Mais especificamente, diremos que $\Psi_{(2)}$, que só tem os elementos 0, 1 e $\not\mid$, é o `alfabeto binário` e as suas letras relevantes serão os ***bits***. Se Σ é um alfabeto,[v] $\mathbf{\Sigma^n}$ será o conjunto das n-uplas σ de letras de Σ. Cada uma delas chamaremos de ***string***, diremos que o seu `comprimento` é n e o denotaremos por $\|\boldsymbol{\sigma}\| := n$. Também, $\mathbf{\Sigma^*}$ será o conjunto dos *strings* de Σ e, se Σ não contiver o símbolo separador $\not\mid$ (p.e., $\Sigma_* \subseteq \Sigma \setminus \{\not\mid\}$), os *strings* por ele formados serão as `palavras`.[vi] O `vocabulário` é o conjunto de todas as palavras de um alfabeto Σ, isto é, $\mathbf{\Sigma_*^*}$. Qualquer subconjunto do vocabulário será um `problema de decisão` e as suas palavras, as `instâncias do problema de decisão`. Estaremos interessados em *máquinas* que, ao receberem alguma `entrada` (isto é, uma palavra), decidam se ela é ou não uma instância do problema de decisão.

Ainda, uma `fita de uma–via` sobre o alfabeto Σ é uma seqüência de letras de Σ indexada pelos números naturais $\mathbb{N}$ e, caso a indexemos pelos inteiros $\mathbb{Z}$, temos uma `fita de duas–vias`. Esta indexação definirá as *posições*, ou seja, os `endereços` da fita. Uma *Máquina de Turing* terá, para cada uma das suas fitas, uma `cabeça de leitura` que estará posicionada em um dos seus endereços e apontando para a correspondente letra. Esta cabeça de leitura poderá ir à direita ou à esquerda e fornecerá à máquina (i.e., `lerá`) a letra por ela endereçada. Mas se a fita for de `leitura–escrita`, a cabeça também poderá escrever uma letra do alfabeto sobre o endereço da cabeça de leitura. Se isto não for o caso, a fita será chamada de `só–leitura`. Entretanto, a fita será de `só–escrita` sse não puder fornecer à máquina a letra endereçada pela cabeça de leitura. Todas as Máquinas de Turing têm uma `fita de trabalho` de *duas–vias* que é de *leitura–escrita* no alfabeto binário $\Psi_{(2)}$. No começo da computação, esta fita poderá fornecer alguma informação útil à execução da máquina, a qual chamaremos de `constantes da computação`, com a estrita restrição de não poder variar com as entradas da computação. Outro tipo de fita é a de `acesso direto` na qual, para ler (`leitura–direta`) ou escrever (`escrita–direta`) em uma das suas posições, não há necessidade de se ir com a cabeça de leitura até a mesma, bastando escrever o número do endereço na palavra à esquerda da posição da cabeça de leitura da fita de trabalho. Então, a máquina em tempo "*um*" posicionará a sua cabeça de leitura no referido endereço.

Consideraremos toda *Máquina de Turing* como sendo *determinística*. A parte não–determinística virá da leitura de uma fita adicional. Excetuando-se a *fita de trabalho*, todas as outras fitas das máquinas serão de *só–leitura* e *uma–via*, devendo começar a computação no primeiro endereço e serão escolhidas entre as três abaixo:

- `fita de entrada` ($\boldsymbol{\rho}$) na qual, no começo da computação, deverá estar escrita somente a *palavra de entrada* x, precedida por um símbolo separador e completada, ao final, pela seqüência infinita de símbolos separadores.

- `fita de testemunha` ($\boldsymbol{\pi}$) de *leitura–direta*, que deverá conter palavras que variam conjuntamente com a entrada x e que podem (e neste caso pensá-las-emos como sendo uma `prova`), ou não, ajudar na computação, operando assim a parte *não–determinística* do problema de decisão.

- `fita de probabilidade` ($\boldsymbol{\tau}$) com o alfabeto binário $\Psi_{(2)_*}$, em que só aparecem as letras relevantes, (i.e., não tem o símbolo separador). Na verdade é uma seqüência de *bits* $\mathbb{Z}_2$, que

[iv] Portanto, um *alfabeto* tem todas as suas letras distintas entre si, assim como podemos distinguir o símbolo separador $\not\mid$ das demais *letras relevantes*.

[v] Isto é, conjunto de letras relevantes com ou sem o símbolo separador.

[vi] Ou seja, as palavras são os *strings* que só têm letras relevantes, portanto não são formadas pelo símbolo separador $\not\mid$.

teoricamente devem ser escolhidos com probabilidade uniforme.

Uma `Máquina de Turing` deve ser pensada como um *programa de computador*. Ela é uma seqüência numerada e finita de comandos, chamados de `estados` e que devem ser executados ordenadamente, a menos de possíveis desvios. Cada estado pode conter, para cada fita da máquina, um dos quatro comandos abaixo:

- dependendo da letra lida no endereço da cabeça de leitura da fita, desviar para um outro estado;
- escrever na fita uma das letras do alfabeto da fita;
- ir com a cabeça de leitura para o endereço à direita ou à esquerda na fita;[vii] e
- ir com a cabeça de leitura para o endereço codificado na palavra à esquerda da cabeça de leitura da fita de trabalho.

É importante ver que a leitura e/ou a escrita nos dois primeiros itens depende do tipo de permissão definida pela máquina para cada uma das suas fitas, assim como o último item depende da permissão de *acesso direto* da fita.

Já uma Máquina de Turing, ao ler uma fita de só–leitura, pode não querer distinguir as letras relevantes (i.e., as letras, a menos do símbolo separador $\not\flat$). Isto acontece se todo comando de leitura tratar igualmente todas as letras relevantes, podendo apenas variar a instrução para o símbolo separador. Seria como se a fita tivesse alfabeto $\Psi_{(1)}$. Neste caso, a máquina fica restrita a saber o comprimento das palavras, não podendo decodificá-las. Diremos então que a máquina tem a fita como `só–leitura–indistinguível`. A Máquina de Turing `pára a computação` se no primeiro item a máquina é desviada para um estado que porventura não exista. Ao parar a computação, uma máquina `aceita a entrada` x fornecida na fita de entrada sse a cabeça de leitura da fita de trabalho estiver sobre um símbolo separador $\not\flat$. O `tempo` de execução desta máquina é o número dos estados executados até ela parar. Estaremos sempre interessados no tempo de parada para aceite de entradas.

I.2–2. Queremos agora utilizar *propriedades probabilísticas* sobre *modelos computacionais* de tal forma que possamos "reconhecer" problemas de decisão, dividindo-os assim em *classes de complexidade*. Definamos, então, o que chamaremos de modelos computacionais e propriedades probabilísticas. Os nossos `modelos computacionais` serão de Máquinas de Turing que sempre se valem da fita de entrada ρ e de trabalho, nas quais não haverá limitações, e sobre elas iremos definir:

- a utilização ou não das fitas de testemunha π e de probabilidade τ;
- o alfabeto das fitas de entrada ρ e de testemunha π; e
- restrições sobre quais *aceites das entradas* da máquina consideraremos como `válidos`, como por exemplo:
  - quanto à limitação no tempo de execução e
  - quanto à limitação no número de leituras nas fitas de testemunha π e de probabilidade τ.

Tendo um *modelo computacional* fixado, definiremos `propriedades probabilísticas` sobre as fitas de teste-

[vii] Se a fita for de uma–via, a cabeça de leitura estiver no primeiro endereço e o comando for ir à esquerda, então nada acontecerá.

munha π e de probabilidade τ [viii] com respeito aos *aceites válidos* das palavras dadas como entrada. Estas *propriedades probabilísticas* são compostas por duas propriedades mutuamente contraditórias, uma para definir as palavras do alfabeto da fita de entrada ρ da máquina que devem pertencer ao problema de decisão e outra para definir as palavras que não devem pertencer. Então, sobre eles definimos uma `classe de complexidade` como sendo composta pelos problemas de decisão L de alfabeto Σ, para os quais existe uma Máquina de Turing M deste *modelo computacional* que também tenha a fita de entrada ρ no alfabeto Σ, tal que as palavras que são instâncias de L satisfaçam a primeira *propriedade probabilística* e as palavras de Σ que não são de L satisfaçam a segunda *propriedade*. Observe o seguinte: se as duas propriedades probabilísticas são complementares entre si, toda máquina deste modelo computacional sempre define um problema de decisão. Caso contrário, ou seja, quando as duas propriedades formam um "*gap*", isto já não é verdade "*a priori*". Neste caso, somente as máquinas do modelo computacional que não tenham entradas que caiam neste *gap* de propriedades definirão um problema de decisão. Isto é, máquinas que tenham todos os seus *aceites válidos* das entradas de tal forma que sempre satisfaçam uma das duas *propriedades probabilísticas*.

Assim, uma *classe de complexidade* é a família dos *problemas de decisão* definidos quando fixamos um *modelo computacional* e as *propriedades probabilísticas*. Ainda, diremos que a Máquina de Turing `reconhece` o problema de decisão por ela definido como sendo da referida *classe de complexidade*. Na Teoria da Complexidade convencional, como veremos a seguir, são definidas classes de complexidade com propriedades probabilísticas complementares entre si. Mas o nosso trabalho se centra na classe $\mathcal{PCP}$ e buscaremos máquinas que reproduzam este "*gap*" nas propriedades.

I.2–3. Esta definição tornar-se-á mais clara com os exemplos e, portanto, descreveremos agora alguns *modelos computacionais* e as respectivas *propriedades probabilísticas*, com as quais definiremos as *classes de complexidade* por nós utilizadas. Seja $\mathcal{F}(n)$ uma família de funções naturais. A primeira classe de complexidade é a `determinística de tempo em` $\boldsymbol{\mathcal{F}(n)}$ e é denotada por $\boldsymbol{\mathcal{TIME}(\mathcal{F}(n))}$. Ela será o conjunto dos problemas de decisão em que as suas *instâncias* são as entradas aceitas por Máquinas de Turing que só usam as fitas obrigatórias (de entrada ρ e de trabalho) e, para algum $t(n) \in \mathcal{F}(n)$, cada palavra de entrada x no alfabeto da fita de entrada ρ só será considerada como aceita, se a sua computação for em tempo não superior a $t(\|x\|)$. Temos ainda a classe $\boldsymbol{\mathcal{NTIME}(\mathcal{F}(n))}$ que é `não–determinística de tempo em` $\boldsymbol{\mathcal{F}(n)}$. O seu *modelo computacional* será de Máquinas de Turing que rodam sobre a fita de testemunha π,[ix] em que também será necessário existir $t(n) \in \mathcal{F}(n)$, tal que uma entrada x vai ser considerada aceita (ou será válida a sua aceitação) se a máquina o fizer em tempo no máximo $t(\|x\|)$. Definimos a *classe de complexidade* como sendo dos problemas de decisão L que tenham `esquema de provas` para máquinas M deste modelo computacional, isto é, para toda palavra x no alfabeto da fita de entrada de M,

$$x \in L \quad \text{sse} \quad \Pr_{\pi}\{\ M \text{ aceita } x\ \} > 0.$$

Aí a probabilidade é tomada uniformemente sobre todas as possíveis fitas de testemunha π. Veja que o número das possíveis fitas de testemunha é infinito e, rigorosamente falando, não podemos tomá-las uniformemente. Tomamos assim cada uma das suas letras uniformemente. Mas, como o tempo de aceite é limitado por $t(n)$, considera-se somente os $t(\|x\|)$ primeiros endereços da fita de testemunha. Deste modo, tendo em vista apenas a parte relevante da fita de testemunha, ela passa a ser escolhida uniformemente. Este mesmo critério será utilizado outras vezes.

Portanto, tudo isto significa dizer que L é o conjunto das palavras x que têm pelo menos uma testemunha π_x, que é chamada de `prova` da pertinência de x em L, e faz com que a máquina o aceite em tempo no máximo $t(\|x\|)$. As duas classes de complexidade mais importantes são a de `tempo polinomial` e a `não–determinística de tempo polinomial`, respectivamente:

$$\boldsymbol{\mathcal{P}} \ := \ \mathcal{TIME}\big(\ \text{Poli}(n)\ \big)$$

[viii] Isto se o *modelo computacional* admitir a utilização de tais fitas.
[ix] Além, é claro, das fitas de entrada ρ e de trabalho.

e

$$\boldsymbol{\mathcal{NP}} \;:=\; \mathcal{NTIME}\big(\,\mathrm{Poli}(n)\,\big).$$

Se L é um problema de decisão sobre um alfabeto, o seu `problema de decisão complementar` (**co–$\boldsymbol{L}$**) é definido pelas palavras deste alfabeto que não estão em L. Se $\mathcal{X}$ é uma classe de complexidade, a `classe de complexidade complementar` (**co–$\boldsymbol{\mathcal{X}}$**) são os problemas de decisão complementares aos de $\mathcal{X}$. Seja uma Máquina de Turing M que *reconhece* um problema de decisão L como sendo de $\mathcal{P}$, tendo o limite de tempo $t(n) \in \mathrm{Poli}(n)$. Então grava-se a função $t(n)$, ponto a ponto, na fita de trabalho como uma *constante da computação*.[x] Se temos uma entrada de comprimento n, então em tempo polinomial, podemos ler $t(n)$ e simular a computação de M nos $t(n)$ primeiros passos. Se até lá M não aceitou a entrada, aceitamo-la. Assim podemos decidir co–L em tempo polinomial no comprimento da entrada e temos que $\mathcal{P} = \text{co–}\mathcal{P}$. Observe que não podemos fazer a mesma coisa com a classe $\mathcal{NP}$.

Trabalharemos, a seguir, com um pouco de probabilidade.

I.3 Classes de Complexidade Probabilísticas

Neste ponto, veremos *propriedades probabilísticas* que são dadas sobre *esquemas de provas robustas*, com as quais definiremos a classe de complexidade $\mathcal{PCP}$, generalizaremos o resto das classes de complexidade e definiremos também os testes probabilísticos necessários para a prova do Teorema do $\mathcal{PCP}$ I.4–4. Esperamos que, ao final deste, fique mais clara a vantagem de definirmos as classes de complexidade pelos modelos computacionais e propriedades probabilísticas, o que, à primeira vista, pode parecer desnecessário.

I.3–1. Seja M uma Máquina de Turing que tem as fitas de entrada ρ com alfabeto Σ e de trabalho e possivelmente as fitas de probabilidade τ e de testemunha π com alfabeto Φ. Neste caso, um problema de decisão $L \subseteq \Sigma^*$ tem `esquema de provas robustas` para M sse existe $\varepsilon \in (0,1)$, tal que

$$\text{se} \quad x \in L \quad \text{então} \quad \Pr_{\pi}\Big\{\, \Pr_{\tau}\{\, M \text{ aceita } x \,\} = 1 \,\Big\} > 0$$

e

$$\text{se} \quad x \in \Sigma^* \smallsetminus L \quad \text{então} \quad \Pr_{\pi}\Big\{\, \Pr_{\tau}\{\, M \text{ aceita } x \,\} < \varepsilon \,\Big\} = 1.$$

Lembremos inicialmente que as probabilidades são tomadas uniformemente. Chamaremos ainda ε de `taxa de erro da robustez`.

Vejamos que temos aqui um tipo de "*gap*" que corresponde à *taxa de erro da robustez* ε. Note que, se a máquina M **não** tem a fita de probabilidade τ, então $\Pr_\tau\{\, M \text{ aceita } x \,\} < \varepsilon$ sse M não aceita a entrada x. Logo, para as classes não–determinísticas, dizer que um problema de decisão L tem *esquema de provas* para M é a mesma coisa que dizer que L tem *esquema de provas robustas* para M. Por outro lado, se pudéssemos pegar $\varepsilon := 1$, perceba que $\Pr_\pi\{\, \Pr_\tau\{\, M \text{ aceita } x \,\} = 1 \,\} > 0$ e $\Pr_\pi\{\, \Pr_\tau\{\, M \text{ aceita } x \,\} < \varepsilon \,\} = 1$ são complementares. Neste caso, toda Máquina de Turing do modelo computacional definiria um problema de decisão. Como isto não acontece, temos aí algumas máquinas que não definem nenhum problema de decisão. Note que, até agora, definimos *propriedades probabilísticas* de tal modo que, para cada Máquina de Turing de um *modelo computacional*, temos um *problema de decisão* a ela associado (ou definido), mas isto não precisa ser de praxe.

Continuando, suponha que a máquina M **não** tenha a fita de testemunha π. Então $\Pr_\pi\{\, \Pr_\tau\{\, M \text{ aceita } x \,\} = 1 \,\} > 0$ sse $\Pr_\tau\{\, M \text{ aceita } x \,\} = 1$. Por outro lado,

[x] Vide definição no Parágrafo I.2–1.

$\mathcal{P}r_\pi\{\ \mathcal{P}r_\tau\{\ M \text{ aceita } x\ \} < \varepsilon\ \} = 1$ sse $\mathcal{P}r_\tau\{\ M \text{ aceita } x\ \} < \varepsilon$. Deste modo, para uma família de funções naturais $\mathcal{F}(n)$, a classe de complexidade `aleatória de tempo em` $\boldsymbol{\mathcal{F}(n)}$, ou seja, $\boldsymbol{\mathcal{RTIME}(\mathcal{F}(n))}$, será definida pelos problemas de decisão L que tenham *esquema de provas robustas* por uma Máquina de Turing M do modelo computacional de máquinas com fita de probabilidade τ [xi] e que têm aceites válidos em tempo não superior a uma função de $\mathcal{F}(n)$. Temos também a classe `aleatória de tempo polinomial`,[xii] que muitas vezes é chamada de `Monte Carlo`, e é dada por

$$\boldsymbol{\mathcal{RP}} := \mathcal{RTIME}\big(\ \text{Poli}(n)\ \big).$$

Voltemos ao "*gap*". Neste caso, conforme descrito, temos que, se $x \in L$, M sempre aceita e, se $x \notin L$, M aceita **erradamente** com probabilidade menor do que a *taxa de erro da robustez* ε. Como vimos, a máquina M, para definir um problema de decisão com o *esquema de provas robustas*, precisa produzir um *gap* de fração $1-\varepsilon$ nas fitas de probabilidade τ. Ele é a probabilidade da certeza que podemos ter na computação. Note que, se trocássemos o "$< \varepsilon$" por "$= 0$" na segunda propriedade, recairíamos na classe não–determinística, no caso geral, e na classe determinística, se não temos a fita de testemunha π.

I.3–2. Observe que só usaremos *modelos computacionais* nos quais as limitações que dependam do comprimento da entrada sempre serão tomadas sobre $\mathcal{O}$ ou Poli. Assim, se temos uma máquina de um modelo computacional, outra máquina que simule a computação desta máquina em um número constante z de vezes, continua sendo do mesmo modelo computacional.[xiii] Com isto, para uma máquina de um modelo computacional que tenha um esquema de provas robustas para um problema de decisão com *taxa de erro da robustez* ε (ou seja, a máquina tem uma certeza $1-\varepsilon$ de **não** aceitar erradamente entradas que não sejam *instância* deste problema de decisão), temos uma outra máquina deste mesmo modelo computacional que tem o mesmo esquema de provas robustas para este problema de decisão, mas com uma *taxa de erro de robustez* de ε^z (ou seja, uma certeza na computação de $1-\varepsilon^z$) tão pequena quanto se queira, mas constante no comprimento da entrada. Portanto, problemas de decisão de uma classe de complexidade sobre esquema de provas robustas podem ser resolvidos com taxa de erro de robustez tão pequena quanto se queira, sem com isto mudarem de *classe de complexidade*. Mais ainda, podemos *iterar* um número constante de computações com esquema de provas robustas que continuamos no esquema de provas robustas e no mesmo modelo computacional.

I.3–3. Começamos a descrever a *classe de complexidade* $\boldsymbol{\mathcal{PCP}}$ (*Probabilistically Checkable Proofs*). Para $R(n)$ e $Q(n)$, famílias de funções naturais, o *modelo computacional* da classe de complexidade $\mathcal{PCP}(R(n), Q(n))$ é de Máquinas de Turing M que têm as suas computações divididas em duas partes: a `fase de endereçamento` e a `fase de decisão`. Só consideraremos aceites válidos em computações que terminem em tempo polinomial no comprimento da entrada, em cada uma das duas fases. Devem, ainda, existir $r(n) \in \mathcal{O}(R(n))$ e $q(n) \in \mathcal{O}(Q(n))$, tais que a *fase de endereçamento* de M, utilizando-se das fitas de entrada ρ e de probabilidade τ,[xiv] lê a entrada x e no máximo $r(\|x\|)$ *bits* aleatórios da fita de probabilidade τ e grava na fita de trabalho $q(\|x\|)$ *endereços da fita de testemunha* π e possivelmente outras informações derivadas da entrada. Por sua vez, a *fase de decisão* de M, através da fita de trabalho e conhecendo em *leitura–direta* **somente** as $q(\|x\|)$ letras do alfabeto da *fita de testemunha* π nos endereços fornecidos, deverá atestar a aceitação ou não de x. Finalizando, a classe de complexidade $\boldsymbol{\mathcal{PCP}(R(n), Q(n))}$ é formada pelos problemas de decisão que têm *esquema de provas robustas* para Máquinas de Turing do modelo computacional descrito acima.

Trocando por palavras, se L é um problema de decisão da classe de complexidade $\mathcal{PCP}(R(n), Q(n))$, para cada entrada x de comprimento n, se ela é instância de L, existe uma *prova robusta* π_x (na fita de testemunha) em que a *fase de endereçamento*, computando em tempo polinomial em n e lendo da ordem de $R(n)$ *bits* aleatórios (na fita de probabilidade τ), escolhe da ordem de $Q(n)$ letras da fita de testemunha π. Já

[xi] E as fitas de entrada ρ e de trabalho.

[xii] Note que há textos em que a definição corresponde à classe co–$\mathcal{RP}$.

[xiii] Esta simulação de repetidas computações de uma máquina é conveniente quando lemos a fita de probabilidade τ, pois, em cada uma das computações, o resultado pode variar.

[xiv] Além, é claro, da fita de trabalho.

a *fase de decisão*, lendo somente tais letras da fita de testemunha e as informações contidas na fita de trabalho, deve, em tempo polinomial em n, sempre aceitar a entrada como sendo do problema de decisão L. Por outro lado, com uma entrada x que não é *instância* de L, qualquer que seja a testemunha π, que neste sentido não pode ser uma prova, ou seja, é uma *"testemunha falsa"* (fornecida na fita de testemunha π), **com uma chance menor do que ε**, (referente a fita de probabilidade τ), a fase de endereçamento deve escolher letras de π que façam com que a fase de decisão aceite a entrada, ou seja, que a fase de decisão seja **enganada**. Se $R(n)$ ou $Q(n)$ é $\{0\}$, seria como se M não utilizasse, respectivamente, as fitas de probabilidade τ ou de testemunha π.

Assim, para ilustrar, conforme os comentários anteriores, é fácil ver que

$$\begin{aligned} \mathcal{P} &= \mathcal{PCP}\big(\ 0, 0\ \big), \\ \mathcal{NP} &= \mathcal{PCP}\big(\ 0,\ \mathrm{Poli}(n)\ \big)\ \text{ e} \\ \mathcal{RP} &= \mathcal{PCP}\big(\ \mathrm{Poli}(n),\ 0\ \big). \end{aligned}$$

A primeira igualdade é imediata. Na segunda, basta lembrar que, se temos uma Máquina de Turing da classe $\mathcal{NP}$, podemos gravar o seu tempo de execução nos aceites ($t(n)$) na fita de trabalho como uma *constante da computação*, conforme feito para provar que $\mathcal{P} = \text{co–}\mathcal{P}$, no final do Parágrafo I.2–3. A *fase de endereçamento* da máquina de $\mathcal{PCP}$ fornece os $t(n)$ primeiros endereços da fita de testemunha π e, assim, a *fase de decisão* pode simular toda a computação da máquina $\mathcal{NP}$. Na verdade, este texto buscará uma igualdade bem melhor do $\mathcal{PCP}$ com a classe $\mathcal{NP}$. A última igualdade também é imediata.

Esta definição decorre de trabalhos nas áreas de Complexidade, de Codificação e de Criptografia. Na verdade, para cada problema de decisão, buscamos um conveniente *esquema de provas robustas*, que nos permite classificá-lo como pertencendo à classe $\mathcal{PCP}$. Estas provas também são chamadas de `transparentes` e `holográficas` e têm a propriedade de, ao se "mentir" um pouco, esta "mentira" deverá obrigatoriamente se espalhar, formando assim um "*gap*", de tal sorte que ao se checar a *testemunha*, com uma probabilidade grande, deve-se achar a "mentira".

A nossa definição da classe $\mathcal{PCP}$ é de `tipo não–adaptativo`, sendo assim mais fraca do que a primeira definição em Arora, Lund, Motwani, Sudan e Szegedy [ALM+92], que é de `tipo adaptativo`. Na verdade, as outras definições posteriores também são *não–adaptativas*, pois obrigam a *fase de endereçamento* a escolher todas as letras da testemunha de uma só vez, conhecendo somente a entrada e os *bits* aleatórios. Com isto, não podemos, à medida que vamos lendo as letras da testemunha, ir decidindo os novos endereços a serem consultados. Mas para nós, a menos de um dos testes, a *fase de endereçamento* será ainda mais restrita, pois, só com a informação do comprimento da entrada e com os *bits* aleatórios, ela deve decidir de uma só vez todos os endereços a serem consultados na testemunha. Esta computação será então denominada de `tipo totalmente não–adaptativo`.

I.4 Introdução ao Teorema do $\mathcal{PCP}$

Introduziremos, agora, o Teorema do $\mathcal{PCP}$ I.4–4, ou seja $\mathcal{NP} = \mathcal{PCP}(\log(n), 1)$, e partiremos à primeira parte da sua demonstração.

I.4–1. Será de fundamental importância o Teorema de Cook–Levin I.4–2, que nos diz que o *problema* 3SAT é $\mathcal{NP}$–completo. Um problema de decisão L é dito `difícil` (ou *polinomialmente difícil*) para uma classe de complexidade $\mathcal{X}$ (i.e., $L \in \boldsymbol{\mathcal{X}}$–`difícil`) sse, para todo *problema de decisão* de $\mathcal{X}$, existe uma Máquina de Turing (ou uma `redução de tempo polinomial`) que, em tempo polinomial na entrada, transforma instâncias deste problema de decisão em instâncias de L e entradas que não são instâncias deste problema de decisão, em entradas que também não são instâncias de L. Se além disto, ainda $L \in \mathcal{X}$, ele será chamado de `completo` (ou *polinomialmente completo*) para $\mathcal{X}$ (i.e., $L \in \boldsymbol{\mathcal{X}}$–`completo`).

Uma `fórmula booleana` $\varphi(\vec{\xi})$ é uma expressão aritmética com operadores `e` ($\wedge$), `ou` ($\vee$) e `não` ($\neg$), em que as $m \in \mathbb{N}_*$ variáveis $\vec{\xi} := (\xi_1, \ldots, \xi_m)$ são booleanas, isto é, correm sobre *falso* e *verdadeiro*.[xv] Um `literal` é uma das variáveis booleanas (ξ_i), ou a sua negação ($\neg\xi_i$). Uma `cláusula` é a disjunção ($\vee$) de literais. Já uma fórmula **3FNC** (em `três forma normal conjuntiva`) é a conjunção ($\wedge$) de cláusulas com exatamente três literais. Note agora que, se evitarmos redundâncias, uma fórmula 3FNC de m variáveis tem um número polinomial em m de cláusulas; mais ainda, podemos representá-la na fita binária $\Psi_{(2)}$ em uma palavra de comprimento também polinomial em m. Portanto, para uma codificação conveniente das fórmulas 3FNC, o `problema` **3SAT** é formado pelas fórmulas 3FNC *satisfazíveis*. Como é de imediata observação, 3SAT está na classe $\mathcal{NP}$, visto que a satisfação de uma valoração das m variáveis pode ser facilmente verificada em tempo polinomial em m, sendo que ela forma um *esquema de provas* para o problema.

Finalizando, diremos que a fórmula booleana φ `é satisfazível fixando-se` $\boldsymbol{x}$ sse x é uma seqüência de n elementos de $\mathbb{Z}_2$ (ou seja, uma palavra do alfabeto binário $\Psi_{(2)}$ [xvi]) e φ é satisfazível para uma valoração em que as n primeiras variáveis representam x. Veja que podemos construir em tempo polinomial em m uma outra fórmula φ_x, tal que é satisfazível sse φ é satisfazível fixando-se x. Chamaremos este passo de `incorporação de` $\boldsymbol{x}$ pela fórmula φ. Também podemos transformar em tempo polinomial em m uma fórmula booleana de m variáveis em uma outra em 3FNC com $\mathcal{O}(m)$ variáveis, mas preservando a satisfazibilidade.

Teorema de Cook–Levin I.4–2. *O problema de decisão* 3SAT *é $\mathcal{NP}$–completo.*

Prova. (Prova sucinta.) Seja uma Máquina de Turing M de $\mathcal{NP}$ que, para aceitar *instâncias do problema de decisão*, roda em tempo $t(n)-2$ que, por sua vez, é no máximo polinomial no comprimento n de cada *instância*. Estamos portanto supondo que, para tal *problema de decisão* em $\mathcal{NP}$, conhecemos tanto a máquina como o seu tempo de execução. Sem perda de generalidade, tomamos M rodando sobre apenas uma *fita de uma–via* e de alfabeto Σ. Então, codificamos M e $t(n)$ na fita de trabalho como uma *constante da computação*,[xvii] já que são fixos para a computação, e iremos mostrar uma construção, em tempo polinomial em n, que, para cada n, faz uma fórmula booleana $\varphi_{M,n}$ em 3FNC. Esta fórmula deve ser *satisfazível fixando-se* x [xviii] sse x é uma *instância do problema de decisão* definido por M, ou seja, existe uma prova π_x fazendo M aceitar x. Perceba que isto é mais forte do que uma simples *redução polinomial*. Dado x na fita de entrada ρ, apenas calculando-se o comprimento n, construímos $\varphi_{M,n}$. Podemos então, lendo x, incorporá-lo a $\varphi_{M,n}$ [xviii] em tempo polinomial em n, tal que $\varphi_{M,n}$ se torne indicativa da pertinência (ou não) de x como instância do problema de decisão, ou seja, obtemos assim a *redução polinomial*. Assim sendo, provamos o teorema.

Observe apenas que aqui não estamos muito interessados em conhecer a prova π_x, bastando-nos somente saber da sua existência. Para construir $\varphi_{M,n}$, codificamos numa tabela $t(n) \times t(n)$, toda uma possível computação de M, sendo que as linhas representarão as letras dos endereços alcançáveis na fita por M, incluindo-se aí um marcador no começo da fita, e as colunas, cada passo de M. Neste formato, cada entrada (ou posição) da tabela deve conter: a letra naquele endereço e instante da fita; um indicador da presença (ou não) da cabeça de leitura sobre esta posição, e caso afirmativo, o número do *estado* de M. O segredo agora é reparar que podemos verificar **localmente** se a tabela representa uma verdadeira computação de M ao aceitar x para alguma prova π_x, olhando-se somente:

- para a primeira linha, para assegurar que a cabeça de leitura está na primeira posição da fita e no primeiro estado da máquina e que a palavra x é dada como entrada. Veja que no começo da computação, a fita deve ter o formato $\not{\text{b}} \cdot x \cdot \not{\text{b}} \cdot \pi_x \cdot \not{\text{b}} \cdot \cdots$,[xix] em que π_x é a testemunha de x;

[xv] Durante todo o texto, iremos propositadamente confundir *falso* com o 0 e o *verdadeiro* com o 1.
[xvi] Portanto, sem o símbolo separador $\not{\text{b}}$.
[xvii] Veja a definição no Parágrafo I.2–1.
[xviii] Conforme mencionado no Parágrafo I.4–1.
[xix] Aqui, o símbolo "$\cdot$" representa a `concatenação de` *strings*.

- para todos os quadros 2×2 da tabela, vendo se as duas posições da fita estão, com respeito ao estado da máquina, em consonância com as mesmas duas posições, no passo seguinte da computação;
- se a primeira coluna é composta só de marcadores de começo de fita (portanto, não deve ter computação sobre elas);
- se não tem computação chegando à última coluna; e
- se na última linha, o estado da máquina é parado, com a cabeça de leitura sobre um símbolo separador $\not\emptyset$, representando o *aceite*.

Portanto, em cada entrada (ou posição) da tabela, definimos um número fixo de variáveis booleanas (ou seja, um `bloco de variáveis`) que devem representar a respectiva entrada da tabela. Acrescentamos a estes blocos de variáveis, mais n, que agora devem representar x. Assim, constata-se facilmente que, em tempo polinomial em n, podemos construir uma fórmula booleana $\varphi_{M,n}$ que verifica localmente as asserções acima. Ou seja, uma satisfação de $\varphi_{M,n}$ que fixa x, é uma representação das entradas (ou posições) de uma tabela, a qual simula a computação de M ao aceitar x, para alguma prova π_x. Portanto, $\varphi_{M,n}$ é satisfazível fixando-se x sse x é *instância do problema de decisão* definido por M.

Só a título de ilustração, daremos aqui apenas um exemplo das possíveis variáveis de $\varphi_{M,n}$. Para cada $i, j \in \{1 \dots t(n)\}$ e $l \in \{1 \dots n\}$, declare os blocos de variáveis:

$\vec{x}_l$ – representando a letra em Σ na l-ésima posição da entrada x;

$\vec{v}_{i,j}$ – representando a letra em Σ no tempo j e no endereço i da fita; e

$\vec{s}_{i,j}$ – indicando a veracidade (ou não) de M estar no tempo j e com a cabeça de leitura no endereço i da fita. Caso seja verdade, fornece também o número do *estado* da máquina M neste tempo j.

Deste modo, temos os blocos de variáveis dados por:

$$\begin{aligned} \vec{x}_l &:= (x_{l,1}, \dots, x_{l,k}), \\ \vec{v}_{i,j} &:= (v_{i,j,1}, \dots, v_{i,j,k}) \text{ e} \\ \vec{s}_{i,j} &:= (s_{i,j,1}, \dots, s_{i,j,e}), \end{aligned}$$

sendo que $k := \lceil \log_2(\#\Sigma) \rceil$ e e é o teto do logaritmo na base dois do número de *estados* da máquina M (incluindo-se aí um estado de parada e ainda um indicador da não presença da cabeça de leitura). Agora, em tempo também polinomial em n, podemos transformar $\varphi_{M,n}$ de modo a ter formato 3FNC, possivelmente acrescendo-se algumas variáveis, mas preservando a *satisfazibilidade*. Temos, assim, a tabela bem "representada", como queríamos.

[I.4–2] ■

Esta mesma idéia será utilizada duas outras vezes. A primeira é agora no lema que demonstrará a primeira parte do Teorema do $\mathcal{PCP}$ I.4–4. Outros limitantes superiores da classe $\mathcal{PCP}(R(n), Q(n))$ também podem ser achados em Goldreich e Håstad [GH96].

Lema I.4–3. *Para famílias de funções naturais $R(n)$ e $Q(n)$, temos*

$$\mathcal{PCP}\big(\ R(n), Q(n)\ \big) \quad \subseteq \quad \mathcal{NTIME}\Big(\ 2^{\mathcal{O}(R(n))} \cdot \mathrm{Poli}(n)\ \Big).$$

Prova. Evidentemente, esta demonstração tem como base a do Teorema de Cook–Levin I.4–2. Seja M uma Máquina de Turing de $\mathcal{PCP}(R(n), Q(n))$, que roda lendo $r(n) \in \mathcal{O}(R(n))$ *bits* da fita de probabilidade τ, para cada *instância* de comprimento n. Tome $Z(n) := \mathcal{O}(\, 2^{\mathcal{O}(R(n))} \cdot \mathrm{Poli}(n)\,)$. Então, para cada n, construiremos com tempo em $Z(n)$ uma fórmula booleana $\varphi_{M,n}$ com variáveis em número também em $Z(n)$, de tal modo que uma entrada x de comprimento n é *instância do problema de decisão* definido por M sse $\varphi_{M,n}$ é *satisfazível fixando-se* x. Portanto, a construção e o teste da satisfazibilidade de $\varphi_{M,n}$ é sem dúvida um *problema de decisão não–determinístico* de solução em tempo em $Z(n)$, o que demonstra o teorema.

Levando-se em conta somente a parte significativa, temos $2^{r(n)}$ possíveis fitas de probabilidade τ distintas. Descreveremos um algoritmo que se repete a cada uma destas fitas e tem o intuito, passo-a-passo, de ir:

- montando uma tabela em que constem todos os endereços da fita de testemunha π que por ventura sejam possíveis de serem fornecidos pela fase de endereçamento de M à sua fase de decisão. Cada um deles deve apontar para um bloco de variáveis que representarão uma letra da fita de testemunha π; e

- construindo a fórmula booleana $\varphi_{M,n}$.

Então, seja a Máquina de Turing que reproduza cada uma destas $2^{r(n)}$ possíveis fitas de probabilidade τ e, para cada uma delas:

- simule a fase de endereçamento de M e, para cada endereço da fita de testemunha π fornecido:

  - veja se o endereço pertence à tabela que está sendo montada. Caso **não** pertença a ela, declare um novo bloco de variáveis (que possam representar uma letra de fita de testemunha π) e abra uma nova entrada (ou posição) na tabela com este endereço, apontando-o a este novo bloco de variáveis, e

  - selecione, ordenadamente, o bloco de variáveis apontado; e

- complete a construção da fórmula booleana $\varphi_{M,n}$, representando nela a computação da fase de decisão de M, conforme o Teorema de Cook–Levin I.4–2. Aqui, todas as leituras na fita de testemunha π serão tomadas sobre os blocos de variáveis selecionados no item anterior (ainda referente a esta mesma fita de probabilidade τ).

Deve-se também declarar n blocos de variáveis de modo a representar uma entrada de comprimento n na fita de entrada ρ. Então, no último item, também devemos tomar as leituras da fita de entrada ρ sobre estes blocos de variáveis.

Note que as fases de endereçamento e de decisão de M têm tempo polinomial em n e, assim sendo, a tabela tem o seu tamanho limitado em $Z(n)$. Portanto, a tarefa mais custosa desta Máquina de Turing será procurar o endereço na tabela. Como repetimos isto $2^{r(n)} \cdot \mathrm{Poli}(n)$ vezes, o tempo total deve ser limitado em $Z(n)$. Não é difícil perceber que o número de variáveis de $\varphi_{M,n}$ é da ordem do tempo gasto para a sua construção. A demonstração termina ao verificarmos que $\varphi_{M,n}$ é *satisfazível fixando-se* x sse existe uma fita de testemunha π, tal que M aceita x com probabilidade "um" (sobre as fitas de probabilidade τ). Temos assim $\varphi_{M,n}$ com a propriedade que desejávamos.

[I.4–3] ■

Note que $2^{\mathcal{O}(\log(n))} \subseteq \text{Poli}(n)$ e disto tiramos, direto do lema acima, que $\mathcal{NP} = \mathcal{NTIME}\big(\, 2^{\mathcal{O}(\log(n))}\text{Poli}(n) \,\big) \supseteq \mathcal{PCP}\big(\, \log(n), \text{Poli}(n) \,\big) \supseteq \mathcal{PCP}\big(\, \log(n), 1 \,\big)$. Concluímos assim a parte fácil do

Teorema do $\mathcal{PCP}$ I.4–4 ([ALM+92])**.**

$$\mathcal{NP} \;=\; \mathcal{PCP}\big(\, \log(n), 1 \,\big).$$

A segunda parte deste teorema será provada na Seção IV.4 *Composição de Testes sobre a Classe de Complexidade $\mathcal{PCP}$* e se deve a Arora, Lund, Motwani, Sudan e Szegedy [ALM+92]. Na verdade, ela será uma decorrência de tudo mais que veremos no texto, mas antes, apenas a título ilustrativo, trataremos da primeira grande conseqüência deste teorema.

I.5 Inexistência de Esquema de Aproximação de Tempo Polinomial para a Classe $\mathcal{MAX}-\mathcal{SNP}$

Houve um grande "*boom*" na *Área de Otimização Combinatória* tendo como base o Teorema do $\mathcal{PCP}$ I.4–4, surgindo, dia após dia, mais e mais resultados. Como estes resultados são conseqüência de um teorema que é da *Área de Complexidade Computacional*, eles não poderiam deixar de ser de caráter "destrutivo de possibilidades", ou seja, indicando a *inexistência computacional* de máquinas para resolver certas tarefas. Através do Teorema do $\mathcal{PCP}$ I.4–4, sabemos hoje de diversos fatores de **inaproximabilidade** em tempo polinomial sobre os mais importantes problemas da Otimização Combinatória. Mas, talvez o mais abrangente seja ainda o resultado já fornecido no artigo de Arora, Lund, Motwani, Sudan e Szegedy [ALM+92], que descreveremos nesta seção.

A Otimização Combinatória representa, há muito, uma importante área da Teoria da Computação, entretanto, a sua inserção na Área de Complexidade Computacional é mais recente e se deve primeiramente a Papadimitriou e Yannakakis em [PY88]. Este foi um passo decisivo, visto que, por outro lado e na mesma época, a Teoria da Complexidade Computacional começou a ir em direção às *provas interativas*, que mesclam propriedades algébricas com probabilísticas. Nesta junção, estava criado um caminho fértil concomitantemente às duas áreas e buscamos reproduzir o seu ápice neste texto. Quanto aos demais resultados obtidos em Otimização Combinatória, há diversas referências importantes, mas salientamos Babai em [Bab94].

Agora começaremos dando uma noção de uma aproximação em tempo polinomial, para depois fornecermos um exemplo de uma classe de complexidade em otimização combinatória, sobre a qual verificaremos a *inexistência de esquema de aproximação de tempo polinomial.*

I.5–1. Para dois alfabetos Σ e Φ, seja um *problema de decisão* $L \subseteq \Sigma_*^* := (\Sigma \setminus \{\not{b}\})^*$ [xx] e uma função

$$\boldsymbol{W} \;:\; L \times \Phi^* \;\;\to\;\; \mathbb{R}_+,$$

tal que, para toda palavra $x \in L$, a função restrita à segunda coordenada

$$W_x \;:\; \Phi^* \;\;\to\;\; \mathbb{R}_+$$

é *limitada.* Deste modo, W será chamada de `função peso` e sobre ela definimos o `problema de otimização`,

[xx] Lembre-se que Σ_* é o alfabeto Σ sem o símbolo separador $\not{b}$ e Σ_*^*, o seu *vocabulário*, conforme definido no Parágrafo I.2–1.

que é associado a uma das suas duas `funções valor ótimo`,

$$\begin{array}{rccc} \textbf{MAX–}\boldsymbol{W}: & L & \to & \mathbb{R}_+ \\ & x & \mapsto & \sup\limits_{\pi\in\Phi^*}(W_x(\pi)) \end{array}$$

e

$$\begin{array}{rccc} \textbf{MIN–}\boldsymbol{W}: & L & \to & \mathbb{R}_+ \\ & x & \mapsto & \inf\limits_{\pi\in\Phi^*}(W_x(\pi))\,. \end{array}$$

Vejamos agora um exemplo para a função valor ótimo máximo.

Exemplo do Problema de Otimização MAX–3SAT E–A. *Seja o problema de decisão* $3\text{FNC}\subseteq\Sigma_*^*$ [xxi] codificado *de alguma maneira interessante sobre um alfabeto* Σ*, de tal modo que toda fórmula booleana* $\varphi\in 3\text{FNC}$ *tenha comprimento polinomial no seu número de variáveis. Assim, para uma fórmula booleana* $\varphi\in 3\text{FNC}$ *de* n *variáveis, definimos a* função peso $\mathbf{3SAT_\varphi(\pi)}$ *pelo número de cláusulas de* φ *que são satisfeitas pela valoração* $\pi\in\Psi_{(2)_*}^n$.[xxii] *Entretanto, no caso de* π *não pertencer a* $\Psi_{(2)_*}^n$*, então* $3\text{SAT}_\varphi(\pi)$ *deve ser* zero. *Por convenção, as cláusulas repetidas também serão contadas na sua multiplicidade. Concluímos assim que a* função valor ótimo $\mathbf{MAX–3SAT(\varphi)}$ *designa o número máximo de cláusulas de* φ *que podem ser satisfeitas simultaneamente. Portanto, se* φ *é satisfazível,* $\text{MAX–3SAT}(\varphi)$ *é igual ao número de cláusulas que* φ *tem.*

I.5–2. Voltemos às Máquinas de Turing, mas agora deixando de lado as computações sobre *problemas de decisão*. Para isto, precisamos definir a:

- `fita de saída` ($\boldsymbol{\theta}$) como sendo de *uma-via* e de *só–escrita*,[xxiii] a qual, no início da computação, deve conter somente símbolos separadores $\not\emptyset$.

Então, consideraremos como `saída` da computação o *string* que, ao final da computação, estiver à esquerda da cabeça de leitura da *fita de saída* θ. Pode, assim, uma Máquina de Turing M com esta fita computar mais do que uma simples resposta **sim** ou **não**. Tal resposta será representada por $\boldsymbol{M(x)}$, sendo que x é a entrada. Veja que, apesar de não termos entrado nos detalhes, já utilizamos este tipo de computação para falarmos das *reduções*, sobre as quais definimos os *problemas de decisão difíceis e completos* para uma classe de complexidade.[xxiv] Se esta máquina M está associada a um problema de otimização definido sobre uma função peso W, então as entradas x, tais que $W(M(x))>0$, serão chamadas de `solução`.

Continuando, sejam $\alpha\in[0,1]$ e a *função peso* W definida no Parágrafo I.5–1. Diremos que o seu *problema de otimização* associado tem uma $\boldsymbol{\alpha}$–`aproximação de tempo polinomial` sse existe uma Máquina de Turing M com uma fita de entrada ρ de alfabeto Σ e outra de saída θ de alfabeto Φ [xxv] e de tempo polinomial no comprimento da entrada, tal que, para toda entrada $x\in\Sigma_*^*$, temos, conforme for o caso,

$$W_x(\,M(x)\,)\;\;\geq\;\;(1-\alpha)\;\text{MAX–}W(x),$$

ou

$$W_x(\,M(x)\,)\;\;\leq\;\;(1+\alpha)\;\text{MIN–}W(x).$$

[xxi] Note que 3FNC são as fórmulas booleanas em três forma normal conjuntiva e que foram definidas no Parágrafo I.4–1.
[xxii] Veja aqui que, como o alfabeto binário $\Psi_{(2)_*}:=\Psi_{(2)}\smallsetminus\{\not\emptyset\}$ não tem o símbolo separador, a valoração π é exatamente uma seqüência de *bits* em $\mathbb{Z}_2^n$.
[xxiii] Vide definições no Parágrafo I.2–1.
[xxiv] Vide o Parágrafo I.4–1.
[xxv] Além, é claro, da *fita de trabalho*.

Ou seja, M calcula em tempo polinomial uma solução para x com perda de no máximo uma fração α do *valor ótimo* MAX–$W(x)$ ou MIN–$W(x)$. Observe que um problema de otimização tem uma *0–aproximação de tempo polinomial* sse a solução de valor ótimo puder ser computada em tempo polinomial. Como sabemos, nem sempre este é o caso.

Exemplo de uma $\frac{1}{2}$–Aproximação de Tempo Polinomial para o MAX–3SAT E–B. *Vejamos que* MAX–3SAT *tem uma* $\frac{1}{2}$–aproximação de tempo polinomial. *É imediato; seja uma Máquina de Turing que, a cada variável de uma fórmula booleana φ dada como entrada, vê o número de cláusulas de φ que têm um literal formado exatamente por esta variável e, também, pela sua negação. Portanto, ela toma a valoração desta variável como sendo a que satisfaz o maior número de cláusulas. Posteriormente, despreza todas as cláusulas em que esta variável aparece e repete o processo. Perceba que isto pode ser feito em tempo polinomial na entrada.*

I.5–3. A *Teoria da Complexidade Computacional* tem como princípio básico buscar a prova de $\boldsymbol{\mathcal{P} \neq \mathcal{NP}}$. Entretanto, e bem mais forte do que isto, na prática, se este não for o caso, a grande maioria dos resultados da teoria, ou deixam de valer, ou perdem o seu "sentido". Um exemplo clássico em que quase todo o "sentido" depende desta diferença está nesta *Área de Complexidade Computacional em Otimização Combinatória*. Vejamos, então, um dos grandes resultados do Teorema do $\mathcal{PCP}$ I.4–4. Pelo comentário acima, ele não perde absolutamente em importância pela sua hipótese.

Lema da Inaproximabilidade para o MAX–3SAT I.5–4 ([ALM$^+$92])**.** *Existe uma constante $\alpha \in (0, 1/2)$, tal que o problema de otimização* MAX–3SAT *não tem uma α–aproximação de tempo polinomial, a menos que $\mathcal{P} = \mathcal{NP}$.*

Prova. Seja um problema de decisão $L \in \mathcal{NP}$. Pelo Teorema do $\mathcal{PCP}$ I.4–4, temos $\mathcal{NP} = \mathcal{PCP}(\log(n), 1)$ e existe uma Máquina de Turing M de $\mathcal{PCP}$ com *taxa de erro da robustez*[xxvi] $\varepsilon \in (0, 1)$ e que lê $r(n) \in \mathcal{O}(\log(n))$ *bits* na fita de probabilidade τ e $q \in \mathcal{O}(1)$ letras na fita de testemunha π, para decidir se uma entrada de comprimento n é ou não instância de L.

Fixe uma entrada x para M de comprimento n. Tome um bloco de variáveis $\vec{\xi}_i$ para representar cada letra efetivamente útil da fita de testemunha π. Veja que, ao fixarmos cada uma das $2^{r(n)} \in \text{Poli}(n)$ fitas de probabilidade τ significativas, temos que a fase de decisão de M passa a ser determinística sobre os q blocos de variáveis $\vec{\xi}_{i_1}, \cdots, \vec{\xi}_{i_q}$ escolhidos pela fase de endereçamento de M. Ora, neste caso, a fase de decisão de M define uma função booleana $F_{x,\tau}$ sobre estes blocos de variáveis.[xxvii] Logo, podemos representá-la por uma fórmula booleana $\varphi_{x,\tau} \in$ 3FNC, acrescendo-se, possivelmente, algumas variáveis, mas sem com isto alterar a sua *satisfazibilidade*. Queremos agora supor que, independentemente da fita de probabilidade τ, $\varphi_{x,\tau}$ tenha sempre o mesmo número de cláusulas. Podemos fazer isto, pois o número de variáveis de $\varphi_{x,\tau}$ é constante para n, tendo assim o número de cláusulas limitado por um z, também constante. Para as fórmulas $\varphi_{x,\tau}$ que tenham menos do que z cláusulas, completa-se com cláusulas novas sobre variáveis também novas. Assim, como estas cláusulas são sempre satisfazíveis, não alteram a *satisfazibilidade* de $\varphi_{x,\tau}$. Seja, então, a conjunção

$$\varphi_x \quad := \quad \bigwedge_{\tau} \varphi_{x,\tau}$$

e suponha φ_x com $c(n) \in \text{Poli}(n)$ cláusulas. Veja que tudo isto pode ser feito em tempo polinomial em n.

Perceba agora que, se $x \in L$, então

$$\mathop{\mathcal{P}r}_{\pi}\Big\{ \mathop{\mathcal{P}r}_{\tau}\{\ M \text{ aceita } x\ \} = 1 \Big\} > 0.$$

[xxvi] Vide o Parágrafo I.3–3.

[xxvii] Isto é, $F_{x,\tau}$ é uma *função booleana* que tem como variáveis estes blocos de variáveis. Cada uma delas deve correr sobre $\mathbb{Z}_2$ e o contra–domínio também deve ser $\mathbb{Z}_2$.

Ou seja, existe uma fita de testemunha π, tal que, para toda fita de probabilidade τ, π "satisfaz" $\varphi_{x,\tau}$. Logo, φ_x é satisfazível. Por outro lado, se $x \notin L$, então

$$\Pr_{\pi}\left\{ \Pr_{\tau}\{ M \text{ aceita } x \} < \varepsilon \right\} = 1$$

e, qualquer que seja a valoração que se "dê" à fita de testemunha π (e incluídas aí as demais variáveis), mais do que uma fração $1-\varepsilon$ das fitas de probabilidade τ terão as suas fórmulas $\varphi_{x,\tau}$ não satisfeitas. Ou seja, mais do que uma fração $(1-\varepsilon)/z$ das cláusulas de φ_x **não** podem ser satisfeitas, independentemente da valoração.

Como só um destes dois casos pode acontecer, construímos um *gap* de fração $(1-\varepsilon)/z$ sobre as cláusulas que podem ser satisfeitas, isto é, um "*gap*" nos possíveis *valores ótimos*. Terminamos a demonstração ao vermos que se sabemos satisfazer cláusulas em tempo polinomial na entrada, a menos de uma fração $(1-\varepsilon)/z$ do *valor ótimo*, então podemos decidir se x está ou não em L (em tempo polinomial). Portanto, $L \in \mathcal{P}$ e $\mathcal{P} = \mathcal{NP}$.

[I.5–4] ■

I.5–5. Veja que as classes de complexidade na Área de Otimização Combinatória não são mais definidas sobre problemas de decisão e sim sobre problemas de otimização referentes às funções peso. Estamos, agora, não só interessados no tempo de execução das Máquinas de Turing, como também na sua capacidade em aproximar os *valores ótimos*, e, por isto, passaremos a chamá-las de `classes de aproximabilidade`. Fagin em [Fag74] fez uma brilhante caracterização das clássicas classes de complexidade através da descrição sintática e/ou lingüística sobre a lógica moderna de segunda ordem. Em particular, ele redefiniu a classe $\mathcal{NP}$ e Kolaitis e Vardi [KV87] valeram-se dela para fazer uma versão restrita, denotada por $\mathcal{SNP}$.[xxviii] Esta caracterização sintática permitiu a Papadimitriou e Yannakakis [PY88] definirem as `classes de aproximabilidade` $\boldsymbol{\mathcal{MAX}-\mathcal{NP}}$ e $\boldsymbol{\mathcal{MAX}-\mathcal{SNP}}$. Além disto, [PY88] trouxeram uma definição de `redução em aproximação` de um problema em otimização para outro. Esta redução preserva linearmente as *aproximações* em tempo polinomial. Com ela, podemos recuperar a noção usual de *problemas de decisão difíceis e completos*, agora para as classes de aproximabilidade.[xxix] Mais recentemente, tornou-se comum redefinir as classes de aproximabilidade $\mathcal{MAX}-\mathcal{NP}$ e $\mathcal{MAX}-\mathcal{SNP}$, fechando-as por esta *redução em aproximação*. Por sinal, com este fecho, acabaram entrando nestas classes os problemas de *minimização* correspondentes. Não é o nosso objetivo entrar nos detalhes quanto a estas classes de aproximabilidade, entretanto, para ilustrar melhor, veja que $\mathcal{MAX}-\mathcal{SNP} \subseteq \mathcal{MAX}-\mathcal{NP}$ e que os problemas MAX–SAT, MAX–INDEP–SET e MIN–COBERT são de $\mathcal{MAX}-\mathcal{NP}$ e $\mathcal{MAX}-\mathcal{SNP}$–difíceis. Por sua vez, MAX–CUT, MAX–kSAT (para $k \in \mathbb{N}$ e $k \geq 2$ fixo), MAX–INDEP–SET–b e MIN–COBERT–b (para $b \in \mathbb{N}_*$ fixo, sendo o grau máximo) são $\mathcal{MAX}-\mathcal{SNP}$–completos.[xxx]

Diremos que uma classe de aproximabilidade tem `esquema de aproximação de tempo polinomial` sse todos os seus problemas de otimização têm uma α–aproximação de tempo polinomial, qualquer que seja o $\alpha \in (0,1)$ tomado. Na verdade, perceba que se uma classe de aproximabilidade **não** tem *esquema de aproximação de tempo polinomial*, então nenhum problema de otimização difícil[xxxi] para esta classe pode ter uma α–aproximação de tempo polinomial, para todos os α's em $(0,1)$. Como já foi dito, MAX–3SAT $\in \mathcal{MAX}-\mathcal{SNP}$ e, assim, temos

Teorema da Inaproximabilidade em $\boldsymbol{\mathcal{MAX}-\mathcal{SNP}}$ I.5–6. *A classe de aproximabilidade $\mathcal{MAX}-\mathcal{SNP}$ (e obviamente $\mathcal{MAX}-\mathcal{NP}$ também) não tem esquema de aproximação de tempo polinomial, a menos que $\mathcal{P} = \mathcal{NP}$.*

■

[xxviii] O $\mathcal{S}$ da classe $\mathcal{SNP}$ advém de uma redefinição "*strict*" da classe $\mathcal{NP}$.

[xxix] Veja maiores detalhes no Parágrafo I.4–1.

[xxx] Não forneceremos aqui as definições destes problemas de otimização, entretanto, elas são evidentes e constam de diversos textos da área. Vide por exemplo [Pap94].

[xxxi] O problema de otimização deve ser *difícil* para a redução em aproximação tratada acima.

I.5–7. Portanto, para todo os importantes problemas de otimização da classe de aproximabilidade $\mathcal{MAX}$–$\mathcal{SNP}$–difícil (como os de $\mathcal{MAX}$–$\mathcal{NP}$–difícil) **não** conseguimos computação em tempo polinomial que forneça solução tão próxima quanto desejada da solução ótima. Ou seja, todos os problemas de $\mathcal{MAX}$–$\mathcal{SNP}$–difíceis têm um *fator de inaproximabilidade* que independe do comprimento da entrada, sendo que, a partir deste, não podemos achar, de modo "barato", soluções com valores mais próximos do que este fator do valor ótimo.

Outros autores também procuraram introduzir classes de aproximabilidade, além, é claro, de Papadimitriou e Yannakakis em [PY88]. Por exemplo, Ausiello, Crescenzi e Protasi [ACP94] buscaram definições mais "computacionais" e criaram as classes de aproximabilidade $\boldsymbol{\mathcal{PTAS}}$ e $\boldsymbol{\mathcal{APX}}$. A primeira é a classe de aproximabilidade maximal com *esquema de aproximação de tempo polinomial* e a segunda é formada pelos problemas de otimização que têm uma α–aproximação de tempo polinomial, para algum $\alpha \in (0,1)$. Neste sentido, temos os seguintes resultados:

$$\begin{aligned} \mathcal{PTAS} &\subseteq \mathcal{APX}, \\ \mathcal{MAX}\text{–}\mathcal{SNP} &\subseteq \mathcal{APX}, \\ \mathcal{MAX}\text{–}\mathcal{NP} \supseteq \mathcal{MAX}\text{–}\mathcal{SNP} &\nsubseteq \mathcal{PTAS} \text{ e} \\ \mathcal{PTAS} \cap \mathcal{MAX}\text{–}\mathcal{SNP}\text{–difícil} &= \varnothing \end{aligned}$$

Para se obter mais detalhes da inter–relação entre estas classe de aproximabilidade, veja Khanna, Motwani, Sudan e Vazirani em [KMSV95].

Perceba que existem muitos outros grandiosos resultados em Otimização Combinatória decorrentes do Teorema do $\mathcal{PCP}$ I.4–4, mas nos concentramos no mais simples e, talvez, importante.

Capítulo II

Codificações Algébricas

Este capítulo será como um guia ao que faremos a seguir, principalmente quanto aos *testes probabilísticos*. Tendo como fundo Sudan [Sud95], baseearemos toda a nossa demonstração do Teorema do $\mathcal{PCP}$ I.4–4 nos *testes sobre códigos*. Todavia, alertamos para o fato que a Teoria dos Códigos é deveras mais abrangente do que poderíamos pretender apresentar no texto. Antes de entrarmos na especificação dos *códigos*, talvez seja aconselhável darmos uma visão mais ampla da sua utilização através da seção

II.1 Estrutura Global da Demonstração do Teorema do $\mathcal{PCP}$

Antes de anunciarmos o Teorema do $\mathcal{PCP}$ I.4–4 no Capítulo I *Esquema de Provas Robustas*, já havíamos visto que

$$\mathcal{NP} \quad \supseteq \quad \mathcal{PCP}\big(\ \log(n), 1\ \big)$$

saía como corolário do Lema I.4–3. Portanto, todo o nosso trabalho daqui para frente no texto será demonstrar a outra inclusão, qual seja,

$$\mathcal{NP} \quad \subseteq \quad \mathcal{PCP}\big(\ \log(n), 1\ \big).$$

Buscaremos agora exatamente "mapear" esta demonstração, ou seja, dar uma visão geral dela. Basicamente a prova se divide em três itens distintos, descritos a seguir, e tem os seus objetivos alcançados nas seções do Capítulo IV *Classe de Complexidade* $\mathcal{PCP}$, conforme mencionaremos mais abaixo:

A – a prova do Corolário do $\mathcal{PCP}$ para Leitura de $\mathcal{O}(1)$ Letras da Testemunha IV.2–4, i.e.

$$\mathcal{NP} \quad \subseteq \quad \mathcal{PCP}\big(\ \mathrm{Poli}(n),\ 1\ \big);$$

B – a prova do Corolário do $\mathcal{PCP}$ para Leitura de $\mathcal{O}(\log(n))$ *Bits* Aleatórios IV.3–4, i.e.

$$\mathcal{NP} \quad \subseteq \quad \mathcal{PCP}\big(\ \log(n),\ \log\log(n)\ \big);\quad \text{e}$$

C – a prova e a utilização do Teorema da Composição de Testes IV.4–2 sobre os Itens A e B.

Na verdade, nos Itens A e B necessitaremos provar resultados mais específicos do que os referidos corolários, que são o Teorema do Teste com Leitura Constante de *Bits* na Pseudo–Fita de Testemunha IV.2–2 e o Teorema do Teste com Leitura Logarítmica de *Bits* Aleatórios IV.3–2. Não obstante, tais corolários são mais ilustrativos e fornecem uma melhor noção de grandeza com relação aos teoremas.

No Item C, descrito na Seção IV.4 *Composição de Testes sobre a Classe de Complexidade* $\mathcal{PCP}$, primeiramente provaremos o Teorema da Composição de Testes IV.4–2 e logo em seguida, na Página 62, aplicando-o por duas vezes, demonstraremos o Teorema do $\mathcal{PCP}$ I.4–4. Veja que, neste ponto, tal demonstração já se tornará muito simples. O Teorema da Composição de Testes IV.4–2 nos informa que, se por hipótese sabemos da pertinência de uma conveniente reestruturação do problema de decisão 3SAT a uma classe de complexidade $\mathcal{PCP}$, temos como tese a inclusão de uma outra classe $\mathcal{PCP}$ nesta primeira (tendo ela apenas uma pequena expansão técnica). Não entraremos nos detalhes sobre o Teorema da Composição de Testes IV.4–2, mas é de fácil observação que ele só é satisfatório quando nos permitir incluir uma classe $\mathcal{PCP}$ "maior" em uma outra "menor". Pois bem, os Itens A e B fornecem exatamente as hipóteses necessárias para a utilização, por duas vezes, do Teorema da Composição de Testes IV.4–2 e obtemos assim um resultado um pouco mais restrito do que

$$\mathcal{PCP}\big(\ \log(n),\ \log\log(n)\ \big) \quad \subseteq \quad \mathcal{PCP}\big(\ \log(n), 1\ \big).$$

Mas veja que, aplicando-se mais uma vez o Item B, temos o que queríamos, ou seja,

$$\mathcal{NP} \quad \subseteq \quad \mathcal{PCP}\big(\ \log(n), 1\ \big).$$

Em todo teste podemos separar a última parte da computação, em que o teste efetivamente e deterministicamente decide a aceitação, ou não, da entrada. Computação esta que será chamada de *fase de decisão* e o seu tempo, de *tempo de decisão*.[i] O Teorema da Composição de Testes IV.4–2 é bastante engenhoso e, partindo da idéia do Teorema de Cook–Levin I.4–2, simula a fase de decisão da computação de um teste $\mathcal{PCP}$ "dispendioso" com a computação completa de um outro teste. Isto pode ser mais "econômico" se a fase de decisão do primeiro teste também o for e consegue-se descer em magnitude o limite no número de leituras de letras na testemunha, o tamanho dos blocos de leitura na testemunha e, por fim, o tempo de decisão. Deve-se, para isto, acoplar convenientemente um teste que resolva o problema de decisão 3SAT configurado de acordo com as hipóteses do Teorema da Composição de Testes IV.4–2.

Continuando, a despeito da Seção IV.4 *Composição de Testes sobre a Classe de Complexidade* $\mathcal{PCP}$ supra citada, as demais seções do Capítulo II *Codificações Algébricas*, do Capítulo III *Testes Probabilísticos* e do Capítulo IV *Classe de Complexidade* $\mathcal{PCP}$ provarão paralelamente os Itens A e B. Os dois itens se alicerçam sobre os testes probabilísticos do Capítulo III. Estes testes tentam, em um primeiro passo, verificar se uma determinada testemunha (ou um pedaço dela) é uma codificação predeterminada, ou se pelo menos a testemunha está muito próxima a esta codificação. Em um segundo passo, os testes procuram assegurar-se se reconhecemos tal codificação como uma *prova satisfatória* para a entrada do problema. A diferenciação mais precisa entre estes dois passos estará na Seção III.1 *Introdução aos Testes* e o conteúdo de tais testes serão descritos, respectivamente, na Seção III.2 *Testes de Proximidade da Codificação* e na Seção III.3 *Testes de Reconhecimento da Codificação*, tanto com relação ao Item A, como com o Item B. Também pode-se notar facilmente que tais testes são baseados sobre as codificações que introduziremos neste Capítulo II *Codificações Algébricas*.

Dados os testes, seguiremos para o Capítulo IV *Classe de Complexidade* $\mathcal{PCP}$. Primeiramente, na Seção IV.1 *Variantes da Classe de Complexidade* $\mathcal{PCP}$ será necessário estender a noção da classe $\mathcal{PCP}$ de modo a adaptá-la ao Teorema da Composição de Testes IV.4–2. Basicamente precisamos trabalhar sobre a testemunha com blocos de letras (variando estes de tamanho conforme o comprimento da entrada) e também ter um limitante para o tempo de decisão. Por fim, em uma próxima meta, também precisamos saber resolver o

[i] Veja mais detalhes no Parágrafo III.1–2.

problema de decisão 3SAT fixando-se previamente e convenientemente a satisfação das primeiras variáveis. Finalmente na Seção IV.2 *Aritmetização de Fórmulas Booleanas para Leitura Constante de Letras na Testemunha* e na Seção IV.3 *Aritmetização de Fórmulas Booleanas para Leitura Logarítmica de* Bits *Aleatórios* concluímos, respectivamente, os Itens A e B.

Veremos primeiro o Item A, em que precisamos basicamente obter um teste para o problema de decisão 3SAT com leitura constante no número de letras da testemunha. Este item encontra-se sintetizado na Seção IV.3 *Aritmetização de Fórmulas Booleanas para Leitura Logarítmica de* Bits *Aleatórios* e é totalmente baseado em funcionais lineares e produtos tensoriais. Então, sobre a Codificação por Funcional Linear C–I, conseguimos tal objetivo fazendo:

- No teste de proximidade da codificação:
  - Repete-se o Teste de Linearidade T–V um número constante de vezes. Ele é um caso particular do Teste de Linearidade T–IV, que, por sua vez, tem a sua corretude como conseqüência do Teorema de Linearidade III.2–1.
- No teste de reconhecimento da codificação:
  - Utiliza-se o Teste do Produto Tensorial T–X por duas vezes.
  - E uma vez o Teste da Nulidade de Funcionais Lineares T–I.

Por sua vez, o Item B é totalmente baseado em polinômios e nas suas nulidades em um domínio restrito. Grosso modo, buscamos nele um teste para o problema de decisão 3SAT com leitura logarítmica de *bits* aleatórios. A sua descrição sucinta se encontra na Seção IV.2 *Aritmetização de Fórmulas Booleanas para Leitura Constante de Letras na Testemunha* e é baseada na Codificação Polinomial com Grau nas Variáveis Baixo C–IV e na Codificação Polinomial com Grau Total Baixo C–VIII, que são, por sua vez, generalizações respectivamente da Codificação Polinomial com Grau nas Variáveis Baixo C–III ⟨Segunda versão⟩ (e Codificação Polinomial com Grau nas Variáveis Baixo C–II ⟨Primeira versão⟩) e da Codificação Polinomial com Grau Total Baixo C–V. Os seus passos são compostos por:

- No teste de proximidade da codificação:
  - Repete-se o Teste Polinomial de Grau Total Baixo T–IX um número constante de vezes. Este teste é extraído do Teste Polinomial de Grau Total Baixo T–VIII, entretanto a demonstração da sua corretude é o tópico mais técnico do texto. Por isto, deixá-la-emos em separado, ocupando todo o Capítulo V *Polinômios de Grau Baixo*, e será uma decorrência imediata do Teorema da Proximidade a um Polinômio de Grau Total Baixo V.1–5.
- No teste de reconhecimento da codificação:
  - Também repete-se o Teste da Nulidade de Polinômios de Grau nas Variáveis Baixo sobre Domínio Restrito T–XIV um número constante de vezes. Ele é um caso particular do Teste da Nulidade de Polinômios de Grau nas Variáveis Baixo sobre Domínio Restrito T–XIII, que, por sua vez, é o resultado do Teste da Soma sobre Polinômios de Uma Variável e de Grau Baixo T–XI e do Teste da Soma sobre Polinômios de Grau nas Variáveis Baixo T–XII.

Terminamos assim o nosso resumo da demonstração do Teorema do $\mathcal{PCP}$ I.4–4, esperando que ele sirva como acompanhante pelo restante do texto.

II.2 Introdução às Codificações

A idéia da *codificação* vem de problemas na *comunicação de dados (a distância)*, dentro da Teoria da Computação. Como se sabe, toda comunicação, com maior ou menor intensidade, está sujeita a "ruídos". Um exemplo simples de codificação são os *dígitos de controle*. O nosso CPF tem os dois últimos algarismos como dígitos de controle. Eles são tomados como uma função dos números precedentes que, por sinal, formam o "verdadeiro" número do CPF. Portanto, pode-se checar, com alta probabilidade, se um digitador entrou no computador com o número do CPF certo. Caso este não seja o caso, pede-se ao digitador repetir a operação.

A *Teoria dos Códigos* começa a tomar corpo quando somos mais exigentes. Em comunicações de dados a distância, ruídos de transmissão são freqüentes. Então, queremos evitar de, sempre que detectamos um problema, sermos obrigados a repetir a operação. Ou seja, precisamos corrigir, com alta probabilidade e sem muito "esforço", a informação fornecida com ruídos. Para isto, a idéia é sempre espalhar (i.e., *codificar*) o universo das palavras possíveis de serem comunicadas em uma estrutura (algébrica) conveniente.

Já no nosso caso, utilizamos, a grosso modo, as idéias de Teoria dos Códigos para basearmos os *testes probabilísticos*. Também estamos interessados no espalhamento fornecido pela codificação, embora, estaremos totalmente em outro contexto. Entraremos nos detalhes das diferenças no próximo capítulo.

Serão utilizadas durante os próximos capítulos as seguintes notações: $\boldsymbol{K}$ como um corpo finito de característica prima $\boldsymbol{p}$ e de $\boldsymbol{k}$ elementos; $\boldsymbol{m, d, \ell} \in \mathbb{N}_*$ como, respectivamente, o número de variáveis, o grau máximo do polinômio (para o grau total ou o grau nas variáveis) e o tamanho de uma restrição sobre K; $\boldsymbol{D} \subseteq \boldsymbol{H} \subseteq K$ com $d+1$ e ℓ elementos, respectivamente, e, por fim, $\boldsymbol{f, g, h} \in {}^{H^m}K$ como funções de H^m a valores em K. Queremos que $d < p$ e $d < \ell \leq k$.

II.2–1. Sejam A um conjunto não vazio com uma probabilidade associada $\mathcal{P}r$ e Σ um alfabeto. Então, para $x, y \in {}^{A}\Sigma$, tomemos a métrica $\boldsymbol{\Delta(x, y)} := \#\{i \in A \| x(i) \neq y(i)\}$ e a pseudo–métrica $\boldsymbol{\mu(x, y) := \mu_A(x, y)} := \mathcal{P}r_{i \in A}\,\{\ x(i) \neq y(i)\ \}$, ambas sobre ${}^{A}\Sigma$. Note que, se A é finito e com probabilidade uniforme, então $\mu(x, y) = \Delta(x, y)/\#A$ e é também uma métrica que assume valores em $[0, 1]$. Neste caso, chamá-la-emos de `Proximidade de Hamming` e utilizá-la-emos pelo resto do texto. Por sua vez, $\boldsymbol{\lambda(x, y) := \lambda_A(x, y)} := 1 - \mu(x, y) := \mathcal{P}r_{i \in A}\,\{\ x(i) = y(i)\ \}$ chamaremos de `Distância de Hamming`.[ii] Para $\alpha \in (0, 1]$, diremos que x e y são $\boldsymbol{\alpha}$`–próximos` sse $\mu(x, y) < \alpha$ (i.e., $\lambda(x, y) > 1 - \alpha$). Se x e y são palavras e estamos interessados em que estejam *próximas*, então a α–proximidade nos garante que temos menos do que uma fração α de letras distintas (i.e., letras "ruins"). Por outro lado, se $\alpha \in [0, 1)$, também diremos que x e y são $\boldsymbol{\alpha}$`–distantes` sse não são $(1 - \alpha)$–próximos.[iii] Neste caso, se x e y são palavras e desejamos que sejam distantes, então a α–distância é a fração máxima de letras iguais (i.e., letras não desejadas). Se $C \subseteq {}^{A}\Sigma$, então a $\boldsymbol{\alpha}$`–vizinhança de` $\boldsymbol{C}$, ($\boldsymbol{\mathscr{V}_\alpha(C)} := \left\{\ y \in {}^{A}\Sigma \ \|\ \exists x \in C, \lambda(x, y) > 1 - \alpha\ \right\}$), são os elementos de ${}^{A}\Sigma$ que são α–próximos a algum elemento de C. Neste texto, a partir de agora, assumiremos que $\boldsymbol{\Sigma}$, $\boldsymbol{\Phi}$, $\boldsymbol{\Upsilon}$, $\boldsymbol{\Sigma}_n$ e $\boldsymbol{\Phi}_n$ são alfabetos puros, para todos os n's em $\mathbb{N}$.

Continuando, para uma função $f \colon \mathbb{N}_* \to \mathbb{N}_*$ estritamente crescente e para todo $n \in \mathbb{N}_*$, $C_n \subseteq {\Phi_n}_*^{f(n)} := (\Phi_n \smallsetminus \{\not b\})^{f(n)}$ (um conjunto de palavras de comprimento $f(n)$ em Φ_n, ou seja, sem o símbolo separador $\not b$), $F_n \subseteq \Sigma_*^n := (\Sigma \smallsetminus \{\not b\})^n$ (outro conjunto de palavras de comprimento n em Σ e, também, sem o símbolo separador) e $\alpha(n) \in [0, 1)$, tome $F := \bigcup_{n \in \mathbb{N}_*} F_n$ e $C := \bigcup_{n \in \mathbb{N}_*} C_n$. Então a função injetora

$$\boldsymbol{\varphi :\ F \subseteq \Sigma_*^* \subseteq \Sigma^* \ \hookrightarrow\ C \subseteq \bigcup_{n \in \mathbb{N}_*} \Phi_{n*}^* \subseteq \bigcup_{n \in \mathbb{N}_*} \Phi_n^*}$$

é uma $\boldsymbol{\alpha(n)}$`–codificação de` $\boldsymbol{F}$ `em` $\boldsymbol{C}$ (ou simplesmente, de $\boldsymbol{F}_n$ em $\boldsymbol{C}_n$) sse cada imagem de F_n por φ está

[ii] Note apenas que, infelizmente, a *Distância de Hamming* tem um sentido oposto ao convencional de distâncias métricas, visto que, se x e y tem distância 0, eles não só não são iguais, como são totalmente distintos.
[iii] Isto é, $\lambda(x, y) \leq \alpha$, ou $\mu(x, y) \geq 1 - \alpha$.

contida em C_n (i.e., $\varphi[\![F_n]\!] \subseteq C_n$) e, para todo $n \in \mathbb{N}_*$ suficientemente grande e para todo $x, y \in F_n$, $\varphi(x)$ e $\varphi(y)$ [iv] são $\alpha(n)$–distantes[v] e, portanto, $\varphi(y) \notin \mathfrak{V}_{1-\alpha(n)}(\varphi(x))$. Assim, quanto menor for $\alpha(n)$, mais esparsa estará a codificação (i.e., a imagem de F pelo φ), permitindo um melhor reconhecimento dos códigos distintos. Algumas vezes utilizaremos os alfabetos Φ_n iguais a um único alfabeto Φ. Com isto teremos a função $\varphi\colon F \subseteq \Sigma_*^* \subseteq \Sigma^* \hookrightarrow C \subseteq \Phi_*^* \subseteq \Phi^*$ com a propriedade de levar palavras distintas e de mesmo comprimento n no alfabeto Σ, para também palavras de mesmo comprimento de Φ, mas $\alpha(n)$–distantes. Note que, neste caso, $C := \bigcup_{n\in\mathbb{N}_*} C_n$ seria um *problema de decisão* em Φ^*.

Se φ é uma $\alpha(n)$–codificação, então, para todo $a \in \Phi_n$, existe no máximo um $x \in F_n$, tal que a está na $\frac{1-\alpha(n)}{2}$–vizinhança de $\varphi(x)$.[vi] Observe antes que, da definição, se existe tal x, a e $\varphi(x)$ devem ter o mesmo comprimento. Com efeito, suponha $x, x' \in F_n$ tais que $\varphi(x)$ e $\varphi(x')$ sejam $\frac{1-\alpha(n)}{2}$–próximos de a. Como μ é pseudo–métrica,

$$\mu(\varphi(x),\varphi(x')) \;\leq\; \mu(\varphi(x),a) + \mu(\varphi(x'),a) \;<\; \frac{1-\alpha(n)}{2} + \frac{1-\alpha(n)}{2} \;=\; 1-\alpha(n).$$

Logo, $\varphi(x)$ e $\varphi(x')$ são $(1-\alpha(n))$–próximos, portanto não são $\alpha(n)$–distantes e, pela definição de *codificação*, eles são iguais.

As codificações serão de fundamental importância para a nossa tarefa de provar o Teorema do $\mathcal{PCP}$ I.4–4. Por seu lado, elas inevitavelmente decorrem de propriedades algébricas. Por isto, para cada $n \in \mathbb{N}_*$, fixamos convenientemente um corpo finito K de k elementos e de característica p e um conjunto não vazio $H \subseteq K$ de ℓ elementos. A idéia será escolher um "bom" m (que também variará com n) e codificar uma seqüência x em H^n, por uma função $f_x \in {}^{H^m}K$. Para isto, ordena-se a gosto o conjunto H e fornece-se f_x *ponto-a-ponto*, isto é, como uma seqüência indexada por H^m, $(\ f_x(a)\ )_{a\in H^m}$. Estamos, então, codificando uma palavra x no alfabeto $\Psi_{(\ell)}$ e de comprimento n, em uma outra f_x no alfabeto $\Psi_{(k)}$ e de comprimento ℓ^m. Vamos ao primeiro exemplo.

Codificação por Funcional Linear C–I ($\frac{1}{\ell}$–codificação)**.** *Codificaremos a seqüência $x \in H^n$ pelo* `funcional linear` *dado pelo* `produto interno`

$$f_x(\xi) \;:=\; \boldsymbol{\mathcal{F}_x(\xi)} \;:=\; \xi \cdot x^{\perp},$$

sendo que $\xi := (\xi_1, \ldots, \xi_n)$ é uma variável e $x^\perp$ é o vetor transposto*. Note que f_x está em ${}^{H^n}K$. Para $x, y \in H^n$ distintos, $x-y \neq 0$, então para $a \in H^n$, $f_x(a) = f_y(a)$ sse $a\cdot(x-y)^\perp = 0$ sse a está no hiper–plano perpendicular ao vetor não nulo $x-y$. Como um hiper–plano tem uma fração $1/\ell$ de pontos de H^n, $\lambda_{H^n}(f_x, f_y) = 1/\ell$ e obtemos a $\frac{1}{\ell}$–*codificação de $\Psi_{(\ell)}^n$ em $\Psi_{(k)}^{\ell^n}$.

Observe que esta codificação (i.e., $1/\ell$), assim como os alfabetos $\Psi_{(\ell)}$ e $\Psi_{(k)}$, podem ser constantes para n. Apesar disto ser o ideal para nós, não é de praxe. Por outro lado, temos algo de ruim, qual seja, o comprimento exponencial da palavra codificada. Teremos bastante trabalho na busca de outras codificações que minorem este problema e não atrapalhem muito no resto. Para isto, a ferramenta mais poderosa são, sem dúvida nenhuma, os *polinômios*.

[iv] Note que, pela definição, $\varphi(x)$ e $\varphi(y)$ têm o mesmo comprimento.
[v] Ou seja, $\lambda(\varphi(x),\varphi(y)) \leq \alpha(n)$, ou $\mu(\varphi(x),\varphi(y)) \geq 1-\alpha(n)$.
[vi] Isto é, $\varphi(x)$ e a são $\frac{1-\alpha(n)}{2}$–próximos; não são $\frac{1+\alpha(n)}{2}$–distantes; $\mu(\varphi(x),a) < (1-\alpha(n))/2$; ou, $\lambda(\varphi(x),a) > (1+\alpha(n))/2$.

II.3 Polinômios

II.3–1. Entraremos então no detalhamento dos `polinômios`. Estamos interessados exatamente nos valores dos polinômios sobre os *pontos*[vii] e, por isto, deliberadamente sempre cometemos o abuso de confundir as *funções polinomiais* com os *polinômios* propriamente ditos. Portanto, assumiremos sempre o grau em cada variável menor do que p, a característica do corpo K de k elementos. Tomaremos também ℓ como a cardinalidade de $H \subseteq K$ e ainda $D \subseteq H$ terá $d+1$ elementos, com $d < p$ e $d < \ell \leq k$. Se h é um polinômio, $\boldsymbol{\partial_{[\xi]}(h)}$ é o `grau do polinômio` $\boldsymbol{h}$ `na variável` $\boldsymbol{\xi}$ e $\boldsymbol{\partial(h)}$ é o `grau total do polinômio`. Como é comum, polinômios constantes e não nulos têm grau 0 e o polinômio nulo tem grau negativo (ou melhor, infinito negativo). Os `conjuntos dos polinômios de` $\boldsymbol{m}$ `variáveis` $\boldsymbol{K[\xi_1,\ldots,\xi_m]_d}$, $\boldsymbol{K[\xi_1,\ldots,\xi_m]_{\langle d_1,\ldots,d_m\rangle}}$ e $\boldsymbol{K[\xi_1,\ldots,\xi_m]_{\langle d\rangle}}$, são, respectivamente, `os de grau total no máximo` $\boldsymbol{d}$, `os de grau na variável` $\boldsymbol{\xi_i}$ no máximo $\boldsymbol{d_i}$, para todo $i \in \{1\ldots m\}$, e `os de grau nas variáveis no máximo` $\boldsymbol{d}$.[viii] Se quisermos restringi-los a um domínio $J \subseteq K^m$, denotá-los-emos respectivamente por $\boldsymbol{K[(\xi_1,\ldots,\xi_m)\in J]_d}$, $\boldsymbol{K[(\xi_1,\ldots,\xi_m)\in J]_{\langle d_1,\ldots,d_m\rangle}}$ e $\boldsymbol{K[(\xi_1,\ldots,\xi_m)\in J]_{\langle d\rangle}}$.

Observação. *Nesta observação e nos teoremas subseqüentes não será necessário supor o corpo K finito, bastando tomar finito $H \subseteq K$. Se $h \in K[\xi \in H]_d$ é um polinômio não nulo de grau na variável ξ no máximo d, então h tem no máximo d raízes em H. Mais ainda, sobre a* Proximidade de Hamming, *$\mu_H(h,0) \geq 1 - d/\#H$, em que $0 \in K[\xi \in H]_d$ é o polinômio nulo. Podemos também dizer que h e 0 são $\frac{d}{\#H}$–distantes em H, ou $\lambda_H(h,0) \leq d/\#H$.*

Teorema de Schwartz II.3–2 ([Sch80])**.** *Sejam $J := \prod_{i=1}^m J_i$, tal que $J_i \subseteq K$ é finito e não vazio, e $h_m \in K[(\xi_1,\ldots,\xi_m)\in J]$, um polinômio não nulo de m variáveis e restrito ao domínio J. Tome então recursivamente, para cada $i \in \{m\ldots 1\}$ em ordem decrescente,*

- *$d_i := \partial_{[\xi_i]}(h_i)$, o grau de h_i na variável ξ_i, e*
- *$h_{i-1} \in K[\xi_1,\ldots,\xi_{i-1}]$, tal que, pela* Divisão de Euclides*,*

$$h_i(\xi_1,\ldots,\xi_i) \;:=\; h_{i-1}(\xi_1,\ldots,\xi_{i-1})\cdot\xi_i^{d_i} + r(\xi_1,\ldots,\xi_i),$$

sendo que $\partial_{[\xi_i]}(r) < d_i$, se $d_i > 0$, e $r = 0$, caso contrário.

Então h_m e 0 são $\left(\sum_{i=1}^m \frac{d_i}{\#J_i}\right)$–distantes em J, (ou seja, $\lambda_J(k,0) \leq \sum_{i=1}^m \frac{d_i}{\#J_i}$, ou $\mu_J(k,0) \geq 1 - \sum_{i=1}^m \frac{d_i}{\#J_i}$).

Prova. Nas hipóteses do teorema, seja $H_i := \prod_{j=1}^i J_j$. Procuraremos provar por indução em $i \in \{1\ldots m\}$ que $\lambda_{H_i}(h_i,0) \leq \sum_{j=1}^i (d_j/\#J_j)$. Como para $i := 1$ é imediato, dada a observação acima, suporemos válido para $i-1$. Defina

$$g_{\xi_1,\ldots,\xi_{i-1}}(\xi_i) \;:=\; h_i(\xi_1,\ldots,\xi_i) \;:=\; h_{i-1}(\xi_1,\ldots,\xi_{i-1})\cdot\xi_i^{d_i} + r(\xi_1,\ldots,\xi_i).$$

[vii] Entendemos por `pontos` de uma função os elementos do domínio.
[viii] Este caso seria o mesmo do anterior, em que todos os d_i's são iguais a d.

Portanto,

$$\begin{aligned}
\lambda_{H_i}(h_i,0) &:= \mathop{\mathcal{P}r}_{(x_1,\ldots,x_i)\in H_i}\{\ h_i(x_1,\ldots,x_i)=0\ \}\\
&= \mathop{\mathcal{P}r}_{(x_1,\ldots,x_i)\in H_i}\{\ h_{i-1}(x_1,\ldots,x_{i-1})=0 \quad \text{e} \quad h_i(x_1,\ldots,x_i)=0\ \}\\
&\quad + \mathop{\mathcal{P}r}_{(x_1,\ldots,x_i)\in H_i}\{\ h_{i-1}(x_1,\ldots,x_{i-1})\neq 0 \quad \text{e} \quad g_{x_1,\ldots,x_{i-1}}(x_i)=0\ \}\\
&\leq \mathop{\mathcal{P}r}_{(x_1,\ldots,x_i)\in H_i}\{\ h_{i-1}(x_1,\ldots,x_{i-1})=0\ \}\\
&\quad + \mathop{\mathcal{P}r}_{(x_1,\ldots,x_i)\in H_i}\{\ x_i \text{ é uma raiz do polinômio não nulo } g_{x_1,\ldots,x_{i-1}}\in K[\xi_i]_d\ \}\\
&\leq \mathop{\mathcal{P}r}_{(x_1,\ldots,x_{i-1})\in H_{i-1}}\{\ h_{i-1}(x_1,\ldots,x_{i-1})=0\ \}+\frac{d_i}{\#J_i} \quad \leq \quad \sum_{j=1}^{i}\frac{d_j}{\#J_j}.
\end{aligned}$$

[II.3–2] ■

Vejamos agora os corolários, que são uma imediata conseqüência do teorema e sobre os quais basearemos todo o nosso trabalho.

Corolário da Distância dos Polinômios de Grau Baixo II.3–3. *Se $f,g\in K[(\xi_1,\ldots,\xi_m)\in H^m]_{\langle d_1,\ldots,d_m\rangle}\subseteq {}^{H^m}K$ são polinômios distintos de m variáveis e de grau na variável ξ_i no máximo d_i, para todo $i\in\{1\ldots m\}$, então também são $\sum_{i=1}^{j}\frac{d_i}{\ell}$–distantes em H^m, (i.e., $\lambda_{H^m}(f,g)\leq\sum_{i=1}^{j}d_i/\ell$, ou $\mu_{H^m}(f,g)\geq 1-\sum_{i=1}^{j}d_i/\ell$).*

■

Corolário da Distância dos Polinômios de Grau nas Variáveis Baixo II.3–4. *Se $f,g\in K[(\xi_1,\ldots,\xi_m)\in H^m]_{\langle d\rangle}\subseteq {}^{H^m}K$ são polinômios distintos de m variáveis e de grau nas variáveis no máximo d, então também são $\frac{md}{\ell}$–distantes em H^m, (i.e., $\lambda_{H^m}(f,g)\leq md/\ell$, ou $\mu_{H^m}(f,g)\geq 1-md/\ell$).*

■

Corolário da Distância dos Polinômios de Grau Total Baixo II.3–5. *Se $f,g\in K[(\xi_1,\ldots,\xi_m)\in H^m]_{d}\subseteq {}^{H^m}K$ são polinômios distintos de m variáveis e de grau total no máximo d, então também são $\frac{d}{\ell}$–distantes em H^m, (i.e., $\lambda_{H^m}(f,g)\leq d/\ell$, ou $\mu_{H^m}(f,g)\geq 1-d/\ell$).*

■

II.4 Codificações

II.4–1. Como $D\subseteq H$ tem cardinalidade $d+1$, para $\vec{i}:=(i_1,\ldots,i_m)$, uma matriz

$$x \quad := \quad (x_{\vec{i}})_{\vec{i}\in D^m} \quad \in \quad H^{(d+1)^m},$$

será chamada de `cubo de dimensão` $\boldsymbol{m}$ `e tamanho` $\boldsymbol{d+1}$ `de` $\boldsymbol{H}$. Por outro lado, x pode ser visto como uma função f_x de $D^m\subseteq H^m$ a H e chamá-la-emos de uma `representação do cubo` $\boldsymbol{x}$ ou de uma `função definida`

no cubo de dimensão m e de tamanho $d+1$.

Codificação Polinomial com Grau nas Variáveis Baixo C–II ⟨Primeira versão⟩ ($\frac{md}{\ell}$–codificação)**.** *Escolheremos convenientes m e d, tais que $d < p$, $md \leq \ell$ e $n \leq (d+1)^m$.*[ix] *Com isto, podemos assumir $x \in H^n$ "$\subseteq$" $H^{(d+1)^m}$, ou seja, podemos estendê-lo a um* cubo de dimensão m e de tamanho $d+1$

$$x \quad := \quad (x_{\vec{i}})_{\vec{i} \in \{0 \ldots d\}^m} \quad \in \quad H^{(d+1)^m},$$

para $\vec{i} := (i_1, \ldots, i_m)$. Tomemos $f_x \in K[(\xi_1, \ldots, \xi_m) \in H^m]_{\langle d \rangle} \subseteq {}^{H^m}K$, tal que o cubo x *defina os coeficientes dos monômios de f_x, isto é,*

$$f_x(\xi_1, \ldots, \xi_m) \quad := \quad \sum_{\vec{i} \in \{0 \ldots d\}^m} x_{\vec{i}} \xi_1^{i_1} \cdots \xi_m^{i_m}.$$

Então, do Corolário da Distância dos Polinômios de Grau nas Variáveis Baixo II.3–4*, concluímos a $\frac{md}{\ell}$–*codificação de $\Psi_{(\ell)}^n$ em $\Psi_{(k)}^{\ell^m}$. *Conforme observado no Parágrafo* II.2–1*, uma função $f \in {}^{H^m}K$ está no máximo em uma $\frac{1-md/\ell}{2}$–vizinhança de algum polinômio de m variáveis e de grau nas variáveis no máximo d.*[x]

II.4–2. Lembrando que $d < p$ e $d < \ell \leq k$, tomemos uma função f'_x de D^m em H que *represente* o *cubo de dimensão m e de tamanho $d+1$*,

$$x \quad := \quad (x_{\vec{i}})_{\vec{i} \in D^m} \quad \in \quad H^{(d+1)^m}.$$

Sendo assim, $f'_x(\vec{i}) = x_{\vec{i}}$, para todo $\vec{i} := (i_1, \ldots, i_m) \in D^m \subseteq H^m$. É de conhecimento geral que, utilizando-se a `Interpolação de Lagrange`, temos a construção, por uma Máquina de Turing com alfabeto k-ário $\Psi_{(k)}$ e de tempo polinomial em m e d, do *único polinômio* $f_x \in K[(\xi_1, \ldots, \xi_m) \in H^m]_{\langle d \rangle} \subseteq {}^{H^m}K$ que estende f'_x de D^m para todo o H^m.

Codificação Polinomial com Grau nas Variáveis Baixo C–III ⟨Segunda versão⟩ ($\frac{md}{\ell}$–codificação)**.** *Na verdade, esta* codificação *é análoga à anterior, mudando somente o método de construção. Escolha novamente convenientes m e d, tais que $d < p$, $d < \ell \leq k$, $md \leq \ell$ e $n \leq (d+1)^m$ e assumiremos $x \in H^n$ "$\subseteq$" $H^{(d+1)^m}$. Logo podemos tomar uma* função f'_x *de D^m a H* que representa *a extensão de x ao* cubo de dimensão m e de tamanho $d+1$. *Conforme observado acima, pela* Interpolação de Lagrange*, existe um único* polinômio de m variáveis e de grau nas variáveis no máximo d, *$f_x \in K[(\xi_1, \ldots, \xi_m) \in H^m]_{\langle d \rangle} \subseteq {}^{H^m}K$, que estende f'_x de D^m para todo o H^m. A $\frac{md}{\ell}$–*codificação $\Psi_{(\ell)}^n$ em $\Psi_{(k)}^{\ell^m}$ *decorre conforme a Codificação Polinomial com Grau nas Variáveis Baixo* C–II ⟨*Primeira versão*⟩ (*$\frac{md}{\ell}$–codificação).*

Vejamos agora a próxima codificação. Logo mais, utilizá-la-emos para construir outra codificação.

Codificação Polinomial com Grau nas Variáveis Baixo C–IV ($\frac{2}{\log\log(n)}$–codificação)**.** *Queremos codificar de maneira conveniente $x \in \mathbb{Z}_2^n$, ou seja, uma* palavra de n bits. *Tome $d(n) := \lceil \log(n) \rceil - 1$, $m(n) := \lceil (d(n)+1)/\log(d(n)+1) \rceil$, $p(n)$ um primo entre $(d(n)+1)^2$ e $2(d(n)+1)^2$, $k(n) := p(n)^z$ (para $z \in \mathbb{N}_*$) e $\ell(n)$ entre $(d(n)+1)^2$ e $k(n)$. Seja $D_n \subseteq H_n \subseteq K_n$, tal que $\#D_n =: d(n)+1$ (e, pela definição, $\#H_n =: \ell(n)$ e $\#K_n =: k(n)$). Portanto,*

$$\#D_n^{m(n)} \quad := \quad (d(n)+1)^{m(n)} \quad = \quad \mathsf{e}^{m(n)\log(d(n)+1)}$$

$$= \quad \mathsf{e}^{\left\lceil \frac{d(n)+1}{\log(d(n)+1)} \right\rceil \log(d(n)+1)} \quad \geq \quad \mathsf{e}^{d(n)+1} \quad \geq \quad \mathsf{e}^{\lceil \log(n) \rceil} \quad \geq \quad n$$

[ix] Observe que, se tomarmos d fixo, m e ℓ podem ser de ordem logarítmica em n.

[x] Isto é, existe no máximo um $g \in K[(\xi_1, \ldots, \xi_m) \in H^m]_{\langle d \rangle}$, tal que f e g são $\frac{1-md/\ell}{2}$–próximos.

e temos uma sobrejeção $D_n^{m(n)} \overset{\sigma}{\twoheadrightarrow} \{1\dots n\}$. *Por outro lado, temos a* inclusão canônica $\mathbb{Z}_2 \hookrightarrow \mathbb{Z}_{p(n)} \hookrightarrow K_n$. *Conjugando a sobrejeção e a inclusão com a função* $f'_x\colon \{1\dots n\} \to \mathbb{Z}_2$ *que* representa $x := (x_i)_{i\in\{1\dots n\}} \in \mathbb{Z}_2^n$, *temos* f''_x *de* $D_n^{m(n)} \subseteq H_n^{m(n)}$ *a* K_n. *Note que*

$$\begin{aligned}
\frac{m(n)d(n)}{\ell(n)} &\leq \frac{\left\lceil \frac{d(n)+1}{\log(d(n)+1)} \right\rceil d(n)}{(d(n)+1)^2} \leq \frac{\left\lceil \frac{d(n)+1}{\log(d(n)+1)} \right\rceil}{d(n)+1} \leq \frac{1}{\log(d(n)+1)} + \frac{1}{d(n)+1} \\
&\leq \frac{1}{\log\log(n)} + \frac{1}{\log(n)} \leq \frac{2}{\log\log(n)} \leq 1, \\
m(n)d(n) &= \left\lceil \frac{\lceil\log(n)\rceil}{\log\lceil\log(n)\rceil} \right\rceil (\lceil\log(n)\rceil - 1) \leq \left(\frac{\lceil\log(n)\rceil}{\log\log(n)} + 1 \right) \lceil\log(n)\rceil \\
&\leq \frac{\lceil\log(n)\rceil^2}{\log\log(n)} + \lceil\log(n)\rceil \in \mathcal{O}\left(\frac{\log(n)^2}{\log\log(n)} \right) \subseteq \mathcal{O}(\mathrm{Poli}(\log(n))), \\
m(n)\log(\ell(n)) &\leq m(n)\log(k(n)) := m(n)\log(p(n)^z) \leq \left\lceil \frac{d(n)+1}{\log(d(n)+1)} \right\rceil z\log(2(d(n)+1)^2) \\
&\leq \left(\frac{d(n)+1+\log(d(n)+1)}{\log(d(n)+1)} \right) 3z\log(d(n)+1) \\
&= 3z\left(\lceil\log(n)\rceil + \log\lceil\log(n)\rceil\right) \in \mathcal{O}(\log(n)) \quad e \\
\ell(n)^{m(n)} &= \mathsf{e}^{m(n)\log(\ell(n))} \leq \mathsf{e}^{3z(\lceil\log(n)\rceil+\log\lceil\log(n)\rceil)} \leq (n\mathsf{e}(\log(n)+1))^{3z} \\
&\in \mathcal{O}(n^{3z}\log(n)^{3z}) \subseteq \mathcal{O}(\mathrm{Poli}(n)).
\end{aligned}$$

Portanto, temos para n suficientemente grande, $m(n)d(n) \leq \ell(n)$. *Como* $d(n) < p(n)$ *e* $d(n) < \ell(n) \leq k(n)$, *analogamente à Codificação Polinomial com Grau nas Variáveis Baixo* C–III ⟨*Segunda versão*⟩ (*$\frac{md}{\ell}$–codificação), existe um único polinômio* $f_x \in K_n[(\xi_1,\dots,\xi_{m(n)}) \in H_n^{m(n)}]_{\langle d(n)\rangle}$ *de* $m(n)$ *variáveis e de grau nas variáveis no máximo* $d(n)$, *que estende* f''_x. *Assim, escolhendo-se* $\{1\dots n\} \overset{\sigma^{-1}}{\hookrightarrow} D_n^{m(n)}$ *como uma das inversões de* σ, *temos* $f_x \circ \sigma^{-1}(i) = x_i$, *o i-ésimo elemento de x. Assim, concluímos a* $\frac{2}{\log\log(n)}$–codificação de $\Psi^n_{(2)}$ em palavras de comprimento $\ell(n)^{m(n)} \in \mathcal{O}(n^{3z}\log(n)^{3z})$ sobre o alfabeto $\Psi_{\left(2^{z\lceil\log(n)\rceil^{2z}}\right)}$. *Para finalizar, é bom lembrar que* $2/\log\log(n)$ *tende para* 0 *quando n cresce, logo temos uma* α*–codificação, para todo* $\alpha \in (0,1)$, *ao se tomar n suficientemente grande.*

Codificação Polinomial com Grau Total Baixo C–V (*$\frac{d}{\ell}$*–codificação)**.** *Primeiro veremos que um polinômio de m variáveis e de grau total no máximo d tem no máximo* $\binom{m+d}{d} = \binom{m+d}{m}$ *monômios. Cada monômio pode ser representado por d bolinhas (•) e m traços verticais (|) devidamente ordenados. O número de bolinhas (•) entre o* $(i-1)$*-ésimo e o i-ésimo traço (|), representa o grau da i-ésima variável neste monômio. As bolinhas (•) que porventura venham no final, representam o quanto falta para o grau do monômio chegar a d. Então, das* $m+d$ *possíveis posições, escolhemos d (ou m) para serem as bolinhas (•) (ou traços (|)), e temos o número* $\binom{m+d}{d} = \binom{m+d}{m}$.

Visto isto, igualmente à Codificação Polinomial com Grau nas Variáveis Baixo C–II ⟨*Primeira versão*⟩ (*$\frac{md}{\ell}$–codificação), escolha convenientes m e d, tais que* $d < p$, $d \leq \ell$ *e* $n \leq \binom{m+d}{d}$.[xi] *Assumindo-se* $x \in H^n$ "$\subseteq$" $H^{\binom{m+d}{d}}$ *podemos tomar* $f_x \in K[(\xi_1,\dots,\xi_m) \in H^m]_d \subseteq {}^{H^m}K$ *um polinômio de m variáveis e de*

[xi] Observe que, se tomarmos d fixo, m pode ser de ordem da raiz d-ésima de n, sendo ainda ℓ e k constantes. Então m não é mais de ordem logarítmica em n, mas por sua vez, a distância da codificação não depende mais de m.

grau total no máximo d, tal que f_x tenha como coeficientes os valores de x. Então, do Corolário da Distância dos Polinômios de Grau Total Baixo II.3–5 *concluímos a $\frac{d}{\ell}$–*codificação de $\Psi_{(\ell)}^n$ em $\Psi_{(k)}^{\ell^m}$. *Conforme observado no Parágrafo* II.2–1, *uma função $f \in {}^{H^m}K$ está no máximo em uma $\frac{1-d/\ell}{2}$–vizinhança de algum polinômio de m variáveis e de grau total no máximo d.*[xii]

Codificação Polinomial com Grau Total no Máximo 1 C–VI ⟨Primeira versão⟩ ($\frac{1}{2}$–codificação). *Tome $d(n) := 1$, $\ell(n) := k(n) := p(n) := 2$, ou seja $H_n := K_n := \mathbb{Z}_2$, e $m(n) := n - 1$. Logo $\binom{m(n)+d(n)}{d(n)} = m(n) + 1 = n$. Para $x := (x_i)_{i\in\{1\ldots n\}} \in \mathbb{Z}_2^{n+1}$, tome $f_x \in \mathbb{Z}_2[\xi_1, \ldots, \xi_n]_1$, tal que*

$$f_x(\xi_1, \ldots, \xi_n) \quad := \quad x_0 + \sum_{i\in\{1\ldots n\}} x_i\xi_i.$$

Conforme a Codificação Polinomial com Grau Total Baixo C–V *($\frac{d}{\ell}$–codificação), obtemos a $\frac{d(n-1)}{\ell(n-1)} = \frac{1}{2}$–*codificação $\Psi_{(2)}^n$ em $\Psi_{(2)}^{2^{n-1}}$. *Voltamos assim a obter uma codificação constante com um comprimento exponencial da palavra codificada.*

Codificação Polinomial com Grau Total no Máximo 1 C–VII ⟨Segunda versão⟩ ($\frac{1}{2}$–codificação). *Sejam $z(n) := 2^{\lceil \log_2(n)\rceil}$ e as* inclusões canônicas

$$\mathbb{Z}_2^n \quad \hookrightarrow \quad \mathbb{Z}_2^{z(n)} \quad \hookrightarrow \quad \mathbb{Z}_2^{\lceil \log_2(n)+1\rceil \left\lceil \frac{z(n)}{\log_2(n)+1}\right\rceil}.$$

Então, conforme a Codificação Polinomial com Grau Total no Máximo 1 C–VI ⟨*Primeira versão*⟩ ($\frac{1}{2}$–*codificação), note que, para cada um dos $\left\lceil \frac{z(n)}{\log_2(n)+1}\right\rceil \in \mathcal{O}(\frac{n}{\log(n)})$, temos uma $\frac{1}{2}$–*codificação de $\mathbb{Z}_2^{\lceil \log_2(n)\rceil+1}$ em $\mathbb{Z}_2^{2^{\lceil \log_2(n)\rceil}} =: \mathbb{Z}_2^{z(n)}$. *Componha todas estas codificações, bloco por bloco, e obtemos uma codificação em $\mathbb{Z}_2^{z(n)\left\lceil \frac{z(n)}{\log_2(n)+1}\right\rceil}$. Ou seja, uma $\frac{1}{2}$–*codificação de $\Psi_{(2)}^n$ em palavras de comprimento $z(n)\left\lceil \frac{z(n)}{\log_2(n)+1}\right\rceil \in \mathcal{O}(n^2)$ no alfabeto $\Psi_{(2)}$.

Esta codificação é muito boa, a menos de ser dividida em muitos blocos de polinômios. Na anterior isto não acontece, mas o número de variáveis $m(n) \in \mathcal{O}(n)$. Adaptaremos então a Codificação Polinomial com Grau nas Variáveis Baixo C–IV ($\frac{2}{\log\log(n)}$–codificação).

Codificação Polinomial com Grau Total Baixo C–VIII ($\frac{2}{\log\log(n)}$–codificação). *Note que um polinômio de m variáveis e de grau nas variáveis no máximo s tem o grau total no máximo $d := ms$. Portanto, conforme a Codificação Polinomial com Grau nas Variáveis Baixo* C–IV (*$\frac{2}{\log\log(n)}$–codificação), sejam $s(n) := \lceil \log(n)\rceil - 1$, $m(n) := \lceil (s(n)+1)/\log(s(n)+1)\rceil$, $p(n)$ um primo entre $(s(n)+1)^2$ e $2(s(n)+1)^2$, $k(n) := p(n)^z$ (para $z \in \mathbb{N}_*$), $\ell(n)$ entre $(s(n)+1)^2$ e $k(n)$ e $d(n) := m(n)s(n)$. Temos assim, nos polinômios de $K_n[(\xi_1, \ldots, \xi_{m(n)}) \in H_n^{m(n)}]_{d(n)}$ a $\frac{2}{\log\log(n)}$–*codificação de $\Psi_{(2)}^n$ em palavras de comprimento $\ell(n)^{m(n)} \in \mathcal{O}(n^{3z}\log(n)^{3z})$ sobre $\Psi_{\left(2^{z\lceil \log(n)\rceil^{2z}}\right)}$. Cumpre-nos lembrar também que vimos na Codificação Polinomial com Grau nas Variáveis Baixo* C–IV (*$\frac{2}{\log\log(n)}$–codificação) que $2(m(n)+1)\log(k(n)) \in \mathcal{O}(\log(n))$ e $d(n) := m(n)s(n) \in \mathcal{O}(\mathrm{Poli}(\log(n)))$, visto que nos será de grande serventia no próximo capítulo.*

Esta codificação será suficiente e falaremos agora dos *testes probabilísticos*.

[xii] Isto é, existe no máximo um $g \in K[(\xi_1, \ldots, \xi_m) \in H^m]_d$, tal que f e g são $\frac{1-d/\ell}{2}$–próximos.

Capítulo III

Testes Probabilísticos

Até agora estávamos vendo a *codificação* do ponto de vista da *Teoria dos Códigos*. A idéia principal é: dada uma palavra x que desejamos transmitir de um computador a outro, buscamos uma *codificação* em que, mesmo que tenhamos algum ruído de comunicação entre os dois computadores, ao receber a mensagem errada pode-se, com uma alta probabilidade, recuperar x corretamente. Para isto a *distância* entre as codificações é essencial. Todavia o nosso enfoque de *codificação* será outro. Dada uma palavra de entrada x, desejamos saber se ela é ou não uma *instância* de um determinado *problema de decisão*, utilizando a ajuda de uma fita de probabilidade e de um *esquema de provas robustas* π_x,[i] mas consultando poucos *bits* aleatórios e lendo um número constante de letras da testemunha.

Considere um problema de decisão definido por uma Máquina de Turing não probabilística, de tempo polinomial e "estritamente" *não–determinística*, (se existir!?),[ii] com um *esquema de provas* $\widehat{\pi}_x$ para instâncias x's. Observe que o comprimento da prova $\widehat{\pi}_x$ não pode ser da ordem de $\log(n)$ (muito menos constante); isto porque, caso contrário, poderíamos simular em tempo polinomial todas as possíveis testemunhas $\widehat{\pi}$ até acharmos $\widehat{\pi}_x$ (se ela existir), obtendo assim uma máquina determinística. Note também que precisamos de toda a informação de $\widehat{\pi}_x$, visto que toda letra não lida poderia, de antemão, ser desprezada. É importante lembrar que, qualquer que seja a codificação do alfabeto binário $\Psi_{(2)}$ no alfabeto k-ário $\Psi_{(k)}$, se lemos d letras da codificação, ela não trará mais informação do que $d\log_2(k)$ *bits*. Por isto, na *Teoria dos Códigos*, codificamos sobre palavras e/ou alfabetos maiores ainda, mas a *distância* entre elas nos permite diferenciá-las, mesmo que algumas letras sejam trocadas. Isto é bom se não temos restrição quanto ao número de leituras da codificação, o que não é o nosso caso.

Por outro lado, para um *esquema de provas* $\widehat{\pi}_x$, se estamos em problemas de decisão de *máquinas probabilísticas*, podemos talvez não ser obrigados a ler todo o $\widehat{\pi}_x$. Contudo, se quisermos ter constante o número de leituras em $\widehat{\pi}_x$, o número de letras não lidas *aumenta* (pelo nosso comentário anterior), e assim, a certeza da corretude **muito provavelmente** deixará de ser constante. Não desejamos isto e todo o nosso trabalho será para evitar tal "conseqüência danosa". Portanto, pelo que foi descrito acima, gostaríamos de sentir o leitor convencido que, dado um *esquema de provas* $\widehat{\pi}_x$ (não–determinístico), não nos resolverá tomar um *esquema de provas robustas* π_x que seja simplesmente uma codificação das provas $\widehat{\pi}_x$, pois **não podemos**

[i] Veja a definição no Parágrafo I.3–1.
[ii] Isto é, se $\mathcal{P} \neq \mathcal{NP}$.

(e não precisaremos) recuperar todo o $\widehat{\pi}_x$. Vejamos o que podemos fazer então.

Descobriu-se que para algumas codificações, como por exemplo a linear e as polinomiais, temos *testes probabilísticos* que certificam, sem muito esforço, se uma testemunha fornecida é ou não (próxima) da codificação. Seria como se codificássemos nomes de alunos de alguma classe. Rodando-se tais testes podemos não saber de quem é a codificação, mas sabemos se estamos ou não falando de alguém da classe. Esta codificação deve ser de tal modo que, agora, com esta informação, também sem muito esforço, utilizamos características da classe e da codificação para descobrir de qual aluno é a testemunha. Como visto, semanticamente os testes são de dois tipos distintos e serão chamados respectivamente de: `testes de proximidade da codificação` e `testes de reconhecimento da codificação`. Lembre-se que todos estes testes, como as suas conjugações, devem estar no *esquema de provas robustas*.

Conforme o parágrafo anterior, precisamos de uma codificação para o *esquema de provas* $\widehat{\pi}_x$ de tal modo que, para uma entrada x, facilmente nos asseguremos se uma testemunha é ou não da codificação, ou seja, é ou não igual à codificação de algum dos $\widehat{\pi}_y$ (*teste de proximidade da codificação*) e, depois, se esta testemunha representa de fato a prova $\widehat{\pi}_x$, ou seja, é uma prova para a *instância* x, (*teste de reconhecimento da codificação*). Para conseguir isto, além da codificação especial, precisamos, na verdade, compor diversos testes. Assim, o nosso problema de decisão terá um *esquema de provas robustas* π_x que será a composição da codificação de $\widehat{\pi}_x$, com os *esquemas de provas robustas* de cada um dos seus testes.

Portanto, para cada $n \in \mathbb{N}_*$, fazemos uma codificação para cada *instância* x de comprimento n. O *teste de proximidade da codificação*, informado do comprimento n de uma entrada x, certifica-se, com alta probabilidade, se a palavra da fita de testemunha está $\alpha(n)$–próxima a uma das codificações. O *teste de reconhecimento da codificação*, de posse de x, verifica, com alta probabilidade, se esta codificação corresponde a uma prova para x. Note que a palavra da fita de testemunha pode estar apenas $\alpha(n)$–próxima de uma codificação, e portanto, o segundo teste pode ser enganado ao lê-lo, isto é, para cada letra da testemunha lida, ele tem uma probabilidade no máximo de $\alpha(n)$ de ser enganado. Como queremos continuar no *esquema de provas robustas*, tomaremos o teste de proximidade da codificação para $\alpha(n) := \alpha$ constante para n, tal que possa ser tão pequeno quanto se queira. Por sua vez, como o *teste de reconhecimento da codificação* lê um número constante para n de letras da fita de testemunha, limitamos esta possibilidade de **erro** (i.e., a *taxa de erro da robustez*) a uma fração também constante, ou seja, ao conjugarmos os dois testes, continuamos no *esquema de provas robustas*.[iii] Entremos nos detalhes

III.1 Introdução aos Testes

Para definir os `testes` precisamos mudar um pouco a nossa visão de uma *computação*. Sejam funções naturais $k(n)$, $p(n)$, $\ell(n)$, $d(n)$ e $m(n)$ e conjuntos H_n e K_n, todos indexados por $n \in \mathbb{N}_*$ e nas condições do Parágrafo II.3–1 (onde definimos os polinômios). Acrescente a eles $\boldsymbol{S_n} \subseteq H_n$ e $\boldsymbol{s(n)} := \#S_n - 1$. A partir de agora, também suporemos todos eles computáveis em tempo polinomial em n.

III.1–1. O nosso objetivo é construir testes "econômicos" para, por exemplo:

- verificar se um vetor $\vec{x} \in \mathbb{Z}_{p(n)}^{m(n)} := K_n^{m(n)}$ dado como entrada é ou não nulo e
- verificar se o polinômio $f \in K_n[(\xi_1, \ldots, \xi_{m(n)}) \in H_n^{m(n)}]_{\langle d(n)\rangle}$ dado como entrada satisfaz ou não $f\restriction_{S_n^{m(n)}} = 0$.

[iii] Veja mais detalhes sobre a taxa de erro da robustez em computações iteradas no Parágrafo I.3–2.

Note que no último teste, se $s(n) \geq d(n)$, então $f\!\upharpoonright_{S_n^{m(n)}} = 0$ [iv] sse f é nulo (i.e., $f = 0$), visto que f tem grau nas variáveis no máximo $d(n)$. Veremos que isto é fácil de se testar, mas poderia não o ser, nos casos em que $s(n) < d(n)$.

Queremos assim salientar que fica natural pensarmos em computações trabalhando diretamente sobre K_n (e/ou H_n e K_n^z, para algum $z \in \mathbb{N}_*$ constante para n). Neste sentido, teríamos as fitas com alfabetos de tamanho $k(n)$ (e/ou $\ell(n)$ e $k(n)^z$) e o tempo de execução passaria a depender somente de $m(n)$, no primeiro caso, e de $d(n)$, $m(n)$ e $s(n)$, no segundo. Como n faz parte da entrada, nós sabemos que não podemos ter nenhuma fita (e portanto a respectiva computação) variando com n. Por outro lado, conhecendo-se n e calculando-se os parâmetros que dependam só de n (p.e., $k(n)$, $p(n)$, $\ell(n)$, $d(n)$, $m(n)$, $s(n)$, S_n, H_n e K_n), podemos simular computações com os alfabetos das fitas variando, conforme o nosso desejo. Esta simulação será feita lendo-se a cada vez um `bloco de letras` da fita. Passaremos então a usar de tal artifício, mudando assim a nossa medida de tempo de computação e de número de leituras sobre a fita de testemunha π. Este expediente facilitar-nos-a centrar a atenção somente na parte **importante** da computação. Entretanto, nunca podemos deixar de lembrar que para se chegar ao número de leituras "real" devemos multiplicá-lo por $\log(k(n))$ (ou $\log(\ell(n))$, ou $\log(zk(n))$), assim também como para se chegar ao tempo de execução.

Lembramos que sempre consideraremos Σ, Φ, Υ, Σ_n e Φ_n como alfabetos, para todo $n \in \mathbb{N}$. Além do que foi visto até aqui, a partir de agora, para uma *$\alpha(n)$–codificação* φ de $F_n := \Upsilon_*^n := \Upsilon_n \setminus \{\not b\}$ em $C_n := \varphi[\![F_n]\!] \subseteq \Sigma_{n*}^* := (\Sigma_n \setminus \{\not b\})^*$ [v] (i.e., C_n é a imagem de φ por F_n que, por sua vez, são todas as palavras de comprimento n de Υ), esqueceremos da distância $\alpha(n)$ e da função injetora

$$\varphi\colon F := \bigcup_{n \in \mathbb{N}_*} F_n := \Upsilon_*^* \hookrightarrow C := \bigcup_{n \in \mathbb{N}_*} C_n \subseteq \bigcup_{n \in \mathbb{N}_*} \Sigma_{n*}^*$$

e só nos preocuparemos com a sua imagem C. Por abuso de notação, chamaremos C (ou ainda C_n) de uma `codificação` sobre o alfabeto Σ_n. Muitas vezes queremos que as entradas das computações sejam restritas a *codificações*. Nestes casos, gostaríamos de ter fitas com alfabetos Σ_n, mas só admitiremos palavras escritas nelas que sejam de $C_n \subseteq \Sigma_{n*}^*$ ao invés de todas as de Σ_{n*}^*.[vi] Lembremos mais uma vez que, se os alfabetos Σ_n's forem fixos em Σ, então $C \subseteq \Sigma_*^*$ será um problema de decisão.

Para explicar melhor estes abusos conceituais, definimos agora o que chamaremos de *pseudo–fitas*. Cada pseudo–fita, como veremos melhor no decorrer do capítulo, é um modo diferente de vermos as velhas fitas de entrada ρ e de testemunha π (ou de pedaços delas). Podemos, então, ou supor que as *pseudo–fitas* têm alfabetos Σ_n's (variando com $n \in \mathbb{N}_*$), ou que elas representariam informações recebidas na fita de trabalho por alguma sub-rotina. Neste último caso, seria como se, dado um endereço, ao invés de lermos a pseudo–fita, calculássemos e fornecêssemos na fita de trabalho o que deveria estar escrito naquele endereço dela. Assim, para $C := \bigcup_{n \in \mathbb{N}_*} C_n$ uma *codificação* sobre os alfabetos Σ_n, definimos:

- `pseudo–fita de parâmetro de entrada` de *só–leitura–indistinguível* (ou seja, com alfabeto $\Psi_{(1)}$). Nela lê-se o *parâmetro de entrada* $n \in \mathbb{N}_*$. Podemos pensar que recebemos n na fita de trabalho por um pré-processamento, como por exemplo, de um teste anterior, ou, a partir de um cálculo sobre a fita de entrada ρ com *só–leitura–indistinguível*.

- `pseudo–fita de teste` ($\boldsymbol{\varrho}$) de *só–leitura–direta* e com alfabeto Σ_n. Esta pseudo–fita também é uma espécie de entrada, que só deve conter palavras da codificação $C_n \subseteq \Sigma_{n*}^*$.

- `pseudo–fita de testemunha` ($\boldsymbol{\pi}$) de *só–leitura–direta* e com alfabeto Φ_n. Ela tem uma função semelhante à da fita de testemunha.

[iv] Denotaremos $f\!\upharpoonright_{S_n^{m(n)}}$ como sendo a função f com o domínio restrito a $S_n^{m(n)}$, conforme visto no Parágrafo I.1–1.

[v] Portanto, F_n e C_n só contêm palavras, ou seja, *strings* sem o símbolo separador $\not b$.

[vi] Note que, pela definição da codificação, todas as palavras de C_n têm o mesmo comprimento.

A entrada dos dados para os testes será feita pelas pseudo–fitas de parâmetro de entrada e de teste ϱ. Pela pseudo–fita de parâmetro de entrada lemos o `parâmetro entrada` n. Sobre o *parâmetro de entrada* calculamos, em tempo polinomial em n, os demais `parâmetros do teste`, que devem depender só de n. Dentre eles está o tamanho do alfabeto da pseudo–fita de teste ϱ, o qual é lido posteriormente como entrada para o teste, e o tamanho da pseudo–fita de testemunha π. Lembre-se que a pseudo–fita de teste contém somente palavras da codificação $C_n \subseteq \Sigma_n^*$.

É sempre bom ter em mente que o nosso objetivo é propiciar conjunções de testes. Portanto, propriedades testadas em um teste provavelmente serão hipóteses de outro teste. Logo, a pseudo–fita de teste ϱ de um teste pode ser um pedaço da pseudo–fita de testemunha π de outro teste. Note que, para tornar o tempo de computação similar ao que desejamos, deveremos ainda supor que temos uma `pseudo–fita de trabalho` com o mesmo alfabeto de cada uma das outras pseudo–fitas. Apesar de sempre estarmos nesta hipótese, não iremos mais nos referenciar a isto. Os testes poderão ainda se utilizar da fita de probabilidade τ, do modo mais conveniente possível.

III.1–2. Daremos agora uma outra definição para o *esquema de provas robustas* com uma pequena variação da fornecida no Parágrafo I.3–1. Seja M uma Máquina de Turing utilizando-se da fita de probabilidade τ e das pseudo–fitas de parâmetro de entrada, de testemunha π (com alfabeto Φ_n) e de teste ϱ (com a codificação C_n sobre o alfabeto Σ_n). Sejam, também, $L_n \subseteq C_n$ uma sub-codificação de C_n, $L := \bigcup_{n\in\mathbb{N}_*} L_n$ e $\alpha(n)$ uma função de $\mathbb{N}_*$ a valores em $[0,1)$. Portanto, M tem `esquema de provas` $\boldsymbol{\alpha(n)}$`–robustas para` $\boldsymbol{L}$ `restrito a` $\boldsymbol{C}$ (ou ainda para L_n restrito a C_n) sse existe $\varepsilon \in (0,1)$ tal que, para todo $n \in \mathbb{N}_*$, temos

$$\text{se} \quad x \in L_n \quad \text{então} \quad \mathop{\mathcal{P}r}_{\pi}\Big\{ \mathop{\mathcal{P}r}_{\tau}\{\ M \text{ aceita } x\ \} = 1 \Big\} > 0$$

e

$$\text{se} \quad x \in C_n \smallsetminus \mathbf{\mathcal{V}}_{\alpha(n)}(L_n) \ \text{[vii]} \quad \text{então} \quad \mathop{\mathcal{P}r}_{\pi}\Big\{ \mathop{\mathcal{P}r}_{\tau}\{\ M \text{ aceita } x\ \} < \varepsilon \Big\} = 1.$$

Aqui a probabilidade é tomada uniformemente, n e x são fornecidos na pseudo–fita de parâmetro de entrada e na pseudo–fita de teste ϱ, respectivamente, e π é uma pseudo–fita de testemunha. Se $\alpha(n)$ é constante em zero, omiti-lo-emos. Lembre-se que ε é chamado de *taxa de erro da robustez*.

Cumpre-nos observar que a definição admite o segundo "*gap*" nas aceitações de M. O primeiro *gap* está descrito no Parágrafo I.3–1 e se refere à *taxa de erro da robustez* $\varepsilon \in (0,1)$. Já o segundo decorre de permitirmos, mas não obrigarmos, que M aceite (ou rejeite) palavras $\alpha(n)$–próximos de palavras de L_n e que não estejam em L_n (i.e., palavras em $(\mathbf{\mathcal{V}}_{\alpha(n)}(L_n) \cap C_n) \smallsetminus L_n$). A outra diferença da primeira definição de *esquema de provas robustas* é a possibilidade de existirem pseudo–fitas, variando assim os alfabetos.

III.1–3. Descreveremos os *modelos computacionais* dos testes sobre as fitas e pseudo–fitas descritas acima. Queremos minimizar o número de leituras na fita de probabilidade e nas pseudo–fitas de teste e de testemunha, como também o tempo de execução da fase de decisão. Com efeito, voltaremos a dividir as computações por *fases*, conforme feito no Parágrafo I.3–3. A *fase de decisão* será ainda a mesma, mas precisamos redividir a *fase de endereçamento* em duas. Estas são as fases:

- `fase de pré-processamento dos parâmetros`. A partir da *pseudo–fita de parâmetro de entrada* lê-se o número n e calculam-se os demais *parâmetros do teste* em tempo polinomial em n para fornecê-los na fita de trabalho. Entre outras coisas, obtêm-se assim os tamanhos dos alfabetos das demais pseudo–fitas. A função desta fase não varia muito de teste para teste. Com isto, apesar dela existir para todos os testes, não iremos mais nos preocupar em descrevê-la.

- `fase de endereçamento`. Com os dados fornecidos na fita de trabalho, lê-se a fita de probabilidade τ para fornecer, também na fita de trabalho, endereços das pseudo–fitas de teste

[vii] Ou seja, são os $x \in C_n$ que não são $\alpha(n)$–próximos a ninguém de L_n (i.e., $\lambda(x, L_n) \leq 1 - \alpha(n)$, ou $\mu(x, L_n) \geq \alpha(n)$).

ϱ e de testemunha π. O tempo de execução também deve ser polinomial em n.

- `fase de decisão`. Lendo **somente** as letras das pseudo–fitas de teste ϱ e de testemunha π constantes nos endereços fornecidos pela *fase de endereçamento* e ajudada pelas informações fornecidas na fita de trabalho, esta fase deve decidir o aceite ou não da entrada.

O tempo de computação da última fase será chamado de `tempo de decisão ponderado`. Veja que este tempo se refere à computação sobre os alfabetos Σ_n e Φ_n que variam. Dentro do nosso espírito de só contarmos o tempo da computação "relevante", o tempo de decisão ponderado é a menos do tempo da decodificação dos respectivos alfabetos. Por outro lado, ao levarmos em conta esta decodificação, temos o `tempo de decisão bruto`, que se refere ao *tempo real de computação da fase de decisão*.

Sejam as famílias de funções naturais $R(n)$, $E(n)$, $Q(n)$ e $T(n)$. Queremos definir o *modelo computacional* das Máquinas de Turing no formato até agora visto e que tenham $r(n) \in \mathcal{O}(R(n))$, $e(n) \in \mathcal{O}(E(n))$, $q(n) \in \mathcal{O}(Q(n))$ e $t(n) \in \text{Poli}(T(n))$, tais que só consideraremos os aceites de entradas que tenham *parâmetro de entrada* n, se o seu *tempo de decisão ponderado* não ultrapassar $t(n)$. Mais ainda, tal computação deve ler no máximo $r(n)$ *bits* aleatórios, $e(n)$ letras Σ_n e $q(n)$ letras Φ_n, respectivamente, da fita de probabilidade τ e das pseudo–fitas de teste ϱ e de testemunha π. Fica assim claro que $R(n)$ ou $Q(n)$ sendo $\{0\}$ significa dizer que não podemos utilizar, respectivamente, a fita de probabilidade τ e a pseudo–fita de testemunha ϱ.

III.1–4. Diremos que um `teste` (isto é, uma Máquina de Turing) é $\boldsymbol{\alpha(n)}$–`seguro–(` $\boldsymbol{R(n), E(n), C_n \subseteq \Sigma_n^*, Q(n), \Phi_n, T(n)}$ `) para` $\boldsymbol{L}$ (ou ainda para L_n) sse ele é deste *modelo computacional* descrito acima e também tem um *esquema de provas* $\alpha(n)$*–robustas para* L *restrito a* C. Neste sentido, $C_n \subseteq \Sigma_n^*$ designa a codificação que está restrita à pseudo–fita de teste ϱ com alfabeto Σ_n, já Φ_n representa o alfabeto da pseudo–fita de testemunha π. Se $\alpha(n)$ é constante em zero, novamente omiti-lo-emos.

Observe que o *esquema de provas* $\alpha(n)$*–robustas* requer a existência da *taxa de erro da robustez* $\varepsilon \in (0,1)$ que independa de $n \in \mathbb{N}_*$. Mas como já foi dito no Parágrafo I.3–2, podemos fazê-la ser tão pequena quanto se queira, apenas repetindo o teste um número constante para n de vezes. O importante é lembrar que tais repetições não alteram a $\alpha(n)$*–segurança do teste*. Isto nos facilitará na composição de um número constante de testes, como queremos. Por outro lado, apesar de ε precisar independer de n, preocupar-nos-emos somente que ele independa de n's suficientemente grandes. Isto porque os nossos *modelos computacionais* têm as restrições que dependem de n dadas por $\mathcal{O}$ e Poli. Novamente, com o mesmo argumento utilizado no Parágrafo I.3–2, se estamos em um *modelo computacional*, podemos fazer pré-computações determinísticas para um número constante de n's, sem com isto sairmos deste mesmo *modelo computacional*.

Como mencionado no começo da seção, temos semanticamente dois tipos de *testes*: os *testes de proximidade da codificação* e os *testes de reconhecimento da codificação*. Nos `testes de proximidade da codificação` não haverá a restrição da codificação C_n na entrada da pseudo–fita de teste ϱ. Ou seja, sempre tomaremos $C_n := \Sigma_{n*}^* := (\Sigma_n \setminus \{\not\emptyset\})^*$, para todo $n \in \mathbb{N}_*$, e, portanto, o teste não é ajudado pela codificação. Além disto, buscaremos testes em que podemos tomar $\alpha(n)$ constante em $\alpha \in [0,1)$ tão pequeno quanto se queira. Já nos `testes de reconhecimento da codificação`, os $\alpha(n)$ serão constantes em zero. Com isto a resposta deve ser exata, não permitindo assim que o teste aceite palavras apenas $\alpha(n)$–próximas de L_n. Contudo, nos dois casos sempre *procuraremos* tomar $E(n)$ e $Q(n)$ constante em $\{1\}$ e minimizar o número de leituras de *bits aleatórios* (referente a $R(n)$) e o *tempo de decisão ponderado* (referente a $T(n)$). Portanto, o teste só poderá ler uma quantidade constante para n de letras Σ_n e Φ_n nas pseudo–fitas de teste ϱ e de testemunha π.

Assim, quando um *teste de reconhecimento da codificação* M_2 para L_n, com *taxa de erro da robustez* ε_2, necessita ter como entrada uma codificação C_n, utilizar-nos-emos de uma prévia computação de um *teste de proximidade da codificação* M_1, com *taxa de erro da robustez* ε_1, para atestarmos a codificação C_n. Note que esta composição de M_1 e M_2 não é perfeita, visto que M_1 aceita palavras $\alpha(n)$–próximos de C_n. Contudo, conforme a descrição anterior, o teste M_2 (que é um *teste de reconhecimento da codificação*) só irá ler um número constante e de letras da codificação e, então, cada letra tem a probabilidade $\alpha(n)$ de "ser lida errada" (i.e., $\alpha(n)$

é a probabilidade de se ler uma letra que não é da codificação de C_n). Lembre-se que $\alpha(n)$ em M_1 (que é um *teste de proximidade da codificação*) é constante em α e que pode ser tomado tão pequeno quanto se queira. Com efeito, a conjugação dos testes tem a *taxa de erro da robustez* em $e\alpha+\varepsilon_1+\varepsilon_2$ e podemos tomá-la constante e menor do que 1, como queremos. Temos assim uma descrição "bem por cima" de um *teste de reconhecimento da codificação* para L_n que não necessita que a entrada seja restrita à codificação C_n.

III.1–5. Só para ilustrar, começaremos dando três exemplos de *testes de reconhecimento da codificação*, os quais são uma decorrência imediata das *codificações* até agora vistas. Depois, na seção seguinte, falamos dos *testes de proximidade da codificação* e, finalizando, completaremos os *testes de reconhecimento da codificação* na última seção. Estes três testes *reconhecem uma* determinada *codificação*, ou, em outras palavras, reconhecem a igualdade de uma codificação fornecida com uma outra pré-determinada. Para tanto, no nosso caso, qual seja, de codificações *algébricas*, basta-nos saber verificar a *nulidade* das codificações. Relembremos que $\Psi_{(n)}$ é o alfabeto n-ário[viii] e comecemos pelos *funcionais lineares*,

Teste da Nulidade de Funcionais Lineares **T–I** (seguro–$\big(m(n)\log(\ell(n)), 1, C_n \subseteq \Psi^*_{(k(n))}, 0, \varnothing, 1\big)$). *Suporemos que $\ell(n) := \#H_n \geq 2$, para todo $n \in \mathbb{N}_*$.*

Entrada: *Conforme visto na Codificação por Funcional Linear* C–I*, seja, para todo $n \in \mathbb{N}_*$, $C_n \subseteq \Psi^*_{(k(n))}$ a codificação dos funcionais lineares f's restritos a H_n, em que $f(\xi) := \mathcal{F}_x(\xi) := \xi \cdot x^\perp$, para algum $x \in K_n^{m(n)}$, e f é fornecido* `ponto-a-ponto`*. Entendemos por uma função fornecida* ponto-a-ponto *a função fornecida na pseudo–fita pela seqüência $\big(\ f(u)\ \big)_{u\in H_n^{m(n)}} \subseteq K_n^{\ell(n)^{m(n)}}$. Na entrada é dado o número n na pseudo–fita de parâmetro de entrada e $f \in C_n$ na pseudo–fita de teste ϱ, que tem alfabeto $\Psi_{(k(n))}$.*

Teste: *O teste verifica se a entrada $f \in C_n$ é uma função nula (i.e., $f = 0$) com segurança–$\big(m(n)\log(\ell(n)), 1, C_n \subseteq \Psi^*_{(k(n))}, 0, \varnothing, 1\big)$. Perceba, ainda, que quando $\ell(n) := k(n) := 2$ é constante, temos o teste seguro–$\big(m(n), 1, C_n \subseteq \Psi^*_{(2)}, 0, \varnothing, 1\big)$.*

Testemunha: *Não há.*

Fase de Endereçamento:

- *Escolhe-se aleatoriamente $u \in H_n^{m(n)}$. Para isto, lêem-se $m(n)\log_2(\ell(n))$* bits *aleatórios.*
- *Fornece-se o endereço de $f(u)$ na pseudo–fita de teste à* fase de decisão.

Fase de Decisão:

- *Lê-se $f(u)$ em um único ponto na pseudo–fita de teste.*
- *Verifica-se se $f(u) = 0$, respondendo conforme o caso.*

Vimos que a Codificação por Funcional Linear C–I é uma $\frac{1}{\ell(n)}$–codificação.[ix] Como $\ell(n) \geq 2$, temos a $\frac{1}{2}$–codificação, ou seja, todas as suas palavras estão $\frac{1}{2}$–distantes.[ix] Portanto, se f não é nulo, isto é, $f := \mathcal{F}_x$, com $x \in H_n^{m(n)}$ não nulo, a probabilidade de tomarmos $u \in H_n^{m(n)}$, tal que $f(u) := \mathcal{F}_x(u) = 0$ é no máximo $1/2$. Estamos assim, como queremos, no *esquema de provas robustas*[x] e provamos que o teste é correto. Note que, neste teste, temos uma natural decorrência para um teste que verifique a igualdade entre

[viii] O alfabeto $\Psi_{(n)}$ tem n letras além do símbolo separador $\not\mid$.
[ix] Vide definição no Parágrafo II.2–1.
[x] Para maiores informações, leia o Parágrafo III.1–2.

dois funcionais lineares.

III.1–6. Quanto aos *polinômios*, tomando-se exatamente o mesmo algoritmo do teste anterior e com a mesma prova de corretude, definiremos dois outros *testes de reconhecimento da codificação*, mudando-se somente a codificação $C_n \subseteq \Sigma^*$ que restringe as palavras de entrada na pseudo–fita de teste ϱ. Vejamos primeiro o dos polinômios com o grau nas variáveis baixo,

Teste da Nulidade de Polinômios de Grau nas Variáveis Baixo T–II (seguro–$\big(m(n)\log(\ell(n)), 1, C_n \subseteq \Psi^*_{(k(n))}, 0, \varnothing, 1\big)$). *Suporemos que, para $n \in \mathbb{N}_*$ suficientemente grande, existe $\varepsilon \in (0,1)$, tal que $m(n)d(n)/\ell(n) \leq \varepsilon$.*

TESTE: *Conforme visto na Codificação Polinomial com Grau nas Variáveis Baixo* C–III ⟨*Segunda versão*⟩ *dos polinômios de grau nas variáveis baixo, seja, para todo $n \in \mathbb{N}_*$, $C_n \subseteq \Psi^*_{(k(n))}$ a codificação dos polinômios f's $m(n)$-ários com o domínio restrito a $H_n^{m(n)}$ e de grau nas variáveis no máximo $d(n)$ (i.e., em $f \in K_n[(\xi_1, \ldots, \xi_{m(n)}) \in H_n^{m(n)}]_{\langle d(n)\rangle}$), em que f é fornecido ponto-a-ponto. Então, o teste verifica se a entrada $f \in C_n$ é uma função nula (i.e., $f = 0$) com segurança–$\big(m(n)\log(\ell(n)), 1, C_n \subseteq \Psi^*_{(k(n))}, 0, \varnothing, 1\big)$.*

A sua descrição é a mesma do Teste da Nulidade de Funcionais Lineares T–I *(seguro–$\big(m(n)\log(\ell(n)), 1, C_n \subseteq \Psi^*_{(k(n))}, 0, \varnothing, 1\big)$).*

Se trocarmos os polinômios de grau nas variáveis baixo por polinômios de grau total baixo, teremos o teste análogo,

Teste da Nulidade de Polinômios de Grau Total Baixo T–III (seguro–$\big(m(n)\log(\ell(n)), 1, C_n \subseteq \Psi^*_{(k(n))}, 0, \varnothing, 1\big)$). Agora *suporemos que, para $n \in \mathbb{N}_*$ suficientemente grande, exista $\varepsilon \in (0,1)$, tal que $d(n)/\ell(n) \leq \varepsilon$.*

TESTE: *Conforme visto na Codificação Polinomial com Grau Total Baixo* C–V*, seja, para todo $n \in \mathbb{N}_*$, $C_n \subseteq \Psi^*_{(k(n))}$ a codificação dos polinômios f's $m(n)$-ários com o domínio restrito a $H_n^{m(n)}$ e de grau total no máximo $d(n)$ (i.e., em $f \in K_n[(\xi_1, \ldots, \xi_{m(n)}) \in H_n^{m(n)}]_{d(n)}$), em que f é fornecido ponto-a-ponto. Então, o teste verifica se a entrada $f \in C_n$ é uma função nula (i.e., $f = 0$), com segurança–$\big(m(n)\log(\ell(n)), 1, C_n \subseteq \Psi^*_{(k(n))}, 0, \varnothing, 1\big)$.*

A sua descrição é a mesma do Teste da Nulidade de Funcionais Lineares T–I *(seguro–$\big(m(n)\log(\ell(n)), 1, C_n \subseteq \Psi^*_{(k(n))}, 0, \varnothing, 1\big)$).*

III.2 Testes de Proximidade da Codificação

Nesta seção apresentaremos os *testes de proximidade da codificação* que necessitaremos. Eles serão tomados sobre as codificações que apresentamos no Capítulo II *Codificações Algébricas* e que, por sua vez, se baseiam em funcionais lineares e em polinômios. Procuraremos testes em que: o valor de segurança $\alpha(n)$ seja constante para n e tão pequeno quanto se queira; o número de leituras nas pseudo–fitas de teste ϱ e de testemunha π seja constante para n; e o número de leituras de *bits* aleatórios e o tempo de decisão ponderado sejam minimizados, conforme já foi discutido no Parágrafo III.1–4.

Daremos inicialmente os *testes de linearidade*. No primeiro deles, tomaremos uma função f de K^m em K e testaremos se ela é um funcional linear m-ário. Para tanto, somente para este caso, restringiremos $\ell := k := p$, ou seja, o corpo $H := K := \mathbb{Z}_p$ é de característica igual ao seu tamanho.

Teste de Linearidade T–IV (α–seguro–$\big(z(n)m(n)\log(p(n)), z(n), \Psi^*_{(p(n))}, 0, \varnothing, z(n)\log(p(n))\big)$**.** *Para cada $n \in \mathbb{N}_*$, sejam $\alpha(n) \in (\ 0, \min\left\{\frac{1}{3}; \frac{1}{p(n)}\right\}\ ]$ e $z(n) \geq 1/\log\left(\frac{p(n)}{p(n)-\alpha(n)}\right)$. Neste instante suporemos que temos $m(n)$, $p(n)$ e $z(n)$ calculáveis em tempo polinomial em n. Neste teste a pseudo–fita de teste ϱ tem alfabeto $\Psi_{(p(n))}$.*

Entrada: *É dado o número n na pseudo–fita de parâmetro de entrada e a função f de $\mathbb{Z}_{p(n)}^{m(n)}$ a $\mathbb{Z}_{p(n)}$ fornecida ponto-a-ponto na pseudo–fita de teste ϱ. Note que a pseudo–fita de teste tem alfabeto $\Psi_{(p(n))}$ e não é restrita a uma codificação.*

Teste: *Este teste é $\alpha(n)$–seguro–$\big(z(n)m(n)\log(p(n)), z(n), \Psi^*_{(p(n))}, 0, \varnothing, z(n)\log(p(n))\big)$ para funcionais lineares $m(n)$-ários sobre o corpo $\mathbb{Z}_{p(n)}$. Note agora que, se $p(n) := p$ é constante para n e $\alpha \in (\ 0, 1/(2p)\ ]$ também constante, temos um teste α–seguro–$(m(n), 1, \Psi_{(p)}, 0, \varnothing, 1)$. Na verdade, como queremos $\alpha(n)$ constante para n, p também o deverá ser.*

Testemunha: *Não há.*

Fase de Endereçamento:

- *Repete-se $z(n)$ vezes:*
  - *Escolhem-se aleatoriamente $u, v \in \mathbb{Z}_{p(n)}^{m(n)}$. Para isto, lêem-se $2m(n)\log_2(p(n))$* bits *aleatórios a cada vez.*
  - *Fornecem-se à* fase de decisão *os endereços $f(u)$, $f(v)$ e $f(u+v)$ na pseudo–fita de teste.*

Fase de Decisão:

- *Repete-se $z(n)$ vezes:*
  - *Lêem-se $f(u), f(v)$ e $f(u+v)$ (i.e., são três pontos a cada vez) na pseudo–fita de teste.*
  - *Testa-se se $f(u) + f(v) = f(u+v)$, respondendo* não*, quando for o caso.*
- *Responde-se* sim*.*

Primeiro definiremos por $\boldsymbol{K^{m\prime}}$ o conjunto dos *funcionais lineares* de K^m a K, portanto, o K–*espaço dual a* K^m. Para verificarmos este teste, veremos o

Teorema de Linearidade III.2–1. *Para $K := \mathbb{Z}_p$, $\alpha \in (\ 0, \min\left\{\frac{1}{3}; \frac{1}{p}\right\}\ ]$ e f de K^m em K,*

$$\text{se} \quad \Pr_{u,v \in K^m}\{\ f(u) + f(v) \neq f(u+v)\ \} < \frac{\alpha}{p} \quad \text{então} \quad f \in \mathfrak{V}_\alpha\left(K^{m\prime}\right).$$

Portanto, se f não está na α–vizinhança (i.e., não é α–próximo) de algum funcional linear em $K^{m\prime}$, então, para uma probabilidade sobre u e v de no mínimo α/p, temos $f(u) + f(v) \neq f(u+v)$.

III.2–2. Então, deste teorema checaremos a corretude do Teste de Linearidade T–IV. Seja ele M.

Para tanto, é de imediata observação que precisamos somente verificar que temos uma probabilidade constante de M aceitar uma função que não é $\alpha(n)$–próxima a um funcional linear em $\mathbb{Z}_{p(n)}^{m(n)'}$. Para $n \in \mathbb{N}_*$, suponha que $f \notin \mathcal{V}_{\alpha(n)}\left(\mathbb{Z}_{p(n)}^{m(n)'}\right)$, ou seja, $\mu(f, K^{m(n)'}) \geq \alpha(n)$ e, pelo teorema,

$$\mathcal{P}r\{\ M \text{ aceita } f, \text{ para } z(n) := 1\ \} \quad = \quad \underset{u,v \in \mathbb{Z}_{p(n)}^{m(n)}}{\mathcal{P}r}\{\ f(u) + f(v) = f(u+v)\ \} \quad \leq \quad 1 - \frac{\alpha(n)}{p(n)}.$$

Portanto, como as $z(n)$ repetições do teste são independentes,

$$\mathcal{P}r\{\ M \text{ aceita } f\ \} \quad \leq \quad \left(1 - \frac{\alpha(n)}{p(n)}\right)^{z(n)}.$$

Então, $z(n) \geq 1/\log\left(\frac{p(n)}{p(n)-\alpha(n)}\right)$, temos $-\frac{1}{z(n)} \geq -\log\left(\frac{p(n)}{p(n)-\alpha(n)}\right) = \log\left(1 - \frac{\alpha(n)}{p(n)}\right)$ e assim $1 - \frac{\alpha(n)}{p(n)} \leq \frac{1}{\mathsf{e}^{\frac{1}{z(n)}}}$. Logo

$$\mathcal{P}r\{\ M \text{ aceita } f\ \} \quad \leq \quad \left(1 - \frac{\alpha(n)}{p(n)}\right)^{z(n)} \quad \leq \quad \frac{1}{\mathsf{e}}.$$

E concluímos a verificação do teste.

Teste de Linearidade T–V (α–seguro–$(n, 1, \Psi_{(2)}, 0, \varnothing, 1)$)**.** *Conforme a Codificação por Funcional Linear* C–I*, sejam* $\ell(n) := k(n) := p(n) := 2$*, então* $H_n := K_n := \mathbb{Z}_2$ *e* $m(n) := n$*. Para* $\alpha(n) := \alpha \in (\ 0, 1/3\ ]$ *seja* $z(n) := z \geq 1/\log(2/(2-\alpha))$*, (logo constantes para* n*). Obtemos o teste* α*–seguro–*$(n, 1, \Psi_{(2)}, 0, \varnothing, 1)$ *que verifica a linearidade.*

Isto realmente é muito bom, pois além de termos a segurança α constante para n, o alfabeto da pseudo–fita de teste ϱ, também o é. Como nem tudo pode ser perfeito, note que queremos ler $\mathcal{O}(\log(n))$ *bits* aleatórios, o que não ocorre. Para tanto, utilizaremos mais tarde os polinômios. Antes, vamos à

Observação. *Como* $K = \mathbb{Z}_p$*, para* $f\colon K^m \to K$*,* $f \in K^{m\prime}$ *sse* $f(u) + f(v) = f(u+v)$*, para todo* $u, v \in K^m$*.*

Agora veremos a

Prova do Teorema de Linearidade III.2–1. Para $a \in K^m$ sejam $\widehat{f}(a) :=$ Maioria$\{\ f(a+u) - f(u) \parallel u \in K^m\ \}$,[xi] sendo que eventuais empates na maioria serão desempatados ao acaso, e $P_a := \mathcal{P}r_{u \in K^m}\left\{\ f(a+u) - f(u) = \widehat{f}(a)\ \right\}$. Note que, da definição de $\widehat{f}$ pela "maioria", $P_a \geq 1/p$. Usaremos este argumento abaixo.

Primeiro queremos provar que $\widehat{f}$ é linear. Para todo $a \in K^m$, como $a + u \in K^m$ é aleatório, se $u \in K^m$ também o for, pela hipótese

$$\underset{u,v \in K^m}{\mathcal{P}r}\{\ f(a+u) + f(v) \neq f(a+u+v)\ \} \quad < \quad \frac{\alpha}{p} \quad \leq \quad \frac{\alpha}{2}$$

e

$$\underset{u,v \in K^m}{\mathcal{P}r}\{\ f(u) + f(a+v) \neq f(a+u+v)\ \} \quad < \quad \frac{\alpha}{p} \quad \leq \quad \frac{\alpha}{2}.$$

[xi] A Maioria é a *moda estatística*, ou seja, o evento de maior ocorrência.

Logo,

$$\begin{aligned}
1-\alpha &< \Pr_{u,v\in K^m}\{\, f(a+u)+f(v)=f(a+u+v)=f(u)+f(a+v)\,\}\\
&= \Pr_{u,v\in K^m}\{\, f(a+u)-f(u)=f(a+u+v)-f(u)-f(v)=f(a+v)-f(v)\,\}\\
&\le \Pr_{u,v\in K^m}\{\, f(a+u)-f(u)=f(a+v)-f(v)\,\}\\
&\le \sum_{X\in K}\left(\Pr_{u\in K^m}\{\, f(a+u)-f(u)=X\,\}\right)^2\\
&= \left(\Pr_{u\in K^m}\left\{\, f(a+u)-f(u)=\widehat{f}(a)\,\right\}\right)^2+\sum_{X\in K\smallsetminus\{\widehat{f}(a)\}}\left(\Pr_{u\in K^m}\{\, f(a+u)-f(u)=X\,\}\right)^2\\
&\le P_a^2+\left(\sum_{X\in K\smallsetminus\{\widehat{f}(a)\}}\Pr_{u\in K^m}\{\, f(a+u)-f(u)=X\,\}\right)^2\\
&= P_a^2+\left(\Pr_{u\in K^m}\left\{\, f(a+u)-f(u)\neq\widehat{f}(a)\,\right\}\right)^2\\
&= P_a^2+(1-P_a)^2 \;\le\; \max\{P_a;1-P_a\}\,(P_a+(1-P_a)) \;=\; \max\{P_a;1-P_a\}\,.
\end{aligned}$$

Portanto, $1-\alpha < P_a$, ou $1-\alpha < 1-P_a$, (i.e., $\alpha > P_a$). Como da hipótese $\alpha \le 1/p \le P_a := \mathcal{P}r_{u\in K^m}\left\{\, f(a+u)-f(u)=\widehat{f}(a)\,\right\}$, temos $P_a > 1-\alpha$, para todo $a\in K^m$.

Com isto, para $a,b\in K^m$, lembrando que $a+u, b+u\in K^m$ também são aleatórios, se $u\in K^m$ o for, temos

$$\Pr_{u\in K^m}\left\{\, \widehat{f}(a)+\widehat{f}(b)+f(u)=f(a+u)+\widehat{f}(b)\,\right\} \;=\; P_a \;>\; 1-\alpha,$$

$$\Pr_{u\in K^m}\left\{\, f(a+u)+\widehat{f}(b)=f(a+u+b)\,\right\} \;=\; P_b \;>\; 1-\alpha$$

e

$$\Pr_{u\in K^m}\left\{\, f(a+u+b)=\widehat{f}(a+b)+f(u)\,\right\} \;=\; P_{a+b} \;>\; 1-\alpha.$$

Logo

$$\Pr_{u\in K^m}\left\{\, \widehat{f}(a)+\widehat{f}(b)=\widehat{f}(a+b)\,\right\} \;=\; \Pr_{u\in K^m}\left\{\, \widehat{f}(a)+\widehat{f}(b)+f(u)=\widehat{f}(a+b)+f(u)\,\right\} \;>\; 1-3\alpha \;\ge\; 0,$$

e, como independe de u e temos probabilidade positiva, $\widehat{f}(a)+\widehat{f}(b)=\widehat{f}(a+b)$, para todo $a,b\in K^m$. Então, pela observação acima, como queríamos, temos $\widehat{f}$ linear. Disto concluímos que $\mu(f,K^{m\prime})\le\mu(f,\widehat{f})$.

Falta-nos ver que f e $\widehat{f}$ são α–próximos. Suponha que não sejam, e teremos $\mathcal{P}r_{u\in K^m}\left\{\, f(u)\neq\widehat{f}(u)\,\right\}=\mu(f,\widehat{f})\ge\alpha$. Mas, para todo $u\in K^m$, $\mathcal{P}r_{v\in K^m}\left\{\, \widehat{f}(u)=f(u+v)-f(v)\,\right\}=P_u\ge 1/p$. Portanto, $\mathcal{P}r_{u,v\in K^m}\{\, f(u)\neq f(u+v)-f(v)\,\}\ge\alpha/p$, contrariando a hipótese. Então $f\in\mathfrak{V}_\alpha\left(K^{m\prime}\right)$.

[III.2–1] ■

Agora daremos dois tipos de *testes polinomiais*. Estes verificam a proximidade a polinômios e darão um pouco de trabalho para atestarmos as suas corretudes. Contudo, este será o tópico do último capítulo. Comecemos pelo

Teste Polinomial de Grau nas Variáveis Baixo T–VI (α–seguro–$\left(z(n)m(n)\log(\ell(n)),z(n),\Psi_{(k(n))},z(n),\Psi_{((d(n)+1)k(n))},z(n)d(n)\right)$)**.** *Para cada constante* $\alpha\in(\,0,1\,)$

suficientemente pequena, existe uma variável $z(n) \in \mathcal{O}(m(n)/\alpha)$ *que também é calculável em tempo polinomial em* n*. Vejamos*

ENTRADA: *É dado o número* n *na pseudo–fita de parâmetro de entrada e a função* f *de* $H_n^{m(n)}$ *a* K_n *fornecida ponto-a-ponto na pseudo–fita de teste* ϱ*. Note que a pseudo–fita de teste tem alfabeto* $\Psi_{(k(n))}$ *e não é restrita a uma codificação.*

TESTE: *O teste é* α*–seguro-*$\big(z(n)m(n)\log(\ell(n)), z(n), \Psi_{(k(n))}, z(n), \Psi_{((d(n)+1)k(n))}, z(n)d(n)\big)$ *para polinômios* $m(n)$*-ários com o domínio restrito a* $H_n^{m(n)}$ *e de grau nas variáveis no máximo* $d(n)$ *(i.e., em* $K_n[(\xi_1, \ldots, \xi_{m(n)}) \in H_n^{m(n)}]_{\langle d(n)\rangle}$*).*

TESTEMUNHA: *Para todo* $i \in \{1 \ldots m(n)\}$ *e* $\vec{u}^i := (u_1, \ldots, u_{i-1}, u_{i+1}, \ldots, u_{m(n)}) \in H_n^{m(n)-1}$*, a pseudo–fita de testemunha* π *deve conter em apenas uma letra os* $d(n)+1$ *coeficientes do polinômio unário de grau no máximo* $d(n)$*,* $g_{\vec{u}^i} \in K_n[\xi_i]_{d(n)}$*, que representaria a função* $f(u_1, \ldots, u_{i-1}, \xi_i, u_{i+1}, \ldots, u_{m(n)})$ *na variável* ξ_i*. Fica assim visto que a pseudo–fita de testemunha tem alfabeto* $\Psi_{((d(n)+1)k(n))}$*.*

FASE DE ENDEREÇAMENTO:

- *Fixa e fornecem-se à* fase de decisão $a_0, \ldots, a_{d(n)} \in H_n$ *distintos.*
- *Repete-se* $z(n)$ *vezes:*
  - *Escolhem-se aleatoriamente* $i \in \{1 \ldots m(n)\}$ *e* $\vec{u} := (u_1, \ldots, u_{m(n)}) \in H_n^{m(n)}$ *e fornece-se* u_i *à* fase de decisão*. Nisto, lêem-se* $\log_2(m(n)) + m(n)\log_2(\ell(n)) \in \mathcal{O}(m(n)\log(\ell(n)))$ bits *aleatórios a cada vez.*
  - *Fornece-se à* fase de decisão *o endereço de* $f(\vec{u})$ *na pseudo–fita de teste.*
  - *Fornece-se à* fase de decisão *o endereço dos coeficientes do polinômio* $g_{\vec{u}^i}(\xi_i)$ *na pseudo–fita de testemunha.*

FASE DE DECISÃO:

- *Repete-se* $z(n)$ *vezes:*
  - *Lê-se* $f(\vec{u})$ *(i.e., um ponto a cada vez) na pseudo–fita de teste.*
  - *Em tempo polinomial em* $d(n)$*, calcula-se* $g_{\vec{u}^i}(u_i)$ *através do polinômio* $g_{\vec{u}^i} \in K_n[\xi_i]_{d(n)}$ *fornecido (em um único ponto a cada vez) pelas* $d(n)+1$ *coordenadas na pseudo–fita de testemunha.*
  - *Testa-se se* $f(\vec{u}) = g_{\vec{u}^i}(u_i)$*, respondendo* NÃO*, se for o caso.*
- *Responde-se* SIM.

A sua corretude é devida a Friedl, Hátsági e Shen [FHS94] e será comentada no Capítulo V *Polinômios de Grau Baixo*. No entanto, perceba que nós precisaremos de algo um pouco mais forte, sendo que $z(n)$ seja constante (conjuntamente com α). De qualquer forma, veja

Teste Polinomial de Grau nas Variáveis Baixo T–VII (α–seguro–$\big(\log(n)^2, \log(n), \Psi_{\left(2^{\lceil \log(n) \rceil^2}\right)}, \log(n), \Psi_{\left(2^{\lceil \log(n) \rceil^3}\right)}, \log(n)\big)$)**.** *Pela Codificação Polinomial com Grau nas Va-*

riáveis Baixo C–IV*, para* $z := 1$*, podemos codificar palavras de* $\Psi_{(2)}^n$ *em palavras de comprimento* $\mathcal{O}(n^2\log(n)^2)$ *sobre o alfabeto* $\Psi_{\left(2\lceil\log(n)\rceil^2\right)}$*. Toma-se novamente* $d(n) := \lceil\log(n)\rceil - 1$*,* $m(n) := \lceil(d(n)+1)/\log(d(n)+1)\rceil$*,* $\ell(n) := (d(n)+1)^2$ *e* $k(n) := p(n)$ *um primo entre* $\ell(n)$ *e* $2\ell(n)$*.*[xii] *Portanto* $\#H_n =: \ell(n)$ *e* $K_n := \mathbb{Z}_{p(n)}$*. Para* α *e* $z(n) \in \mathcal{O}(m(n)/\alpha)$ *nas condições do Teste Polinomial de Grau nas Variáveis Baixo* T–VI *(*α*–seguro–*$\big(z(n)m(n)\log(\ell(n)), z(n), \Psi_{(k(n))}, z(n), \Psi_{((d(n)+1)k(n))}, z(n)d(n)\big)$*) acima, temos*

$$z(n) \;\in\; \mathcal{O}(m(n)) \;\subseteq\; \mathcal{O}\left(\left\lceil\frac{d(n)+1}{\log(d(n)+1)}\right\rceil\right) \;\subseteq\; \mathcal{O}\left(\frac{\lceil\log(n)\rceil}{\log\lceil\log(n)\rceil}\right) \;\subseteq\; \mathcal{O}(\log(n)).$$

Conforme visto na Codificação Polinomial com Grau nas Variáveis Baixo C–IV*,* $m(n)\log(\ell(n))$ *e* $d(n) \in \mathcal{O}(\log(n))$*. Logo* $z(n)m(n)\log(\ell(n)) \in \mathcal{O}(\log(n)^2)$ *e* $z(n)d(n) \in \mathcal{O}(\mathrm{Poli}(\log(n)))$*, (i.e., majorado poli–logaritmicamente em* n*). Portanto,*

TESTE: *O teste é* α*–seguro–*$\big(\log(n)^2, \log(n), \Psi_{\left(2\lceil\log(n)\rceil^2\right)}, \log(n), \Psi_{\left(2\lceil\log(n)\rceil^3\right)}, \log(n)\big)$ *para polinômios* $m(n)$*-ários com o domínio restrito a* $H_n^{m(n)}$ *e de grau nas variáveis no máximo* $d(n)$ *(i.e., em* $K_n[(\xi_1,\ldots,\xi_{m(n)}) \in H_n^{m(n)}]_{\langle d(n)\rangle}$*).*

Queremos observar duas coisas. Em primeiro lugar, no teste anterior, $z(n) \in \mathcal{O}(\log(n))$ é um fator das famílias $R(n)$, $E(n)$, $Q(n)$ e $T(n)$. Queremos reduzir $R(n)$ para $\{\log(n)\}$ e transformar $E(n)$, $Q(n)$ e $T(n)$ em $\{1\}$. Evidentemente $z(n)$ é o nosso maior problema(!) e transformando-o em constante para n, teremos o que queremos. Isto ocorrerá no próximo teste, mas a demonstração da sua corretude já é totalmente diferente, como era de se esperar.

A segunda observação decorre do fato que, no Teste Polinomial de Grau nas Variáveis Baixo T–VI, podemos facilmente dispensar a testemunha, passando assim $Q(n)$ para $\{0\}$. A perda seria em $E(n)$ que passaria a ser $\{z(n)(d(n)+1)\}$. Com isto, teríamos uma perda quadrática sobre o Teste Polinomial de Grau nas Variáveis Baixo T–VII, ou seja, $E(n) := \{\log(n)^2\}$. Como pode ser feito isto? É simples. Ao invés de lermos o polinômio $g_{\vec{u}^i} \in K_n[\xi_i]_{d(n)}$ na pseudo–fita de testemunha π, construímo-lo em tempo polinomial em $d(n)$, a partir da leitura de qualquer dos $d(n)+1$ pontos de $f(u_1,\ldots,u_{i-1},\xi_i,u_{i+1},\ldots,u_{m(n)})$ na pseudo–fita de teste ϱ. Talvez o teste desta maneira ficasse melhor, mas perdemos a propriedade da primeira observação, qual seja, fazendo-se $z(n)$ constante para n, ainda assim $E(n)$ não seria constante. Façamos, então, $z(n)$ constante.

Teste Polinomial de Grau Total Baixo T–VIII (α–seguro–$\big((2m(n)+1)\log(k(n)), 1, \Psi_{(k(n))}, 1, \Psi_{((d(n)+1)k(n))}, d(n)\big)$**.** *Assumiremos que neste teste os domínios das funções não são restritos, assim,* $\ell(n) = k(n)$ *e* $H_n = K_n$*. Ainda, para todo* $n \in \mathbb{N}_*$ *suficientemente grande, temos* $m(n) \geq 2$*,* $d(n) \geq 4$*,* $k(n) \geq \max\{2^{27}3^75^5; 132d(n)^3\}$ *e* $p(n) \geq 2d(n)+1$*. Então, para cada constante* $\alpha \in (\,0, 1/2^{28}3^75^5\,]$*, existe uma outra constante* $z \in \Omega(1/\alpha)$*, tal que*

ENTRADA: *É dado o número* n *na pseudo–fita de parâmetro de entrada e a função* f *de* $K_n^{m(n)}$ *a* K_n *fornecida ponto-a-ponto na pseudo–fita de teste* ϱ*. Note que a pseudo–fita de teste tem alfabeto* $\Psi_{(k(n))}$ *e não é restrita a uma codificação.*

TESTE: *O teste é* α*–seguro–*$\big((2m(n)+1)\log(k(n)), 1, \Psi_{(k(n))}, 1, \Psi_{((d(n)+1)k(n))}, d(n)\big)$ *para polinômios* $m(n)$*-ários de grau total no máximo* $d(n)$ *(i.e., em* $K_n[\xi_1,\ldots,\xi_{m(n)}]_{d(n)}$*).*

TESTEMUNHA: *Para todo* $u, v \in K_n^{m(n)}$*, a pseudo–fita de testemunha* π *deve conter em apenas uma letra os* $d(n)+1$ *coeficientes do polinômio* $g_{u,v} \in K_n[\zeta]_{d(n)}$ *de grau no máximo* $d(n)$*, que representaria a função* $f(\zeta u + v)$*. Note que a pseudo–fita de testemunha tem alfabeto* $\Psi_{((d(n)+1)k(n))}$*.*

[xii] Veja que sempre existe um primo entre um número natural e o seu dobro e, nestes caso, ele pode ser achado em tempo polinomial em n.

FASE DE ENDEREÇAMENTO:

- *Repete-se z vezes:*
  - *Escolhem-se aleatoriamente* $u, v \in K_n^{m(n)}$ *e* $l \in K_n$. *Para isto, lêem-se* $(2m(n)+1)\log(k(n))$ bits *aleatórios a cada vez.*
  - *Fornece-se à* fase de decisão *o endereço de* $f(lu+v)$ *na pseudo–fita de teste.*
  - *Fornece-se à* fase de decisão *o endereço na pseudo–fita de testemunha dos* $d(n)+1$ *coeficientes de* $g_{u,v}(\zeta)$.

FASE DE DECISÃO:

- *Repete-se z vezes:*
  - *Lê-se* $f(lu+v)$ *na pseudo–fita de teste em um único ponto a cada vez.*
  - *Em tempo polinomial em* $d(n)$, *calcula-se* $g_{u,v}(l)$ *através do polinômio* $g_{u,v} \in K_n[\zeta]_{d(n)}$ *fornecido (em um ponto só a cada vez) pelas* $d(n)+1$ *coordenadas na pseudo–fita de testemunha.*
  - *Testa-se se* $f(lu+v) = g_{u,v}(l)$, *respondendo* NÃO, *se for o caso.*
- *Responde-se* SIM.

A corretude do teste será vista no Capítulo V *Polinômios de Grau Baixo* e a nossa prova se deve a Arora [Aro94], mas ela também se baseará em Sudan [Sud92]. Todavia, cumpre-nos lembrar que ela, na verdade, decorre de Arora, Lund, Motwani, Sudan e Szegedy [ALM+92] e que, por sua vez, seguiram a de Arora e Safra [AS92].

Na Codificação Polinomial com Grau Total no Máximo 1 C–VI ⟨Primeira versão⟩ precisaríamos tomar $m(n) := n-1$ e novamente leríamos $\mathcal{O}(n)$ *bits* aleatórios. Já na Codificação Polinomial com Grau Total no Máximo 1 C–VII ⟨Segunda versão⟩, para cada um dos $\mathcal{O}(n/\log(n))$ blocos, poderíamos usar o Teste Polinomial de Grau Total Baixo T–VIII, sendo que para cada bloco temos um bom comportamento; no entanto, ao compô-los o teste fica pior do que o teste anterior. Portanto veja

Teste Polinomial de Grau Total Baixo T–IX (α–seguro–$\big(\log(n), 1, \Psi_{(8\lceil\log(n)\rceil^6)}, 1, \Psi_{(8\lceil\log(n)\rceil^8)}, \log(n)\big)$)**.** *Nele utilizamos a Codificação Polinomial com Grau Total Baixo* C–VIII, *que codifica palavras de* $\Psi_{(2)}^n$ *em palavras de comprimento polinomial em n sobre o alfabeto* $\Psi_{(8\lceil\log(n)\rceil^6)}$, *(pois tomamos* $z := 3$*). Portanto, sejam* $s(n) := \lceil\log(n)\rceil - 1$, $m(n) := \lceil (s(n)+1)/\log(s(n)+1)\rceil$, $p(n)$ *um primo entre* $(s(n)+1)^2$ *e* $2(s(n)+1)^2$, $\ell(n) := k(n) := p(n)^3$ *e* $d(n) := m(n)s(n)$.

É imediato vermos que, para n suficientemente grande, temos $m(n) \geq 2$, $d(n) \geq 4$, $p(n) \geq (s(n)+1)^2 \geq 2d(n)+1$ *e* $k(n) := p(n)^3 \geq \max\{2^{27}3^75^5; 132d(n)^3\}$. *Ainda, vimos na Codificação Polinomial com Grau Total Baixo* C–VIII *que* $2(m(n)+1)\log(k(n)) \in \mathcal{O}(\log(n))$, $d(n) \in \mathcal{O}(\mathrm{Poli}(\log(n)))$ *e* $(d(n)+1)k(n) \leq (s(n)\lceil(s(n)+1)/(\log(s(n)+1))\rceil + 1)\,8(s(n)+1)^6 \leq (s(n)+1)^2 8(s(n)+1)^6 = 8\lceil\log(n)\rceil^8$. *Logo, para* $\alpha \in (\,0, 1/2^{28}3^75^5\,]$, *pelo Teste Polinomial de Grau Total Baixo* T–VIII *(*α*–seguro–*$\big((2m(n)+1)\log(k(n)), 1, \Psi_{(k(n))}, 1, \Psi_{((d(n)+1)k(n))}, d(n)\big)$*), temos*

TESTE: *O teste é* α*–seguro–*$\big(\log(n), 1, \Psi_{(8\lceil\log(n)\rceil^6)}, 1, \Psi_{(8\lceil\log(n)\rceil^8)}, \log(n)\big)$ *para polinômios* $m(n)$*-ários de grau*

total no máximo $d(n)$ (i.e., em $K_n[\xi_1,\ldots,\xi_{m(n)}]_{d(n)}$).

III.3 Testes de Reconhecimento da Codificação

Voltando aos *testes de reconhecimento da codificação*. Assim como nos Testes de Linearidade T–V, Polinomial de Grau nas Variáveis Baixo T–VI e Polinomial de Grau Total Baixo T–VIII, tomaremos testes em que: o valor de segurança $\alpha(n)$ seja constante em zero; o número de leituras nas pseudo–fitas de teste ϱ e de testemunha π sejam constantes para n; e o número de leituras de *bits* aleatórios e o tempo de decisão ponderado sejam minimizados; conforme foi discutido no Parágrafo III.1–4. Começaremos pelo reconhecimento de *produtos tensoriais* de funcionais lineares.

III.3–1. Sejam $m, w \in \mathbb{N}_*$, as variáveis $\vec{\xi} := (\xi_1,\ldots,\xi_m)$ e $\vec{\zeta} := (\zeta_1,\ldots,\zeta_w)$ e os vetores $u := (u_1,\ldots,u_m)$, $a := (a_1,\ldots,a_m) \in K^m$ e $v := (v_1,\ldots,v_w)$, $b := (b_1,\ldots,b_w) \in K^w$. Então definimos o `produto tensorial`

$$\big(\boldsymbol{\xi_i}\big)_{\boldsymbol{i\in\{1\ldots m\}}} \otimes \big(\boldsymbol{\zeta_j}\big)_{\boldsymbol{j\in\{1\ldots w\}}} \quad := \quad \big(\xi_i\zeta_j\big)_{(i,j)\in\{1\ldots m\}\times\{1\ldots w\}} \quad \in \quad K^{mw},$$

e aqui, pensamos $\big(\xi_i\zeta_j\big)_{(i,j)}$ como um vetor de mw entradas e que tenha m `blocos` (ou linhas), cada um de tamanho w (representando as colunas). Se temos o vetor $c := \big(c_{i,j}\big)_{(i,j)\in\{1\ldots m\}\times\{1\ldots w\}} \in K^{mw}$, $I \subseteq \{1\ldots m\}$ e $J \subseteq \{1\ldots w\}$, então

$$\boldsymbol{\mathcal{L}^1_I(c)} \quad := \quad \boldsymbol{\mathcal{L}^1_I\big(c_{i,j}\big)_{(i,j)}} \quad := \quad \big(c_{i,j}\big)_{(i\in I\ ,\ j\in\{1\ldots w\})}$$

é ter c restrito aos i-ésimos blocos (ou linhas), com $i \in I$, e

$$\boldsymbol{\mathcal{L}^2_J(c)} \quad := \quad \boldsymbol{\mathcal{L}^2_J\big(c_{i,j}\big)_{(i,j)}} \quad := \quad \big(c_{i,j}\big)_{(i\in\{1\ldots m\}\ ,\ j\in J)}$$

são as $(j \in J)$-ésimas colunas de c.

Lembremos que na Codificação por Funcional Linear C–I definimos o *funcional linear* através do *produto interno* $\mathcal{F}_a(\vec{\xi}) := \vec{\xi}\cdot a^\perp$. Disto temos as propriedades que usaremos logo abaixo

$$\mathcal{F}_c(\vec{\xi}\otimes\vec{\zeta}) \quad = \quad \mathcal{F}_{\left(\mathcal{F}_{\mathcal{L}^1_{\{i\}}(c)}(\vec{\zeta})\right)_i}(\vec{\xi})$$

e

$$\mathcal{F}_{a\otimes b}(\vec{\xi}\otimes\vec{\zeta}) \quad = \quad \mathcal{F}_a(\vec{\xi})\mathcal{F}_b(\vec{\zeta}).$$

Portanto, se $c \neq 0$, existe $i \in \{1\ldots m\}$, tal que o i-ésimo bloco não é nulo, i.e. $\mathcal{L}^1_{\{i\}}(c) \neq 0$. Assim, dado que $k := \#K$ e como visto na Codificação por Funcional Linear C–I,

$$\Pr_{v\in K^w}\left\{\ \mathcal{F}_{\mathcal{L}^1_{\{i\}}(c)}(v) = 0\ \right\} \quad = \quad \frac{1}{k}$$

logo,

$$\Pr_{v,v'\in K^w}\left\{\ \mathcal{F}_{\mathcal{L}^1_{\{i\}}(c)}(v) = 0 \quad \text{e} \quad \mathcal{F}_{\mathcal{L}^1_{\{i\}}(c)}(v') = 0\ \right\} \quad = \quad \frac{1}{k^2}.$$

Mas, se $\mathcal{F}_{\mathcal{L}^1_{\{i\}}(c)}(v) \neq 0$, então

$$\Pr_{u,u'\in K^m}\left\{\ \mathcal{F}_{\left(\mathcal{F}_{\mathcal{L}^1_{\{i\}}(c)}(v)\right)_i}(u) = 0 \quad \text{e} \quad \mathcal{F}_{\left(\mathcal{F}_{\mathcal{L}^1_{\{i\}}(c)}(v)\right)_i}(u') = 0\ \right\} \quad = \quad \frac{1}{k^2},$$

e pela primeira propriedade acima, concluímos que

$$\mathop{\mathcal{P}r}_{\substack{u,u'\in K^m\\ v,v'\in K^w}} \{\ \mathcal{F}_c(u\otimes v)=0 \quad \text{e} \quad \mathcal{F}_c(u'\otimes v)=0 \quad \text{e} \quad \mathcal{F}_c(u\otimes v')=0 \quad \text{e} \quad \mathcal{F}_c(u'\otimes v')=0\ \} \quad \leq \quad \frac{2}{k^2} \quad \leq \quad \frac{1}{2}.$$

Note agora que, se queremos testar se $a\otimes b=c$, pela segunda propriedade acima, temos

$$\mathcal{F}_a(\vec{\xi})\mathcal{F}_b(\vec{\zeta})-\mathcal{F}_c(\vec{\xi}\otimes\vec{\zeta}) \quad = \quad \mathcal{F}_{a\otimes b-c}(\vec{\xi}\otimes\vec{\zeta}),$$

e podemos, dados u, u', v e v', verificar se a equação é igual a zero. Este teste é feito conhecendo-se apenas os valores dos funcionais lineares nos 12 pontos definidos por u, u', v e v'. Portanto, se $c\neq a\otimes b$, temos no máximo a probabilidade $1/2$ de sermos enganados. Estamos, assim, no *esquema de provas robustas*[xiii] e provamos a corretude do teste

Teste do Produto Tensorial T–X (seguro–$\big((m(n)+w(n))\log(k(n)),1,C_n\subseteq\Psi^*_{(k(n))},0,\varnothing,1\big)$**.** *Sejam funções naturais* $m(n),w(n),k(n)\in\mathcal{O}(\text{Poli}(n))$. *Então*

ENTRADA: *Seja, para todo* $n\in\mathbb{N}_*$, $C_n\subseteq\Psi^*_{(k(n))}$ *a codificação de três funcionais lineares sobre* K_n, $f_a:=\mathcal{F}_a$, $f_b:=\mathcal{F}_b$ *e* $f_c:=\mathcal{F}_c$, *respectivamente,* $m(n)$, $w(n)$ *e* $m(n)w(n)$*-ários, de tal forma que também sejam fornecidos ponto-a-ponto na pseudo–fita de teste. Logo, são* $a:=(a_1,\ldots,a_{m(n)})\in K_n^{m(n)}$, $b:=(b_1,\ldots,b_{w(n)})\in K_n^{w(n)}$ *e* $c:=(c_1,\ldots,c_{m(n)w(n)})\in K_n^{m(n)w(n)}$ *e assumimos que* n, $m(n)$ *e* $w(n)$ *também são fornecidos na codificação* C_n. *É dado o número* n *na pseudo–fita de parâmetro de entrada e* $f_a,f_b,f_c\in C_n$ *na pseudo–fita de teste* ϱ, *que tem alfabeto* $\Psi_{(k(n))}$.

TESTE: *Este teste verifica se dadas como entrada as funções* f_a, f_b *e* f_c *da codificação* C_n, *temos o* produto tensorial $c=a\otimes b$, *com segurança–*$\big((m(n)+w(n))\log(k(n)),1,C_n\subseteq\Psi^*_{(k(n))},0,\varnothing,1\big)$. *Note que, quando temos* $k(n):=2$ *constante, o teste é seguro–*$\big(m(n)+w(n),1,C_n\subseteq\Psi^*_{(2)},0,\varnothing,1\big)$.

FASE DE ENDEREÇAMENTO:

- *Sorteiam-se* $u,u'\in K_n^{m(n)}$ *e* $v,v'\in K_n^{w(n)}$, *com* $2(m(n)+w(n))\log_2(k(n))$ bits *aleatórios.*
- *Calculam-se* $u\otimes v$, $u'\otimes v$, $u\otimes v'$ *e* $u'\otimes v'$, *em tempo polinomial para* n.
- *Fornecem-se à* fase de decisão *os endereços na pseudo–fita de teste de* $f_a(u)$, $f_a(u')$, $f_a(v)$, $f_a(v')$, $f_b(u)$, $f_b(u')$, $f_b(v)$, $f_b(v')$, $f_c(u\otimes v)$, $f_c(u'\otimes v)$, $f_c(u\otimes v')$ *e* $f_c(u'\otimes v')$.

FASE DE DECISÃO:

- *Lêem-se os 12 pontos da pseudo–fita de teste* ϱ, *através dos endereços fornecidos pela* fase de endereçamento.
- *Calcula-se em tempo constante,* $f_a(u)f_b(v)-f_c(u\otimes v)$, $f_a(u')f_b(v)-f_c(u'\otimes v)$, $f_a(u)f_b(v')-f_c(u\otimes v')$ *e* $f_a(u')f_b(v')-f_c(u'\otimes v')$.
- *Verificam-se se todos são iguais a zero, respondendo conforme o caso.*

III.3–2. Mudando de assunto, a partir de agora procuraremos chegar ao Teste da Nulidade de Polinômios de Grau nas Variáveis Baixo sobre Domínio Restrito T–XIV que, sobre uma conveniente codificação,

[xiii] Vide Parágrafo III.1–2.

verifica se uma função é nula em um domínio restrito. Para tanto, se $J \subseteq H^m$ e f é uma função de H^m a K, então denotaremos $\boldsymbol{\mathcal{S}_J(f)} := \sum_{u \in J} f(u)$. Primeiro queremos testar se $\mathcal{S}_J(f)$ tem um valor pré-determinado. Comecemos supondo f de uma variável. Lembremos que $S_n \subseteq H_n$ e $s(n) := \#S_n - 1$, para $n \in \mathbb{N}_*$, e também que são computáveis em tempo polinomial em n.

Teste da Soma sobre Polinômios de Uma Variável e de Grau Baixo T–XI (seguro–$\big(\log(\ell(n)), 1, C_n \subseteq \Psi^*_{(k(n))}, 1, \Psi_{((d(n)+1)k(n))}, \{d(n), s(n)\}\big)$). *Suporemos que, para $n \in \mathbb{N}_*$ suficientemente grande, existe $\varepsilon \in (0,1)$, tal que $d(n)/\ell(n) \leq \varepsilon$. Note que tomaremos $m(n) := 1$, para todo $n \in \mathbb{N}_*$.*

Entrada: *Seja, para todo $n \in \mathbb{N}_*$, $C_n \subseteq \Psi^*_{(k(n))}$ a codificação dos polinômios f's unários com o domínio restrito a H_n e de grau no máximo $d(n)$ (i.e., em $f \in K_n[\xi \in H_n]_{d(n)}$) fornecidos ponto-a-ponto. É dado o número n na pseudo–fita de parâmetro de entrada e $f \in C_n$ na pseudo–fita de teste ϱ, que tem alfabeto $\Psi_{(k(n))}$.*

Teste: *Este teste verifica se a função natural de entrada $f \in C_n$ tem soma $c_n \in K_n$ em S_n (i.e., $\mathcal{S}_{S_n}(f) = c_n$), com segurança–$\big(\log(\ell(n)), 1, C_n \subseteq \Psi^*_{(k(n))}, 1, \Psi_{((d(n)+1)k(n))}, \{d(n), s(n)\}\big)$.*

Testemunha: *A pseudo–fita de testemunha π deve conter os $d(n)+1$ coeficientes do polinômio unário de grau no máximo $d(n)$, $g \in K_n[\xi]_{d(n)}$, que representaria a função $f(\xi)$. Portanto, a pseudo–fita de testemunha tem alfabeto $\Psi_{((d(n)+1)k(n))}$ e leremos somente uma letra.*

Fase de Endereçamento:

- *Sorteia-se $u \in H_n$ (com* $\log_2(\ell(n))$ bits *aleatórios).*
- *Fornecem-se à* fase de decisão *o endereço na pseudo–fita de teste de $f(u)$ e o endereço na pseudo–fita de testemunha dos $d(n)+1$ coeficientes de g.*

Fase de Decisão:

- *Calcula-se $\mathcal{S}_{S_n}(g)$, em tempo polinomial em $d(n)$ e $s(n)$, com a leitura dos $d(n)+1$ coeficientes de $g \in K_n[\xi]_{d(n)}$ na pseudo–fita de testemunha. Note que, para isto, lemos só uma letra.*
- *Verifica-se se $\mathcal{S}_{S_n}(g) = c_n$, respondendo* não, *se for o caso.*
- *Lê-se $f(u)$ na pseudo–fita de teste (também, em um único ponto).*
- *Calcula-se $g(u)$ em tempo polinomial em $d(n)$.*
- *Compara-se se $g(u) = f(u)$, respondendo conforme o caso.*

Note que podemos dispensar a testemunha, mas para isto leríamos $d(n)+1$ letras na pseudo–fita de teste, o que não nos ajuda muito, pois perdemos o número de leituras constante para n. Precisamos agora observar que o teste só dará errado se a testemunha g não for uma prova, isto é, $g \neq f$. Como obrigatoriamente $g, f \in K_n[\xi \in H_n]_{d(n)}$, pelo Corolário da Distância dos Polinômios de Grau Baixo II.3–3, temos a probabilidade $d(n)/\ell(n) \leq \varepsilon$ de não detectarmos pelo teste quando $g \neq f$. Passemos, então, para mais variáveis.

Teste da Soma sobre Polinômios de Grau nas Variáveis Baixo T–XII (seguro–$\big(m(n)\log(\ell(n)), 1, C_n \subseteq \Psi^*_{(k(n))}, 1, \Psi_{((d(n)+1)m(n)k(n))}, \{m(n)d(n), m(n)s(n)\}\big)$). *Agora suporemos que, para $n \in \mathbb{N}_*$ suficientemente grande, existe $\varepsilon \in (0,1)$, tal que $m(n)d(n)/\ell(n) \leq \varepsilon$.*

ENTRADA: *Seja, para todo $n \in \mathbb{N}_*$, $C_n \subseteq \Psi^*_{(k(n))}$ a codificação dos polinômios f's $m(n)$-ários com o domínio restrito a H_n e de grau nas variáveis no máximo $d(n)$ (i.e., em $f \in K_n[(\xi_1, \ldots, \xi_{m(n)}) \in H_n^{m(n)}]_{\langle d(n)\rangle}$), fornecidos ponto-a-ponto. É dado o número n na pseudo–fita de parâmetro de entrada e $f \in C_n$ na pseudo–fita de teste ϱ, que tem alfabeto $\Psi_{(k(n))}$.*

TESTE: *Este teste verifica se a função natural de entrada $f \in C_n$ tem soma $c_n \in K_n$ em $S_n^{m(n)}$ (i.e., $\mathcal{S}_{S_n^{m(n)}}(f) = c_n$) com segurança–$\big(m(n)\log(\ell(n)), 1, C_n \subseteq \Psi^*_{(k(n))}, 1, \Psi_{((d(n)+1)m(n)k(n))}, \{m(n)d(n), m(n)s(n)\}\big)$.*

TESTEMUNHA: *A pseudo–fita de testemunha π tem alfabeto $\Psi_{((d(n)+1)m(n)k(n))}$ e leremos somente um número constantes em n de letras dela. Ela deve conter, para todo $(u_1, \ldots, u_{m(n)}) \in H_n^{m(n)}$, em apenas uma letra, os $d(n)+1$ coeficientes dos polinômios unários de grau no máximo $d(n)$, $g_{u_1,\ldots,u_{i-1}} \in K_n[\xi_i]_{d(n)}$, para todos os $i \in \{1 \ldots m(n)\}$. Eles representariam*

$$\begin{aligned} c_n &= \mathcal{S}_{S_n}(g) := \sum_{u \in S_n} g(u), \\ g_{u_1,\ldots,u_{i-1}}(u_i) &= \mathcal{S}_{S_n}(g_{u_1,\ldots,u_i}) := \sum_{u \in S_n} g_{u_1,\ldots,u_i}(u) \quad e \\ f(u_1, \ldots, u_{m(n)}) &= g_{u_1,\ldots,u_{m(n)-1}}(u_{m(n)}) \end{aligned}$$

FASE DE ENDEREÇAMENTO:

- *Sorteia-se $(u_1, \ldots, u_{m(n)}) \in H_n^{m(n)}$, com $m(n)\log_2(\ell(n))$* bits *aleatórios.*
- *Fornece-se à* fase de decisão *o endereço na pseudo–fita de teste de $f(u_1, \ldots, u_{m(n)})$ e o endereço na pseudo–fita de testemunha dos $d(n)+1$ coeficientes dos $g_{u_1,\ldots,u_{i-1}}$, para cada $i \in \{1 \ldots m(n)\}$.*

FASE DE DECISÃO:

- *Lêem-se todos os polinômios $g_{u_1,\ldots,u_{i-1}} \in K_n[\xi_i]_{d(n)}$, para todo $i \in \{1 \ldots m(n)\}$, em apenas um ponto.*
- *Calcula-se $\mathcal{S}_{S_n}(g)$ em tempo polinomial em $d(n)$ e $s(n)$.*
- *Verifica-se se $c_n = \mathcal{S}_{S_n}(g)$, respondendo* NÃO, *se for o caso.*
- *Para $i \in \{1 \ldots m(n)-1\}$ repete-se (i.e., $m(n)-1$ vezes):*
  - *Calcula-se $\mathcal{S}_{S_n}(g_{u_1,\ldots,u_i})$ em tempo polinomial em $d(n)$ e $s(n)$ (a cada vez).*
  - *Calcula-se $g_{u_1,\ldots,u_{i-1}}(u_i)$ em tempo polinomial em $d(n)$ (a cada vez).*
  - *Compara-se se $g_{u_1,\ldots,u_{i-1}}(u_i) = \mathcal{S}_{S_n}(g_{u_1,\ldots,u_i})$, respondendo* NÃO, *se for o caso.*
- *Lê-se $f(u_1, \ldots, u_{m(n)})$ na pseudo–fita de teste (também, em um único ponto).*
- *Calcula-se $g_{u_1,\ldots,u_{m(n)-1}}(u_{m(n)})$ em tempo polinomial em $d(n)$.*
- *Compara-se se $f(u_1, \ldots, u_{m(n)}) = g_{u_1,\ldots,u_{m(n)-1}}(u_{m(n)})$, respondendo conforme o caso.*

Note que, na verdade, realizamos $m(n)$ vezes quase o mesmo Teste da Soma sobre Polinômios de

Uma Variável e de Grau Baixo T–XI para uma variável. Se $\mathcal{S}_{S_n^{m(n)}}(f) = c_n$, é imediato que sempre temos uma testemunha correta (isto é, uma *prova*) que faz o teste sempre aceitar, independente de qual fita de probabilidade τ é tomada. Caso contrário, se $\mathcal{S}_{S_n^{m(n)}}(f) \neq c_n$ e sendo aceito pelo teste, em alguma das $m(n)$ repetições do teste de uma variável houve um aceite *errado*. Como visto, cada uma das repetições tem a probabilidade $d(n)/\ell(n)$ de *errar*. Logo, no total, temos a probabilidade de $m(n)d(n)/\ell(n) \leq \varepsilon$ de cometermos o *erro* de aceitar $\mathcal{S}_{S_n^{m(n)}}(f) \neq c_n$. Como $\varepsilon \in (0,1)$ é constante para n, estamos no *esquema de provas robustas*,[xiv] como queríamos.

III.3–3. Sejam $\vec{\xi} := (\xi_1, \ldots, \xi_m)$ e $\vec{\zeta} := (\zeta_1, \ldots, \zeta_m)$. Se $f(\vec{\xi}) \in K[\vec{\xi} \in H^m]_{\langle d \rangle}$ é um polinômio m-ário com o domínio restrito a H^m e de grau nas variáveis no máximo d, queremos testar se ele, restrito a $S^m \subseteq H^m$, é uma função nula (i.e., $f \restriction_{S^m} = 0$). Como já comentado, se $s := \#S - 1 \geq d$, então nos basta verificar se f inteiro é uma função nula, com o Teste da Nulidade de Polinômios de Grau nas Variáveis Baixo T–II. Portanto, sem perda de generalidade, suporemos $s < d$ e $S \subset D := \{0,\ 1,\ 1+1, \ldots, d\} \subseteq H$. Comecemos definindo a função

$$\boldsymbol{I_{\vec{\zeta}}(\vec{\xi})} \colon H^{2m} \to K,$$

que tem variáveis $\vec{\xi}$ e $\vec{\zeta}$ e é tal que, para todo $\vec{u} := (u_1, \ldots, u_m) \in H^m$, então $I_{\vec{u}}(\vec{\xi}) \in K[\vec{\xi} \in H^m]_{\langle d \rangle}$ é a *Interpolação de Lagrange*[xv] dos pontos

$$\left(\vec{i}\ ,\ \prod_{j \in \{1 \ldots m\}} u_j^{i_j} \right)_{\vec{i} := (i_1, \ldots, i_m) \in D^m}.$$

Note-se que operando sobre o alfabeto k-ário $\Psi_{(k)}$, podemos calcular $I_{\vec{u}}(\vec{\xi})$ em tempo polinomial em m e d. Nela obtemos a propriedade que, para todo $\vec{i} \in D^m$, temos

$$I_{\vec{\zeta}}(\vec{i}) \quad = \quad \prod_{j \in \{1 \ldots m\}} \zeta_j^{i_j} \quad \in \quad K[\vec{\zeta} \in H^m]_{\langle d \rangle}$$

e definimos

$$\boldsymbol{f'_{\vec{\zeta}}(\vec{\xi})} \quad := \quad f(\vec{\xi}) I_{\vec{\zeta}}(\vec{\xi}).$$

Portanto, para todo $\vec{i} \in S^m \subseteq D^m$ e $\vec{u} \in H^m$, temos

$$f'_{\vec{\zeta}}(\vec{i}) \quad := \quad f(\vec{i}) I_{\vec{\zeta}}(\vec{i}) \quad \in \quad K[\vec{\zeta} \in H^m]_{\langle d \rangle}$$

e

$$f'_{\vec{u}}(\vec{\xi}) \quad := \quad f(\vec{\xi}) I_{\vec{u}}(\vec{\xi}) \quad \in \quad K[\vec{\xi} \in H^m]_{\langle 2d \rangle}.$$

Definimos também

$$\boldsymbol{f''(\vec{\zeta})} \quad := \quad \mathcal{S}_{S^m}(f'_{\vec{\zeta}}) \quad := \quad \sum_{\vec{i} \in S^m} f'_{\vec{\zeta}}(\vec{i}) \quad \in \quad K[\vec{\zeta} \in H^m]_{\langle d \rangle}.$$

Disto tudo, concluimos que $f \restriction_{S^m} = 0$ sse $f'' = 0$. Veja que f'' é um polinômio m-ário restrito a H^m e de grau nas variáveis no máximo d e, portanto, utilizamos aqui o Teste da Nulidade de Polinômios de Grau nas Variáveis Baixo T–II. Lembre-se agora que $f''(\vec{\zeta}) := \mathcal{S}_{S^m}(f'_{\vec{\zeta}})$, e o que, na verdade, conhecemos é f. Ou seja, f'' está definido sobre f', que por sua vez está definido sobre f. Então, para calcularmos f'' em um só ponto (i.e., $f''(\vec{u})$, para $\vec{u} \in H^m$) precisamos fazer uma somatória de $s^m := (\#S - 1)^m$ pontos de f', o que **não é nada bom** para nós. Contudo, veja que a única informação que o Teste da Nulidade de Polinômios de Grau nas Variáveis Baixo T–II necessita de f'' se refere a um número constante para n de perguntas, do tipo $f''(\vec{u}) = 0$. Neste instante, a solução será lembrar novamente que $f''(\vec{u}) := \mathcal{S}_{S^m}(f'_{\vec{u}})$ e $f'_{\vec{u}}$ é um polinômio m-ário

[xiv] Vide definição no Parágrafo III.1–2.

[xv] Note que $d := \#D - 1$ e a descrição da *Interpolação de Lagrange* está contida no Parágrafo II.4–2.

restrito a H^m e de grau nas variáveis no máximo $2d$. Ou seja, sobre cada pergunta $f''(\vec{u}) = 0$, podemos utilizar o teste acima e verificar se $\mathcal{S}_{S^m}(f'_{\vec{u}}) = 0$. Portanto, $f'_{\vec{u}}(\vec{\xi}) := f(\vec{\xi})I_{\vec{u}}(\vec{\xi})$ e, para cada ponto $\vec{i} \in S^m \subset D^m$, se desejamos saber o valor de $f'_{\vec{u}}(\vec{i})$, basta-nos calcular $I_{\vec{u}}(\vec{i})$ e ler $f(\vec{i})$, em também um só ponto. Assim sendo, se tivermos $2md/\ell \leq \varepsilon$, podemos compor os dois testes[xvi] para verificar se $f\restriction_{S^m} = 0$ e provamos a corretude do

Teste da Nulidade de Polinômios de Grau nas Variáveis Baixo sobre Domínio Restrito T–XIII $(\text{seguro–}\big(m(n)\log(\ell(n)), 1, C_n \subseteq \Psi^*_{(k(n))}, 1, \Psi_{((2d(n)+1)m(n)k(n))}, m(n)d(n)\big))$**.** *Desta vez suporemos, para $n \in \mathbb{N}_*$ suficientemente grande, $s(n) < d(n)$ e que existe $\varepsilon \in (0,1)$, tal que $2m(n)d(n)/\ell(n) \leq \varepsilon$.*

ENTRADA: *Seja, para todo $n \in \mathbb{N}_*$, $C_n \subseteq \Psi^*_{(k(n))}$ a codificação dos polinômios f's $m(n)$-ários com o domínio restrito a $H_n^{m(n)}$ e de grau nas variáveis no máximo $d(n)$ (i.e., em $f \in K_n[(\xi_1, \ldots, \xi_{m(n)}) \in H_n^{m(n)}]_{\langle d(n)\rangle}$), fornecidos ponto-a-ponto. É dado o número n na pseudo–fita de parâmetro de entrada e $f \in C_n$ na pseudo–fita de teste ϱ, que tem alfabeto $\Psi_{(k(n))}$.*

TESTE: *O teste verifica se a entrada $f \in C_n$ restrita a $S_n^{m(n)}$ é uma função nula (i.e., $f\restriction_{S_n^{m(n)}} = 0$) com segurança–$\big(m(n)\log(\ell(n)), 1, C_n \subseteq \Psi^*_{(k(n))}, 1, \Psi_{((2d(n)+1)m(n)k(n))}, m(n)d(n)\big)$.*

ALGORITMO: *Conforme descrito acima, faz-se o Teste da Nulidade de Polinômios de Grau nas Variáveis Baixo* T–II *(seguro–$\big(m(n)\log(\ell(n)), 1, C_n \subseteq \Psi^*_{(k(n))}, 0, \varnothing, 1\big)$) sobre a função $f'' \in K_n[(\zeta_1, \ldots, \zeta_{m(n)}) \in H_n^{m(n)}]_{\langle d(n)\rangle}$ utilizando-se de $\mathcal{O}(m(n)\log(\ell(n)))$* bits *aleatórios e em tempo de decisão ponderado constante para n. Toda vez que este teste pergunta se f'' é zero em algum ponto $\vec{u} := (u_1, \ldots, u_{m(n)}) \in H_n^{m(n)}$ (i.e., $f''(\vec{u}) = 0$), deve-se calcular $I_{\vec{u}}(\xi_1, \ldots, \xi_{m(n)})$ e executar o Teste da Soma sobre Polinômios de Grau nas Variáveis Baixo* T–XII *(seguro–$\big(m(n)\log(\ell(n)), 1, C_n \subseteq \Psi^*_{(k(n))}, 1, \Psi_{((d(n)+1)m(n)k(n))}, \{m(n)d(n), m(n)s(n)\}\big)$) para verificar se $f'_{\vec{u}} \in K_n[(\xi_1, \ldots, \xi_{m(n)}) \in H_n^{m(n)}]_{\langle 2d(n)\rangle}$ soma 0 em $S_n^{m(n)}$ (i.e., $\mathcal{S}_{S^m}(f'_{\vec{u}}) = 0$). Desta vez, quando o teste escolhe $\vec{i} := (i_1, \ldots, i_{m(n)}) \in D_n^{m(n)} \subseteq H_n^{m(n)}$, calcula-se e fornece-se $f'_{\vec{u}}(\vec{i}) := f(\vec{i})I_{\vec{u}}(\vec{i})$, como se estivesse na pseudo–fita de teste ϱ, através da leitura de $f(\vec{i})$ na pseudo-fita de teste ϱ. Note que este segundo teste é chamado um número constante para n de vezes. Logo, o segundo teste utiliza-se também de $\mathcal{O}(m(n)\log(\ell(n)))$* bits *aleatórios, faz leituras nas pseudo–fitas de teste ϱ e de testemunha π em um número constante para n de vezes e gasta* $\text{Poli}(m(n)d(n))$ *no tempo de decisão ponderado.*

TESTEMUNHA: *A pseudo–fita de testemunha π tem alfabeto $\Psi_{((2d(n)+1)m(n)k(n))}$ e deve conter todas as testemunhas que o Teste da Soma sobre Polinômios de Grau nas Variáveis Baixo* T–XII *(seguro–$\big(m(n)\log(\ell(n)), 1, C_n \subseteq \Psi^*_{(k(n))}, 1, \Psi_{((d(n)+1)m(n)k(n))}, \{m(n)d(n), m(n)s(n)\}\big)$) exige, para cada um dos possíveis $f'_{\vec{u}}$'s.*

Vejamos este teste com valores de uma das codificações polinomiais

Teste da Nulidade de Polinômios de Grau nas Variáveis Baixo sobre Domínio Restrito T–XIV $(\text{seguro–}\big(\log(n), 1, C_n \subseteq \Psi^*_{(8\lceil\log(n)\rceil^6)}, 1, \Psi_{(384\lceil\log(n)\rceil^8)}, \log(n)\big))$**.** *Tomaremos como base a Codificação Polinomial com Grau nas Variáveis Baixo* C–IV *(com $z := 3$), a menos de duas constantes multiplicativas que nos serão úteis em um futuro próximo. Portanto, sejam $s(n) := \lceil\log(n)\rceil - 1$, $m(n) := 4\lceil(s(n)+1)/\log(s(n)+1)\rceil$, $p(n)$ um primo entre $(s(n)+1)^2$ e $2(s(n)+1)^2$, $k(n) := p(n)^3$, $\ell(n)$ entre $(s(n)+1)^2$ e $k(n)$ e $d(n) := 6s(n)$. Lembre-se que $\#S_n =: s(n)+1$, $\#H_n =: \ell(n)$, $\#K_n =: k(n)$ e $S_n \subseteq H_n \subseteq K_n$.*

[xvi] Acreditamos que a esta altura, fica claro, a possibilidade de fazermos tais composições dos testes, conforme comentado no Parágrafo III.1–4.

Concluímos assim que $s(n) < d(n)$ *e, na Codificação Polinomial com Grau nas Variáveis Baixo* C–IV*, vimos que existe* $\varepsilon \in (0,1)$*, tal que* $2m(n)d(n)/\ell(n) \leq 96/\log\log(n) \leq \varepsilon$*, para* n *suficientemente grande,* $m(n)\log(\ell(n)) \in \mathcal{O}(\log(n))$*,* $m(n)d(n) \in \mathcal{O}(\text{Poli}(\log(n)))$ *e* $(2d(n)+1)m(n)k(n) \leq 12(s(n)+1)4\lceil(s(n)+1)/(\log(s(n)+1))\rceil\, 8(s(n)+1)^6 \leq 384\lceil\log(n)\rceil^8$*. Logo, pelo Teste da Nulidade de Polinômios de Grau nas Variáveis Baixo sobre Domínio Restrito* T–XIII *(seguro–*$\big(m(n)\log(\ell(n)), 1, C_n \subseteq \Psi^*_{(k(n))}, 1, \Psi_{((2d(n)+1)m(n)k(n))}, m(n)d(n)\big)$*) acima, temos*

TESTE: *Seja, para todo* $n \in \mathbb{N}_*$*,* $C_n \subseteq \Psi^*_{\left(8\lceil\log(n)\rceil^6\right)}$ *a codificação dos polinômios* f*'s* $m(n)$*-ários com o domínio restrito a* $H_n^{m(n)}$ *e de grau nas variáveis no máximo* $d(n)$ *(i.e., em* $f \in K_n[(\xi_1,\ldots,\xi_{m(n)}) \in H_n^{m(n)}]_{\langle d(n)\rangle}$*) fornecidos ponto-a-ponto. Então, este teste verifica se a entrada* $f \in C_n$ *restrita a* $S_n^{m(n)}$ *é uma função nula (i.e.,* $f\restriction_{S_n^{m(n)}} = 0$*), com segurança–*$\big(\log(n), 1, C_n \subseteq \Psi^*_{\left(8\lceil\log(n)\rceil^6\right)}, 1, \Psi_{\left(384\lceil\log(n)\rceil^8\right)}, \log(n)\big)$.

Com estes testes, já podemos provar, mais diretamente, o Teorema do $\mathcal{PCP}$ I.4–4. Vejamos o próximo capítulo.

Capítulo IV

Classe de Complexidade $\mathcal{PCP}$

Voltaremos, agora, o nosso enfoque ao assunto que foi inicializado no Capítulo I *Esquema de Provas Robustas*, em particular, na Seção I.4 *Introdução ao Teorema do $\mathcal{PCP}$*. O intuito é exatamente fechar a prova do Teorema do $\mathcal{PCP}$ I.4–4. Para tanto, valemo-nos dos testes definidos no Capítulo III *Testes Probabilísticos* sobre as codificações tratadas no Capítulo II *Codificações Algébricas*. Com isto, chegamos aos dois pontos "altos" do texto, quais sejam, as *aritmetizações das fórmulas booleanas* e o Teorema da Composição de Testes IV.4–2.

IV.1 Variantes da Classe de Complexidade $\mathcal{PCP}$

Aprofundando-nos nos *testes*, queremos recaracterizá-los de forma a nos possibilitar definir dois outros tipos de *classes de complexidade $\mathcal{PCP}$*. Para isto, começaremos mudando levemente o *modelo computacional*[i] dos testes seguros–$\big($ $R(n), E(n), C_n \subseteq \Sigma_n^*, Q(n), \Phi_n, T(n)$ $\big)$.[ii] Esta mudança será suficiente na primeira redefinição da classe $\mathcal{PCP}$. Logo a seguir, na segunda redefinição, mudamos também a descrição das *propriedades probabilísticas*.[i]

IV.1–1. Veja que em um teste podemos ter a *entrada de dados* sendo fornecida pela fita de entrada ρ. Simplesmente, a pseudo–fita de parâmetro de entrada seria o fornecimento direto do **parâmetro de entrada n** através de um cálculo sobre a fita de entrada ρ com *só–leitura–indistinguível*.[iii] Para isto, tanto o tempo deste cálculo como o próprio n devem ser polinomiais no comprimento da entrada. Podemos, assim, ler livremente a fita de entrada ρ, utilizando-a como se fosse a pseudo–fita de teste ϱ, mas agora com alfabeto fixo Σ, sem *leitura–direta* e sem limitação no número de leituras. Entretanto, tal leitura só deve ser feita nesta nova:

[i] Note que, conforme declarado no Parágrafo I.2–2, uma classe de complexidade é definida pelo seu *modelo computacional* e pelas suas *propriedades probabilísticas*.

[ii] Veja a introdução de testes $\alpha(n)$–seguros–$\big($ $R(n), E(n), C_n \subseteq \Sigma_n^*, Q(n), \Phi_n, T(n)$ $\big)$ no Parágrafo III.1–4.

[iii] Vide definição no Parágrafo I.2–1.

- `fase de pré-processamento da entrada`. Lê-se x na fita de entrada ρ e fornece-se, na fita de trabalho, resultados de pré-processamentos sobre o que foi lido. Neste momento, pode-se obter ajuda dos parâmetros do teste e o tempo de computação deve ser polinomial em n.

Executamos a *fase de pré-processamento da entrada* entre as *fases de pré-processamento dos parâmetros* e *de endereçamento*. Portanto, os nossos testes terão as *quatro fases* bem delineadas e serão chamados de `tipo não–adaptativo`. Por sua vez, a menos de uma exceção, todos eles, ou não se utilizam da *fase de pré-processamento da entrada*, por não terem a fita de entrada ρ, ou terão a propriedade que a *fase de endereçamento* independe da *fase de pré-processamento da entrada*, sendo assim, elas podem ser computadas em ordem inversa. Neste caso, definiremos o teste como de `tipo totalmente não–adaptativo`. Já os testes de `tipo adaptativo` serão aqueles em que há a necessidade de interação entre as fases, o que não será permitido por nós.

Continuando a descrever o nosso novo *modelo computacional*, veja que ao voltarmos a ter a entrada sendo fornecida pela fita de entrada ρ, que tem o seu alfabeto fixo em Σ, também perdemos o interesse em restringi-la a uma codificação C_n e em controlar o número de leituras sobre ela. Passa, então, o nosso modelo computacional a ser análogo ao dos testes seguros–$\big(\, R(n), \mathrm{Poli}(n), \Sigma, Q(n), \Phi_n, T(n) \,\big)$, conforme foi definido no Parágrafo III.1–3. Qual seja, são as Máquinas de Turing com estas quatro fases, tais que façam no máximo $r(n) \in \mathcal{O}(R(n))$ e $q(n) \in \mathcal{O}(Q(n))$ leituras na fita de probabilidade τ e na pseudo–fita de testemunha π, respectivamente, e tenham os seus aceites restritos aos de *tempo de decisão ponderado*[iv] limitados por $t(n) \in \mathrm{Poli}(T(n))$, em que n é o *parâmetro de entrada*.

IV.1–2. Seja agora a primeira redefinição da classe $\mathcal{PCP}$. Com efeito, a classe de complexidade

$$\boldsymbol{\mathcal{PCP}_{\Phi_n}\big(\, R(n),\ Q(n),\ T(n) \,\big)}$$

é a dos problemas de decisão $L \subseteq \Sigma^*$ que têm um teste seguro–$\big(\, R(n),\ \mathrm{Poli}(n),\ \Sigma,\ Q(n),\ \Phi_n,\ T(n) \,\big)$ para L, sendo que a entrada de dados é dada pela fita de entrada ρ, conforme o *modelo computacional* descrito acima. Deste modo, as *propriedades probabilísticas* desta classe de complexidade são dadas pelas Máquinas de Turing com *esquema de provas robustas*.[v] Portanto, se temos o *tempo de decisão bruto*[vi] limitado polinomialmente em n (i.e., $\mathcal{O}(\,\log(k(n))) \cdot \mathrm{Poli}(T(n)\,) \subseteq \mathrm{Poli}(n)$ [vii]), então

$$\mathcal{PCP}_{\Psi_{(k(n))}}\big(\, R(n),\ Q(n),\ T(n) \,\big) \quad \subseteq \quad \mathcal{PCP}\big(\, R(n),\ \log(k(n)) \cdot Q(n) \,\big).$$

Ou ainda, se tomamos o alfabeto da pseudo–fita de testemunha fixo em Φ,

$$\mathcal{PCP}_{\Phi}\big(\, R(n),\ Q(n),\ n \,\big) \quad = \quad \mathcal{PCP}\big(\, R(n),\ Q(n) \,\big).$$

Esta classe de complexidade é mais restrita e serve exatamente para associarmos os testes à velha definição de classe $\mathcal{PCP}$. Entretanto, vejamos agora uma outra redefinição da classe $\mathcal{PCP}$ que é ainda mais restrita, porém, ela nos permitirá fazer a simulação de outras computações, fato que será de fundamental importância para o Teorema da Composição de Testes IV.4–2, que é um dos nossos objetivos aqui.

IV.1–3. Ampliando o que foi visto no Parágrafo I.4–1, estamos interessados nas fórmulas booleanas $\varphi(\vec{\xi})$ com $c \in \mathbb{N}_*$ cláusulas e que tenham as suas variáveis $\vec{\xi}$ tomadas com $v \in \mathbb{N}_*$ blocos, cada um de tamanho $b \in \mathbb{N}_*$. Ou seja,

$$\vec{\xi} \quad := \quad (\xi_{i,j})_{\substack{i \in \{1 \ldots v\} \cdot \\ j \in \{1 \ldots b\}}}$$

[iv] Conforme definido no Parágrafo III.1–3, o *tempo de decisão ponderado* é o tempo da fase de decisão quando se pensa nele trabalhando sobre o alfabeto da pseudo–fita de testemunha π.

[v] Vide definição no Parágrafo I.3–1.

[vi] Ao contrário do tempo de decisão ponderado, o *tempo de decisão bruto* é o tempo da fase de decisão real, sendo assim, a computação precisa gastar da ordem de $\log(k(n))$ passos para decodificar todas as letras sobre as quais ela trabalha. Veja mais detalhes no Parágrafo III.1–3.

[vii] No caso, este limite polinomial sempre vai acontecer na prática e, talvez, devesse vir através de uma definição limitando polinomialmente $T(n)$ e $k(n)$, visto que sempre estamos interessados em computações de tempo no máximo polinomial.

Neste sentido, se $c(n)$, $v(n)$ e $b(n)$ são funções naturais, então $\mathbf{3FNC(c(n), v(n), b(n))}$ são as fórmulas $\varphi_n(\vec{\xi})$ em 3FNC, tais que tenham $c(n)$ cláusulas e as suas variáveis $\vec{\xi}$ são tomadas com $v(n)$ blocos de tamanho $b(n)$, para os n's em $\mathbb{N}_*$. Já as fórmulas booleanas em $\text{3FNC}(c(n), v(n), b(n))$ que também são de 3SAT chamaremos de $\mathbf{3SAT(c(n), v(n), b(n))}$. Requeremos que na codificação das fórmulas em $\text{3FNC}(c(n), v(n), b(n))$ também seja fornecido o número n, sendo ele o *parâmetro de entrada*, assim como os números $c(n)$, $v(n)$ e $b(n)$. Ainda suporemos sempre $c(n), v(n), b(n) \in \mathcal{O}(\text{Poli}(n))$ e, portanto, estas fórmulas podem ser codificadas em palavras de comprimento polinomial em n e, neste caso, é de fácil observação que $\text{3SAT}(c(n), v(n), b(n))$ pode ser computada por Máquinas de Turing de $\mathcal{NP}$ (que têm tempo polinomial no comprimento da codificação das fórmulas booleanas), as quais, também são de tempo polinomial em n. Por outro lado, é importante notar que $\text{3SAT}(n, 1, n)$ é $\mathcal{NP}$–completo.

No Parágrafo I.4–1 também comentamos quando a fórmula booleana $\varphi(\vec{\xi})$ *é satisfazível fixando-se uma valoração* $\vec{y}$ às primeiras variáveis de $\vec{\xi}$. Utilizar-nos-emos desta idéia de se dividir a satisfação em pedaços para definirmos *propriedades probabilísticas* que têm uma alteração semântica com relação à antiga concepção de *esquema de provas* $\alpha(n)$*–robustas*, que foi fornecida no Parágrafo III.1–2. A diferença está no fato que passamos a só trabalhar com problemas de decisão definidos sobre $\text{3FNC}(c(n), v(n), b(n))$ e que, sobre eles, queremos **mais** do que saber simplesmente se uma de suas fórmulas é satisfazível ou não. Iremos querer também ter um certo controle sobre a satisfazibilidade. A grosso modo, queremos detectar a satisfazibilidade fixando-se alguns dos blocos das variáveis. Na prática, teremos uma fórmula booleana $\varphi(\vec{\xi})$, sortearemos diversos blocões codificados de uma tabela e queremos saber se eles são uma codificação de bloquinhos que são a parte inicial de uma satisfação de $\varphi(\vec{\xi})$, isto é, se a fórmula é satisfazível fixando-se estes bloquinhos. Não querendo nos alongar muito, vamos às definições destas propriedades probabilísticas.

Seja f uma $\alpha(n)$–codificação de $\mathbb{Z}_2^{b(n)}$ em $\Phi_{n*}^* := (\Phi_n \setminus \{\not\emptyset\})^*$.[viii] Então, se $\varphi_n(\vec{\xi}) \in \text{3FNC}(c(n), v(n), b(n))$ para $n \in \mathbb{N}_*$ fixo, defina

$$
\begin{aligned}
L_{\varphi_n} &:= \left\{ \vec{x} := (\vec{x}_1, \ldots, \vec{x}_{v(n)}) \in \mathbb{Z}_2^{v(n)b(n)} \parallel \text{para todo } i \in \{1 \ldots v(n)\}, \vec{x}_i \in \mathbb{Z}_2^{b(n)} \text{ e } \varphi_n(\vec{x}) = 1 \right\}, \\
G_{\varphi_n} &:= \left\{ f_{\vec{x}_1} \cdots f_{\vec{x}_{v(n)}} \parallel (\vec{x}_1, \ldots, \vec{x}_{v(n)}) \in L_{\varphi_n} \right\} \text{ e} \\
\overline{G}_{\varphi_n} &:= \left\{ z_1 \cdots z_{v(n)} \parallel \text{se } (\vec{x}_1, \ldots, \vec{x}_{v(n)}) \in L_{\varphi_n} \text{ então } z_i \notin \mathfrak{V}_{\frac{1-\alpha(n)}{2}}(f_{\vec{x}_i}), \text{ para algum } i \in \{1 \ldots v(n)\} \right\}.
\end{aligned}
$$

Aqui, o símbolo "$\cdot$" é a *concatenação de strings* e $\mathfrak{V}_{\frac{1-\alpha(n)}{2}}(f_{\vec{x}_i})$ são os pontos na $\frac{1-\alpha(n)}{2}$–vizinhança de $f_{\vec{x}_i}$ (ou pontos $\frac{1-\alpha(n)}{2}$–próximos de $f_{\vec{x}_i}$), conforme descrito no Parágrafo II.2–1. Neste mesmo parágrafo, vimos que, por f ser uma $\alpha(n)$–codificação, para $\vec{y}, \vec{y}' \in \mathbb{Z}_2^{b(n)}$ distintos, temos $\mathfrak{V}_{\frac{1-\alpha(n)}{2}}(f_{\vec{y}}) \cap \mathfrak{V}_{\frac{1-\alpha(n)}{2}}(f_{\vec{y}'}) = \varnothing$. Agora diremos que uma Máquina de Turing M tem `esquema de provas robustas com a satisfação fixada por blocos codificados` sse existem $\varepsilon \in (0,1)$ e uma $\alpha(n)$–codificação f conforme definida acima, tais que, para todo $n \in \mathbb{N}_*$ e $\varphi_n(\vec{\xi}) \in \text{3FNC}(c(n), v(n), b(n))$, temos

$$
\text{se} \quad \pi' \in G_{\varphi_n} \quad \text{então} \quad \Pr_{\pi''}\left\{ \Pr_{\tau}\{ M \text{ aceita } x \text{ com testemunha } \pi := \pi' \cdot \pi'' \} = 1 \right\} > 0
$$

e

$$
\text{se} \quad \pi' \in \overline{G}_{\varphi_n} \quad \text{então} \quad \Pr_{\pi''}\left\{ \Pr_{\tau}\{ M \text{ aceita } x \text{ com testemunha } \pi := \pi' \cdot \pi'' \} < \varepsilon \right\} = 1.
$$

Aqui a probabilidade é tomada uniformemente, $\varphi_n(\vec{\xi})$ está descrita na fita de entrada ρ e a pseudo–fita de testemunha π tem alfabeto Φ_n e deve conter a concatenação de π' com π''. Novamente, lembre-se que ε é chamada de *taxa de erro da robustez*.

Diremos, então, que

$$
\text{3SAT}(\, c(n),\, v(n),\, b(n)\,) \quad \in \quad \mathcal{FixPCP}_{\Phi_n}(\, \boldsymbol{R(n)},\, \boldsymbol{Q(n)},\, \boldsymbol{T(n)}\,),
$$

[viii] Pode-se ler a respeito das codificações no Parágrafo II.2–1.

sse existe uma Máquina de Turing no modelo computacional definido no parágrafo anterior e que tem *esquema de provas robustas com a satisfação fixada por blocos codificados*. Logo a seguir utilizar-nos-emos de duas aritmetizações de fórmulas booleanas para verificar a satisfação de $\varphi_n(\vec{\xi})$ fixando-se os blocos codificados. Com isto fornecemos os exemplos deste novo tipo de *classe de complexidade* que, esperamos, possam elucidar melhor o leitor.

IV.1–4. Entretanto, vejamos antes a relação entre estes dois novos tipos de classe de complexidade. Note que, se uma fórmula booleana $\varphi_n(\vec{\xi})$ não é satisfazível, então $\overline{G}_{\varphi_n}$ é formado por todos os possíveis $z_1 \cdots z_{v(n)}$. Fica assim imediato percebermos que, se

$$3\mathrm{SAT}(\ c(n),\ v(n),\ b(n)\ ) \quad \in \quad \mathcal{FixPCP}_{\Phi_n}\big(\ R(n),\ Q(n),\ T(n)\ \big),$$

então

$$3\mathrm{SAT}(\ c(n),\ v(n),\ b(n)\ ) \quad \in \quad \mathcal{PCP}_{\Phi_n}\big(\ R(n),\ Q(n),\ T(n)\ \big).$$

Ou seja,

$$\mathcal{FixPCP}_{\Phi_n}\big(\ R(n),\ Q(n),\ T(n)\ \big) \quad \subseteq \quad \mathcal{PCP}_{\Phi_n}\big(\ R(n),\ Q(n),\ T(n)\ \big).$$

Em particular, relembrando o comentário do final do Parágrafo IV.1–2, se o tempo de decisão bruto for limitado polinomialmente em n, então

$$\mathcal{FixPCP}_{\Psi_{k(n)}}\big(\ R(n),\ Q(n),\ T(n)\ \big) \quad \subseteq \quad \mathcal{PCP}_{\Psi_{k(n)}}\big(\ R(n),\ Q(n),\ T(n)\ \big) \quad \subseteq \quad \mathcal{PCP}\big(\ R(n),\ \log(k(n))\cdot Q(n)\ \big)$$

e, por outro lado, como $3\mathrm{SAT}(n,1,n)$ é $\mathcal{NP}$–completo, sendo

$$3\mathrm{SAT}(\ n,1,n\ ) \quad \in \quad \mathcal{PCP}_{\Psi_{k(n)}}\big(\ R(n),\ Q(n),\ T(n)\ \big),$$

temos que

$$\mathcal{NP} \quad \subseteq \quad \mathcal{PCP}\big(\ R(\mathrm{Poli}(n)),\ \log(k(\mathrm{Poli}(n)))\cdot Q(\mathrm{Poli}(n))\ \big).$$

Utilizar-nos-emos disto mais tarde. Vamos, agora, às aritmetizações.

IV.2 Aritmetização de Fórmulas Booleanas para Leitura Constante de Letras na Testemunha

Buscaremos agora caracterizar aritmeticamente as fórmulas booleanas, de tal modo que cada uma das valorações possa ser representada por elementos *algébricos* (como vetores e polinômios) que nos permitam verificar, através de suas propriedades (como referentes a sua nulidade), se ela é uma satisfação ou não da fórmula booleana.

IV.2–1. Comecemos tomando a fórmula booleana $\varphi(\vec{\xi}) \in 3\mathrm{FNC}(b(n), v(n), b(n))$ e $c \in \{1 \ldots b(n)\}$ e definimos sobre ela

$$\boldsymbol{e_c(\vec{\xi})} \quad := \quad \prod_{z\in\{1,2,3\}} \Big(\ 1 - \chi_{z,c}(\vec{\xi})\ \Big) \quad \in \quad \mathbb{Z}_2[\ \vec{\xi}\ ]_3,$$

sendo que

$$\boldsymbol{\chi_{z,c}(\vec{\xi})} \quad := \quad \begin{cases} \xi_{i,j}, & \text{se o } z\text{-ésimo literal da } c\text{-ésima cláusula de } \varphi(\vec{\xi}) \text{ for exatamente } \xi_{i,j}, \text{ e} \\ 1-\xi_{i,j}, & \text{se o } z\text{-ésimo literal da } c\text{-ésima cláusula de } \varphi(\vec{\xi}) \text{ for exatamente } \neg\xi_{i,j}, \end{cases}$$

com $i \in \{1 \ldots v(n)\}$ e $j \in \{1 \ldots b(n)\}$. Logo

$$e_c(\vec{\xi}) = \ [\text{ a } c\text{-ésima cláusula de } \varphi(\vec{\xi}) \textbf{ não} \text{ é satisfeita pela valoração de } \vec{\xi}\ ].^{[ix]}$$

Então, tomando-se $\varphi(\vec{\xi})$ como uma função booleana vista algebricamente sobre o corpo $\mathbb{Z}_2$, temos

$$\varphi(\vec{\xi}) \ := \prod_{c\in\{1\ldots b(n)\}} \Big(\ 1 - e_c(\vec{\xi})\ \Big) \ := \prod_{c\in\{1\ldots b(n)\}} \Big(\ 1 - \prod_{z\in\{1,2,3\}} \Big(1 - \chi_{z,c}(\vec{\xi})\Big)\ \Big).$$

Com isto, verificamos que

$$\vec{x} := (x_{i,j})_{\substack{i\in\{1\ldots v(n)\}\\ j\in\{1\ldots b(n)\}}} \in \mathbb{Z}_2^{v(n)b(n)} \text{ satisfaz } \varphi(\vec{\xi}) \quad \text{sse} \quad \big(e_c(\vec{x})\big)_{c\in\{1\ldots b(n)\}} \in \mathbb{Z}_2^{b(n)} \text{ é um vetor nulo,}$$

ou seja, para todo $c \in \{1\ldots b(n)\}$, $e_c(\vec{x}) = 0$. Precisamos agora codificar o vetor $\big(e_c(\vec{x})\big)_{c\in\{1\ldots b(n)\}}$ de modo a nos ajudar a verificar a sua nulidade.

Na Codificação por Funcional Linear C–I definimos o *funcional linear* $\mathcal{F}_x(a) := a \cdot x^{\perp}$ e no Parágrafo III.3–1 o *produto tensorial* $(a_i)_i \otimes (b_j)_j := (a_i b_j)_{(i,j)}$. Sabemos também que, para um polinômio $g \in K[\ \vec{\xi}\ ]_d$ de uma variável e de grau no máximo d, existem $g_l \in K^{(v(n)b(n))^l}$, com $l \in \{0\ldots d\}$, tais que

$$g(\vec{\xi}) \ = \ g_0 + \mathcal{F}_{\vec{\xi}}(g_1) + \mathcal{F}_{\vec{\xi}\otimes\vec{\xi}}(g_2) + \cdots + \mathcal{F}_{\vec{\xi}\otimes\ldots\otimes\vec{\xi}}(g_d).$$

Isto significa dizer que os g_l's dependem só do formato de $g(\vec{\xi})$ e não da valoração de $\vec{\xi}$. Mais ainda, se conhecemos os coeficientes de $g(\vec{\xi})$, facilmente calculamos os g_l's. Portanto, para todo $c \in \{1\ldots b(n)\}$, note que $e_c(\vec{\xi}) \in \mathbb{Z}_2[\ \vec{\xi}\ ]_3$. Logo, para todo $\vec{r} := (r_{i,j})_{\substack{i\in\{1\ldots v(n)\}\\ j\in\{1\ldots b(n)\}}} \in \mathbb{Z}_2^{v(n)b(n)}$,

$$\mathcal{F}_{\left(e_c(\vec{\xi})\right)_c}(\vec{r}) \ \in \ \mathbb{Z}_2[\ \vec{\xi}\ ]_3.$$

Por isto, para todo $l \in \{0\ldots 3\}$, existem $g_l\colon \mathbb{Z}_2^{v(n)b(n)} \to \mathbb{Z}_2^{(v(n)b(n))^l}$, tais que, para a variável $\vec{\zeta} := (\zeta_{i,j})_{\substack{i\in\{1\ldots v(n)\}\\ j\in\{1\ldots b(n)\}}}$,

$$\mathcal{F}_{\left(e_c(\vec{\xi})\right)_c}(\vec{\zeta}) \ =: \ g_0(\vec{\zeta}) + \mathcal{F}_{\vec{\xi}}(g_1(\vec{\zeta})) + \mathcal{F}_{\vec{\xi}\otimes\vec{\xi}}(g_2(\vec{\zeta})) + \mathcal{F}_{\vec{\xi}\otimes\vec{\xi}\otimes\vec{\xi}}(g_3(\vec{\zeta})).$$

Observe que, para todo $\vec{r} \in \mathbb{Z}_2^{v(n)b(n)}$, os coeficientes do polinômio $\mathcal{F}_{\left(e_c(\vec{\xi})\right)_c}(\vec{r})$ dependem só de $\vec{r}$ e dos coeficientes dos polinômios $e_c(\vec{\xi})$ e, conhecendo-se $\varphi(\vec{\xi})$, podemos calcular os coeficientes de $e_c(\vec{\xi})$ gastando no máximo tempo polinomial em $v(n)b(n)$. Portanto, se $v(n), b(n) \in \mathcal{O}(\mathrm{Poli}(n))$ e na fase de pré–processamento da entrada entramos com $\varphi(\vec{\xi})$, então, na fase de endereçamento sortearemos os $\vec{r}$'s necessários e computamos, em tempo polinomial em n, os valores dos $g_l(\vec{r})$'s, pois eles independem da valoração de $\vec{\xi}$, ou seja, da testemunha. Portanto, podemos fazê-lo antes da fase de decisão.

Suponha que recebemos as funções $f'\colon \mathbb{Z}_2^{(v(n)b(n))^2} \to \mathbb{Z}_2$, $f''\colon \mathbb{Z}_2^{(v(n)b(n))^3} \to \mathbb{Z}_2$ e, para cada $i \in \{1\ldots v(n)\}$, $f_i\colon \mathbb{Z}_2^{b(n)} \to \mathbb{Z}_2$. Sobre elas defina

$$f(\vec{\zeta}) \ := \sum_{i\in\{1\ldots v(n)\}} f_i(\zeta_{i,1},\ldots,\zeta_{i,b(n)}) \ : \ \mathbb{Z}_2^{v(n)b(n)} \to \mathbb{Z}_2$$

e perceba que, se os f_i's são iguais a $\mathcal{F}_{\mathcal{L}^1_{\{i\}}(\vec{\xi})}$,[x] temos $f = \mathcal{F}_{\vec{\xi}}$. Com isto, queremos testar se existe $\vec{x} \in \mathbb{Z}_2^{v(n)b(n)}$, tal que

[ix] Recordemos que ["propriedade"] é o indicativo da "propriedade", ou seja, é a função característica que tem valor 1 se a "propriedade" é verdadeira e 0, caso contrário. Tudo isto conforme foi definido no Parágrafo I.1–1.

[x] Veja que $\mathcal{L}^1_{\{i\}}(\vec{x}) \in \mathbb{Z}_2^{b(n)}$ é o i-ésimo bloco de $\vec{x}$. Vide o Parágrafo III.3–1.

- temos as igualdades
  - $f_i = \mathcal{F}_{\mathcal{L}^1_{\{i\}}(\vec{x})}$, para todo $i \in \{1 \dots v(n)\}$,
  - $f' = \mathcal{F}_{\vec{x}\otimes\vec{x}}$ e
  - $f'' = \mathcal{F}_{\vec{x}\otimes\vec{x}\otimes\vec{x}}$,
- com $\mathcal{F}_{\left(e_c(\vec{x})\right)_c}$ sendo um funcional linear nulo.

Na verdade, dados corretos f_i's, f' e f'', é muito custoso para nós descobrirmos quem é $\vec{x}$. Mas note que não precisamos disto. Já é muito bom se nos certificarmos de três coisas.

Em primeiro lugar, através do Teste de Linearidade T–V (α–seguro–$(n, 1, \Psi_{(2)}, 0, \varnothing, 1)$), se cada função fornecida é um *funcional linear*, isto é, existem $\vec{y}_i \in \mathbb{Z}_2^{b(n)}$, $\vec{y}' \in \mathbb{Z}_2^{(v(n)b(n))^2}$ e $\vec{y}'' \in \mathbb{Z}_2^{(v(n)b(n))^3}$, tais que

$$\begin{aligned} f_i &= \mathcal{F}_{\vec{y}_i}, \text{ para todo } i \in \{1 \dots v(n)\}, \\ f' &= \mathcal{F}_{\vec{y}'} \text{ e} \\ f'' &= \mathcal{F}_{\vec{y}''}. \end{aligned}$$

Em segundo que, para a concatenação $\vec{x} := \vec{y}_1 \cdots \vec{y}_{v(n)}$ (portanto teríamos, $\vec{y}_i = \mathcal{L}^1_{\{i\}}(\vec{x})$), valem os *produtos tensoriais* $\vec{y}' = \vec{x} \otimes \vec{x}$ e $\vec{y}'' = \vec{x} \otimes \vec{y}'$. Logo, se este for o caso, temos

$$\begin{aligned} f &= \mathcal{F}_{\vec{x}}, \\ f' &= \mathcal{F}_{\vec{x}\otimes\vec{x}} \text{ e} \\ f'' &= \mathcal{F}_{\vec{x}\otimes\vec{x}\otimes\vec{x}}. \end{aligned}$$

Utilizamos, para isto, o Teste do Produto Tensorial T–X (seguro–$\big((m(n) + w(n))\log(k(n)), 1, C_n \subseteq \Psi^*_{(k(n))}, 0, \varnothing, 1\big)$) duas vezes (no primeiro com $m(n) := w(n) := v(n)b(n)$ e no segundo com $m(n) := v(n)b(n)$ e $w(n) := (v(n)b(n))^2$, mas sempre com $k(n) := 2$). Note que, dadas as também variáveis $\vec{v} := (\upsilon_1, \dots, \upsilon_{v(n)b(n)})$ e $\vec{\eta} := (\eta_1, \dots, \eta_{(v(n)b(n))^2})$ e tendo em vista a estrutura interna deste teste, ao executarmos os testes, eles se utilizam somente da valoração sobre os pontos dos funcionais lineares $\mathcal{F}_{\vec{x}}$, $\mathcal{F}_{\vec{y}'}$ e $\mathcal{F}_{\vec{y}''}$ e, sendo assim, podemos tomá-los pelas igualdades

$$\begin{aligned} \mathcal{F}_{\vec{x}}(\vec{\zeta}) &= f(\vec{\zeta}), \\ \mathcal{F}_{\vec{y}'}(\vec{v} \otimes \vec{\zeta}) &= f'(\vec{v} \otimes \vec{\zeta}), \\ \mathcal{F}_{\vec{y}'}(\vec{\eta}) &= f'(\vec{\eta}) \text{ e} \\ \mathcal{F}_{\vec{y}''}(\vec{v} \otimes \vec{\eta}) &= f''(\vec{v} \otimes \vec{\eta}). \end{aligned}$$

Logo, fornecemos ao teste os valores lidos nos pontos dos f_i's, f' e f'', sem que com isto seja necessário que conheçamos $\vec{x}$, $\vec{y}'$ e $\vec{y}''$.

Em terceiro lugar e para finalizar, valemo-nos do Teste da Nulidade de Funcionais Lineares T–I (seguro–$\big(m(n)\log(\ell(n)), 1, C_n \subseteq \Psi^*_{(k(n))}, 0, \varnothing, 1\big)$) (com $\ell(n) := k(n) := 2$ e $m(n) := v(n)b(n)$) para verificar se o *funcional linear*

$$\begin{aligned} \mathcal{F}_{\left(e_c(\vec{x})\right)_c}(\vec{\zeta}) &=: g_0(\vec{\zeta}) + \mathcal{F}_{\vec{x}}(g_1(\vec{\zeta})) + \mathcal{F}_{\vec{x}\otimes\vec{x}}(g_2(\vec{\zeta})) + \mathcal{F}_{\vec{x}\otimes\vec{x}\otimes\vec{x}}(g_3(\vec{\zeta})) \\ &= g_0(\vec{\zeta}) + f(g_1(\vec{\zeta})) + f'(g_2(\vec{\zeta})) + f''(g_3(\vec{\zeta})) \end{aligned}$$

é nulo. Isto nos garantiria que $\left(e_c(\vec{x})\right)_c$ é um vetor nulo e, portanto, concluiríamos que $\vec{x}$ satisfaz $\varphi(\vec{\xi})$. Aqui, da estrutura interna do teste, cada g_l só vai ser valorado sobre os pontos $\vec{r} \in \mathbb{Z}_2^{v(n)b(n)}$ sorteados. Fica assim válido o que discutimos antes e podemos calcular cada um dos $g_l(\vec{r})$'s na fase de endereçamento.

Perceba que esta computação é a única no processo que **não** é de *tipo totalmente não–adaptativo*. Na execução do Teste da Nulidade de Funcionais Lineares T–I (seguro–$\big(m(n)\log(\ell(n)),1,C_n \subseteq \Psi^*_{(k(n))},0,\varnothing,1\big)$) necessitamos calcular os g_l's tendo à mão a entrada $\varphi(\vec{\xi})$, pois, a partir destes, iremos localizar o endereço dos pontos dos f_i's, f' e f'' na testemunha. Neste sentido, a *fase de pré–processamento da entrada* deve ser executada antes da *fase de endereçamento*. De qualquer forma, as fases continuam bem divididas e, assim, o teste ainda é de *tipo não–adaptativo*. Vide maior comentário no Parágrafo IV.1–1.

Teorema do Teste com Leitura Constante de *Bits* na Pseudo–Fita de Testemunha IV.2–2. *Se $v \in \mathcal{O}(1)$ e $b(n) \in \mathcal{O}(\mathrm{Poli}(n))$ é uma função natural, então*

$$3\mathrm{SAT}(\ b(n),\ v,\ b(n)\ ) \ \in\ \mathcal{F}\!\mathit{ix}\mathcal{PCP}_{\Psi_{(2)}}\big(\ b(n)^3,\ 1,\ 1\ \big) \ \subseteq\ \mathcal{PCP}_{\Psi_{(2)}}\big(\ b(n)^3,\ 1,\ 1\ \big).$$

Prova. Note que a prova já foi feita no parágrafo anterior. Entretanto,

Entrada: É fornecida na fita de entrada ρ, com alfabeto binário $\Psi_{(2)}$, a fórmula $\varphi(\vec{\xi}) \in 3\mathrm{FNC}(b(n),v,b(n))$. Ela está codificada em comprimento polinomial em n e, lendo-a, obtemos o valor do parâmetro de entrada n e os parâmetros do teste v e $b(n)$.

Testemunha: Se $\vec{x} \in \mathbb{Z}_2^{vb(n)}$ é uma satisfação para $\varphi(\vec{\xi})$, a pseudo–fita de testemunha π, com alfabeto binário $\Psi_{(2)}$, deverá conter os funcionais lineares $\mathcal{F}_{\mathcal{L}^1_{\{i\}}(\vec{x})}$,[xi] [xii] para cada $i \in \{1\ldots v\}$, $\mathcal{F}_{\vec{x}\otimes\vec{x}}$ e $\mathcal{F}_{\vec{x}\otimes\vec{x}\otimes\vec{x}}$.[xiii] Por outro lado, note que em todos os testes descritos abaixo não necessitamos de testemunhas.

Algoritmo: Lêem-se n como parâmetro de entrada e v e $b(n)$ como parâmetros do teste. Conforme descrito acima, executa-se $v+2$ vezes o Teste de Linearidade T–V (α–seguro–$(n,1,\Psi_{(2)},0,\varnothing,1)$), duas o Teste do Produto Tensorial T–X (seguro–$\big((m(n)+w(n))\log(k(n)),1,C_n \subseteq \Psi^*_{(k(n))},0,\varnothing,1\big)$) e uma o Teste da Nulidade de Funcionais Lineares T–I (seguro–$\big(m(n)\log(\ell(n)),1,C_n \subseteq \Psi^*_{(k(n))},0,\varnothing,1\big)$).

Teste: Perceba que, ao fazermos o Teste da Nulidade de Funcionais Lineares T–I (seguro–$\big(m(n)\log(\ell(n)),1,C_n \subseteq \Psi^*_{(k(n))},0,\varnothing,1\big)$) sobre f'' (que tem $(vb(n))^3$ variáveis), gastamos $(vb(n))^3$ *bits* aleatórios da fita de probabilidade τ. Logo, temos o teste em $\mathcal{PCP}_{\Psi_{(2)}}\big(\ b(n)^3,\ 1,\ 1\ \big)$. Agora, as v primeiras aplicações do Teste de Linearidade T–V (α–seguro–$(n,1,\Psi_{(2)},0,\varnothing,1)$) nos garantem o *esquema de provas robustas com a satisfação fixada por blocos codificados*[xiv] pensado sobre a Codificação por Funcional Linear C–I (para $\ell = 2$). Portanto, este teste também está em $\mathcal{F}\!\mathit{ix}\mathcal{PCP}_{\Psi_{(2)}}\big(\ b(n)^3,\ 1,\ 1\ \big)$.

[IV.2–2] ■

Então, simplesmente fixando-se $b(n) := n$ e $v := 1$, veja que

Corolário do Teste com Leitura de $\mathcal{O}(1)$ Letras da Testemunha IV.2–3. *Temos*

$$3\mathrm{SAT}(\ n,1,n\ ) \ \in\ \mathcal{PCP}_{\Psi_{(2)}}\big(\ n^3,1,1\ \big).$$

■

[xi] Aqui, $\mathcal{L}^1_{\{i\}}(\vec{x}) \in \mathbb{Z}_2^{b(n)}$ é o i-ésimo bloco de $\vec{x}$ e $\mathcal{F}_{\mathcal{L}^1_{\{i\}}(\vec{x})}$ é o seu produto interno. Vide o Parágrafo III.3–1 e a Codificação por Funcional Linear C–I.

[xii] Note que os $\mathcal{F}_{\mathcal{L}^1\{i\}(\vec{x})}$'s são as codificações que fixam a satisfação dos bloco de tamanho $b(n)$. Vide maiores detalhes no Parágrafo IV.1–3.

[xiii] Veja a definição do produto tensorial $x \otimes \vec{x}$ na Codificação por Funcional Linear C–I.

[xiv] Vide o Parágrafo IV.1–3.

Agora, pela observação do Parágrafo IV.1–4, concluímos que

Corolário do $\mathcal{PCP}$ para Leitura de $\mathcal{O}(1)$ Letras da Testemunha IV.2–4.

$$\mathcal{NP} \quad \subseteq \quad \mathcal{PCP}\big(\ \mathrm{Poli}(n),\ 1\ \big).$$

■

Este resultado, apesar de ser posterior e **extremamente surpreendente**, é em si, muito mais simples do que o seguinte. Ele sozinho nos diz que podemos checar testemunhas vendo apenas um número constante de *bits* e com probabilidade tão grande quanto se queira. Mas gostaríamos de ressaltar que ele não é suficiente para provar os resultados obtidos em *Otimização Combinatória*, como por exemplo a *inaproximabilidade* vista na Seção I.5 *Inexistência de Esquema de Aproximação de Tempo Polinomial para a Classe* $\mathcal{MAX}\text{–}\mathcal{SNP}$. O problema é que não podemos simular em tempo polinomial todos os $\mathcal{O}(n^3)$ *bits* aleatórios. Para isto, precisamos abaixá-lo para $\mathcal{O}(\log(n))$ e acontecerá na próxima construção *aritmética* das fórmulas booleanas em 3FNC.

IV.3 Aritmetização de Fórmulas Booleanas para Leitura Logarítmica de *Bits* Aleatórios

IV.3–1. Seja de novo $\varphi(\vec{\xi}) \in \mathrm{3FNC}(b(n), v(n), b(n))$ e defina, para todo $i_1, i_2, i_3 \in \{1 \dots v(n)\}$ e $c, j_1, j_2, j_3 \in \{1 \dots b(n)\}$

$$\boldsymbol{e_{\vec{\xi}}(c, i_1, j_1, i_2, j_2, i_3, j_3)} \quad := \quad [\text{ a } c\text{-ésima cláusula de } \varphi(\vec{\xi}) \text{ tem os seus três literais } \textbf{não} \text{ satisfeitos pela valoração de } \vec{\xi} \text{ e eles são, respectivamente, constituídos pelas variáveis } \xi_{i_1,j_1},\ \xi_{i_2,j_2} \text{ e } \xi_{i_3,j_3}\ ].^{[\mathrm{xv}]}$$

Portanto,

$$\prod_{\substack{i_1,i_2,i_3\in\{1\dots v(n)\}\\ j_1,j_2,j_3\in\{1\dots b(n)\}}} \Big(\ 1 - e_{\vec{\xi}}(c, i_1, j_1, i_2, j_2, i_3, j_3)\ \Big) \quad = \quad [\text{ a } c\text{-ésima cláusula de } \varphi(\vec{\xi}) \text{ é satisfeita pela valoração de } \vec{\xi}\ ].$$

Logo,

$$\varphi(\vec{\xi}) \quad = \prod_{\substack{i_1,i_2,i_3\in\{1\dots v(n)\}\\ c,j_1,j_2,j_3\in\{1\dots b(n)\}}} \Big(\ 1 - e_{\vec{\xi}}(c, i_1, j_1, i_2, j_2, i_3, j_3)\ \Big).$$

Ou seja, para $\vec{x} := (x_{i,j})_{\substack{i\in\{1\dots v(n)\}\\ j\in\{1\dots b(n)\}}} \in \mathbb{Z}_2^{v(n)b(n)}$, $\varphi(\vec{x}) = 1$ sse, para todo $i_1, i_2, i_3 \in \{1 \dots v(n)\}$ e $c, j_1, j_2, j_3 \in \{1 \dots b(n)\}$, $e_{\vec{x}}(c, i_1, j_1, i_2, j_2, i_3, j_3) = 0$, ou ainda,

$$\vec{x} \in \mathbb{Z}_2^{v(n)b(n)} \text{ satisfaz } \varphi(\vec{\xi}) \quad \text{sse} \quad e_{\vec{x}} = 0.$$

Buscaremos, então, aritmetizar $e_{\vec{x}}$ via polinômios e verificar a sua nulidade.

[xv] Aqui, ["propriedade"] é o indicativo da "propriedade" e foi definido no Parágrafo I.1–1.

Para começar, se $z \in \{1,2,3\}$, $i \in \{1 \dots v(n)\}$ e $c, j \in \{1 \dots b(n)\}$, defina

$$\boldsymbol{a_{(z,i)}(c,j)} \quad := \quad [\text{ a } c\text{-ésima cláusula de } \varphi(\vec{\xi}) \text{ tem exatamente a variável } \xi_{i,j} \text{ como sendo o seu } z\text{-ésimo literal }] \text{ e}$$

$$\boldsymbol{b_{(z,i)}(c,j)} \quad := \quad [\text{ a } c\text{-ésima cláusula de } \varphi(\vec{\xi}) \text{ tem exatamente } \neg\xi_{i,j} \text{ como sendo o seu } z\text{-ésimo literal }].$$

Cumpre-nos lembrar que $a_{(z,i)}(c,j)$ e $b_{(z,i)}(c,j)$ dependem somente do formato de $\varphi(\vec{\xi})$ e não de uma possível valoração de $\vec{\xi}$. Com isto concluímos que, para $x_{i,j} \in \mathbb{Z}_2$,

$$a_{(z,i)}(c,j) + x_{i,j}\big(\, b_{(z,i)}(c,j) - a_{(z,i)}(c,j) \,\big) = [\text{ a } c\text{-ésima cláusula de } \varphi(\vec{\xi}) \text{ tem o } z\text{-ésimo literal constituído pela variável } \xi_{i,j} \text{ (i.e., com } (\neg\xi_{i,j}) \text{ ou sem } (\xi_{i,j}) \text{ a negação) e tal literal } \textbf{não} \text{ é satisfeito por } x_{i,j} \,]$$

e, portanto,

$$e_{\vec{\xi}}(c, i_1, j_1, i_2, j_2, i_3, j_3) \quad = \quad \prod_{z\in\{1,2,3\}} \Big(\, a_{(z,i_z)}(c,j_z) + \xi_{i_z,j_z}\big(\, b_{(z,i_z)}(c,j_z) - a_{(z,i_z)}(c,j_z) \,\big) \, \Big).$$

Note que, nesta equação, as operações, como em todo o resto, são feitas sobre $\mathbb{Z}_2$. Porém, a equação foi formulada de tal forma que podemos utilizar as operações de qualquer anel comutativo com unidade, como por exemplo de $\mathbb{Z}$ e/ou de um $\mathbb{Z}_t$ e/ou um corpo K, que o resultado é sempre o mesmo, (variando em 0 ou 1). Agora, conforme as Codificações Polinomial com Grau nas Variáveis Baixo C–IV e Polinomial com Grau Total Baixo C–VIII, para $z := 3$, tome $s(n) := \lceil \log(b(n)) \rceil - 1$, $m(n) := \lceil (s(n)+1)/\log(s(n)+1) \rceil$, $p(n)$ um primo entre $(s(n)+1)^2$ e $2(s(n)+1)^2$, $\ell(n) := k(n) := p(n)^3$ e $d(n) := m(n)s(n)$. Lembre-se que $\#S_n =: s(n)+1$, $\#K_n =: k(n)$ e $S_n \subseteq H_n := K_n$. Sejam as variáveis $\vec{\zeta}_z := (\zeta_{z,1}, \dots, \zeta_{z,m(n)})$, com $z \in \{1,2,3\}$, e $\vec{\eta} := (\eta_1, \dots, \eta_{m(n)})$. Podemos, assim, para cada $i \in \{1 \dots v(n)\}$, codificar o i-ésimo bloco de uma valoração $\vec{x} \in \mathbb{Z}_2^{v(n)b(n)}$ [xvi] de $\varphi(\vec{\xi})$, no polinômio

$$\boldsymbol{f_{\mathcal{L}^1_{\{i\}}(\vec{x})}(\vec{\zeta}_z)} \quad \in \quad K_n[\, \vec{\zeta}_z \in K_n^{m(n)} \,]_{\langle s(n)\rangle} \quad \subseteq \quad K_n[\, \vec{\zeta}_z \in K_n^{m(n)} \,]_{d(n)}.$$

Da mesma maneira codificamos, para cada $z \in \{1,2,3\}$ e $i \in \{1 \dots v(n)\}$, as funções $a_{(z,i)}$ e $b_{(z,i)}$, quando as vemos como pertencentes a $\mathbb{Z}_2^{2m(n)}$, respectivamente, nos polinômios

$$\boldsymbol{g_{a_{(z,i)}}(\vec{\eta},\vec{\zeta}_z)}, \quad \boldsymbol{g_{b_{(z,i)}}(\vec{\eta},\vec{\zeta}_z)} \quad \in \quad K_n[\, (\vec{\eta},\vec{\zeta}_z) \in K_n^{2m(n)} \,]_{\langle s(n)\rangle}.$$

Na Codificação Polinomial com Grau nas Variáveis Baixo C–IV também vimos que existe a sobrejeção $S_n^{m(n)} \xrightarrow{\sigma\;\;} \{1 \dots b(n)\}$, tal que, para todo $z \in \{1,2,3\}$, $i \in \{1 \dots v(n)\}$ e $c, j \in \{1 \dots b(n)\}$, temos $f_{\mathcal{L}^1_{\{i\}}(\vec{x})}(\sigma^{-1}(j)) = x_{i,j}$, $g_{a_{(z,i)}}(\sigma^{-1}(c), \sigma^{-1}(j)) = a_{(z,i)}(c,j)$ e $g_{b_{(z,i)}}(\sigma^{-1}(c), \sigma^{-1}(j)) = b_{(z,i)}(c,j)$. Logo, sobre $S_n^{m(n)}$, recuperamos nas codificações os valores de $\mathcal{L}^1_{\{i\}}(\vec{x})$ e dos $a_{(z,i)}(c,j)$'s e $b_{(z,i)}(c,j)$'s.

Definiremos, para cada valoração $\vec{x} \in \mathbb{Z}_2^{v(n)b(n)}$ de $\varphi(\vec{\xi})$ e $i_1, i_2, i_3 \in \{1 \dots v(n)\}$, o polinômio

$$\begin{aligned} \boldsymbol{F_{\vec{x},i_1,i_2,i_3}(\vec{\eta},\vec{\zeta}_1,\vec{\zeta}_2,\vec{\zeta}_3)} \quad &:= \quad \prod_{z\in\{1,2,3\}} \Big(\, g_{a_{(z,i_z)}}(\vec{\eta},\vec{\zeta}_z) + f_{\mathcal{L}^1_{\{i_z\}}(\vec{x})}(\vec{\zeta}_z)\big(\, g_{b_{(z,i_z)}}(\vec{\eta},\vec{\zeta}_z) - g_{a_{(z,i_z)}}(\vec{\eta},\vec{\zeta}_z) \,\big) \, \Big) \\ &\in \quad K_n[\, (\vec{\eta},\vec{\zeta}_1,\vec{\zeta}_2,\vec{\zeta}_3) \in K_n^{4m(n)} \,]_{\langle 6s(n)\rangle}. \end{aligned}$$

O mais importante é notar que, pelos comentários acima,

$$\vec{x} \in \mathbb{Z}_2^{v(n)b(n)} \text{ satisfaz } \varphi(\vec{\xi}) \quad \text{sse} \quad F_{\vec{x},i_1,i_2,i_3}\restriction_{S_n^{4m(n)}} = 0, \text{ para todo } i_1, i_2, i_3 \in \{1 \dots v(n)\}.$$

[xvi] Veja sobre o i–ésimo bloco de $\vec{x}$ (i.e., $\mathcal{L}^1_{\{i\}}(\vec{x}) := (x_{i,j})_{j\in\{1\dots b(n)\}} \in \mathbb{Z}_2^{b(n)}$) no Parágrafo III.3–1.

Observe inicialmente que não obrigatoriamente $F_{\vec{x},i_1,i_2,i_3} = 0$ (i.e., $F_{\vec{x},i_1,i_2,i_3}$ é uma função nula em todo o domínio), dado que $\#S_n =: s(n) + 1$ e $F_{\vec{x},i_1,i_2,i_3}$ é um polinômio de grau nas $4m(n)$ variáveis no máximo $6s(n)$. Mais ainda, perceba que conhecendo-se a fórmula $\varphi(\vec{\xi})$, podemos, em tempo polinomial em n, calcular os $g_{a_{(z,i_z)}}$'s e $g_{b_{(z,i_z)}}$'s, que independem da valoração $\vec{x}$. Por outro lado, se $\vec{c}, \vec{j}_1, \vec{j}_2, \vec{j}_3 \in K_n^{m(n)}$ e queremos calcular $F_{\vec{x},i_1,i_2,i_3}(\vec{c}, \vec{j}_1, \vec{j}_2, \vec{j}_3)$, podemos fazê-lo em uma computação sobre o alfabeto $k(n)$-ário $\Psi_{(k(n))}$ em *tempo ponderado* constante para n e lendo apenas os três pontos $f_{\mathcal{L}^1_{\{i_1\}}(\vec{x})}(\vec{j}_1)$, $f_{\mathcal{L}^1_{\{i_2\}}(\vec{x})}(\vec{j}_2)$ e $f_{\mathcal{L}^1_{\{i_3\}}(\vec{x})}(\vec{j}_3)$. Ou seja, podemos simular um teste que necessite de $F_{\vec{x},i_1,i_2,i_3}$ na pseudo–fita de testemunha π, tendo apenas os valores de $f_{\mathcal{L}^1_{\{i_1\}}(\vec{x})}(\vec{j}_1)$, $f_{\mathcal{L}^1_{\{i_2\}}(\vec{x})}(\vec{j}_2)$ e $f_{\mathcal{L}^1_{\{i_3\}}(\vec{x})}(\vec{j}_3)$'s. Isto se dá com perda da ordem de $\mathcal{O}$ no número de leituras nesta pseudo–fita e no *tempo de decisão ponderado.*[xvii] Portanto, esta simulação continua no mesmo *modelo computacional*.

Teorema do Teste com Leitura Logarítmica de *Bits* Aleatórios IV.3–2. *Se $v \in \mathcal{O}(1)$ e $b(n) \in \mathcal{O}(\mathrm{Poli}(n))$ é uma função natural, então*

$$\begin{aligned} \mathrm{3SAT}(\ b(n),\ v,\ b(n)\ ) &\in \mathcal{FixPCP}_{\Psi_{(384\lceil \log(b(n))\rceil^8)}}\big(\ \log(b(n)),\ 1,\ \log(b(n))\ \big) \\ &\subseteq \mathcal{PCP}_{\Psi_{(384\lceil \log(b(n))\rceil^8)}}\big(\ \log(b(n)),\ 1,\ \log(b(n))\ \big). \end{aligned}$$

Prova. Dado tudo o que já foi descrito no parágrafo anterior e continuando o que já foi definido, só nos basta aprofundar no algoritmo do teste.

Entrada: É fornecida na fita de entrada ρ, com alfabeto fixo Σ, a fórmula $\varphi(\vec{\xi}) \in \mathrm{3FNC}(b(n), v, b(n))$. Ela está codificada em comprimento polinomial em n e, lendo-a, obtém-se o valor do parâmetro de entrada n e os parâmetros do teste $b(n)$ e v.

Testemunha: Se $\vec{x} \in \mathbb{Z}_2^{vb(n)}$ é uma satisfação para $\varphi(\vec{\xi})$, a pseudo–fita de testemunha π, com alfabeto $\Psi_{\left(384\lceil \log(b(n))\rceil^8\right)}$, deverá conter os polinômios $f_{\mathcal{L}^1_{\{i\}}(\vec{x})}(\vec{\zeta}) \in K_n[\ \vec{\zeta} \in K_n^{m(n)}\ ]_{d(n)}$,[xviii] para cada $i \in \{1 \dots v\}$.[xix] Por outro lado, a pseudo–fita de testemunha deve conter também as testemunhas dos testes referidos abaixo.

Algoritmo: Lêem-se n como parâmetro de entrada e v e $b(n)$ como os parâmetros do teste e fazem-se dois tipos de testes:

- Executa-se o Teste Polinomial de Grau Total Baixo T–IX (α–seguro–$\big(\log(n), 1, \Psi_{\left(8\lceil \log(n)\rceil^6\right)}, 1, \Psi_{\left(8\lceil \log(n)\rceil^8\right)}, \log(n)\big)$) para verificar se $f_{\mathcal{L}^1_{\{i\}}(\vec{x})}(\vec{\zeta})$ está α–próximo de um polinômio de $K_n[\ \vec{\zeta} \in K_n^{m(n)}\ ]_{d(n)}$, para todo $i \in \{1 \dots v\}$ e sendo $\alpha \in (0, 1/2^{28}3^75^5]$ e que pode ser tomado tão pequeno quanto se queira e constante para n. Como repetimos este teste um número também constante para n de vezes (i.e., v vezes), continuamos no *esquema de provas robustas*,[xx] como gostaríamos. Quanto a esta iteração, temos uma melhor discussão no Parágrafo III.1–4.

- Utiliza-se o Teste da Nulidade de Polinômios de Grau nas Variáveis Baixo sobre Domínio Restrito T–XIV (seguro–$\big(\log(n), 1, C_n \subseteq \Psi^*_{\left(8\lceil \log(n)\rceil^6\right)}, 1, \Psi_{\left(384\lceil \log(n)\rceil^8\right)}, \log(n)\big)$) para verificar

[xvii] Vide o Parágrafo III.1–3.

[xviii] Veja que definimos no Parágrafo III.3–1 $\mathcal{L}^1_{\{i\}}(\vec{x}) \in \mathbb{Z}_2^{b(n)}$ como sendo o i-ésimo bloco de $\vec{x}$.

[xix] Note que os $f_{\mathcal{L}^1_{\{i\}}(\vec{x})}$'s são as codificações que fixam a satisfação dos bloco de tamanho $b(n)$. Vide maiores detalhes no Parágrafo IV.1–3.

[xx] Vide definição no Parágrafo III.1–2.

se $F_{\vec{x},i_1,i_2,i_3}\restriction_{S_n^{4m(n)}} = 0$, para todo $i_1, i_2, i_3 \in \{1 \dots v\}$. Como já foi mencionado, para cada ponto $\vec{c}, \vec{j}_1, \vec{j}_2, \vec{j}_3 \in K_n^{m(n)}$ do qual desejamos conhecer a imagem $F_{\vec{x},i_1,i_2,i_3}(\vec{c}, \vec{j}_1, \vec{j}_2, \vec{j}_3)$, deveremos calculá-la sobre os valores de $f_{\mathcal{L}^1_{\{i_1\}}(\vec{x})}(\vec{j}_1)$, $f_{\mathcal{L}^1_{\{i_2\}}(\vec{x})}(\vec{j}_2)$ e $f_{\mathcal{L}^1_{\{i_3\}}(\vec{x})}(\vec{j}_3)$. Novamente, pelo Parágrafo III.1–4, vemos que continuamos no *esquema de provas robustas*, dado: que α pode ser tomado constante para n e tão pequeno quanto se queira; que iteramos este teste v^3 vezes (que também é constante para n) e que cada teste lê um número constante para n de letras da pseudo–fita de teste ϱ, isto é, de pontos $F_{\vec{x},i_1,i_2,i_3}(\vec{\eta}, \vec{\zeta}_1, \vec{\zeta}_2, \vec{\zeta}_3)$.

TESTE: Pela composição destes testes, temos o teste em $\mathcal{PCP}_{\Psi_{\left(384\lceil\log(b(n))\rceil^8\right)}}\big(\log(b(n)),\ 1,\ \log(b(n))\big)$. Por outro lado, as v vezes que rodamos o Teste Polinomial de Grau Total Baixo T–IX (α–seguro–$\big(\log(n), 1, \Psi_{\left(8\lceil\log(n)\rceil^6\right)}, 1, \Psi_{\left(8\lceil\log(n)\rceil^8\right)}, \log(n)\big)$) também nos fornecem o *esquema de provas robustas com a satisfação fixada por blocos codificados*[xxi] sobre a codificação definida acima. Logo, também estamos em $\mathcal{FixPCP}_{\Psi_{\left(384\lceil\log(b(n))\rceil^8\right)}}\big(\log(b(n)),\ 1,\ \log(b(n))\big)$.

[IV.3–2] ■

Assim, novamente tomando-se $b(n) := n$ e $v := 1$, é imediato que

Corolário do Teste com Leitura de $\mathcal{O}(\log(n))$ *Bits* Aleatórios IV.3–3. *Temos*

$$3\mathrm{SAT}(\ n, 1, n\ ) \quad \in \quad \mathcal{PCP}_{\Psi_{\left(384\lceil\log(n)\rceil^8\right)}}\big(\log(n),\ 1,\ \log(n)\big).$$

■

Mais uma vez, lembrando-se do Parágrafo IV.1–4, segue que

Corolário do $\mathcal{PCP}$ para Leitura de $\mathcal{O}(\log(n))$ *Bits* Aleatórios IV.3–4.

$$\mathcal{NP} \quad \subseteq \quad \mathcal{PCP}\big(\log(n),\ \log\log(n)\big).$$

■

O nosso único problema neste corolário, é o fato do número de leituras na pseudo–fita de testemunha π não ser constante para n e isto decorre do Corolário IV.3–3, por ter o alfabeto $\Psi_{\left(384\lceil\log(n)\rceil^8\right)}$ na sua pseudo–fita de testemunha π não constante para n. Portanto, a computação que tem o número de leituras na pseudo–fita de testemunha π constante, quando é operada sobre este alfabeto, deixa de sê-lo à luz da definição original da classe de complexidade $\mathcal{PCP}$, que tem número *real de leituras*. Mas isto vai ser resolvido na seção seguinte, com o Teorema da Composição de Testes IV.4–2.

IV.4 Composição de Testes sobre a Classe de Complexidade $\mathcal{PCP}$

Neste momento, chegamos à "alma" da parte computacional do Teorema do $\mathcal{PCP}$ I.4–4. Procuraremos, agora, *compor testes* de modo a economizar leituras na fita de testemunha, sem que isto aumente muito o

[xxi] Vide o Parágrafo IV.1–3.

número de *bits* aleatórios lidos.

IV.4–1. O Teorema de Cook–Levin I.4–2 mostra-nos que podemos expressar as possíveis computações de uma Máquina de Turing através de uma fórmula booleana em 3FNC. Esta fórmula deverá ter o número de variáveis da ordem do quadrado do tempo de execução da máquina. A propriedade desta fórmula é a que ela é satisfazível fixando-se a representação da entrada da computação[xxii] sse a máquina aceita a mesma entrada. Queremos nos valer da mesma idéia para que um teste possa decidir a aceitação ou não, na *fase de decisão*, de outro teste, sem que este tenha a necessidade de ler toda a testemunha.

Em resumo, desejamos compor dois testes, sendo que o primeiro M tem pequeno tempo de decisão ponderado[xxiii] e o segundo N sabe resolver o problema 3SAT de forma "barata" no número de leituras na pseudo–fita de testemunha π_N. Executamos primeiro as *fases de pré–processamento dos parâmetros*, *de pré–processamento da entrada* e *de endereçamento* do primeiro teste M, para depois simularmos a computação da sua *fase de decisão* através do segundo teste N. Veja que a fase de decisão é determinística e tem como "entrada", a grosso modo, somente os parâmetros de entrada e do teste e as letras lidas na pseudo–fita de testemunha π_M, que, por sinal, são sobre os endereços sorteados pela sua fase de endereçamento.

Portanto, como já dissemos, podemos codificar a computação da fase de decisão de M em uma fórmula booleana $\varphi(\vec{\xi})$ de 3FNC conveniente, tal que temos *aceite* de M sse esta fórmula é satisfazível fixando-se a representação da entrada da sua computação. Na verdade, não perdemos em nada ao *incorporarmos*[xxiv] a representação dos parâmetros de entrada e do teste na fórmula $\varphi(\vec{\xi})$. Ou seja, o segundo teste N constrói $\varphi(\vec{\xi})$ na sua fase de endereçamento, que é satisfazível fixando-se a representação das letras contidas na pseudo–fita de testemunha π_M (sobre os endereços fornecidos na fase de endereçamento de M) sse o primeiro teste M tem a computação de aceite na sua fase de decisão. Cumpre-nos lembrar que as letras da pseudo–fita de testemunha π_M (de M) são de um alfabeto Ψ_n e, portanto, a sua representação binária deve se dar por um bloco de *bits*.

Veja que tudo o que fizemos neste capítulo já nos possibilita realizar tal composição de testes. Mas, sobre as classes de complexidade usuais quais seriam os nossos problemas? Eles são de dois tipos e relativos ao segundo teste N. Em primeiro lugar, o teste N sabe verificar se $\varphi(\vec{\xi})$ é satisfazível (i.e., a computação da fase de decisão de M é de aceite). Porém, como vamos fazer para verificar se esta satisfação fixa a representação das letras da testemunha de M sorteadas na sua fase de endereçamento? Por isto, N precisa saber mais do que verificar apenas a satisfazibilidade de fórmulas booleanas em 3FNC! O nosso segundo problema é que estas letras são sorteadas aleatoriamente sobre uma testemunha que não podemos conhecer de antemão. Entretanto, temos como nosso maior aliado o fato das letras, que queremos que representem a parte fixa da satisfação, serem fornecidas como testemunha. Podemos, então, ao invés de solicitar como testemunha uma letra no alfabeto da pseudo–fita de testemunha π_M, requerer diretamente uma codificação robusta da referida letra.

Ou seja, à primeira vista, resolveremos estes problemas da seguinte maneira. Perceba que as Máquinas de Turing da classe $\mathcal{NP}$ para o problema de decisão 3SAT tomam como prova (i.e., uma testemunha correta) a própria satisfação $\vec{x}$ da fórmula de entrada $\varphi(\vec{\xi})$. Já os testes de $\mathcal{PCP}$ tomam como prova uma codificação robusta $\pi_{\vec{x}}$ de uma satisfação $\vec{x}$. Mas, esta codificação é de tal modo que nos é muito custoso recuperar $\vec{x}$. Portanto, ao ser verificado por N que $\varphi(\vec{\xi})$ é satisfazível, fica-nos muito difícil saber se esta satisfação está fixando a representação das letras escolhidas por M na pseudo–fita de testemunha π_M. É por este motivo que exigimos que o teste N tenha *esquema de provas robustas com a satisfação fixada por blocos codificados*,[xxv] ou seja, se $\vec{x} := \vec{y}_1 \cdots \vec{y}_v$ é uma satisfação, então a prova seria $\pi_{\vec{y}_1} \cdots \pi_{\vec{y}_v} \cdot \overline{\pi}_{\vec{x}}$.

Neste momento devemos pensar que a testemunha de M é uma tabela de letras no alfabeto da sua pseudo–fita de testemunha π_M (e/ou tabela de blocos $\vec{y}$'s, que são as representações binárias destas letras). Sobre

[xxii] Uma fórmula booleana é satisfazível fixando-se algumas constantes sse ela é satisfazível para uma valoração que respeite estas constantes. Veja mais detalhes no Parágrafo I.4–1.

[xxiii] Vide o Parágrafo III.1–3.

[xxiv] Incorporamos algumas constantes a uma fórmula booleana quando ela passa a ser satisfazível sse a anterior o é fixando-se estas constantes. Também vimos a definição no Parágrafo I.4–1.

[xxv] Vide definição no Parágrafo IV.1–3.

tal tabela serão sorteadas algumas das suas entradas que seriam computadas na fase de decisão de M. Temos, então, que na composição dos dois testes precisamos tomar a prova como uma tabela de mesmo tamanho, mas que cada uma das suas entradas seja agora a codificação $\pi_{\vec{y}}$ da respectiva entrada $\vec{y}$ da tabela original. Assim, na fase de endereçamento de M sortearemos algumas entradas da sua tabela, que hipoteticamente conteriam os blocos $\vec{y}_1, \ldots, \vec{y}_z$, mas encontramos lá as codificações $\pi_{\vec{y}_1}, \ldots, \pi_{\vec{y}_z}$. Logo, ao acrescentarmos a esta tabela uma outra que nos forneça a codificação $\pi_{\vec{y}_{z+1}} \cdots \pi_{\vec{y}_v} \cdot \overline{\pi}_{\vec{x}}$, em que $\vec{x} := \vec{y}_1 \cdots \vec{y}_v$ é uma satisfação de $\varphi(\vec{\xi})$ que fixe $\vec{y}_1, \ldots, \vec{y}_v$, temos o *esquema de provas robustas* para utilizarmos na execução do teste N, quando simula a fase de decisão de M.

Agora, depois desta discussão "a grosso modo", retomaremos o assunto, entrando em "um pouco" mais de detalhes.

Teorema da Composição de Testes IV.4–2. *Se $k'(n)$ é uma função natural, $R(n)$, $Q(n)$, $T(n)$, $R'(n)$, $Q'(n)$ e $T'(n)$ são famílias de funções naturais e, para todo $b(n) \in \mathcal{O}(\mathrm{Poli}(\log(k'(n)) \cdot T'(n)))$ e $v(n) \in \mathcal{O}(Q'(n))$, temos uma função natural $k_{bv}(n)$, tal que*

$$3\mathrm{SAT}(\ b(n),\ v(n),\ b(n)\ ) \quad \in \quad \mathcal{FixPCP}_{\Psi_{(k_{bv}(n))}}(\ R(n),\ Q(n),\ T(n)\ ),$$

então,

$$\mathcal{PCP}_{\Psi_{(k'(n))}}(\ R'(n),\ Q'(n),\ T'(n)\ ) \quad \subseteq \quad \bigcup_{k_{bv}(n)} \mathcal{PCP}_{\Psi_{(k_{bv}(n))}}(\ R(n)+R'(n),\ Q(n),\ T(n)\ ).$$

Prova. Portanto, suponha que temos um teste M de $\mathcal{PCP}_{\Psi_{(k'(n))}}(\ R'(n), Q'(n), T'(n)\ )$. Note primeiro que o seu *tempo de decisão bruto*[xxvi] está em $\mathcal{O}(\log(k'(n)) \cdot \mathrm{Poli}(T'(n)))$. Isto se deve ao fato do *tempo de decisão ponderado*[xxvi] ser da ordem de $\mathrm{Poli}(T'(n))$ e da computação da fase de decisão de M gastar tempo da ordem de $\log(k'(n))$ para trans–codificar cada letra de $\Psi_{(k'(n))}$. Há, então, uma fórmula booleana $\varphi(\vec{\xi})$ com $b(n) \in \mathcal{O}(\ (\log(k'(n)) \cdot \mathrm{Poli}(T'(n)))^2\ )$ variáveis, que expressa a computação da *fase de decisão* de M. Novamente, a idéia central é que, se temos outra máquina N que sabe resolver fórmulas booleanas, ela poderia decidir o resultado da computação da fase de decisão de M. Deve-se, então, executar as outras fases de M e, na hora de executar a fase de decisão, roda-se a máquina N sobre esta fórmula booleana $\varphi(\vec{\xi})$. Se a máquina M é rápida na execução da sua fase de decisão, esta fórmula booleana tem poucas variáveis e repare que ela será a entrada de N. Como é só na fase de decisão que iremos ler a pseudo–fita de testemunha π_M, se a máquina N for econômica na leitura da testemunha, temos o que procuramos.

Veja que a entrada da fase de decisão de M são as informações na fita de trabalho (entre elas os endereços) e as letras da pseudo–fita de testemunha π_M (que suporemos ser de número $v(n) - 1 \in \mathcal{O}(Q'(n))$). Evidentemente precisamos *fixar as variáveis*[xxvii] de $\varphi(\vec{\xi})$ que representam esta entrada, pois, apenas deste modo, a simulação da fase de decisão por $\varphi(\vec{\xi})$ fica bem definida. As informações da fita de trabalho não são problema, pois N pode efetivamente lê-las e *incorporá-las* a $\varphi(\vec{\xi})$.[xxvii] O nosso cuidado deve ser com as letras que M leria na pseudo–fita de testemunha π_M. Note que, nos testes que definimos, não conseguimos recuperar "economicamente" a prova, por ela ser robusta. Neste sentido, não temos condição de certificar se alguma prova robusta codifica uma satisfação que fixe as variáveis que representam as testemunhas para M. Como resolvemos este problema?

A solução está no fato óbvio que cada letra do alfabeto $\Psi_{k'(n)}$ da pseudo–fita de testemunha π_M poder ser representada em menos variáveis do que toda a computação da fase de decisão de M. Então, ao invés de supormos $\varphi(\vec{\xi})$ com $b(n)$ variáveis, supô-la-emos com $v(n)$ blocos de $b(n)$ variáveis cada um. Ou seja,

[xxvi] O *tempo de decisão ponderado* é o tempo da fase de decisão, levando-se em conta que ela trabalha sobre o alfabeto da pseudo–fita de testemunha π, e o *tempo de decisão bruto* é o tempo real de execução da fase de decisão, conforme foi definido no Parágrafo III.1–3.

[xxvii] Vide o Parágrafo I.4–1.

$\varphi(\vec{\xi})$ teria um total de $v(n)b(n)$ variáveis. As $b(n)$ variáveis do último bloco representariam a computação propriamente dita da fase de decisão de M. Já, cada um dos $v(n)-1$ primeiros blocos representariam cada uma das $v(n)-1$ letras da pseudo–fita de testemunha que M deveria ler. Mais ainda, não perdemos em generalidade ao supormos que a fórmula $\varphi(\vec{\xi})$ está em 3FNC e também que contém só $b(n)$ cláusulas, isto quando passamos a tomar $b(n)$ em Poli($\log(k'(n)) \cdot T'(n)$). Portanto, $\varphi(\vec{\xi}) \in$ 3FNC($b(n), v(n), b(n)$).

Com isto, exigimos agora que N seja de $\mathcal{F}\mathit{ix}\mathcal{PCP}_{\Psi_{(k_{bv}(n))}}\big(\ R(n), Q(n), T(n)\ \big)$ e, deste modo, saiba resolver $\varphi(\vec{\xi})$ com *esquema de provas robustas com a satisfação fixada por blocos codificados.*[xxviii] Seja esta codificação dada por $\pi_{\vec{y}_1}\cdots\pi_{\vec{y}_{v(n)}}\cdot\overline{\pi}_{\vec{x}}$, para uma satisfação $\vec{x} := \vec{y}_1\cdots\vec{y}_{v(n)}$ da fórmula booleana $\varphi(\vec{\xi})$, sendo que cada $\vec{y}_i$ é um bloco em $\mathbb{Z}_2^{b(n)}$. Queremos então construir a computação do teste $\boldsymbol{N\circ M}$, que é a composição de N por M. Logo, perceba que a testemunha para N também é dividida por blocos (de $\pi_{\vec{y}_1}$ até $\overline{\pi}_{\vec{x}}$) e que, assim, eles não precisam estar necessariamente gravados deste modo consecutivo na pseudo–fita de testemunha $\pi_{N\circ M}$. Por outro lado, cumpre-nos recordar que cada um dos $v(n)-1$ primeiros blocos devem representar uma letra no alfabeto $\Psi_{(k'(n))}$ da pseudo–fita de testemunha π_M. É por este motivo que, simulando a fase de endereçamento de M, que sorteia os endereços das letras $y_1,\ldots,y_{v(n)-1}$, as quais M leria na sua fase de decisão, a fase de endereçamento do teste composto $N\circ M$ deve "montar a testemunha" (ou seja, restringir a pseudo–fita de testemunha $\pi_{N\circ M}$ aos endereços que correspondem a) $\pi_{\vec{y}_1}\cdots\pi_{\vec{y}_{v(n)}}\cdot\overline{\pi}_{\vec{x}}$, sobre a qual passamos a simular a computação de N. Mas antes disto, ainda na fase de endereçamento de $N\circ M$, pode-se efetivamente montar a fórmula booleana $\varphi(\vec{\xi})$ que representa, de um lado, a computação da fase de decisão de M e, de outro lado, é a entrada do teste N. Completamos, assim, a fase de endereçamento e passamos à fase de decisão do teste composto $N\circ M$, repetindo os passos de N, agora sobre esta "testemunha montada".

É absolutamente imediato que o teste composto $N\circ M$ está no *modelo computacional* de $\mathcal{PCP}_{\Psi_{(k_{bv}(n))}}\big(\ R(n)+R'(n), Q(n), T(n)\ \big)$ [xxix] e também continua com o *esquema de provas robustas*,[xxx] já que a sua *taxa de erro da robustez*[xxx] é a soma das taxas de erro da robustez dos testes M e N. Veja mais detalhes sobre o porque podemos fazer isto no Parágrafo I.3–2. Também é imediato perceber que o teste composto $N\circ M$ é de *tipo não–adaptativo.*[xxxi] Por outro lado, se M e N são de *tipo totalmente não–adaptativo*,[xxxi] com um pouco mais de cuidado, vemos que a composição continua a sê-lo. Concluímos, assim, a prova.

[IV.4–2] ■

Demonstraremos finalmente o Teorema do $\mathcal{PCP}$ I.4–4, que está no Capítulo I *Esquema de Provas Robustas*, na Página 13.

Prova do Teorema do $\mathcal{PCP}$ I.4–4. Provaremos logo a seguir a inclusão

$$\mathcal{PCP}_{\Psi_{(384\lceil\log(n)\rceil^8)}}\big(\ \log(n),\ 1,\ \log(n)\ \big) \ \subseteq\ \mathcal{PCP}_{\Psi_{(2)}}\big(\ \log(n),\ 1,\ 1\ \big).$$

Assim sendo, do Corolário do Teste com Leitura de $\mathcal{O}(\log(n))$ *Bits* Aleatórios IV.3–3, temos que

$$\text{3SAT}(\ n,1,n\ ) \ \in\ \mathcal{PCP}_{\Psi_{(384\lceil\log(n)\rceil^8)}}\big(\ \log(n),\ 1,\ \log(n)\ \big) \ \subseteq\ \mathcal{PCP}_{\Psi_{(2)}}\big(\ \log(n),\ 1,\ 1\ \big).$$

Conforme foi observado no final do Parágrafo IV.1–4, 3SAT$(n,1,n)$ é $\mathcal{NP}$–completo e pela definição da classe de complexidade $\mathcal{PCP}$, obtemos o que nos falta para provar o Teorema do $\mathcal{PCP}$ I.4–4, qual seja,

$$\mathcal{NP} \ \subseteq\ \mathcal{PCP}\big(\ \log(n),\ 1\ \big).$$

[xxviii] A definição está no Parágrafo IV.1–3.

[xxix] Vide o Parágrafo IV.1–2.

[xxx] Veja as definições no Parágrafo I.3–1.

[xxxi] Todos os nossos testes são de *tipo não–adaptativo*, dado que eles são divididos nas quatro fases. Já os testes em que podemos sortear os endereços (na fase de endereçamento) antes de se ler a entrada (na fase de pré–processamento da entrada), são chamados de *tipo totalmente não–adaptativo.* Veja a definição no Parágrafo IV.1–1.

Portanto, aplicaremos o Teorema da Composição de Testes IV.4–2 duas vezes, sempre para $Q(n) := Q'(n) := \{1\}$. Primeiramente, pelo Teorema do Teste com Leitura Logarítmica de *Bits* Aleatórios IV.3–2, para todo $v \in \mathcal{O}(1)$ e toda função natural $b(n) \in \mathcal{O}(\text{Poli}(\log(n)))$, temos

$$\text{3SAT}(\ b(n), v, b(n)\ ) \quad \in \quad \mathcal{FixPCP}_{\Psi_{(384\lceil \log(b(n))\rceil^8)}}\big(\ \log\log(n),\ 1,\ \log\log(n)\ \big).$$

Então

$$\mathcal{PCP}_{\Psi_{(384\lceil \log(n)\rceil^8)}}\big(\ \log(n),\ 1,\ \log(n)\ \big) \quad \subseteq \bigcup_{b(n)\in \text{Poli}(\log(n))} \mathcal{PCP}_{\Psi_{(384\lceil \log(b(n)\rceil^8)}}\big(\ \log\log(n)+\log(n),\ 1,\ \log\log(n)\ \big).$$

Agora, pelo Teorema do Teste com Leitura Constante de *Bits* na Pseudo–Fita de Testemunha IV.2–2, para todo $v' \in \mathcal{O}(1)$ e toda função natural $b'(n) \in \mathcal{O}(\text{Poli}(\log\log(n)))$, temos

$$\text{3SAT}(\ b'(n), v', b'(n)\ ) \quad \in \quad \mathcal{FixPCP}_{\Psi_{(2)}}\big(\ \text{Poli}(\log\log(n)),\ 1,\ 1\ \big).$$

Logo,

$$\mathcal{PCP}_{\Psi_{(384\lceil \log(b(n)\rceil^8)}}\big(\ \log(n),\ 1,\ \log\log(n)\ \big) \quad \subseteq \quad \mathcal{PCP}_{\Psi_{(2)}}\big(\ \text{Poli}(\log\log(n))+\log(n),\ 1,\ 1\ \big),$$

para todo $b(n) \in \text{Poli}(\log(n))$. Destes dois resultados terminamos a demonstração, já que obtemos o que nos faltava, ou seja,

$$\mathcal{PCP}_{\Psi_{(384\lceil \log(n)\rceil^8)}}\big(\ \log(n),\ 1,\ \log(n)\ \big) \quad \subseteq \quad \mathcal{PCP}_{\Psi_{(2)}}\big(\ \log(n),\ 1,\ 1\ \big).$$

[I.4–4] ■

Fechamos, deste modo, a parte computacional da prova do Teorema do $\mathcal{PCP}$ I.4–4. Esquecê-la-emos a partir de agora, visto que, para concluirmos a prova, resta-nos apenas ver as *propriedades algébrico–probabilísticas* que constam do próximo capítulo.

Capítulo V

Polinômios de Grau Baixo

Neste capítulo, daremos os subsídios necessários para nós chegarmos à corretude do Teste Polinomial de Grau Total Baixo T–VIII. Será fácil de se observar isto como uma conseqüência imediata do resultado alcançado com o objetivo central deste capítulo, qual seja, provar o Teorema da Proximidade a um Polinômio de Grau Total Baixo V.1–5. Este teorema nos garante uma proximidade constante a polinômios de grau total baixo olhando-se somente para a proximidade a polinômios sobre as "linhas" do seu domínio. Também há uma versão bem simplificada deste teorema, em que trocamos a necessidade dos polinômios de grau total baixo, por polinômios de grau nas variáveis baixo. Entretanto, não a apresentaremos aqui, apesar dela nos provar que o Teste Polinomial de Grau nas Variáveis Baixo T–VI é correto.

Seguindo o intuito acima delineado, começaremos introduzindo as definições básicas na Seção V.1 *Proximidade e Polinômios*, em que ao final, podemos enunciar o Teorema da Proximidade a um Polinômio de Grau Total Baixo V.1–5. A seguir, na Seção V.2 *Introdução à Proximidade aos Polinômios de Grau Baixo* fornecemos as primeiras propriedades associando proximidade e polinômios, em particular, daremos condições de proximidade suficientes para polinômios de grau total baixo. Continuando, na Seção V.3 *Proximidade aos Polinômios de Duas Variáveis*, demonstraremos para o caso particular de dimensão 2, ou seja, quando o domínio é um plano, que obtemos a proximidade constante a polinômios de grau baixo (seja grau total, como grau nas variáveis) olhando-se localmente somente sobre linhas do domínio. Já na Seção V.4 *Proximidade aos Polinômios de Grau Total Baixo*, recorreremos ao fato de sabermos fazer no plano exatamente aquilo que queremos obter para um domínio de dimensão maior para, finalmente, provarmos o Teorema da Proximidade a um Polinômio de Grau Total Baixo V.1–5. Termina, deste modo, a nossa tarefa! Mas, para finalizar, na Seção V.5 *Propriedades Algébricas e Probabilísticas*, entramos nos detalhes técnicos que preferimos deixar passar em branco nas seções precedentes, a fim de amenizar o texto, o quanto possível.

V.1 Proximidade e Polinômios

V.1–1. Primeiramente vejamos algumas convenções. Mas antes, lembraremos novamente que: $\boldsymbol{K}$ é um corpo finito de característica prima $\boldsymbol{p}$ e de $\boldsymbol{k}$ elementos; $\boldsymbol{m, d} \in \mathbb{N}_*$ são, respectivamente, o número de variáveis e o grau total máximo de um polinômio. Suporemos $d < p$ e $d < k$. Tomemos $\mathbb{K}^{\boldsymbol{i}} := K[\xi_1, \ldots, \xi_i]_d$, como o conjunto dos polinômios de i variáveis e de grau total no máximo d, $\widetilde{\mathbb{K}}^{\boldsymbol{i}} := K[\xi_1, \ldots, \xi_i]_{\langle d \rangle}$, como o conjunto dos polinômios de i variáveis e de grau nas variáveis no máximo d e $\widetilde{\mathbb{K}}^{\boldsymbol{i}}_{\boldsymbol{j}}$, como sendo as funções em ${}^{K^i}K$, tais que, na j–ésima coordenada, são polinômios de uma variável e de grau no máximo d, ou seja, fixando-se as variáveis a menos da j–ésima, temos polinômios em $K[\varepsilon_j]_d$.

Por outro lado, sejam $\mathbb{L}$ o *conjunto das linhas de* K^m e $\mathbb{S}$ o *conjunto dos planos de* K^m. Veremos cada `linha` $l \in \mathbb{L}$ como sendo uma função de K a K^m, $l(\xi) := (l_1(\xi), \ldots, l_m(\xi))$, tal que todo $l_i(\xi) \in K[\xi]_1$ é um polinômio linear de uma variável. Já um `plano` $s \in \mathbb{S}$ será visto como o conjunto das linhas nele contidas. Se $x \in K^m$, $s \in \mathbb{S}$ e $l \in \mathbb{L}$, então $\mathbb{L}_{\langle \boldsymbol{x} \rangle} := \{l \in \mathbb{L} \| x \in l\}$ é o conjunto das linhas de $\mathbb{L}$ que passam por x, $\boldsymbol{s}_{\langle \boldsymbol{x} \rangle} := s \cap \mathbb{L}_{\langle x \rangle}$ são as linhas do plano s que passam por x, $\mathbb{S}_{\langle \boldsymbol{x} \rangle} := \left\{s \in \mathbb{S} \| s_{\langle x \rangle} \neq \varnothing\right\}$ são os planos de $\mathbb{S}$ que têm linhas que passam por x e $\mathbb{S}_{\langle \boldsymbol{l} \rangle} := \{s \in \mathbb{S} \| l \in s\}$ são os planos de $\mathbb{S}$ que contém a linha l. A partir de agora, neste capítulo, sempre tomaremos $f \in {}^{K^m}K$ uma função de domínio K^m e contra-domínio K.

V.1–2. Denotaremos a `distância de` $\boldsymbol{f}$ `aos polinômios de grau total no máximo` $\boldsymbol{d}$ por

$$\boldsymbol{\lambda(f)} \;:=\; \lambda_{K^m}(f) \;:=\; \lambda_{K^m}(f, \mathbb{K}^m) \;:=\; \max_{h \in \mathbb{K}^m} \lambda_{K^m}(f, h) \;:=\; \max_{h \in \mathbb{K}^m} \underset{x \in K^m}{\mathcal{P}\mathrm{r}} \{\, f(x) = h(x) \,\}$$

e tomaremos $\boldsymbol{\mathcal{P}^f} \in \mathbb{K}^m$ como sendo esta melhor `aproximação polinomial de` $\boldsymbol{f}$ `sobre o domínio` $\boldsymbol{K^m}$, ou seja, $\lambda(f) := \lambda\left(f, \mathcal{P}^f\right)$. Note que a função $\mathcal{P}^f$ pode não ser única, mas isto não nos atrapalhará.

Faremos o mesmo para uma restrição à linha $l \in \mathbb{L}$. Seja $\boldsymbol{f_{(l)}} := f \circ l \in {}^{K}K$, a função de K a K e que compõe f com l. Neste sentido, a `aproximação polinomial de` $\boldsymbol{f}$ `sobre` $\boldsymbol{l}$ será a função $\boldsymbol{\mathcal{P}^{f,l}}\colon l \subseteq K^m \to K$, tal que $\mathcal{P}^{f,l}_{(l)} := \mathcal{P}^{f,l} \circ l \;\in\; \mathbb{K}^1$ e $\lambda_l(f) = \lambda_l\left(f\restriction_l, \mathcal{P}^{f,l}\right) =: \lambda_K\left(f_{(l)}, \mathcal{P}^{f,l}_{(l)}\right)$, em que

$$\boldsymbol{\lambda_l(f)} \;:=\; \lambda_K\left(f_{(l)}, \mathbb{K}^1\right) \;:=\; \max_{h \in \mathbb{K}^1} \lambda_K\left(f_{(l)}, h\right) \;:=\; \max_{h \in \mathbb{K}^1} \underset{t \in K}{\mathcal{P}\mathrm{r}} \left\{\, f_{(l)}(t) = h(t) \,\right\}.$$

Cumpre-nos lembrar que aqui, muitas vezes, a linha l precisa ser vista como sendo a representação dos seus pontos em K^m, qual seja, a sua imagem por K (i.e., $l\text{“} := \text{”}l\,[\![K]\!] \subseteq K^m$). Observe também que

$$\frac{d+1}{k} \;\leq\; \lambda_l(f) \;=\; \lambda_l\left(f\restriction_l, \mathcal{P}^{f,l}\right) \;=\; \underset{x \in l}{\mathcal{P}\mathrm{r}} \left\{\, f(x) = \mathcal{P}^{f,l}(x) \,\right\} \;=\; \underset{x \in l}{\mathcal{E}xp} \left[\, f(x) = \mathcal{P}^{f,l}(x) \,\right].^{[\mathrm{i}]}$$

Ou seja, $\mathcal{P}^{f,l}_{(l)}$ é o polinômio de uma variável e de grau no máximo d que mais se aproxima de f quando restrito a l, (isto é, que mais se aproxima de $f_{(l)}$).

Do mesmo modo, se $s \in \mathbb{S}$ é um plano, definiremos a `aproximação polinomial de` $\boldsymbol{f}$ `sobre` $\boldsymbol{s}$ pela função

$$\boldsymbol{\mathcal{P}^{f,s}} \;\in\; H_s \;:=\; \left\{\, h\colon s \subseteq K^m \to K \;\|\; h_{(l)} := h \circ l \;\in\; \mathbb{K}^1, \text{ para toda linha } l \in s \,\right\},$$

tal que $\lambda_s(f) = \lambda_s\left(f\restriction_s, \mathcal{P}^{f,s}\right)$ e

$$\boldsymbol{\lambda_s(f)} \;:=\; \max_{h \in H_s} \lambda_s\left(f\restriction_s, h\right) \;:=\; \max_{h \in H_s} \underset{x \in s}{\mathcal{P}\mathrm{r}} \left\{\, f(x) = h(x) \,\right\}.$$

[i] Veja que $\boldsymbol{\mathcal{E}xp}$ é a *esperança* e será definida no Parágrafo V.5–2, que se localiza logo mais a frente na Seção V.5 *Propriedades Algébricas e Probabilísticas*. Cumpre-nos lembrar que [“propriedade”] é o indicativo da “propriedade”, ou seja, é a função característica que tem valor 1 se a “propriedade” é verdadeira e 0, caso contrário, conforme foi definido no Parágrafo I.1–1.

Note primeiramente que $h \in H_s$ é um polinômio de duas variáveis e de grau total no máximo d (i.e., em $\mathbb{K}^2$) transladado para $s \subseteq K^m$. Depois que, mais uma vez aqui, o plano s (como conjunto de linhas $l \in s$), por um abuso notacional, também representará a imagem dos seus pontos em K^m, isto é, $s\text{“}:=\text{”}\bigcup s := \bigcup_{l\in s} l := \bigcup_{l\in s} l\,[\![K]\!] \subseteq K^m$.

V.1–3. Definiremos, então, para $L \subseteq \mathbb{L}$, a `razão de proximidade de` $\boldsymbol{f}$ `em` $\boldsymbol{L}$ por

$$\boldsymbol{\Lambda_L(f)} \;:=\; \underset{l\in L}{\mathcal{E}xp}(\ \lambda_l(f)\ )\,.$$

Quando $L = \mathbb{L}$, omiti-lo-emos. Se $S \subseteq \mathbb{S}$, chamaremos

$$\boldsymbol{\Lambda_S(f)} \;:=\; \underset{s\in S}{\mathcal{E}xp}(\ \Lambda_s(f)\ ) \;:=\; \underset{s\in S}{\mathcal{E}xp}\left(\ \underset{l\in s}{\mathcal{E}xp}(\ \lambda_l(f)\ )\ \right)$$

de `razão de proximidade de` $\boldsymbol{f}$ `nos planos de` $\boldsymbol{S}$.

V.1–4. Para nós transformarmos as definições acima, do caso de polinômios de grau total baixo para polinômios de grau nas variáveis baixo, precisamos observar que m variáveis definem, de modo óbvio, $\boldsymbol{m}$ `coordenadas (cartesianas)`. Seja, então, $\mathbb{C} \subseteq \mathbb{S}$ o conjunto das linhas paralelas às m coordenadas cartesianas. Chamá-lo-emos de conjunto das `linhas coordenadas`. Neste caso, definimos a `razão de proximidade de` $\boldsymbol{f}$ `pelas coordenadas` por

$$\boldsymbol{\widetilde{\Lambda}(f)} \;:=\; \Lambda_{\mathbb{C}}(f).$$

Mas note, não é difícil verificar que

$$\widetilde{\Lambda}(f) \;=\; \underset{i\in\{1\ldots m\}}{\mathcal{E}xp}\left(\ \lambda(f,\widetilde{\mathbb{K}}_i^m)\ \right) \;:=\; \underset{i\in\{1\ldots m\}}{\mathcal{E}xp}(\ \lambda(f,h_i)\ )\,,$$

sendo que cada h_i é tomado, de forma a maximizar $\lambda(f,h_i)$, sobre as funções que são polinômios de uma variável e de grau no máximo d sobre todas as linhas paralelas à i-ésima coordenada (i.e., $h_i \in \widetilde{\mathbb{K}}_i^m$).

Por fim, denotaremos a `distância de` $\boldsymbol{f}$ `aos polinômios de grau nas variáveis no máximo` $\boldsymbol{d}$ por

$$\boldsymbol{\widetilde{\lambda}(f)} \;:=\; \widetilde{\lambda}_{K^m}(f) \;:=\; \lambda_{K^m}(f,\widetilde{\mathbb{K}}^m) \;:=\; \max_{h\in\widetilde{\mathbb{K}}^m} \lambda_{K^m}(f,h) \;:=\; \max_{h\in\widetilde{\mathbb{K}}^m} \underset{x\in K^m}{\mathcal{P}r}\{\ f(x)=h(x)\ \}\,.$$

Deixe-nos, agora, enunciar o principal teorema deste capítulo, que se deve a Arora, Lund, Motwani, Sudan e Szegedy e o qual provaremos nas seções subseqüentes.

Teorema da Proximidade a um Polinômio de Grau Total Baixo V.1–5 ([ALM+92])**.** *Suponha que temos $m \geq 2$, $d \geq 4$, $k \geq \max\{b; 132d^3\}$, $p \geq 2d+1$ e $\delta \in (0, 1/b]$, em que $\boldsymbol{b} := 2^{27}3^75^5$. Para uma função qualquer f de K^m a K,*

$$\textit{se}\quad \Lambda(f) > 1-\delta \quad \textit{então}\quad \lambda(f) > 1-2\delta.$$

Temos assim, pela contra–positiva, uma demonstração imediata do Teste Polinomial de Grau Total Baixo T–VIII, que é o grande objetivo deste capítulo. Com efeito, veja que se f é $(1-\alpha)$–distante (i.e., $\lambda(f) \leq 1-\alpha$), então $\Lambda(f) \leq 1-\alpha/2$ é constante para n e representa a probabilidade do teste *aceitar erradamente*.

Neste momento, antes de nós irmos diretamente às demonstrações, recomendamos fortemente ao leitor ver, no final do capítulo, a Seção V.5 *Propriedades Algébricas e Probabilísticas*, que traz os pré-requisitos necessários para compreendê-las melhor.

V.2 Introdução à Proximidade aos Polinômios de Grau Baixo

Vejamos alguns, e muito úteis, critérios que nos garantem quando uma função é um polinômio de m variáveis e de grau total no máximo d, isto é, condições de proximidade suficientes para pertinência em $\mathbb{K}^m := K[\xi_1, \ldots, \xi_m]_d$.

Lema V.2–1. *Sejam $d' \in \{d \ldots (p-1)\}$, $h \in K[\xi_1, \ldots, \xi_m]_{d'}$ e qualquer um dos três casos abaixo:*

(i) $\underset{l \in \mathbb{L}_{\langle x \rangle}}{\mathcal{P}r} \left\{ \lambda_l(h) > \dfrac{d'}{k} \right\} := \underset{l \in \mathbb{L}_{\langle x \rangle}}{\mathcal{P}r} \left\{ \underset{t \in K}{\mathcal{P}r} \left\{ h_{(l)}(t) = \mathcal{P}^{h,l}_{(l)}(t) \right\} > \dfrac{d'}{k} \right\} > \dfrac{d'}{k},$
para algum $x \in K^m$;

(ii) $\Lambda(h) := \underset{l \in \mathbb{L}}{\mathcal{E}xp}(\lambda_l(h)) := \underset{l \in \mathbb{L}}{\mathcal{E}xp} \left(\underset{x \in l}{\mathcal{P}r} \{ h(x) = \mathcal{P}^{h,l}(x) \} \right) > 1 - \left(1 - \dfrac{d'}{k}\right)^2;$
ou

(iii) $\Lambda(h) = 1.$

Então concluímos que $h \in \mathbb{K}^m := K[\xi_1, \ldots, \xi_m]_d$.

Prova. **(i)** Por translação, provaremos somente para $x := 0$. Como $h \in K[\xi_1, \ldots, \xi_m]_{d'}$ e $d' < p$, para $i \in \{0 \ldots d'\}$, defina $h^{(i)} \in K[\xi_1, \ldots, \xi_m]_i$, tal que

- $h := \sum_{i \in \{0 \ldots d'\}} h^{(i)}$ e
- $h^{(i)}$ é `homogêneo de grau` i, para todo $i \in \{1 \ldots d'\}$.[ii]

Isto é, se $t \in K$ e $x \in K^m$, então $h^{(i)}(t \cdot x) = t^i x$. Na verdade, cada $h^{(i)}$ é formado pelos monômios de h que têm grau total exatamente i.

Faremos a associação de $x \in K^m_*$ [iii] com $l \in \mathbb{L}_{\langle 0 \rangle}$ por $l(t) = t \cdot x$, para todo $t \in K$. Fixe $x \in K^m_*$ com $\lambda_l(h) > d'/k$. Como $h \in K[\xi_1, \ldots, \xi_m]_{d'}$, $h_{(l)} \in K[\xi]_{d'}$ e sendo $\mathcal{P}r_{t \in K} \left\{ h_{(l)}(t) = \mathcal{P}^{h,l}_{(l)}(t) \right\} =: \lambda_l(h) > d'/k$, pelo Corolário da Distância dos Polinômios de Grau Baixo II.3–3, temos que $h_{(l)} = \mathcal{P}^{h,l}_{(l)} \in \mathbb{K}^1$. Mas, observe que

$$h_{(l)}(\xi) = h \circ l\,(\xi) = h(\xi \cdot x) = \sum_{i \in \{0 \ldots d'\}} h^{(i)}(\xi \cdot x) = \sum_{i \in \{0 \ldots d'\}} h^{(i)}(x) \cdot \xi^i \in \mathbb{K}^1 := K[\xi]_d.$$

Ou seja, para $i \in \{d+1 \ldots d'\}$, temos $h^{(i)}(x) = 0$ e, pela hipótese,

$$\underset{x \in K^m_*}{\mathcal{P}r} \left\{ h^{(i)}(x) = 0 \right\} \geq \underset{l \in \mathbb{L}_{\langle 0 \rangle}}{\mathcal{P}r} \left\{ \lambda_l(h) > \frac{d'}{k} \right\} > \frac{d'}{k},$$

visto que $h^{(i)}(0) = 0$. Como $h^{(i)}(\vec{0}) = 0$, pelo Parágrafo V.5–8,

$$\lambda_{K^m}\left(h^{(i)}, 0\right) := \underset{x \in K^m}{\mathcal{P}r} \left\{ h^{(i)}(x) = 0 \right\} \geq \underset{x \in K^m_*}{\mathcal{P}r} \left\{ h^{(i)}(x) = 0 \right\} > \frac{d'}{k}.$$

[ii] Note que $h^{(0)}$ é constante.
[iii] Lembremos que K^m_* é o espaço K^m menos a origem.

Logo, pelo Corolário da Distância dos Polinômios de Grau Total Baixo II.3–5, $h^{(i)} = 0$, (para $i \in \{d+1 \dots d'\}$). Portanto, $h \in \mathbb{K}^m$.

(ii) Note que como toda linha de $\mathbb{L}$ tem exatamente k pontos de K^m e todo ponto de K^m está contido na mesma quantidade de linhas de $\mathbb{L}$, pelo Parágrafo V.5–4, temos que $\mathbb{L}$ multi–particiona igualitariamente K^m [iv] e

$$\underset{x\in K^m}{\mathcal{E}xp}\left(\underset{l\in\mathbb{L}_{\langle x\rangle}}{\mathcal{E}xp}\ (\ \lambda_l(h)\ )\right) \;=\; \underset{l\in\mathbb{L}}{\mathcal{E}xp}(\ \lambda_l(h)\ ) \;=:\; \Lambda(h) \;>\; 1-\left(1-\frac{d'}{k}\right)^2.$$

Ou seja, existe pelo menos um $x \in K^m$, tal que

$$\underset{l\in\mathbb{L}_{\langle x\rangle}}{\mathcal{E}xp}\ (\ \lambda_l(h)\ ) \;>\; 1-\left(1-\frac{d'}{k}\right)^2,$$

e, pelo Parágrafo V.5–6,

$$\underset{l\in\mathbb{L}_{\langle x\rangle}}{\mathcal{P}r}\left\{\ \lambda_l(h) \;>\; \frac{d'}{k}\ \right\} \;>\; \frac{d'}{k}.$$

Portanto, $h \in \mathbb{K}^m$, pelo item acima.

(iii) Note apenas que este é um caso particular do item anterior.

[V.2–1] ■

V.2–2. Para $t \in K$, definiremos $\boldsymbol{f_{[t]}}$ como a função $f(\xi_1, \dots, \xi_m)$ restrita ao hiper–plano em que a primeira variável ξ_1 está fixa a t.

Corolário V.2–3. *Se $p \geq 2d+1$ e $\Lambda(f) = 1$, também temos $f \in \mathbb{K}^m := K[\xi_1, \dots, \xi_m]_d$.*[v]

Prova. Por indução sobre $m \in \mathbb{N}_*$. Para $m := 1$ é imediato. Tome $t_0, \dots, t_d \in K$ distintos. Sobre os hiper–planos que fixam a primeira coordenada ξ_1 aos t_i's, aplique a indução e concluímos que f restrito a cada um dos hiper–planos é um polinômio de grau total d, ou melhor, $f_{[t_i]} \in \mathbb{K}^{m-1}$. Pela *Interpolação de Lagrange*,[vi] podemos obter um polinômio de grau total $2d$, $h \in K[\xi_1, \dots, \xi_m]_{2d}$, tal que $h_{[t_i]} = f_{[t_i]}$, para todo $i \in \{0 \dots d\}$. Mais ainda, note que h restrita às linhas transversais aos hiper–planos é um polinômio de grau d. Como $\Lambda(f) := \mathcal{E}xp_{l\in\mathbb{L}}(\ \lambda_l(f)\ ) = 1$, para toda linha $l \in \mathbb{L}$, f restrita a ela também é um polinômio de grau d. Mas, nas linhas transversais aos hiper–planos, h e f coincidem exatamente nos $d+1$ hiper–planos. Logo, $h = f$ (em todo o domínio). Basta, agora, utilizarmos a Lema V.2–1.(iii), já que $2d \leq p-1$.

[V.2–3] ■

Note que $h \in \widetilde{\mathbb{K}}^m := K[\xi_1, \dots, \xi_m]_{(d)} \subseteq K[\xi_1, \dots, \xi_m]_{md}$. Assim, se $\Lambda(h) > 1 - \left(1 - \frac{md}{k}\right)^2$, diretamente do Lema V.2–1.(ii), concluímos que $h \in \mathbb{K}^m := K[\xi_1, \dots, \xi_m]_d$. Formalizando,

Teorema V.2–4. *Se $p \geq md+1$, $h \in \widetilde{\mathbb{K}}^m$ e $\Lambda(h) > 1 - \left(1 - \frac{md}{k}\right)^2$, então $h \in \mathbb{K}^m$.*

■

[iv] Veja que neste parágrafo também encontramos a definição de *"multi–particiona igualitariamente"*.
[v] O importante é notar que aqui não precisamos saber de antemão se f é um polinômio, como ocorre com o Lema V.2–1.
[vi] Vide o Parágrafo II.4–2.

Sejam, agora, $k \geq 2md$ e $\delta \leq 1/4 \leq (1 - md/k)^2$. Portanto, se

$$\Lambda(h) \;>\; 1-\delta \;\geq\; 1-\left(1-\frac{md}{k}\right)^2,$$

temos,

Corolário V.2–5. *Para $\delta \in (0, 1/4]$, $k \geq 2md$ e $p \geq md+1$, se $h \in \widetilde{\mathbb{K}}^m$ e $\Lambda(h) > 1-\delta$, então $h \in \mathbb{K}^m$.*

■

Corolário V.2–6. *Sejam $\delta, \beta \in (0,1)$, tais que $\delta + \beta \leq 1/4$, $k \geq 2md$ e $p \geq md+1$. Se $\widetilde{\lambda}(f) > 1-\delta$ e $\Lambda(f) > 1-\beta$, então $\lambda(f) > 1-\delta$.*

Prova. Seja $h \in \widetilde{\mathbb{K}}^m$, tal que $\lambda(f,h) := \widetilde{\lambda}(f) > 1-\delta$. Então, veja que

$$\begin{aligned}
\Lambda(h) \;&:=\; \underset{l\in\mathbb{L}}{\mathcal{E}xp}\big(\,\lambda_l(h,\mathcal{P}^{h,l})\,\big) \;\geq\; \underset{l\in\mathbb{L}}{\mathcal{E}xp}\big(\,\lambda_l(h,f)+\lambda_l(f,\mathcal{P}^{h,l})-1\,\big)\\
&\geq\; \underset{l\in\mathbb{L}}{\mathcal{E}xp}\big(\,\lambda_l(h,f)\,\big)+\underset{l\in\mathbb{L}}{\mathcal{E}xp}\big(\,\lambda_l(f,\mathcal{P}^{f,l})\,\big)-1\\
&=:\; \underset{l\in\mathbb{L}}{\mathcal{E}xp}\big(\,\lambda_l(h,f)\,\big)+\Lambda(f)-1 \;=\; \lambda(h,f)+\Lambda(f)-1 \;>\; 1-(\delta+\beta).
\end{aligned}$$

Aqui, a igualdade se deve ao Parágrafo V.5–4, onde vemos, novamente, que $\mathbb{L}$ forma uma multi–partição igualitária de K^m. Portanto, do Corolário V.2–5, $h \in \mathbb{K}^m$ e $\lambda(f) \geq \lambda(f,h) = \widetilde{\lambda}(f) > 1-\delta$.

[V.2–6] ■

V.3 Proximidade aos Polinômios de Duas Variáveis

Provaremos aqui o Teorema da Proximidade a um Polinômio de Grau Total Baixo V.1–5, para o caso particular em que o domínio é um plano (dimensão dois), portanto, com o número de variáveis $m = 2$. Este resultado será a base para subirmos na dimensão, o que ocorrerá na seção seguinte.

Corolário da Proximidade a um Polinômio de Duas Variáveis de Grau Baixo V.3–1. *Tomando-se $m = 2$, $\delta \in (0, 1/20]$, $d \geq 4$ e $k \geq 132d^3$, temos*

$$\textit{se}\quad \widetilde{\Lambda}(f) > 1-\delta \quad \textit{então}\quad \widetilde{\lambda}(f) > 1-\frac{19\delta}{2} \geq 1-10\delta.$$

Antes de demonstrá-lo, veja uma imediata conseqüência para o caso de grau total baixo.

Corolário da Proximidade a um Polinômio de Duas Variáveis que tem Grau Total Baixo V.3–2. *Sejam $m = 2$, $\delta \in (0, 1/84]$, $d \geq 4$, $k \geq 132d^3$ e $p \geq 2d+1$. Assim,*

$$\textit{se}\quad \Lambda(f) > 1-\delta \quad \textit{então}\quad \lambda(f) > 1-20\delta.$$

Prova. Antes, defina o `conjunto das linhas paralelas` $\mathbb{Z}$ como sendo formado pelos conjuntos maximais de linhas parelelas[vii] em K^m. Então, como em nosso caso $m = 2$, veja que as linhas paralelas de algum elemento de $\mathbb{Z}$ cobrem o plano K^2. Ainda, cada elemento do conjunto das linhas paralelas tem o mesmo número de linhas de $\mathbb{L}$ e cada linha está contida em um único elemento de $\mathbb{Z}$. Com isto, do Parágrafo V.5–4, temos que $\mathbb{Z}$ forma uma partição igualitária para $\mathbb{L}$ e

$$1-\delta \;<\; \Lambda(f) \;:=\; \underset{l\in\mathbb{L}}{\mathcal{E}xp}(\,\lambda_l(f)\,) \;=\; \underset{z\in\mathbb{Z}}{\mathcal{E}xp}\left(\,\underset{l\in z}{\mathcal{E}xp}(\,\lambda_l(f)\,)\,\right) \;=:\; \underset{z\in\mathbb{Z}}{\mathcal{E}xp}(\,\Lambda_z(f)\,).$$

Veja primeiro que $7\delta/3 \le 1/63 < 1/2$. Agora escolha $z \in \mathbb{Z}$ que maximize $\Lambda_z(f)$. Então, respectivamente, pelos Parágrafos V.5–9 e V.5–6,

$$\underset{z'\in\mathbb{Z}\smallsetminus\{z\}}{\mathcal{E}xp}(\,\Lambda_{z'}(f)\,) \;>\; 1-2\delta$$

e

$$\underset{z'\in\mathbb{Z}\smallsetminus\{z\}}{\mathcal{P}r}\left\{\,\Lambda_{z'}(f) > 1-\frac{40\delta}{19}\,\right\} \;>\; 1-\frac{19}{20} \;\ge\; 0.$$

Logo, existe $z' \in \mathbb{Z}$ distinto de z, tal que

$$\Lambda_z(f) \;\ge\; \Lambda_{z'}(f) \;>\; 1-\frac{40\delta}{19}.$$

Agora, reparametrizando-se f, temos que

$$\widetilde{\Lambda}(f) \;>\; 1-\frac{40\delta}{19}$$

e, como $40\delta/19 \le 1/20$, podemos aplicar o Corolário da Proximidade a um Polinômio de Duas Variáveis de Grau Baixo V.3–1 para concluir que

$$\widetilde{\lambda}(f) \;>\; 1-20\delta.$$

Como $k \ge 132d^3 \ge 4d$ e $p \ge 2d+1$ e ainda $21\delta \le 1/4$, podemos utilizar o Corolário V.2–6 e obter

$$\lambda(f) \;>\; 1-20\delta.$$

[V.3–2] ■

Primeiramente, veja este teorema de Arora e Safra. Ele nos afirma que duas funções de duas variáveis que são polinômios unários e de grau no máximo d, respectivamente, nas suas linhas (i.e., pertence à $\widetilde{\mathbb{K}}_1^2$) e nas suas colunas (i.e., em $\widetilde{\mathbb{K}}_2^2$) e que são δ–próximas entre si, também são, necessariamente, 5δ–próximos a polinômios de grau nas variáveis no máximo d. Ou seja,

Teorema da Proximidade de Funções de Duas Variáveis e de Coordenadas Polinomiais V.3–3 ([AS92])**.** *Sejam $m = 2$, $\delta \in (0, 1/10]$, $d \ge 4$, $k \ge 132d^3$, $h_{\varepsilon_y}(\varepsilon_x) := h(\varepsilon_x, \varepsilon_y) \in \widetilde{\mathbb{K}}_1^2$ e $g_{\varepsilon_x}(\varepsilon_y) := g(\varepsilon_x, \varepsilon_y) \in \widetilde{\mathbb{K}}_2^2$. Assim,*

$$\textit{se}\quad \lambda(g,h) > 1-\delta \quad \textit{então}\quad \widetilde{\lambda}(g) > 1-\frac{17\delta}{4} \ge 1-5\delta \quad \textit{e}\quad \widetilde{\lambda}(h) > 1-\frac{17\delta}{4} \ge 1-5\delta.$$

Podemos, então, verificar a

Prova do Corolário V.3–1. Sejam $h \in \widetilde{\mathbb{K}}_1^2$ e $g \in \widetilde{\mathbb{K}}_2^2$ que mais se aproximam de f, ou seja,

$$\begin{aligned}\lambda(f,h) &:= \lambda(f, \widetilde{\mathbb{K}}_1^2)\\ \lambda(f,g) &:= \lambda(f, \widetilde{\mathbb{K}}_2^2).\end{aligned}$$

[vii] Duas linhas são *paralelas* se formam um único plano (portanto são distintas) e não têm pontos em comum.

Como $m = 2$ e $\widetilde{\Lambda}(f) > 1 - \delta$,

$$1 - 2\delta \;<\; 2\widetilde{\Lambda}(f) - 1 \;:=\; \lambda(f,h) + \lambda(f,g) - 1 \;\leq\; \lambda(h,g)$$

e ainda, $\lambda(f,h) > 1 - \delta$, ou $\lambda(f,g) > 1 - \delta$. Sem perda de generalidade, suponha o primeiro caso. Assim, como $2\delta \leq 1/10$, pelo Teorema da Proximidade de Funções de Duas Variáveis e de Coordenadas Polinomiais V.3–3, existe $q \in \widetilde{\mathbb{K}}^2$, tal que $\lambda(h,q) > 1 - 17\delta/2$. Logo,

$$\widetilde{\lambda}(f) \;\geq\; \lambda(f,q) \;\geq\; \lambda(f,h) + \lambda(h,q) - 1 \;>\; 1 - \delta + 1 - \frac{17\delta}{2} - 1 \;=\; 1 - \frac{19\delta}{2}.$$

[V.3–1] ■

V.3–4. Feita a parte mais fácil, vamos à demonstração do Teorema da Proximidade de Funções de Duas Variáveis e de Coordenadas Polinomiais V.3–3. Ela se dará pelas proposições que se seguem. Para isto, tome $m = 2$, $\delta \in (0, 1/10]$, $d \geq 4$, $k \geq 132d^3$, $h_{\varepsilon_y}(\varepsilon_x) := h(\varepsilon_x, \varepsilon_y) \in \widetilde{\mathbb{K}}_1^2$ e $g_{\varepsilon_x}(\varepsilon_y) := g(\varepsilon_x, \varepsilon_y) \in \widetilde{\mathbb{K}}_2^2$, tais que

$$\lambda(g,h) \;>\; 1 - \delta,$$

conforme a hipótese do teorema. Também recorreremos ao auxílio dos polinômios $\boldsymbol{A} \in K[\varepsilon_x, \varepsilon_y]$ e $\boldsymbol{B} \in K[\varepsilon_x, \varepsilon_y]$, tais que os graus nas variáveis ε_x e ε_y são exatamente[viii]

$$\begin{aligned} \boldsymbol{x_A} &:= \partial_{[\varepsilon_x]}(A), \\ \boldsymbol{y_A} &:= \partial_{[\varepsilon_y]}(A), \\ \boldsymbol{x_B} &:= \partial_{[\varepsilon_x]}(B) \text{ e} \\ \boldsymbol{y_B} &:= \partial_{[\varepsilon_y]}(B). \end{aligned}$$

Defina, então, o `polinômio divisor para` $\boldsymbol{B}$ por

$$\boldsymbol{\mathcal{B}^B_{[\varepsilon_y]}(\varepsilon_x)} \;:=\; \left(\Delta^B_{[\varepsilon_y]}(\varepsilon_x)\right)^{\max\{y_A - y_B + 1; 0\}},$$

em que $\boldsymbol{\Delta^B_{[\varepsilon_y]}(\varepsilon_x)}$ é tal que independe da variável ε_y e existe um polinômio $r \in K[\varepsilon_x, \varepsilon_y]_{\langle x_B, y_B - 1\rangle}$ com

$$B(\varepsilon_x, \varepsilon_y) \;=\; \Delta^B_{[\varepsilon_y]}(\varepsilon_x) \cdot \varepsilon_y^{y_B} + r(\varepsilon_x, \varepsilon_y).$$

Ou seja, $\Delta^B_{[\varepsilon_y]}(\varepsilon_x)$ seria o coeficiente para o mais alto grau, quando nós vemos B como um polinômio na variável ε_y. Note apenas que, se B não é nulo, o mesmo acontece com $\mathcal{B}^B_{[\varepsilon_y]}$, sendo que ele ainda tem o grau limitado, visto que $\partial_{[\varepsilon_x]}(\mathcal{B}^B_{[\varepsilon_y]}) \leq x_B \max\{y_A - y_B + 1; 0\} \leq x_B(1 + \max\{y_A; 0\})$.

Proposição V.3–5. *Suponha que B não seja o polinômio nulo e $X \subseteq K$ não vazio, de tal forma que*

$$A(x, \varepsilon_y) \;=\; g_x(\varepsilon_y) B(x, \varepsilon_y),$$

para todo $x \in X$. Então existe um polinômio $C \in K[\varepsilon_x, \varepsilon_y]_{\langle x_A + x_B y_A, y_A - y_B\rangle}$, tal que, para todo $x \in \boldsymbol{\mathcal{B}^B_{[\varepsilon_y]}} \boldsymbol{X} := \left\{x \in X \| \Delta^B_{[\varepsilon_y]}(x) \neq 0\right\}$,

$$C(x, \varepsilon_y) \;=\; g_x(\varepsilon_y) \mathcal{B}^B_{[\varepsilon_y]}(x),$$

Mais ainda, se $\#X \geq x_A + 1 + x_B(y_A + 1)$, então o grau de C na variável ε_y é no máximo d (i.e., $\partial_{[\varepsilon_y]}(C) \leq d$).

Prova. Como $B \neq 0$, temos $x_B, y_B \geq 0$ e podemos tomar a *Pseudo–divisão de Euclides* sobre a variável ε_y,

$$\mathcal{B}^B_{[\varepsilon_y]}(\varepsilon_x) A(\varepsilon_x, \varepsilon_y) \;=\; C(\varepsilon_x, \varepsilon_y) B(\varepsilon_x, \varepsilon_y) + R(\varepsilon_x, \varepsilon_y),$$

[viii] Definimos o grau (∂) de um polinômio no Parágrafo II.3–1.

sendo que C e R são polinômios em que $\partial_{[\varepsilon_y]}(R) < \partial_{[\varepsilon_y]}(B) =: y_B$, $\partial_{[\varepsilon_y]}(C) \leq y_A - y_B$ e $\partial_{[\varepsilon_x]}(C) \leq \partial_{[\varepsilon_x]}(\mathcal{B}^B_{[\varepsilon_y]}) + x_A - x_B \leq x_A + x_B \max\{y_A; 0\} = x_A + x_B y_A$, este último, visto que $x_A < 0$ (e, portanto, $x_A := -\infty$), se $\max\{y_A; 0\} = 0$. Ora, se $x \in X$,

$$g_x(\varepsilon_y)\mathcal{B}^B_{[\varepsilon_y]}(x)B(x,\varepsilon_y) \quad = \quad \mathcal{B}^B_{[\varepsilon_y]}(x)A(x,\varepsilon_y) \quad = \quad C(x,\varepsilon_y)B(x,\varepsilon_y) + R(x,\varepsilon_y).$$

Mas, para $x \in \mathcal{B}^B_{[\varepsilon_y]}X \subseteq X$, temos $\Delta^B_{[\varepsilon_y]}(x) \neq 0$ e, portanto, $B(x,\varepsilon_y) \neq 0$. Assim, como a divisão (no caso, por $B(x,\varepsilon_y)$ e também sobre a variável ε_y) é única, temos que $C(x,\varepsilon_y) = g_x(\varepsilon_y)\mathcal{B}^B_{[\varepsilon_y]}(x)$ e $R(x,\varepsilon_y) = 0$. Provamos, assim, a primeira parte.

Suponha agora que $\#\mathcal{B}^B_{[\varepsilon_y]}X \geq x_A + x_B y_A + 1$. Como $\partial_{[\varepsilon_x]}(C) \leq x_A + x_B y_A$, temos que C é a *Interpolação de Lagrange*[ix] de $g_x(\varepsilon_y)\mathcal{B}^B_{[\varepsilon_y]}(x)$ sobre $x_A + x_B y_A + 1$ pontos $x \in \mathcal{B}^B_{[\varepsilon_y]}X$. Logo

$$\partial_{[\varepsilon_y]}(C) \quad = \quad \max_{x \in \mathcal{B}^B_{[\varepsilon_y]}X} \partial_{[\varepsilon_y]}\left(g_x(\varepsilon_y)\mathcal{B}^B_{[\varepsilon_y]}(x)\right) \quad \leq \quad d.$$

Para terminar, note que $\#\left(X \smallsetminus \mathcal{B}^B_{[\varepsilon_y]}\right) := \#\left\{x \in X \| \Delta^B_{[\varepsilon_y]}(x) = 0\right\} \leq x_B$ e, assim sendo, $\#X \geq x_A + 1 + x_B(y_A + 1)$ é suficiente para C ter grau d na variável ε_y.

[V.3–5] ■

Agora, simplesmente trocaremos as variáveis ε_x por ε_y e tomaremos B variando somente com ε_x (logo, $y_B = 0$). Neste caso, $\Delta^B_{[\varepsilon_x]}(\varepsilon_y)$ é uma constante não nula. Portanto $\mathcal{B}^B_{[\varepsilon_x]}(\varepsilon_y)$ também é uma constante não nula e $\mathcal{B}^B_{[\varepsilon_x]}Y = Y$. Disto, é fácil de ver que

Corolário V.3–6. *Sejam $B \in K[\varepsilon_x]$ um polinômio só na variável ε_x e não nulo e $Y \subseteq K$ não vazio, tais que, para todo $y \in Y$,*

$$A(\varepsilon_x, y) \quad = \quad h_y(\varepsilon_x)B(\varepsilon_x).$$

Então existe um polinômio $C \in K[\varepsilon_x, \varepsilon_y]_{\langle x_A - x_B, y_A\rangle}$ tal que, para todo $y \in Y$,

$$C(\varepsilon_x, y) \quad = \quad h_y(\varepsilon_x).$$

Ainda, se $\#Y \geq y_A + 1$, então o grau de C na variável ε_x é no máximo d (i.e., $\partial_{[\varepsilon_x]}(C) \leq d$).

■

V.3–7. Continuemos. Para $x, y \in K$, diremos que eles são `pontos bons entre si` (ou ainda, um `não é um ponto ruim` para o outro) sse $g_x(y) = h_y(x)$.

Proposição V.3–8. *Existem $\boldsymbol{\mathcal{X}} \subseteq \boldsymbol{\mathfrak{X}} \subseteq K$ e $\boldsymbol{\mathcal{Y}} \subseteq \boldsymbol{\mathfrak{Y}} \subseteq K$ tais que*

(i) $\underset{K}{\mathcal{P}r}(\ \mathfrak{X}\ ) \quad > \quad 1 - \dfrac{17\delta}{4} \quad \geq \quad 1 - 5\delta,$

(ii) $\#\mathfrak{Y} \quad = \quad 3d + 1,$

(iii) $\#\mathcal{X} \quad \geq \quad 10d^3 + 1,$

(iv) $\#\mathcal{Y} \quad \geq \quad d + 1,$

[ix] Vide o Parágrafo II.4–2.

(v) *$\mathcal{Y}$ tem pelo menos $d+1$ pontos bons, para cada um dos pontos de $\mathfrak{X}$,*

(vi) *$\mathcal{X}$ tem pelo menos $11d^3+1$ pontos bons, para cada um dos pontos de $\mathcal{Y}$ e*

(vii) *$\mathfrak{Y}$ tem pelo menos $2d+1$ pontos bons, para cada um dos pontos de $\mathcal{X}$.*

Prova. Das hipóteses do Teorema da Proximidade de Funções de Duas Variáveis e de Coordenadas Polinomiais V.3–3, temos que

$$\Pr_{x,y\in K}\{\ x \text{ e } y \text{ são pontos bons entre si }\} \ := \ \Pr_{x,y\in K}\{\ g_x(y)=h_y(x)\ \} \ =: \ \lambda(g,h) \ > \ 1-\delta.$$

Mas, como $k \geq 132d^3 \geq 17\cdot 3d$ e, pelo Parágrafo V.5–6, temos que

$$\Pr_{y\in K}\left\{ \Pr_{x\in K}\{\ x \text{ e } y \text{ são pontos bons entre si }\} > 1-\frac{17\delta}{16} \right\} \ > \ 1-\frac{16}{17} \ = \ \frac{1}{17} \ \geq \ \frac{3d}{k}.$$

Portanto, existe $\mathfrak{Y}\subseteq K$ satisfazendo o item **(ii)**, tal que

$$\Pr_{\substack{y\in\mathfrak{Y}\\ x\in K}}\{\ x \text{ e } y \text{ são pontos bons entre si }\} \ > \ 1-\frac{17\delta}{16}.$$

Novamente pelo Parágrafo V.5–6,

$$\Pr_{x\in K}\left\{ \Pr_{y\in\mathfrak{Y}}\{\ x \text{ e } y \text{ são pontos bons entre si }\} > 1-\frac{1}{4} \right\} \ > \ 1-\frac{17\delta}{4} \ \geq \ 1-5\delta$$

e existe $\mathfrak{X}\subseteq K$ satisfazendo o item **(i)** e

$$\Pr_{y\in\mathfrak{Y}}\{\ x \text{ e } y \text{ são pontos bons entre si }\} \ > \ 1-\frac{1}{4} \ = \ \frac{3}{4} \ \geq \ 1-\frac{1}{3}, \qquad \text{(V.3:1)}$$

para todo $x\in\mathfrak{X}$. Assim, pelo fato de $k\geq 132d^3$ e $5\delta\leq 1/2$, temos

$$\#\mathfrak{X} \ > \ k(1-5\delta) \ \geq \ \frac{k}{2} \ \geq \ 66d^3$$

e existe $\mathcal{X}\subseteq\mathfrak{X}$ com $66d^3+1$ elementos e satisfazendo os itens **(iii)** e **(vii)**. Como

$$\Pr_{\substack{x\in\mathcal{X}\\ y\in\mathfrak{Y}}}\{\ x \text{ e } y \text{ são pontos bons entre si }\} \ > \ 1-\frac{1}{3},$$

mais uma vez pelo Parágrafo V.5–6,

$$\Pr_{y\in\mathfrak{Y}}\left\{ \Pr_{x\in\mathcal{X}}\{\ x \text{ e } y \text{ são pontos bons entre si }\} > 1-\frac{5}{6}=\frac{1}{6} \right\} \ > \ 1-\frac{2}{5} \ = \ \frac{3}{5}.$$

Assim, existe $\mathcal{Y}\subseteq\mathfrak{Y}$ satisfazendo os itens **(iv)** e **(vi)**. Para finalizar, lembraremos que, da Equação (V.3:1), para todo $x\in\mathfrak{X}$,

$$\Pr_{y\in\mathfrak{Y}}\{\ x \text{ e } y \text{ são pontos bons entre si }\} \ > \ \frac{3}{4}$$

e que tiramos uma fração $2/5$ de $\mathfrak{Y}$ para formar $\mathcal{Y}$. Portanto temos, para todo $x\in\mathfrak{X}$,

$$\Pr_{y\in\mathfrak{Y}}\{\ y\in\mathcal{Y} \text{ e, ainda, } x \text{ e } y \text{ são pontos bons entre si }\} \ > \ \frac{3}{4}-\frac{2}{5} \ = \ \frac{7}{20} \ \geq \ \frac{1}{3}$$

e obtemos o item **(v)**.

[V.3–8] ■

Proposição V.3–9. *Veja que:*

(i) *Se $A(\varepsilon_x, y) = h_y(\varepsilon_x)B(\varepsilon_x, y)$, para todo $y \in \mathfrak{Y}$, $y_A \leq 2d$ e $y_B \leq d$, então, para todo $x \in \mathcal{X}$, $A(x, \varepsilon_y) = g_x(\varepsilon_y)B(x, \varepsilon_y)$.*

(ii) *Se $A(x, \varepsilon_y) = g_x(\varepsilon_y)B(x)$, para todo $x \in \overline{\mathcal{X}} \subseteq \mathcal{X}$, tal que $\#\mathcal{X} \smallsetminus \overline{\mathcal{X}} \leq d^3$, $x_A \leq 10d^3$ e $x_B \leq 10d^3 - d$, então, para todo $y \in \mathcal{Y}$, $A(\varepsilon_x, y) = h_y(\varepsilon_x)B(\varepsilon_x)$.*

(iii) *Se $A(\varepsilon_x, y) = h_y(\varepsilon_x)$, para todo $y \in \mathcal{Y}$ e $y_A \leq d$, então, para todo $x \in \mathfrak{X}$, $A(x, \varepsilon_y) = g_x(\varepsilon_y)$.*

Prova. Provemos o item **(ii)** utilizando-nos da Proposição V.3–8.(vi). Analogamente podemos provar os itens **(i)** e **(iii)** através das Proposições V.3–8.(vii) e V.3–8.(v), respectivamente. Fixe $y \in \mathcal{Y}$. Então $A(\varepsilon_x, y)$ e $h_y(\varepsilon_x)B(\varepsilon_x)$ têm grau no máximo $10d^3$. Ora, se $x \in \overline{\mathcal{X}}$ é ponto bom para y,

$$A(x, y) \quad = \quad g_x(y)B(x) \quad = \quad h_y(x)B(x).$$

Veja que, da Proposição V.3–8.(vi), $\mathcal{X}$ tem pelo menos $11d^3 + 1$ pontos bons para y. Mas $\#\mathcal{X} \smallsetminus \overline{\mathcal{X}} \leq d^3$. Portanto, $\overline{\mathcal{X}}$ tem pelo menos $10d^3 + 1$ pontos bons para y. Assim, pelo Corolário da Distância dos Polinômios de Grau Baixo II.3–3, temos

$$A(\varepsilon_x, y) \quad = \quad h_y(\varepsilon_x)B(\varepsilon_x).$$

[V.3–9] ■

Proposição V.3–10. *Existem polinômios $q' \in K[\varepsilon_x, \varepsilon_y]_{\langle 4d^2, 2d\rangle}$ e $q'' \in K[\varepsilon_x, \varepsilon_y]_{\langle 4d^2, d\rangle}$, tais que q'' não é nulo e, para todo $y \in \mathfrak{Y}$,*

$$q'(\varepsilon_x, y) \quad = \quad h_y(\varepsilon_x)q''(\varepsilon_x, y).$$

Tomando-se como certa esta proposição, podemos, finalmente, obter a

Prova do Teorema da Proximidade de Funções de Duas Variáveis e de Coordenadas Polinomiais V.3–3. Sejam os polinômios $q' \in K[\varepsilon_x, \varepsilon_y]_{\langle 4d^2, 2d\rangle}$ e $q'' \in K[\varepsilon_x, \varepsilon_y]_{\langle 4d^2, d\rangle}$, tais que q'' não é nulo e, para todo $y \in \mathfrak{Y}$,

$$q'(\varepsilon_x, y) \quad = \quad h_y(\varepsilon_x)q''(\varepsilon_x, y),$$

fornecidos pela Proposição V.3–10. Pela Proposição V.3–9.(i), para todo $x \in \mathcal{X}$,

$$q'(x, \varepsilon_y) \quad = \quad g_x(\varepsilon_y)q''(x, \varepsilon_y).$$

Pela Proposição V.3–8.(iii), $\#\mathcal{X} \geq 10d^3 + 1 \geq 4d^2 + 1 + 4d^2(2d + 1)$ (pois $d \geq 4$) e da Proposição V.3–5, existe $q''' \in K[\varepsilon_x, \varepsilon_y]_{\langle 10d^3, d\rangle}$, tal que, para todo $x \in \mathcal{B}^B_{[\varepsilon_y]}\mathcal{X}$,

$$q'''(x, \varepsilon_y) \quad = \quad g_x(\varepsilon_y)\mathcal{B}^{q''}_{[\varepsilon_y]}(x).$$

Agora, como $d \geq 4$, temos $\#\mathcal{X} \smallsetminus \mathcal{B}^{q''}_{[\varepsilon_y]}\mathcal{X} \leq \partial_{[\varepsilon_x]}(q'') \leq 4d^2 \leq d^3$ e $\partial_{[\varepsilon_x]}\left(\mathcal{B}^{q''}_{[\varepsilon_y]}\right) \leq 4d^2(2d + 1) \leq 10d^3 - d$. Portanto, da Proposição V.3–9.(ii), para todo $y \in \mathcal{Y}$,

$$q'''(\varepsilon_x, y) \quad = \quad h_y(\varepsilon_x)\mathcal{B}^{q''}_{[\varepsilon_y]}(\varepsilon_x).$$

Pela Proposição V.3–8.(iv), $\#\mathcal{Y} \geq d+1$ e do Corolário V.3–6, existe $q \in K[\varepsilon_x, \varepsilon_y]_{\langle d,d\rangle}$, tal que, para todo $y \in \mathcal{Y}$,

$$q(\varepsilon_x, y) \quad = \quad h_y(\varepsilon_x).$$

Por fim, pela Proposição V.3–9.(iii),

$$q(x, \varepsilon_y) \quad = \quad g_x(\varepsilon_y),$$

para todo $x \in \mathfrak{X}$. Concluímos a demonstração ao ver a Proposição V.3–8.(i) e assim

$$\widetilde{\lambda}(g) \quad \geq \quad \mathop{\mathcal{P}r}_{x,y\in K}\{\, q(x,y) = g_x(y) \,\} \quad \geq \quad \mathop{\mathcal{P}r}_{x\in K}\{\, q(x,\varepsilon_y) = g_x(\varepsilon_y) \,\} \quad > \quad 1 - \frac{17\delta}{4}.$$

Veja que podemos obter uma demonstração análoga para

$$\widetilde{\lambda}(h) \quad > \quad 1 - \frac{17\delta}{4}.$$

[V.3–3] ■

Agora, só nos falta verificar a

Prova da Proposição V.3–10. Comecemos definindo, para $y \in \mathfrak{Y}$,

$$A_y \quad := \quad \left\|1, y, \ldots, y^{2d}, -h_y(\varepsilon_x), -h_y(\varepsilon_x)y, \ldots, -h_y(\varepsilon_x)y^d\right\|_{1\times(3d+2)} \quad \neq \quad 0$$

e

$$A \quad := \quad \left\|\left(A_y\right)_{y\in\mathfrak{Y}}\right\|_{(3d+1)\times(3d+2)},$$

visto que $\#\mathfrak{Y} = 3d+1$, pela Proposição V.3–8.(ii). Portanto, as entradas de A são polinômios na variável ε_x e de grau no máximo d (i.e., em $K[\varepsilon_x]_d$). Seja, então,

$$t \quad := \quad \left\| \begin{array}{c} q'_0(\varepsilon_x) \\ \vdots \\ q'_{2d}(\varepsilon_x) \\ q''_0(\varepsilon_x) \\ \vdots \\ q''_d(\varepsilon_x) \end{array} \right\|_{(3d+2)\times 1}$$

uma solução do sistema homogêneo $A \cdot t = 0$. Pelo Parágrafo V.5–1, temos condição de ver que podemos tomar t como sendo um vetor cujas entradas são polinômios na variável ε_x e de grau no máximo $4d^2 \geq d(3d+2)$ (já que $d \geq 4$). Ou seja, $q''_i, q'_j \in K[\varepsilon_x]_{4d^2}$, para todo $i \in \{0 \ldots d\}$ e $j \in \{0 \ldots 2d\}$. Assim, defina

$$q''(\varepsilon_x, \varepsilon_y) \quad := \quad \sum_{i=0}^{d} q''_i(\varepsilon_x) \cdot \varepsilon_y^i \quad \in \quad K[\varepsilon_x, \varepsilon_y]_{\langle 4d^2, d\rangle}$$

e

$$q'(\varepsilon_x, \varepsilon_y) \quad := \quad \sum_{j=0}^{2d} q'_j(\varepsilon_x) \cdot \varepsilon_y^j \quad \in \quad K[\varepsilon_x, \varepsilon_y]_{\langle 4d^2, 2d\rangle}.$$

Portanto, da solução do sistema $A \cdot t = 0$, temos, para todo $y \in \mathfrak{Y}$,

$$q'(\varepsilon_x, y) \quad = \quad h_y(\varepsilon_x) q''(\varepsilon_x, y).$$

Falta-nos apenas ver que $q''(\varepsilon_x, \varepsilon_y)$ não é o polinômio nulo. Na verdade, tudo também estará bem se provarmos que $q'(\varepsilon_x, \varepsilon_y)$ não é nulo, como é fácil de se ver. Como observado ao final do Parágrafo V.5–1, A tem menos equações do que variáveis e, assim, possui uma solução não nula nas condições acima, completando a prova.

[V.3–10] ■

V.4 Proximidade aos Polinômios de Grau Total Baixo

Provaremos, finalmente, o Teorema da Proximidade a um Polinômio de Grau Total Baixo V.1–5 utilizando, para tanto, o Corolário da Proximidade a um Polinômio de Duas Variáveis que tem Grau Total Baixo V.3–2.

V.4–1. Primeiramente redefiniremos f à $\widehat{f} \in {}^{K^m}K$ ponto-a-ponto, através da *maioria* dos valores no ponto das melhores aproximações polinomiais de f sobre cada linha que passa pelo respectivo ponto, ou seja,

$$\widehat{\boldsymbol{f}}(\boldsymbol{\xi}) \;:=\; \underset{l \in \mathbb{L}_{\langle \xi \rangle}}{\text{Maioria}} \left\{ \; \mathcal{P}^{f,l}(\xi) \; \right\}.$$

Se, por algum acaso, houver empate na *maioria*, desempata-se aleatoriamente.

Note que o Teorema da Proximidade a um Polinômio de Grau Total Baixo V.1–5 nos diz que, a grosso modo, se temos uma função f que é, em média, próxima a polinômios de grau baixo, sempre que restrita às linhas (i.e., $\Lambda(f) > 1-\delta$), então f já é, como um todo, próxima a um polinômio de grau total baixo. Portanto, tendo em vista o resultado abaixo, para nós provarmos o teorema, falta-nos "apenas" verificarmos se $\widehat{f}$ é um polinômio de grau total no máximo d (isto é, $\widehat{f} \in \mathbb{K}^m$).

Proposição V.4–2. *Se $\delta \in (0, 1/2)$, tal que $\Lambda(f) > 1 - \delta$, então $\lambda\left(f, \widehat{f}\right) > 1 - 2\delta$.*

Prova. Como $\mathcal{P}r_{K^m}\left(\; l \; \right)$ e $\mathcal{P}r_{\mathbb{L}}\left(\; \mathbb{L}_{\langle x \rangle} \; \right)$ são constantes para $l \in \mathbb{L}$ e $x \in K^m$, pelo Parágrafo V.5–4, temos que $\mathbb{L}$ multi–particiona igualitariamente[x] K^m e

$$1 - \delta \;<\; \Lambda(f) \;:=\; \underset{l \in \mathbb{L}}{\mathcal{E}xp}\left(\underset{x \in l}{\mathcal{P}r}\left\{ \; f(x) \;=\; \mathcal{P}^{f,l}(x) \; \right\} \right) \;=\; \underset{x \in K^m}{\mathcal{E}xp}\left(\underset{l \in \mathbb{L}_{\langle x \rangle}}{\mathcal{P}r}\left\{ \; f(x) \;=\; \mathcal{P}^{f,l}(x) \; \right\} \right).$$

Portanto, pelo Parágrafo V.5–6,

$$\underset{x \in K^m}{\mathcal{P}r}\left\{ \; \underset{l \in \mathbb{L}_{\langle x \rangle}}{\mathcal{P}r}\left\{ \; f(x) = \mathcal{P}^{f,l}(x) \; \right\} > 1 - \frac{1}{2} = \frac{1}{2} \; \right\} \;>\; 1 - 2\delta.$$

Ou seja,

$$\lambda\left(f, \widehat{f}\right) \;:=\; \underset{x \in K^m}{\mathcal{P}r}\left\{ \; f(x) = \widehat{f}(x) \; \right\} \;>\; 1 - 2\delta,$$

já que $\widehat{f}(x) := z$, se $\mathcal{P}r_{l \in \mathbb{L}_{\langle x \rangle}}\left\{ \; z = \mathcal{P}^{f,l}(x) \; \right\} > 1/2$.

[V.4–2] ■

[x] Note que a definição de *"multi–particiona igualitariamente"* está no referido parágrafo.

Comecemos, então, a provar que $\widehat{f} \in \mathbb{K}^m$, que é tudo que nos falta.

Proposição V.4–3. *Para $m \geq 2$ e $x \in K^m$, temos $\Lambda(f) \leq \Lambda_{\mathbb{S}_{\langle x \rangle}}(f) + 2/k$.*

Prova. Note que

$$\begin{aligned} \#\mathbb{L} &= \frac{\binom{k^m}{2}}{\binom{k}{2}} = k^{m-1}\frac{k^m-1}{k-1} \\ \#\mathbb{L}_{\langle x \rangle} &= \frac{k^m-1}{k-1}. \end{aligned}$$

Portanto,

$$\mathop{\mathcal{P}r}_{\mathbb{L}}\big(\, \mathbb{L}_{\langle x \rangle} \,\big) = \frac{1}{k^{m-1}}$$

e

$$\Lambda_L(f) \geq \Lambda(f) - \mathop{\mathcal{P}r}_{\mathbb{L}}\big(\, \mathbb{L}_{\langle x \rangle} \,\big) = \Lambda(f) - \frac{1}{k^{m-1}} \geq \Lambda(f) - \frac{1}{k},$$

em que $L := \mathbb{L} \smallsetminus \mathbb{L}_{\langle x \rangle}$, qual seja, o conjunto das linhas que não passam por x. Então, cada linha de L forma, com x, um único plano $s \in \mathbb{S}_{\langle x \rangle}$, ou seja, conforme o Parágrafo V.5–4, $\mathbb{S}_{\langle x \rangle}$ pode ser visto como uma partição igualitária de L. E temos, ao voltarmos os olhos para o Parágrafo V.5–8,

$$\begin{aligned} \Lambda_L(f) &:= \mathop{\mathcal{E}xp}_{l \in L}\big(\, \lambda_l(f) \,\big) = \mathop{\mathcal{E}xp}_{s \in \mathbb{S}_{\langle x \rangle}} \left(\mathop{\mathcal{E}xp}_{l \in s \smallsetminus s_{\langle x \rangle}} \big(\, \lambda_l(f) \,\big) \right) \\ &\leq \mathop{\mathcal{E}xp}_{s \in \mathbb{S}_{\langle x \rangle}} \left(\mathop{\mathcal{E}xp}_{l \in s}\big(\, \lambda_l(f) \,\big) + \mathop{\mathcal{P}r}_{s}\big(\, s_{\langle x \rangle} \,\big) \right) \leq \Lambda_{\mathbb{S}_{\langle x \rangle}}(f) + \frac{1}{k} \end{aligned}$$

e, aqui, $s \smallsetminus s_{\langle x \rangle}$ são as linhas de s que não passam por x. Concluindo,

$$\Lambda(f) \leq \Lambda_L(f) + \frac{1}{k} \leq \Lambda_{\mathbb{S}_{\langle x \rangle}}(f) + \frac{2}{k}.$$

[V.4–3] ■

V.4–4. Sejam $x \in K^m$, $t \in K$ e $\delta \in (0,1)$. Diremos que $\boldsymbol{L \subseteq \mathbb{L}_{\langle x \rangle}}$ `é` $\boldsymbol{\delta}$`–coerente com` $\boldsymbol{f}$ `em` $\boldsymbol{x}$ `sobre` $\boldsymbol{t}$ sse, para todo $l \in L$, f e $\mathcal{P}^{f,l}$ são δ–próximos em $l \smallsetminus \{x\}$ (isto é, $\lambda_{l \smallsetminus \{x\}}\big(f, \mathcal{P}^{f,l}\big) > 1 - \delta$) e $\mathcal{P}^{f,l}(x) = t$. Algumas vezes o ponto t não é importante e, neste caso, diremos simplesmente que $\boldsymbol{L}$ `é` $\boldsymbol{\delta}$`–coerente com` $\boldsymbol{f}$ `em` $\boldsymbol{x}$, com o significado que existe $t \in K$, tal que L é δ–coerente com f em x sobre t.

Proposição V.4–5. *Sejam $h \in \mathbb{K}^m$, $\delta \in (0, 1/4]$ e $k \geq 2d + 5$.*

(i) *Se $x \in K^m$ e $L \subseteq \mathbb{L}_{\langle x \rangle}$, tal que, para todo $l \in L$, f e h são δ–próximos em $l \smallsetminus \{x\}$ (isto é, $\lambda_{l \smallsetminus \{x\}}(f,h) > 1 - \delta$), então*

$$L \text{ é } \delta\text{–coerente com } f \text{ em } x \text{ sobre } h(x).$$

(ii) *Se f e h são δ^2–próximos (isto é, $\lambda_{K^m}(f,h) > 1 - \delta^2$), então, para todo $x \in K^m$,*

$$\mathop{\mathcal{P}r}_{l \in \mathbb{L}_{\langle x \rangle}} \{\, \{l\} \text{ é } \delta\text{–coerente com } f \text{ em } x \text{ sobre } h(x) \,\} > 1 - 2\delta.$$

Prova. **(i)** Para $l \in L$, temos $h_{(l)} := h \circ l \in \mathbb{K}^1$ e $\mathcal{P}^{f,l}_{(l)} := \mathcal{P}^{f,l} \circ l \in \mathbb{K}^1$. Tome $K' := K \setminus l^{-1}[\![x]\!]$. Então, pelo Parágrafo V.5–8, como $\lambda_K\left(f_{(l)}, \mathcal{P}^{f,l}_{(l)}\right) \geq \lambda_K\left(f_{(l)}, h_{(l)}\right)$, obtemos

$$\lambda_{K'}\left(f_{(l)}, \mathcal{P}^{f,l}_{(l)}\right) \;\geq\; \lambda_{K'}\left(f_{(l)}, h_{(l)}\right) - \frac{2}{k}.$$

Por uma propriedade probabilística trivial,

$$\begin{aligned}
\lambda_{K'}\left(h_{(l)}, \mathcal{P}^{f,l}\right) &\geq \lambda_{K'}\left(f_{(l)}, h_{(l)}\right) + \lambda_{K'}\left(f_{(l)}, \mathcal{P}^{f,l}_{(l)}\right) - 1 \\
&\geq 2\lambda_{K'}\left(f_{(l)}, h_{(l)}\right) - \frac{2}{k} - 1 \;\geq\; 2\lambda_{l\setminus\{x\}}(f,h) - 1 - \frac{2}{k-1} \\
&> 2 - 2\delta - \frac{2}{k-1} - 1 \;\geq\; \frac{k-3}{k-1} - \frac{1}{2} \;\geq\; \frac{k-5}{2(k-1)} \;\geq\; \frac{d}{k-1},
\end{aligned}$$

lembrando que $\delta \leq 1/4$ e $k \geq 2d+5$. Portanto, pelo Corolário da Distância dos Polinômios de Grau Baixo II.3–3, $h_{(l)} = \mathcal{P}^{f,l}_{(l)}$ e concluímos o resultado.

(ii) Fixe $x \in K^m$ e tome $H := K^m \setminus \{x\}$. Como $\lambda_{K^m}(f,h) > 1-\delta^2$, pelo Parágrafo V.5–9, $\lambda_H(f,h) > 1 - 2\delta^2$. Veja agora que todo ponto em H define uma única linha em $\mathbb{L}_{\langle x\rangle}$. Assim, $\mathbb{L}_{\langle x\rangle}$ forma uma partição igualitária de H e com isto, dado o Parágrafo V.5–4, temos

$$\begin{aligned}
\underset{l\in\mathbb{L}_{\langle x\rangle}}{\mathcal{E}xp}\left(\, \lambda_{l\setminus\{x\}}(f,h)\,\right) &:= \underset{l\in\mathbb{L}_{\langle x\rangle}}{\mathcal{E}xp}\left(\, \underset{x'\in l\setminus\{x\}}{\mathcal{P}r}\{\, f(x') = h(x')\,\}\,\right) \\
&= \underset{x'\in H}{\mathcal{P}r}\{\, f(x') = h(x')\,\} \;=:\; \lambda_H(f,h) \\
&> 1 - 2\delta^2 \;=\; 1 - \delta(2\delta).
\end{aligned}$$

Logo, novamente pelo Parágrafo V.5–6,

$$\underset{l\in\mathbb{L}_{\langle x\rangle}}{\mathcal{P}r}\left\{\, \lambda_{l\setminus\{x\}}(f,h) > 1-\delta \,\right\} \;>\; 1 - 2\delta$$

e, pelo item (i), terminamos a prova.

[V.4–5] ■

V.4–6. A partir de agora, fixaremos $d \geq 4$, $k \geq \max\left\{b; 132d^3\right\}$ [xi] e $p \geq 2d+1$. Assim, estamos supondo que o corpo K é suficientemente grande.

Proposição V.4–7. *Se $m = 2$, $\delta \in (0, 1/3]$ e $\Lambda(f) > 1 - \delta/40$, então, para todo $x \in K^2$,*

$$\underset{l\in\mathbb{L}_{\langle x\rangle}}{\mathcal{P}r}\left\{\, \{l\} \textit{ é } \delta\textit{–coerente com } f \textit{ em } x \textit{ sobre } \mathcal{P}^f(x)\,\right\} \;>\; 1-\delta.$$

Prova. Pelo Corolário da Proximidade a um Polinômio de Duas Variáveis que tem Grau Total Baixo V.3–2, como $\delta/40 \leq 1/120 \leq 1/84$ e $\Lambda(f) > 1 - \delta/40$, temos que $\lambda(f) := \lambda\left(f, \mathcal{P}^f\right) > 1 - \delta/2$. Agora, pela Proposição V.4–5.(ii), como $\mathcal{P}^f \in \mathbb{K}^2$ e $2\delta \leq 1/4$, para todo $x \in K^2$,

$$\underset{l\in\mathbb{L}_{\langle x\rangle}}{\mathcal{P}r}\left\{\, \{l\} \text{ é } \delta\text{–coerente com } f \text{ em } x \text{ sobre } \mathcal{P}^f(x)\,\right\} \;>\; 1-\delta,$$

[xi] Cumpre-nos lembrar que $b := 2^{27}3^75^5$.

como queríamos.

[V.4–7] ■

V.4–8. Precisamos, agora, fazer algumas contas com constantes que iremos definir. Sejam as constantes abaixo, as quais nos ajudarão nos nossos próximos passos,

$$\begin{aligned} \boldsymbol{a} &:= 240 = 2^4 \cdot 3 \cdot 5, \\ \boldsymbol{\delta} &\in \left(0, \frac{1}{b} \right], \\ \boldsymbol{\gamma}^2 &:= 40\left(\delta + \frac{2}{k}\right) \quad \text{e} \\ \boldsymbol{\upsilon} &:= 40\left(\gamma + \frac{1}{a}\right). \end{aligned}$$

Primeiramente, lembremos que $b := 2^{27}3^75^5$ e vejamos que

$$\frac{1}{k+1} \leq \frac{1}{b+1} \leq \frac{1}{2a} < \frac{1}{a}. \tag{V.4:2}$$

Como

$$\gamma^2 := 40\left(\delta + \frac{2}{k}\right) \geq \frac{1}{k} \geq \frac{1}{k^2},$$

e

$$\gamma^2 := 40\left(\delta + \frac{2}{k}\right) \leq 40\frac{3}{b} = \frac{1}{48^2a^4},$$

$$\frac{1}{k} \leq \gamma \leq \frac{1}{48a^2} < \frac{1}{24a} \leq \frac{1}{3}. \tag{V.4:3}$$

Portanto, veja que

$$\gamma \leq \frac{1}{a} \leq \frac{1}{12},$$

logo,

$$1 - 6\gamma \geq \frac{1}{2} \tag{V.4:4}$$

e

$$\upsilon := 40\left(\gamma + \frac{1}{a}\right) \leq 40\frac{2}{a} = \frac{1}{3}. \tag{V.4:5}$$

Como $1/a \leq 1/2$ e $\upsilon \leq 1/2$,

$$\upsilon + \frac{1}{a} \leq 1. \tag{V.4:6}$$

Agora podemos ir às demonstrações.

Proposição V.4–9. *Se $m \geq 2$ e $\Lambda(f) > 1 - \delta$, então, para todo $x \in K^m$, temos*

$$\underset{l \in \mathbb{L}_{\langle x \rangle}}{\mathcal{P}r} \left\{ \{l\} \text{ é } \gamma\text{–coerente com } f \text{ em } x \text{ sobre } \widehat{f}(x) \right\} > 1 - 6\gamma.$$

Prova. Fixe $x \in K^m$. Como $\Lambda(f) > 1 - \delta$,

$$\underset{s \in \mathbb{S}_{\langle x \rangle}}{\mathcal{E}xp}\ (\ \Lambda_s(f)\ ) =: \Lambda_{\mathbb{S}_{\langle x \rangle}}(f) \geq \Lambda(f) - \frac{2}{k} > 1 - \left(\delta + \frac{2}{k}\right) =: 1 - \frac{\gamma^2}{40},$$

primeiro, pela Proposição V.4–3, e depois, pela definição de γ. Portanto, do Parágrafo V.5–6,

$$\underset{s\in\mathbb{S}_{\langle x\rangle}}{\mathcal{P}r}\left\{\ \Lambda_s(f)\ >\ 1-\frac{\gamma}{40}\ \right\}\ >\ 1-\gamma$$

e pela Equação (V.4:3), temos que $\gamma \leq 1/3$. Assim, podemos utilizar da Proposição V.4–7, sobre cada plano $s \in \mathbb{S}_{\langle x\rangle}$, e

$$\underset{s\in\mathbb{S}_{\langle x\rangle}}{\mathcal{P}r}\left\{\ \underset{l\in s_{\langle x\rangle}}{\mathcal{P}r}\ \{\ \{l\}\text{ é }\gamma\text{–coerente com }f\text{ em }x\text{ sobre }\mathcal{P}^{f,s}(x)\ \}>1-\gamma\ \right\}\ >\ 1-\gamma,$$

ou, simplesmente,

$$\underset{s\in\mathbb{S}_{\langle x\rangle}}{\mathcal{P}r}\left\{\ \underset{l\in s_{\langle x\rangle}}{\mathcal{P}r}\ \{\ \{l\}\text{ é }\gamma\text{–coerente com }f\text{ em }x\ \}>1-\gamma\ \right\}\ >\ 1-\gamma.$$

Mas, pelo Parágrafo V.5–10,[xii]

$$\underset{s\in\mathbb{S}_{\langle x\rangle}}{\mathcal{P}r}\left\{\ \underset{\{l,l'\}\in\binom{s_{\langle x\rangle}}{2}}{\mathcal{P}r}\ \{\ \{l,l'\}\text{ é }\gamma\text{–coerente com }f\text{ em }x\ \}>1-4\gamma\ \right\}\ >\ 1-\gamma,$$

ou seja,

$$\underset{s\in\mathbb{S}_{\langle x\rangle}}{\mathcal{E}xp}\left(\ \underset{\{l,l'\}\in\binom{s_{\langle x\rangle}}{2}}{\mathcal{P}r}\ \{\ \{l,l'\}\text{ é }\gamma\text{–coerente com }f\text{ em }x\ \}\ \right)\ >\ 1-5\gamma,$$

pelo Parágrafo V.5–5. Como cada duas linhas distintas de $\mathbb{L}_{\langle x\rangle}$ (ou seja, cada elemento de $\binom{\mathbb{L}_{\langle x\rangle}}{2}$) definem um único plano de $\mathbb{S}_{\langle x\rangle}$, $\mathbb{S}_{\langle x\rangle}$ forma uma partição igualitária de $\binom{\mathbb{L}_{\langle x\rangle}}{2}$. Pelo Parágrafo V.5–4,

$$\underset{\{l,l'\}\in\binom{\mathbb{L}_{\langle x\rangle}}{2}}{\mathcal{P}r}\ \{\ \{l,l'\}\text{ é }\gamma\text{–coerente com }f\text{ em }x\ \}\ >\ 1-5\gamma,$$

e, novamente do Parágrafo V.5–6,

$$\underset{l\in\mathbb{L}_{\langle x\rangle}}{\mathcal{P}r}\left\{\ \underset{l'\in\mathbb{L}_{\langle x\rangle}\smallsetminus\{l\}}{\mathcal{P}r}\ \{\ \{l,l'\}\text{ é }\gamma\text{–coerente com }f\text{ em }x\ \}>1-6\gamma\ \right\}\ >\ 1-\frac{5}{6}\geq 0.$$

Portanto, existe $\widehat{l}\in\mathbb{L}_{\langle x\rangle}$, tal que

$$\underset{l\in\mathbb{L}_{\langle x\rangle}\smallsetminus\{\widehat{l}\}}{\mathcal{P}r}\left\{\ \left\{\widehat{l},l\right\}\text{é }\gamma\text{–coerente com }f\text{ em }x\ \right\}\ >\ 1-6\gamma.$$

Então, todas as linhas válidas nesta probabilidade são coerentes entre si e o são sobre o ponto $\mathcal{P}^{f,\widehat{l}}(x)\in K^m$. Logo,

$$\underset{l\in\mathbb{L}_{\langle x\rangle}}{\mathcal{P}r}\left\{\ \{l\}\text{ é }\gamma\text{–coerente com }f\text{ em }x\text{ sobre }\mathcal{P}^{f,\widehat{l}}(x)\ \right\}\ >\ 1-6\gamma\ \geq\ \frac{1}{2},$$

sendo que a última desigualdade é devida à Equação (V.4:4). Com isto, pela sua definição, $\widehat{f}(x)=\mathcal{P}^{f,\widehat{l}}(x)$ e concluímos a prova.

[V.4–9] ■

V.4–10. Sejam $x\in K^m$, $l\in\mathbb{L}_{\langle x\rangle}$ e $s\in\mathbb{S}_{\langle l\rangle}$. Então, diremos que ***s*** `tem boa coerência para` ***x*** sse

$$\underset{l'\in s_{\langle x\rangle}}{\mathcal{P}r}\left\{\ \{l'\}\text{ é }\gamma\text{–coerente com }f\text{ em }x\text{ sobre }\widehat{f}(x)\ \right\}\ >\ 1-\frac{1}{a}.$$

Por sua vez, o plano ***s*** `é bom sobre` ***x*** `e` ***l*** sse

[xii] Veja, também, no Parágrafo V.5–10 a definição de $\binom{s_{\langle x\rangle}}{2}$.

- s tem boa coerência para x e
- $\underset{x' \in l}{\mathcal{P}r}\{\ s \text{ tem boa coerência para } x'\ \} \ > \ 1 - \dfrac{1}{a}$.

Proposição V.4–11. *Se $m \geq 2$, $x \in K^m$, $l \in \mathbb{L}_{\langle x \rangle}$ e $\Lambda(f) > 1 - \delta$, então existe um plano bom $s \in \mathbb{S}_{\langle l \rangle}$ sobre x e l.*

Prova. Fixe um $x' \in l$ qualquer. Sobre o resultado da Proposição V.4–9,

$$\underset{l' \in \mathbb{L}_{\langle x' \rangle}}{\mathcal{P}r}\left\{\ \{l'\} \text{ é } \gamma\text{–coerente com } f \text{ em } x' \text{ sobre } \widehat{f}(x')\ \right\} \ > \ 1 - 6\gamma,$$

podemos utilizar o Parágrafo V.5–9 e obtemos

$$\underset{l' \in \mathbb{L}_{\langle x' \rangle} \smallsetminus \{l\}}{\mathcal{P}r}\left\{\ \{l'\} \text{ é } \gamma\text{–coerente com } f \text{ em } x' \text{ sobre } \widehat{f}(x')\ \right\} \ > \ 1 - 12\gamma.$$

Então, veja os planos $\{\ s_{\langle x' \rangle} \smallsetminus \{l\} \parallel s \in \mathbb{S}_{\langle l \rangle}\ \}$ formando uma partição igualitária de $\mathbb{L}_{\langle x' \rangle} \smallsetminus \{l\}$. Isto, visto que cada linha de $\mathbb{L}_{\langle x' \rangle} \smallsetminus \{l\}$ define um único plano de $\mathbb{S}_{\langle l \rangle}$ e cada plano de $\mathbb{S}_{\langle l \rangle}$ contém o mesmo número de linhas que passam por x' e são distintas de l. Portanto, pelo Parágrafo V.5–4,

$$\underset{s \in \mathbb{S}_{\langle l \rangle}}{\mathcal{E}xp}\left(\ \underset{l' \in s_{\langle x' \rangle} \smallsetminus \{l\}}{\mathcal{P}r}\left\{\ \{l'\} \text{ é } \gamma\text{–coerente com } f \text{ em } x' \text{ sobre } \widehat{f}(x')\ \right\}\ \right) \ > \ 1 - 12\gamma.$$

Disto temos que

$$\begin{aligned}
&\underset{s \in \mathbb{S}_{\langle l \rangle}}{\mathcal{P}r}\{\ s \text{ tem boa coerência para } x'\ \} \ := \\
&\qquad := \ \underset{s \in \mathbb{S}_{\langle l \rangle}}{\mathcal{P}r}\left\{\ \underset{l' \in s_{\langle x' \rangle}}{\mathcal{P}r}\left\{\ \{l'\} \text{ é } \gamma\text{–coerente com } f \text{ em } x' \text{ sobre } \widehat{f}(x')\ \right\} \ > \ 1 - \frac{1}{a}\ \right\} \\
&\qquad \geq \ \underset{s \in \mathbb{S}_{\langle l \rangle}}{\mathcal{P}r}\left\{\ \underset{l' \in s_{\langle x' \rangle} \smallsetminus \{l\}}{\mathcal{P}r}\left\{\ \{l'\} \text{ é } \gamma\text{–coerente com } f \text{ em } x' \text{ sobre } \widehat{f}(x')\ \right\} \ > \ 1 - \frac{1}{2a}\ \right\} \\
&\qquad > \ 1 - 24a\gamma,
\end{aligned}$$

sendo que a última desigualdade decorre do Parágrafo V.5–6, visto sobre o resultado acima, e a penúltima desigualdade é conseqüência do Parágrafo V.5–9. Mas, para utilizá-la, precisamos ver que, da Equação (V.4: 2), $\mathcal{P}r_{s_{\langle x' \rangle}}(\ l\ ) = 1/(k+1) \leq 1/(2a) < 1/2$.

Como tínhamos fixado $x' \in l$, concluímos que

$$\underset{x' \in l}{\mathcal{P}r}\left\{\ \underset{s \in \mathbb{S}_{\langle l \rangle}}{\mathcal{P}r}\{\ s \text{ tem boa coerência para } x'\ \} > 1 - 24a\gamma\ \right\} \ = \ 1.$$

Logo, da Equação (V.4: 3), obtemos que $24a\gamma < 1$, permitindo-nos usar o Parágrafo V.5–7, e assim

$$\underset{s \in \mathbb{S}_{\langle l \rangle}}{\mathcal{P}r}\left\{\ \underset{x' \in l}{\mathcal{P}r}\{\ s \text{ tem boa coerência para } x'\ \} > 1 - \frac{1}{a}\ \right\} \ > \ 1 - 24a^2\gamma.$$

Portanto, tomando-se a ante-penúltima equação para $x' := x$, temos

$$\begin{aligned}
&\underset{s \in \mathbb{S}_{\langle l \rangle}}{\mathcal{P}r}\{\ s \text{ é bom sobre } x \text{ e } l\ \} \ := \\
&\qquad := \ \underset{s \in \mathbb{S}_{\langle l \rangle}}{\mathcal{P}r}\left\{\ s \text{ tem boa coerência para } x \text{ e } \underset{x' \in l}{\mathcal{P}r}\{\ s \text{ tem boa coerência para } x'\ \} > 1 - \frac{1}{a}\ \right\} \\
&\qquad > \ 1 - \left(24a\gamma + 24a^2\gamma\right) \ \geq \ 1 - 48a^2\gamma \ \geq \ 0,
\end{aligned}$$

pela Equação (V.4:3) e terminamos a prova.

[V.4–11] ■

Proposição V.4–12. *Sejam $m \geq 2$, $x \in K^m$, $l \in \mathbb{L}_{\langle x \rangle}$ e $s \in \mathbb{S}_{\langle l \rangle}$, tal que s é bom sobre x e l. Então $\mathcal{P}^{f,s}(x) = \widehat{f}(x)$.*

Prova. Veja que

$$\begin{aligned} \underset{l' \in s_{\langle x \rangle}}{\mathcal{P}r} \{\ \lambda_{l'}(f) > 1 - 2\gamma\ \} &\geq \underset{l' \in s_{\langle x \rangle}}{\mathcal{P}r} \{\ \lambda_{l' \smallsetminus \{x\}}(f) > 1 - \gamma\ \} \\ &\geq \underset{l' \in s_{\langle x \rangle}}{\mathcal{P}r} \left\{\ \{l'\} \text{ é } \gamma\text{–coerente com } f \text{ em } x \text{ sobre } \widehat{f}(x)\ \right\} \\ &> 1 - \frac{1}{a}, \end{aligned}$$

pela ordem, do Parágrafo V.5–9 (já que, pela Equação (V.4:2), $\mathcal{P}r_{l'}(\ x\ ) = 1/k \leq \gamma < 1/2$), da definição de *coerente* e por s ser bom sobre x e l (i.e., s tem boa coerência para x). Logo, aplicando-se o Parágrafo V.5–5 nesta equação e pela definição de υ, chegamos a

$$\Lambda_{s_{\langle x \rangle}}(f) = \underset{l \in s_{\langle x \rangle}}{\mathcal{E}xp}(\ \lambda_l(f)\ ) > 1 - (2\gamma + \frac{1}{a}) =: 1 - \frac{\upsilon}{40}.$$

Já que $\upsilon \leq 1/3$ (pela Equação (V.4:5)), podemos utilizar, somente sobre o plano s, o resultado acima na Proposição V.4–7. Então, pela definição de *coerente*,

$$\underset{l' \in s_{\langle x \rangle}}{\mathcal{P}r} \left\{\ \mathcal{P}^{f,l'}(x) = \mathcal{P}^{f,s}(x)\ \right\} \geq \underset{l' \in s_{\langle x \rangle}}{\mathcal{P}r} \{\ \{l'\} \text{ é } \upsilon\text{–coerente com } f \text{ em } x \text{ sobre } \mathcal{P}^{f,s}(x)\ \} > 1 - \upsilon.$$

Por outro lado, como s tem boa coerência para x e, novamente, pela definição de *coerente*,

$$\underset{l' \in s_{\langle x \rangle}}{\mathcal{P}r} \left\{\ \mathcal{P}^{f,l'}(x) = \widehat{f}(x)\ \right\} \geq \underset{l' \in s_{\langle x \rangle}}{\mathcal{P}r} \left\{\ \{l'\} \text{ é } \gamma\text{–coerente com } f \text{ em } x \text{ sobre } \widehat{f}(x)\ \right\} > 1 - \frac{1}{a}.$$

Portanto,

$$\underset{l' \in s_{\langle x \rangle}}{\mathcal{P}r} \left\{\ \mathcal{P}^{f,s}(x) = \widehat{f}(x)\ \right\} \geq \underset{l' \in s_{\langle x \rangle}}{\mathcal{P}r} \left\{\ \left(\mathcal{P}^{f,s}(x) = \mathcal{P}^{f,l'}(x)\right) \text{ e } \left(\mathcal{P}^{f,l'}(x) = \widehat{f}(x)\right)\ \right\} > 1 - (\upsilon + \frac{1}{a}) \geq 0,$$

pela Equação (V.4:6). Chegamos, então, a $\mathcal{P}^{f,s}(x) = \widehat{f}(x)$, com queríamos.

[V.4–12] ■

Agora, finalmente,

Proposição V.4–13. *Tome $m \geq 2$ e $\Lambda(f) > 1 - \delta$, assim, $\widehat{f} \in \mathbb{K}^m$.*

Prova. Queremos provar que, para todo $l \in \mathbb{L}$, $\mathcal{P}^{\widehat{f},l} = \widehat{f}$ em l. Então, $\Lambda\left(\widehat{f}\right) = 1$ e, pelo Corolário V.2–3, temos a tese $\widehat{f} \in \mathbb{K}^m$. Logo, tome $l \in \mathbb{L}$. Para todo $x \in l$, pela Proposição V.4–11, dado que $\Lambda(f) > 1 - \delta$, existe um plano bom $s \in \mathbb{S}_{\langle l \rangle}$ sobre x e l e da Proposição V.4–12, $\mathcal{P}^{f,s}(x) = \widehat{f}(x)$. Resta-nos apenas mostrar que $\mathcal{P}^{f,s}(x) = \mathcal{P}^{\widehat{f},l}(x)$. Mas, como s é um plano bom sobre x e l,

$$\underset{x' \in l}{\mathcal{P}r} \{\ s \text{ tem boa coerência para } x'\ \} > 1 - \frac{1}{a},$$

que independe de x. Ora, então, como s tem boa coerência para todo $y \in l$, temos que s é bom sobre y e l. Ou seja,

$$\Pr_{y \in l}\{\ s \text{ é bom sobre } y \text{ e } l\ \} \ > \ 1 - \frac{1}{a}.$$

Portanto, utilizando-nos novamente do raciocínio anterior,

$$\lambda_l\left(\mathcal{P}^{f,s}, \widehat{f}\right) \ := \ \Pr_{y \in l}\left\{\ \mathcal{P}^{f,s}(x) = \widehat{f}(x)\ \right\} \ > \ 1 - \frac{1}{a} \ \geq \ \frac{1}{2} \ \geq \ \frac{1}{2}\left(1 - \frac{d}{k}\right),$$

visto que $a \geq 2$. Então, como $\mathcal{P}^{f,s}$ é um polinômio de duas variáveis e de grau total no máximo d, ele, restrito à linha l, é um polinômio de uma variável e de grau no máximo d. Finalizando, através de um rápido argumento sobre o Corolário da Distância dos Polinômios de Grau Baixo II.3–3, vemos que o próprio $\mathcal{P}^{f,s}$ é a melhor aproximação polinomial de $\widehat{f}$ sobre l. Ou seja, $\mathcal{P}^{f,s} = \mathcal{P}^{\widehat{f},l}$ em l, (e, em particular, sobre $x \in l$), como queríamos.

[V.4–13] ■

Pronto, já temos tudo para obtermos a

Prova do Teorema da Proximidade a um Polinômio de Grau Total Baixo V.1–5. Basta juntar os resultados das Proposições V.4–2 e V.4–13.

[V.1–5] ■

V.5 Propriedades Algébricas e Probabilísticas

Nesta seção, trabalharemos com alguns (mas importantes) "truques" probabilísticos, os quais foram utilizados no decorrer do capítulo. Mas, antes, veremos apenas alguns fatos sobre soluções de *sistemas lineares* pensados sobre polinômios de grau baixo (i.e., quando os escalares são os polinômios). Este assunto nos foi útil na Seção V.3 *Proximidade aos Polinômios de Duas Variáveis* e esmiuçá-lo-emos aqui.

V.5–1. Trabalharemos sobre um *domínio de integridade com unidade* $\mathcal{D}$. Seja o *sistema homogêneo* sobre $\mathcal{D}$

$$A \cdot t \ = \ 0,$$

com $A := A_{m \times n}$, uma matriz não nula de m linhas e de n colunas e t um vetor de n variáveis. Tomemos $B := B_{\ell \times \ell}$ como sendo uma das sub-matrizes quadradas de A de maior tamanho e que ainda tenha determinante não nulo.[xiii]

Sem perda de generalidade, suponha que A e t tenham a seguinte configuração,[xiv]

$$A \ := \ \left\| \begin{array}{c|c} B_{\ell \times \ell} & C_{\ell \times (n-\ell)} \\ \hline D_{(m-\ell) \times n} & \end{array} \right\|_{m \times n}$$

e

$$t \ := \ \left\| \frac{x_{\ell \times 1}}{y_{(n-\ell) \times 1}} \right\|_{n \times 1}.$$

[xiii] Portanto, se $\mathcal{D}$ fosse um corpo, B seria invertível.

[xiv] Aqui, as barras verticais e horizontais representam o local de acoplamento das sub-matrizes.

Portanto, se $\big\|B \mid C\big\| \cdot t = 0$ (em que t é uma solução para $\big\|B \mid C\big\|$), então $D \cdot t = 0$ e, em particular, $A \cdot t = 0$ (em que t também é uma solução para A). Neste caso,

$$0 \;=\; \big\|B \mid C\big\| \cdot t \;=\; \big\|B \mid C\big\| \cdot \left\| \frac{x}{y} \right\| \;=\; B \cdot x + C \cdot y,$$

e, portanto,

$$B \cdot x \;=\; -C \cdot y.$$

Observe que sempre $\ell \leq n$ e, no caso de serem iguais, quanto à notação, estaremos supondo $-C \cdot y = 0$. Agora, fixe

$$y \;:=\; \left\| \begin{matrix} 1 \\ \vdots \\ 1 \end{matrix} \right\|_{(n-\ell)\times 1}$$

e, pela *Regra de Cramer* para

$$x \;:=\; \left\| \begin{matrix} x_1 \\ \vdots \\ x_\ell \end{matrix} \right\|_{\ell\times 1},$$

temos, para todo $i \in \{1 \ldots \ell\}$,

$$\det(B)x_i \;=\; \det \left\| \; \mathcal{L}^2_{\{1\ldots i-1\}}(B) \;\mid -C \cdot y \mid\; \mathcal{L}^2_{\{i+1\ldots \ell\}}(B) \; \right\|,$$

em que $\mathcal{L}^2_{\{1\ldots i-1\}}(B)$ são as $i-1$ primeiras colunas de B e $\mathcal{L}^2_{\{i+1\ldots \ell\}}(B)$ são as $n-i-1$ últimas, conforme definido no Parágrafo III.3–1.

Pensemos, agora, o conjunto dos polinômios de uma variável como um *domínio de integridade*. Portanto, seja $\mathcal{D} := K[\varepsilon]$. Se ainda tivermos A como sendo uma matriz em que todas as suas entradas são polinômios de grau no máximo d (i.e., em $K[\varepsilon]_d$), então $\det(B)$ e $\det(B)x_i$ são polinômios de grau no máximo $\ell d \leq nd$, para todo $i \in \{1 \ldots \ell\}$. Mas veja que, se t é uma solução, $\det(B)t$ também o é. Em particular, se o sistema tem uma solução não trivial (i.e., se t for solução não nula), então existe uma outra solução não trivial (i.e., $\det(B)t \neq 0$), tal que tenha todas as suas entradas como sendo polinômios de grau no máximo nd. Para finalizar, note que, nestas condições, se $m < n$, então obrigatoriamente temos uma solução não nula com entradas em $K[\varepsilon]_d$.

Entremos, então, nas propriedades probabilísticas, que nos foram tão úteis neste capítulo.

V.5–2. Seja ν_X uma *medida finita* sobre o conjunto não vazio X. Sempre que falarmos de subconjuntos de X, supô-los-emos na respectiva σ-álgebra (ou seja, mensuráveis). Do mesmo modo, esperamos sempre estar falando de funções integráveis, quando tiramos as médias. Sejam φ uma `variável booleana` (ou seja, $\varphi\colon X \to \{\textit{verdadeiro, falso}\}$), f uma `variável aleatória` (isto é, $f\colon X \to \mathbb{R}$ [xv]) e $A \subseteq X$.[xvi] Conforme visto com a probabilidade,[xvii] denotaremos por $\nu_X(A)$ a medida ν_X de A e, se A não tiver medida nula e $B \subseteq X$,

$$\boldsymbol{\nu_{A\subseteq X}(\, B \,)} \;:=\; \frac{\nu_X(\, B \cap A \,)}{\nu_X(\, A \,)} \;\geq\; \nu_X(\, B \cap A \,)$$

será a medida ν_X restrita a A. Também denotaremos

$$\boldsymbol{\nu_{x\in A\subseteq X}\{\, \varphi(x) \,\}} \;:=\; \nu_{A\subseteq X}\,(\, \{x \in A \| \varphi(x)\} \,)\,.$$

[xv] Suporemos, ainda, f integrável.
[xvi] Sendo também, A uma σ-álgebra.
[xvii] Veja o Parágrafo I.1–1.

Agora, definimos a sua `medida média` por

$$\overline{\boldsymbol{\nu}}_{\boldsymbol{x\in A\subseteq X}}\big(\ \boldsymbol{f(x)}\ \big) \ := \ \int_{x\in A} f(x)\ \mathrm{d}\nu_{A\ \subseteq X}(x) \ = \ \frac{1}{\nu_X(A)}\int_{x\in A} f(x)\ \mathrm{d}\nu_X(x)$$

e

$$\overline{\boldsymbol{\nu}}_{\boldsymbol{x\in A\subseteq X}}\big[\ \boldsymbol{\varphi(x)}\ \big] \ := \ \overline{\nu}_{x\in A\subseteq X}\big(\ [\ \varphi(x)\ ]\ \big) \ = \ \nu_{x\in A\subseteq X}\{\ \varphi(x)\ \},$$

sendo que $[\ \varphi(x)\ ]$ é a função característica de $\varphi(x)$.[xviii] Novamente, se $A = X$ suprimi-lo-emos. Observe que, se ν_X é uma *probabilidade* $\mathcal{P}r_X$, (ou seja, uma medida, tal que $\mathcal{P}r_X(\ X\ ) = 1$), definimos a sua `esperança` (*condicional*) pela *probabilidade média* e temos

$$\underset{\boldsymbol{x\in A\subseteq X}}{\boldsymbol{\mathcal{E}xp}}\ \big(\ \boldsymbol{f(x)}\ \big) \ := \ \overline{\mathcal{P}r}_{x\in A\subseteq X}\big(\ f(x)\ \big) \ := \ \frac{1}{\mathcal{P}r_X(\ A\ )}\int_{x\in A} f(x)\ \mathrm{d}\underset{X}{\mathcal{P}r}(x)$$

e

$$\underset{\boldsymbol{x\in A\subseteq X}}{\boldsymbol{\mathcal{E}xp}}\ \big[\ \boldsymbol{\varphi(x)}\ \big] \ := \ \overline{\mathcal{P}r}_{x\in A\subseteq X}[\ \varphi(x)\ ] \ = \ \underset{x\in A\subseteq X}{\mathcal{P}r}\{\ \varphi(x)\ \}.$$

Muitas vezes, quando X está subentendido, denotaremos simplesmente

$$\underset{A}{\mathcal{P}r}(\ B\ ) := \underset{A\subseteq X}{\mathcal{P}r}(\ B\ ) \quad \text{e} \quad \underset{x\in A}{\mathcal{E}xp}(\ f(x)\ ) := \underset{x\in A\subseteq X}{\mathcal{E}xp}(\ f(x)\ ).$$

V.5–3. Sejam ν_X e ν_Y medidas sobre X e Y e f' uma variável aleatória para $X\times Y$. A cada $x\in X$ e $y\in Y$, definimos[xix]

$$\mathcal{L}^1_{\{x\}}(A) \ := \ \{\ y'\in Y \parallel (x,y')\in A\ \}$$

e

$$\mathcal{L}^2_{\{y\}}(A) \ := \ \{\ x'\in X \parallel (x',y)\in A\ \},$$

tal que $A\subseteq X\times Y$, de modo a que os $\mathcal{L}^1_{\{x\}}(A)$'s e $\mathcal{L}^2_{\{y\}}(A)$'s não sejam vazios. Então

$$\begin{aligned}
\overline{\nu}_{y\in Y}\left(\ \overline{\nu}_{x\in\mathcal{L}^2_{\{y\}}(A)\subseteq X}\left(\ f'(x,y)\frac{\nu_X(\mathcal{L}^2_{\{y\}}(A))}{\nu_X(x)}\ \right)\ \right) \ &= \\
&= \ \frac{1}{\nu_Y(Y)}\int_{y\in Y}\frac{1}{\nu_X(\mathcal{L}^2_{\{y\}}(A))}\int_{x\in\mathcal{L}^2_{\{y\}}(A)} f'(x,y)\frac{\nu_X(\mathcal{L}^2_{\{y\}}(A))}{\nu_X(x)}\ \mathrm{d}\nu_X(x)\ \mathrm{d}\nu_Y(y)\\
&= \ \frac{1}{\nu_Y(Y)}\int_{y\in Y}\int_{x\in\mathcal{L}^2_{\{y\}}(A)}\frac{f'(x,y)}{\nu_X(x)}\ \mathrm{d}\nu_X(x)\ \mathrm{d}\nu_Y(y)\\
&= \ \frac{1}{\nu_X(x)}\int_{x\in X}\int_{y\in\mathcal{L}^1_{\{x\}}(A)}\frac{f'(x,y)}{\nu_Y(Y)}\ \mathrm{d}\nu_Y(y)\ \mathrm{d}\nu_X(x)\\
&= \ \frac{1}{\nu_X(x)}\int_{x\in X}\frac{1}{\nu_Y(\mathcal{L}^1_{\{x\}}(A))}\int_{y\in\mathcal{L}^1_{\{x\}}(A)} f'(x,y)\frac{\nu_Y(\mathcal{L}^1_{\{x\}}(A))}{\nu_Y(Y)}\ \mathrm{d}\nu_Y(y)\ \mathrm{d}\nu_X(x)\\
&= \ \overline{\nu}_{x\in X}\left(\ \overline{\nu}_{y\in\mathcal{L}^1_{\{x\}}(A)\subseteq Y}\left(\ f'(x,y)\frac{\nu_Y(\mathcal{L}^1_{\{x\}}(A))}{\nu_Y(Y)}\ \right)\ \right).
\end{aligned}$$

Ou seja, nas probabilidades temos

$$\underset{y\in Y}{\mathcal{E}xp}\left(\ \underset{x\in\mathcal{L}^2_{\{y\}}(A)\subseteq X}{\mathcal{E}xp}\left(\ f'(x,y)\,\underset{X}{\mathcal{P}r}\left(\ \mathcal{L}^2_{\{y\}}(A)\ \right)\ \right)\ \right) \ = \ \underset{x\in X}{\mathcal{E}xp}\left(\ \underset{y\in\mathcal{L}^1_{\{x\}}(A)\subseteq Y}{\mathcal{E}xp}\left(\ f'(x,y)\,\underset{Y}{\mathcal{P}r}\left(\ \mathcal{L}^1_{\{x\}}(A)\ \right)\ \right)\ \right).$$

[xviii] Ela também pode ser vista no Parágrafo I.1–1.

[xix] Veja também o Parágrafo III.3–1.

Mais ainda, se $f'(x,y)$ independe de y (i.e., $f'(x,y) := f(x)$), temos

$$\mathop{\mathcal{E}xp}_{y\in Y}\left(\mathop{\mathcal{E}xp}_{x\in\mathcal{L}^2_{\{y\}}(A)\subseteq X}\Big(f(x)\,\mathop{\mathcal{P}r}_{X}\Big(\mathcal{L}^1_{\{y\}}(A)\Big)\Big)\right) \;=\; \mathop{\mathcal{E}xp}_{x\in X}\Big(f(x)\,\mathop{\mathcal{P}r}_{Y}\Big(\mathcal{L}^1_{\{x\}}(A)\Big)\Big).$$

V.5–4. No mesmo contexto, sejam, agora, $\mathcal{P}r_X$ e $\mathcal{P}r_Y$ *probabilidades uniformes*,

$$\widehat{X} \;:=\; \mathop{\overset{\circ}{\bigcup}}_{x\in X}\mathcal{L}^1_{\{x\}}(A) \;:=\; \Big\{\, (\mathcal{L}^1_{\{x\}}(A),x) \parallel x\in X \,\Big\}$$

e

$$\widehat{Y} \;:=\; \mathop{\overset{\circ}{\bigcup}}_{y\in Y}\mathcal{L}^2_{\{y\}}(A) \;:=\; \Big\{\, (\mathcal{L}^2_{\{y\}}(A),y) \parallel y\in Y \,\Big\}.$$

Muitas vezes, por um abuso notacional, denotaremos $x\in\widehat{X}$ e $\mathcal{L}^1_{\{x\}}(A)\in\widehat{X}$ com o significado óbvio. O mesmo valerá para o Y. Suporemos, ainda, $A\subseteq X\times Y$ tomado de tal forma que $\mathcal{P}r_X\Big(\mathcal{L}^2_{\{y\}}(A)\Big)$ e $\mathcal{P}r_Y\Big(\mathcal{L}^1_{\{x\}}(A)\Big)$ sejam constantes nos $\mathcal{L}^2_{\{y\}}(A)\in\widehat{Y}$ e nos $\mathcal{L}^1_{\{x\}}(A)\in\widehat{X}$, respectivamente. Com isto, sempre temos $\mathcal{P}r_X\Big(\mathcal{L}^2_{\{y\}}(A)\Big)=\mathcal{P}r_Y\Big(\mathcal{L}^1_{\{x\}}(A)\Big)$ e

$$\mathop{\mathcal{E}xp}_{y\in Y}\left(\mathop{\mathcal{E}xp}_{x\in\mathcal{L}^2_{\{y\}}(A)\subseteq X}(\,f'(x,y)\,)\right) \;=\; \mathop{\mathcal{E}xp}_{x\in X}\left(\mathop{\mathcal{E}xp}_{y\in\mathcal{L}^1_{\{x\}}(A)\subseteq Y}(\,f'(x,y)\,)\right),$$

ou ainda,

$$\mathop{\mathcal{E}xp}_{y\in Y}\left(\mathop{\mathcal{E}xp}_{x\in\mathcal{L}^2_{\{y\}}(A)\subseteq X}(\,f(x)\,)\right) \;=\; \mathop{\mathcal{E}xp}_{x\in X}(\,f(x)\,).$$

Neste sentido, diremos que $\widehat{\boldsymbol{Y}}$ `multi–particiona igualitariamente` $\widehat{\boldsymbol{X}}$ (ou, simplesmente $\boldsymbol{X}$). Isto significa dizer que Y particiona (via $\mathcal{L}^2_{\{y\}}(A)$, para os $y\in Y$) um número finito de uniões disjuntas de "cópias" de X (mais precisamente $\#\mathcal{L}^1_{\{x\}}(A)$ [xx] "cópias"). Deste modo, cada classe de equivalência desta partição está contida em uma única "cópia" de X e tem um número constante de elementos.[xxi] Assim, concluímos que

$$\mathop{\mathcal{E}xp}_{y\in Y}\left(\mathop{\mathcal{E}xp}_{x\in\mathcal{L}^2_{\{y\}}(A)}(\,f'(x,y)\,)\right) \;=\; \mathop{\mathcal{E}xp}_{x\in X}\left(\mathop{\mathcal{E}xp}_{y\in\mathcal{L}^1_{\{x\}}(A)}(\,f'(x,y)\,)\right),$$

ou

$$\mathop{\mathcal{E}xp}_{y\in Y}\left(\mathop{\mathcal{E}xp}_{x\in\mathcal{L}^2_{\{y\}}(A)}(\,f(x)\,)\right) \;=\; \mathop{\mathcal{E}xp}_{x\in X}(\,f(x)\,).$$

Se $\mathcal{L}^2_{\{y\}}(A)\subseteq X$ for unitário, diremos simplesmente que $\widehat{\boldsymbol{X}}$ `é particionado igualitariamente por` $\widehat{\boldsymbol{Y}}$. Por sua vez, se φ' é uma variável booleana para $X\times Y$, também temos,

$$\mathop{\mathcal{E}xp}_{y\in Y}\left(\mathop{\mathcal{P}r}_{x\in\mathcal{L}^2_{\{y\}}(A)}\{\,\varphi'(x,y)\,\}\right) \;=\; \mathop{\mathcal{E}xp}_{x\in X}\left(\mathop{\mathcal{P}r}_{y\in\mathcal{L}^1_{\{x\}}(A)}\{\,\varphi'(x,y)\,\}\right),$$

[xx] Note que $\#\mathcal{L}^1_{\{x\}}(A)$ é constante em $x\in X$.

[xxi] Por sua vez, também podemos dizer que $\widehat{X}$ multi–particiona igualitariamente $\widehat{Y}$ (ou, simplesmente Y), pois, neste caso, não há distinção entre X e Y.

ou

$$\underset{y\in Y}{\mathcal{E}xp}\left(\underset{x\in\mathcal{L}^2_{\{y\}}(A)}{\mathcal{P}r}\{\ \varphi(x)\ \}\right) \quad = \quad \underset{x\in X}{\mathcal{P}r}\{\ \varphi(x)\ \}.$$

V.5–5. Mudando de assunto, sejam também $\alpha,\beta,\gamma\in\mathbb{R}_+$. Logo, se temos $\beta<1$ e

$$\underset{x\in X}{\mathcal{P}r}\left\{\ \underset{y\in\mathcal{L}^1_{\{x\}}(A)}{\mathcal{P}r}\{\ \varphi'(x,y)\ \}\geq 1-\beta\ \right\} \quad := \quad \underset{x\in X}{\mathcal{E}xp}\left[\ \underset{y\in\mathcal{L}^1_{\{x\}}(A)}{\mathcal{E}xp}[\ \varphi'(x,y)\ ]\geq 1-\beta\ \right] \quad > \quad 1-\alpha,$$

ou,

$$\underset{x\in X}{\mathcal{P}r}\left\{\ \underset{y\in\mathcal{L}^1_{\{x\}}(A)}{\mathcal{P}r}\{\ \varphi'(x,y)\ \}>1-\beta\ \right\} \quad := \quad \underset{x\in X}{\mathcal{E}xp}\left[\ \underset{y\in\mathcal{L}^1_{\{x\}}(A)}{\mathcal{E}xp}[\ \varphi'(x,y)\ ]>1-\beta\ \right] \quad \geq \quad 1-\alpha,$$

então vale

$$\underset{x\in X}{\mathcal{E}xp}\left(\underset{y\in\mathcal{L}^1_{\{x\}}(A)}{\mathcal{P}r}\{\ \varphi'(x,y)\ \}\right) \quad := \quad \underset{x\in X}{\mathcal{E}xp}\left(\underset{y\in\mathcal{L}^1_{\{x\}}(A)}{\mathcal{E}xp}[\ \varphi'(x,y)\ ]\right) \quad > \quad 1-(\beta+\alpha).$$

Em particular, se $A:=X\times Y$, $\mathcal{L}^1_{\{x\}}(A)=Y$, para todo $x\in X$. Com efeito, vejamos apenas o primeiro caso. Tome $x\in X$ e portanto,[xxii]

$$\left[\ \underset{y\in\mathcal{L}^1_{\{x\}}(A)}{\mathcal{E}xp}[\ \varphi'(x,y)\ ]\geq 1-\beta\ \right] \quad = \quad \left[\ \frac{\mathcal{E}xp_{y\in\mathcal{L}^1_{\{x\}}(A)}[\ \varphi'(x,y)\ ]}{1-\beta}\geq 1\ \right] \quad = \quad \begin{cases} 1, & \text{se } \dfrac{\mathcal{E}xp_{y\in\mathcal{L}^1_{\{x\}}(A)}[\ \varphi'(x,y)\ ]}{1-\beta}\geq 1 \text{ e} \\ 0, & \text{caso contrário.}\end{cases}$$

Logo,

$$\frac{\mathcal{E}xp_{y\in\mathcal{L}^1_{\{x\}}(A)}[\ \varphi'(x,y)\ ]}{1-\beta} \quad \geq \quad \left[\ \underset{y\in\mathcal{L}^1_{\{x\}}(A)}{\mathcal{E}xp}[\ \varphi'(x,y)\ ]\geq 1-\beta\ \right],$$

e

$$\underset{x\in X}{\mathcal{E}xp}\left(\frac{\mathcal{E}xp_{y\in\mathcal{L}^1_{\{x\}}(A)}[\ \varphi'(x,y)\ ]}{1-\beta}\right) \quad \geq \quad \underset{x\in x}{\mathcal{E}xp}\left[\ \underset{y\in\mathcal{L}^1_{\{x\}}(A)}{\mathcal{E}xp}[\ \varphi'(x,y)\ ]\geq 1-\beta\ \right] \quad > \quad 1-\alpha.$$

Assim,

$$\underset{x\in X}{\mathcal{E}xp}\left(\underset{y\in\mathcal{L}^1_{\{x\}}(A)}{\mathcal{E}xp}[\ \varphi'(x,y)\ ]\right) \quad > \quad (1-\beta)(1-\alpha) \quad \geq \quad 1-(\beta+\alpha).$$

V.5–6. Agora, se $\beta>0$ e

$$\underset{x\in X}{\mathcal{E}xp}\left(\underset{y\in\mathcal{L}^1_{\{x\}}(A)}{\mathcal{P}r}\{\ \varphi'(x,y)\ \}\right) \quad := \quad \underset{x\in X}{\mathcal{E}xp}\left(\underset{y\in\mathcal{L}^1_{\{x\}}(A)}{\mathcal{E}xp}[\ \varphi'(x,y)\ ]\right) \quad > \quad 1-\beta\alpha,$$

[xxii] Lembre-se primeiro que, conforme definido no Parágrafo I.1–1, a função característica [*afirmação*] é 1 se a *afirmação* é válida e 0 caso contrário.

temos

$$\mathop{\mathcal{P}r}_{x\in X}\left\{\ \mathop{\mathcal{P}r}_{y\in\mathcal{L}^1_{\{x\}}(A)}\{\ \varphi'(x,y)\ \}>1-\beta\ \right\}\ :=\ \mathop{\mathcal{E}xp}_{x\in X}\left[\ \mathop{\mathcal{E}xp}_{y\in\mathcal{L}^1_{\{x\}}(A)}[\ \varphi'(x,y)\ ]>1-\beta\ \right]\ >\ 1-\alpha.$$

Novamente, se $A:=X\times Y$, $\mathcal{L}^1_{\{x\}}(A)=Y$, para todo $x\in X$. Suponha, por absurdo, que

$$\mathop{\mathcal{E}xp}_{x\in X}\left[\ \mathop{\mathcal{E}xp}_{y\in\mathcal{L}^1_{\{x\}}(A)}[\ \varphi'(x,y)\ ]>1-\beta\ \right]\ \leq\ 1-\alpha.$$

Então,

$$\mathop{\mathcal{E}xp}_{x\in X}\left[\ \mathop{\mathcal{E}xp}_{y\in\mathcal{L}^1_{\{x\}}(A)}[\ \neg\varphi'(x,y)\ ]\geq\beta\ \right]\ =\ \mathop{\mathcal{E}xp}_{x\in X}\left[\ \mathop{\mathcal{E}xp}_{y\in\mathcal{L}^1_{\{x\}}(A)}[\ \varphi'(x,y)\ ]\leq 1-\beta\ \right]\ \geq\ \alpha.$$

Agora, analogamente ao parágrafo anterior,

$$\left[\ \mathop{\mathcal{E}xp}_{y\in\mathcal{L}^1_{\{x\}}(A)}[\ \neg\varphi'(x,y)\ ]\geq\beta\right]\ =\ \left[\ \frac{\mathcal{E}xp_{y\in\mathcal{L}^1_{\{x\}}(A)}[\ \neg\varphi'(x,y)\ ]}{\beta}\ \geq 1\right]\ =\ \begin{cases}1, & \text{se } \dfrac{\mathcal{E}xp_{y\in\mathcal{L}^1_{\{x\}}(A)}[\ \neg\varphi'(x,y)\ ]}{\beta}\geq 1 \text{ e}\\[2ex] 0, & \text{caso contrário.}\end{cases}$$

Portanto,

$$\frac{\mathcal{E}xp_{y\in\mathcal{L}^1_{\{x\}}(A)}[\ \neg\varphi'(x,y)\ ]}{\beta}\ \geq\ \left[\ \mathop{\mathcal{E}xp}_{y\in\mathcal{L}^1_{\{x\}}(A)}[\ \neg\varphi'(x,y)\ ]\geq\beta\ \right]$$

e

$$\mathop{\mathcal{E}xp}_{x\in X}\left(\ \frac{\mathcal{E}xp_{y\in\mathcal{L}^1_{\{x\}}(A)}[\ \neg\varphi'(x,y)\ ]}{\beta}\ \right)\ \geq\ \mathop{\mathcal{E}xp}_{x\in X}\left[\ \mathop{\mathcal{E}xp}_{y\in\mathcal{L}^1_{\{x\}}(A)}[\ \neg\varphi'(x,y)\ ]\geq\beta\ \right]\ \geq\ \alpha.$$

Logo,

$$\mathop{\mathcal{E}xp}_{x\in X}\left(\ \mathop{\mathcal{E}xp}_{y\in\mathcal{L}^1_{\{x\}}(A)}[\ \neg\varphi'(x,y)\ ]\ \right)\ \geq\ \beta\alpha$$

e

$$\mathop{\mathcal{E}xp}_{x\in X}\left(\ \mathop{\mathcal{E}xp}_{y\in\mathcal{L}^1_{\{x\}}(A)}[\ \varphi'(x,y)\ ]\ \right)\ \leq\ 1-\beta\alpha,$$

contrariando a hipótese.

V.5–7. Novamente, quando $A:=X\times Y$, como temos $\mathcal{L}^1_{\{x\}}(A)=Y$, para todo $x\in X$. Neste caso, podemos trocar a ordem das *esperanças* e, se $\beta<1$ e $\gamma>0$, obtemos, para

$$\mathop{\mathcal{P}r}_{x\in X}\left\{\ \mathop{\mathcal{P}r}_{y\in Y}\{\ \varphi'(x,y)\ \}\geq 1-\beta\ \right\}\ :=\ \mathop{\mathcal{E}xp}_{x\in X}\left[\ \mathop{\mathcal{E}xp}_{y\in Y}[\ \varphi'(x,y)\ ]\geq 1-\beta\ \right]\ >\ 1-\alpha,$$

ou,

$$\mathop{\mathcal{P}r}_{x\in X}\left\{\ \mathop{\mathcal{P}r}_{y\in Y}\{\ \varphi'(x,y)\ \}>1-\beta\ \right\}\ :=\ \mathop{\mathcal{E}xp}_{x\in X}\left[\ \mathop{\mathcal{E}xp}_{y\in Y}[\ \varphi'(x,y)\ ]>1-\beta\ \right]\ \geq\ 1-\alpha,$$

a tese

$$\mathop{\mathcal{P}r}_{y\in Y}\left\{\ \mathop{\mathcal{P}r}_{x\in X}\{\ \varphi'(x,y)\ \}>1-\gamma\ \right\}\ :=\ \mathop{\mathcal{E}xp}_{y\in Y}\left[\ \mathop{\mathcal{E}xp}_{x\in X}[\ \varphi'(x,y)\ ]>1-\gamma\ \right]\ >\ 1-\frac{\beta+\alpha}{\gamma}.$$

V.5–8. Veremos, agora, mais algumas propriedades para $\mathcal{P}r_X$ (uma probabilidade sobre X), com esperança $\mathcal{E}xp_X$, $A \subseteq X$, tal que $X' := X \smallsetminus A$ não tenha probabilidade nula, e $f\colon X \to [0,1]$ uma variável aleatória. Logo

$$\underset{x\in X'}{\mathcal{E}xp}\,(\ f(x)\ ) - \underset{X}{\mathcal{P}r}(\ A\ ) \ \leq\ \underset{x\in X}{\mathcal{E}xp}(\ f(x)\ ) \ \leq\ \underset{x\in X'}{\mathcal{E}xp}\,(\ f(x)\ ) + \underset{X}{\mathcal{P}r}(\ A\ ).$$

Com efeito,[xxiii]

$$\begin{aligned}
\underset{x\in X}{\mathcal{E}xp}(\ f(X)\ ) &= \underset{x\in X}{\mathcal{E}xp}(\ f(x)[x\notin A]\ ) + \underset{x\in X}{\mathcal{E}xp}(\ f(x)[x\in A]\ ) \\
&\leq \underset{x\in X}{\mathcal{E}xp}(\ f(x)[x\in X']\ ) + \underset{x\in X}{\mathcal{E}xp}(\ [x\in A]\ ) \\
&\leq \underset{x\in X'}{\mathcal{E}xp}\,(\ f(x)\ ) + \underset{x\in X}{\mathcal{P}r}\{\ x\in A\ \} \\
&= \underset{x\in X'}{\mathcal{E}xp}\,(\ f(x)\ ) + \underset{X}{\mathcal{P}r}(\ A\ )
\end{aligned}$$

e, analogamente,

$$\underset{x\in X}{\mathcal{E}xp}(\ 1-f(x)\ ) \ \leq\ \underset{x\in X'}{\mathcal{E}xp}\,(\ 1-f(x)\ ) + \underset{X}{\mathcal{P}r}(\ A\ ),$$

logo

$$\underset{x\in X'\subseteq X}{\mathcal{E}xp}\,(\ f(x)\ ) - \underset{X}{\mathcal{P}r}(\ A\ ) \ \leq\ \underset{x\in X}{\mathcal{E}xp}(\ f(x)\ ).$$

Com isto, se φ é uma variável booleana,

$$\underset{x\in X'}{\mathcal{E}xp}[\ \varphi(x)\ ] - \underset{X}{\mathcal{P}r}(\ A\ ) \ \leq\ \underset{x\in X}{\mathcal{E}xp}[\ \varphi(x)\ ] \ \leq\ \underset{x\in X'}{\mathcal{E}xp}\,[\ \varphi(x)\ ] + \underset{X}{\mathcal{P}r}(\ A\ )$$

ou

$$\underset{x\in X'}{\mathcal{P}r}\{\ \varphi(x)\ \} - \underset{X}{\mathcal{P}r}(\ A\ ) \ \leq\ \underset{x\in X}{\mathcal{P}r}\{\ \varphi(x)\ \} \ \leq\ \underset{x\in X'}{\mathcal{P}r}\{\ \varphi(x)\ \} + \underset{X}{\mathcal{P}r}(\ A\ ).$$

Ainda, se $\varphi(x)$ é falso para todo $x \in A$, então $\mathcal{E}xp_{x\in X}[\ \varphi(x)\ ] \leq \mathcal{E}xp_{x\in X'}[\ \varphi(x)\ ]$ e, se $\varphi(x)$ é verdadeiro para todo $x \in A$, então $\mathcal{E}xp_{x\in X'}[\ \varphi(x)\ ] \leq \mathcal{E}xp_{x\in X}[\ \varphi(x)\ ]$.

V.5–9. Continuando, se X é finito, com mais de dois elementos e com probabilidade $\mathcal{P}r_X$ uniforme, $\#A = 1$ e $\delta \in (0,1]$, então,

$$\text{se}\quad \underset{x\in X}{\mathcal{E}xp}[\ \varphi(x)\ ] = \underset{x\in X}{\mathcal{P}r}\{\ \varphi(x)\ \} > 1-\delta \quad\text{então}\quad \underset{x\in X'}{\mathcal{E}xp}\,(\ \varphi(x)\ ) = \underset{x\in X'}{\mathcal{P}r}\{\ \varphi(x)\ \} > 1-2\delta$$

e

$$\text{se}\quad \underset{x\in X}{\mathcal{E}xp}[\ \varphi(x)\ ] = \underset{x\in X}{\mathcal{P}r}\{\ \varphi(x)\ \} < \delta \quad\text{então}\quad \underset{x\in X'}{\mathcal{E}xp}\,[\ \varphi(x)\ ] = \underset{x\in X'}{\mathcal{P}r}\{\ \varphi(x)\ \} < 2\delta.$$

No primeiro caso, suponha que $\mathcal{P}r_{x\in X}\{\ \varphi(x)\ \} > 1-\delta$. Logo, se $\mathcal{P}r_X(\ A\ ) \geq \delta$, $\mathcal{P}r_{x\in X}\{\ \varphi(x)\ \} > 1-\delta \geq 1-\mathcal{P}r_X(\ A\ )$. Portanto, $\mathcal{P}r_{x\in X}\{\ \varphi(x)\ \} = 1$ e, também, $\mathcal{P}r_{x\in X'}\{\ \varphi(x)\ \} = 1 > 1-2\delta$. Por outro lado, se $\mathcal{P}r_X(\ A\ ) \leq \delta$, do parágrafo anterior, temos que $\mathcal{P}r_{x\in X'}\{\ \varphi(x)\ \} \geq \mathcal{P}r_{x\in X}\{\ \varphi(x)\ \} - \mathcal{P}r_X(\ A\ ) > 1-2\delta$. No segundo caso, basta tomar a negação de φ, como já foi feito anteriormente.

Por sua vez, se ainda mais $\delta \in [\mathcal{P}r_X(\ A\ ), 1/2)$, então, também temos,

$$\text{se}\quad \underset{x\in X'}{\mathcal{E}xp}\,[\ \varphi(x)\ ] = \underset{x\in X'}{\mathcal{P}r}\{\ \varphi(x)\ \} > 1-\delta \quad\text{então}\quad \underset{x\in X}{\mathcal{E}xp}(\ \varphi(x)\ ) = \underset{x\in X}{\mathcal{P}r}\{\ \varphi(x)\ \} > 1-2\delta$$

[xxiii] Veja que $[x \in A]$ é a função característica da pertinência de x a A, conforme definido no Parágrafo I.1–1.

e

$$\text{se} \quad \underset{x\in X'}{\mathcal{E}xp}[\ \varphi(x)\ ] = \underset{x\in X'}{\mathcal{P}r}\{\ \varphi(x)\ \} < \delta \quad \text{então} \quad \underset{x\in X}{\mathcal{E}xp}[\ \varphi(x)\ ] = \underset{x\in X}{\mathcal{P}r}\{\ \varphi(x)\ \} < 2\delta.$$

V.5–10. Por sua vez, definiremos o *conjunto dos subconjuntos de dois elementos de* $A \subseteq X$ por

$$\binom{\boldsymbol{A}}{\boldsymbol{2}} := \{\ \{x,y\} \subseteq A \parallel x \neq y\ \},$$

e, se a probabilidade ainda é uniforme, temos

$$\text{se} \quad \underset{x\in X}{\mathcal{P}r}\{\ \varphi(x)\ \} > 1-\delta \quad \text{então} \quad \underset{\{x,y\}\in\binom{X}{2}}{\mathcal{P}r}\{\ \varphi(x) \ \text{ e } \ \varphi(y)\ \} > 1-4\delta.$$

Vejamos. Se $A := \{x \in X \| \varphi(x)\}$ e $a \in A$, tome novamente $X' := X \smallsetminus \{a\}$. Logo $\mathcal{P}r_{x\in X'}\{\ \varphi(x)\ \} > 1-2\delta$. Portanto,

$$\begin{aligned}\underset{\{x,y\}\in\binom{X}{2}}{\mathcal{P}r}\{\ \varphi(x) \ \text{ e } \ \varphi(y)\ \} &= \frac{\binom{\#A}{2}}{\binom{\#X}{2}} = \frac{\#A(\#A-1)}{\#X(\#X-1)} \geq \left(\frac{\#A-1}{\#X-1}\right)^2 \\ &= \left(\underset{x\in X'}{\mathcal{P}r}\{\ \varphi(x)\ \}\right)^2 > (1-2\delta)^2 \geq 1-4\delta.\end{aligned}$$

Terminamos, assim, os pré-requisitos probabilísticos necessários para este capítulo.

V.6 Epílogo

No presente capítulo descrevemos somente a prova do Teorema da Proximidade a um Polinômio de Grau Total Baixo V.1–5. Como ele nos assegura a corretude do Teste Polinomial de Grau Total Baixo T–VIII, fechamos o nosso objetivo de demonstrar o Teorema do $\mathcal{PCP}$ I.4–4. Fica assim condensado neste capítulo o arsenal "mais técnico" do texto. A esta altura, o leitor já tem em seu poder a demonstração completa, entretanto precisa, muitas vezes, ainda reavaliar todos os seus passos e verificar os intrincados encaixes do encadeamento da prova de

$$\mathcal{NP} \subseteq \mathcal{PCP}\big(\ \log(n), 1\ \big).$$

Tivemos a intenção de propiciar ao leitor um mergulho à beleza oriunda das **provas robustas checáveis probabilisticamente**. Escolhemos para isto uma introdução via a demonstração do resultado mais expressivo da área, qual seja, o Teorema do $\mathcal{PCP}$ I.4–4. Este resultado não só é o maior já obtido, como esperamos que o leitor a esta altura esteja convencido da abrangência dele na área. Entretanto, sempre tivemos como meta um mergulho técnico, apesar de introdutório. Como já afirmado, queremos dizer com isto que não seria esta a melhor maneira de um estudioso despretensioso se inteirar do assunto. Para isto, há na literatura especializada diversos textos, como já mencionado. Não obstante, buscamos cobrir um vácuo, acreditamos, deixado entre tais textos ilustrativos e os textos técnicos (como os originais p.e.) que já pressupõem uma grande familiaridade com o tema que é, por si só, bastante eclético e árduo. Com isto, esperamos que o leitor interessado em *produzir* nesta área esteja, este sim, bem "introduzido". Ficaríamos muito gratos se este objetivo fosse alcançado.

Índice Remissivo

Referências Bibliográficas

[ACP94] G. Ausiello, P. Crescenzi, and M. Protasi. Approximate solution of NP optimization problems. Technical report, Dipartimento di Scienze dell' Informazione, Universita degli Studi di Roma, 1994.

[ALM+92] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and hardness of approximation problems. In *Proc. 33rd Ann. IEEE Symp. on Foundations of Computer Science*, pages 14–23, 1992.

[Aro94] Sanjeev Arora. *Probabilistic Checking of Proofs and Hardness of Approximation Problems.* PhD thesis, U. of California at Berkeley, 1994.

[AS92] Sanjeev Arora and Shamuel Safra. Probabilistic checking of proofs: A new characterization of NP. In *Proc. 33rd Ann. IEEE Symp. on Foundations of Computer Science*, pages 2–13, 1992.

[Bab88] László Babai. Arthur–Merlin Games: A randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36:254–276, 1988.

[Bab90] László Babai. E–Mail and the unexpected power of interaction. In *Proc. 5th IEEE Symp. on Structure in Complexity Theory*, pages 30–44, 1990.

[Bab94] László Babai. Transparent proofs and limits to approximation. Technical Report TR-94-07, University of Chicago, 1994. To appear in the *Proc. First Europ. Congr. Math.*, Birkhäuser.

[BK89] Manuel Blum and Sampath Kannan. Designing programs that check their work. In *Proc. 21st Ann. ACM Symp. on Theory of Computing*, pages 86–97, 1989.

[BLR93] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self–testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47:549–599, 1993.

[BM89] László Babai and Shlomo Moran. Proving properties of interactive proofs by a generalized counting technique. *Information and Computation*, 82:185–197, 1989.

[Fag74] R. Fagin. Generalized first–order spectra and polynomial–time recognizable sets. In R. Karp, editor, *Complexity of Computations.* AMS, 1974.

[FHS94] Katalin Friedl, Zosolt Hátsági, and Alexander Shen. Low–degree tests. In *Proc. 5th Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 57–64, 1994.

[GH96] Oded Goldreich and Johan Håstad. On the message complexity of interactive proof systems. In *Electronic Colloquium on Computational Complexity*, 1996. Report Nr.: TR96–018.

[GLR+91] Peter Gemmell, Richard Lipton, Ronitt Rubinfeld, Madhu Sudan, and Avi Wigderson. Self–testing/correcting for polynomials and for approximate functions. In *Proc. 23rd Ann. ACM Symp. on Theory of Computing*, pages 32–42, 1991.

[Gol94] Oded Goldreich. Probabilistic proof systems (a survey). In *Electronic Colloquium on Computational Complexity*, 1994. Report Nr.: TR94–008.

[GS86] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *Proc. 18th Ann. ACM Symp. on Theory of Computing*, pages 59–68, 1986.

[HPS94] S. Hougardy, H. J. Prömel, and A. Steger. Probabilistically checkable proofs and their consequences for approximation algorithms. *Discrete Mathematics*, 136:175–223, 1994.

[KMSV95] Sanjeev Khanna, Rajeev Motwani, Madhu Sudan, and Umesh Vazirani. On syntatic versus computacional views of approximability. In *Electronic Colloquium on Computational Complexity*, 1995. Report Nr.: TR95–023.

[KS95] Yoshiharu Kohayakawa and José Augusto Ramos Soares. *Demonstrações Transparentes e a Impossibilidade de Aproximações*. Livro do 20º. Colóquio Brasileiro de Matemática. IMPA, Instituto de Matemática Pura e Aplicada, Rio de Janeiro, julho 1995.

[KV87] Phokion G. Kolaitis and Moshe Y. Vardi. The decision problem for the probabilities of higher–order. In *Proc. 19th Ann. ACM Symp. on Theory of Computing*, pages 425–435, 1987.

[LFKN92] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the Association for Computing Machinery*, 39(4):859–868, 1992.

[Lov94] László Lovász. Computation complexity. Lecture notes, 1994.

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Company, New York, 1994.

[PY88] Christos H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. In *Proc. 20th Ann. ACM Symp. on Theory of Computing*, pages 229–234, 1988.

[RS92] Romitt Rubinfeld and Madhu Sudan. Self–testing polynomial functions efficiently and over rational domains. In *Proc. 3rd Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 23–32, 1992.

[Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the Association for Computing Machinery*, 27(4):701–717, 1980.

[Sha92] Adi Shamir. IP = PSPACE. *Journal of the Association for Computing Machinery*, 39(4):869–877, 1992.

[She92] A. Shen. IP = PSPACE: Simplified proof. *Journal of the Association for Computing Machinery*, 39(4):878–880, 1992.

[Sud92] Madhu Sudan. *Efficient Checking of Polynomials and Proofs and the Hardness of Approximation Problems*. PhD thesis, U. of California at Berkeley, 1992.

[Sud95] Madhu Sudan. On the role of algebra in efficient verification of proofs. In *Electronic Colloquium on Computational Complexity*, 1995. Report Nr.: TR95–019.