Aplicações de Técnicas de Reescrita ao Problema da Palavra de Grupos

Luiz Manoel Rocha Gadelha Júnior

Dissertação de Mestrado

Departamento de Ciência da Computação
Universidade de Brasília

LUIZ MANOEL ROCHA GADELHA JÚNIOR

APLICAÇÕES DE TÉCNICAS DE REESCRITA AO PROBLEMA DA PALAVRA DE GRUPOS

Dissertação apresentada como requisito parcial à
obtenção do grau de Mestre.
Curso de Pós-Graduação em Ciência da Computação,
Departamento de Ciência da Computação,
Instituto de Ciências Exatas,
Universidade de Brasília.
Orientador: Prof. Dr. Mauricio Ayala Rincón.

BRASÍLIA
2000

Agradecimentos

Ao Professor Mauricio Ayala Rincón pela dedicação, incentivo e orientação, não apenas durante o mestrado mas também durante o curso de graduação nas atividades de iniciação científica. Estes servirão de inspiração para futura conduta profissional.

Aos Professores Noraí Rocco e Francisco Cartaxo pelas sugestões e pelo acompanhamento do desenvolvimento deste trabalho.

Ao Professor Friedrich Otto da Universität Gh. Kassel por atentamente responder nossas questões.

Aos meus pais, Luiz Manoel Rocha Gadelha e Maria Juvenha Macedo Gadelha, pela educação e princípios éticos e irmãs Melissa (Mel) e Bruna (Baia) pelo apoio e convivência sempre agradável.

Aos colegas de graduação e mestrado.

Resumo. Foi estudado um procedimento para solução do problema da palavra de grupos finitamente apresentados proposto recentemente por Robert Cremanns e Friedrich Otto e que utiliza uma estrutura combinatória chamada de ciclo de palavras para armazenar conjuntos de regras de reescrita simétricas. O procedimento foi implementado para realização de experimentos com diversas apresentações de grupos. É proposta também uma estratégia eficiente para ordenar a aplicação das regras de inferência utilizadas por Cremanns e Otto em seu procedimento.

Abstract. We study a procedure proposed recently by Robert Cremanns and Friedrich Otto that can be applied to solve the word problem of finitely presented groups. It uses a combinatorial structure called word-cycle to store symmetric sets of rewriting rules. This procedure has been implemented to experiment with various group presentations. We also present an efficient strategy to apply in an ordered manner the inference rules used by Cremanns and Otto in their procedure.

Conteúdo

CAPÍTULO 1

Introdução

O estudo de problemas algoritmicos em Álgebra, em particular em estruturas algébricas como grupos e monóides, é relevante tanto no contexto da Computação Simbólica quanto no contexto de Teoria da Computação. A Computação Simbólica trata de representar objetos matemáticos de maneira precisa e de realizar cálculos precisos usando a representação destes objetos. Neste sentido procedimentos eficientes para resolução de problemas computacionais em Teoria dos Grupos são importantes ferramentas no estudo de grupos. A subárea da Computação Simbólica que estuda métodos computacionais para grupos é chamada de Teoria dos Grupos Computacional e têm em Max Dehn um de seus precursores quando em 1911 este publica uma série de problemas de decisão para grupos [**Deh11**] como por exemplo o *problema da palavra*, que consiste de decidir se, dado um grupo definido por um conjunto de geradores e relações, uma palavra nos geradores representa a identidade. Com o advento dos computadores modernos a área se torna bastante ativa a partir de 1960 quando são desenvolvidos procedimentos para diversos fins como por exemplo a manipulação de grupos de permutações.

No contexto da Teoria da Computação o assunto é importante pois muitos dos problemas computacionais para Teoria dos Grupos, como o que será estudado nesta dissertação, são indecidíveis em geral e desta maneira podem ser úteis no estudo de indecidibilidade de outros problemas em Ciência da Computação. Muitos destes problemas são decidíveis e as questões que surgem então estão relacionadas com a complexidade das soluções achadas. Esta área está também fortemente relacionada com as Linguagens Formais e Autômatos como pode se verificar em [**ECH$^+$92**], [**Ott99a**] e [**ST99**]. Por exemplo relações como a de que o problema da palavra de um grupo é uma linguagem regular se e somente se o mesmo é finito [**Ani71**].

Nesta dissertação será estudado um procedimento para o problema da palavra de *grupos finitamente apresentados* proposto recentemente por Friedrich Otto e Robert Cremanns [**CO98**] e que manipula uma estrutura chamada de *ciclo de palavras*. A abordagem usual para se tentar resolver o problema da palavra de um determinado grupo com técnicas de reescrita é tentar *completar* sua apresentação utilizando o procedimento de Knuth-Bendix [**KB70**]. Neste caso é necessário estabelecer uma *ordem de redução* para as palavras no alfabeto dado pelos geradores do grupo, duas palavras representarão o mesmo elemento se suas formas normais forem idênticas. O procedimento de Otto-Cremanns dispensa esta exigência e nos casos em que pára o conjunto de ciclos de palavras resultante pode ser transformado em um sistema de reescrita de palavras *convergente* para qualquer ordem de redução. O que se observou no entanto é que o procedimento de Otto-Cremanns é consideravelmente menos eficiente em tempo que o procedimento de Knuth-Bendix. Uma das razões para tal pode ser explicada pelo fato de que no procedimento de Otto-Cremanns o casamento é realizado sobre ciclos de palavras e no procedimento de Knuth-Bendix o mesmo é feito sobre palavras, o que é menos custoso. O resultado mais relevante da dissertação é uma estratégia para aplicação das regras de inferência do procedimento de Otto-Cremanns que além de ser completa no sentido de que o conjunto de *ciclos de palavras persistentes* pode ser utilizado na solução do *problema da palavra reduzido* do grupo em questão mostrou-se bem mais eficiente na prática do que a aplicação não determinística das mesmas como é feito na descrição genérica do procedimento.

Para fins experimentais foi implementado o procedimento de Otto-Cremanns (sendo esta a primeira implementação do mesmo [**Ott00**]) em linguagem C. Uma versão preliminar implementava a mesma estratégia proposta por Otto e Cremanns e a versão final implementa a estratégia sendo proposta nesta dissertação. Um programa bastante utilizado foi o KBMAG (*Knuth-Bendix for Monoids and Automatic Groups*) [**EHR91**] de D. Holt que implementa o procedimento de Knuth-Bendix para palavras e foi utilizado para completar apresentações de grupos. A descrição da implementação do procedimento de Otto-Cremanns está no apêndice A.

1. Motivação

Nesta seção descreveremos um problema de cunho prático que é indecidível. A demonstração de indecidibilidade é realizada reduzindo o problema ao problema da palavra em monóides (ou equivalentemente ao problema da palavra em sistemas de reescrita de palavras) e que exemplifica a importância do conhecimento de propriedades de estruturas algébricas em Ciência da Computação. Será assumido o conhecimento da notação básica e de conceitos de Linguagens Formais e Autômatos que podem ser consultados em [**HU79**] e [**Sud97**].

O Problema da Correspondência de Post é um exemplo bastante prático de problema indecidível. Dado um conjunto de dominós da forma $\frac{a_1 a_2 ... a_n}{b_1 b_2 ... b_m}$, onde $a_1...a_n$ e $b_1...b_m$ são palavras, o problema consiste de determinar se é possível colocar os dominós em uma seqüência tal que se tenha a mesma palavra na parte de cima e na parte de baixo dos dominós. A formalização do problema é dada a seguir.

Definição 1.1. *Um* **sistema de correspondência de Post** *(SCP) consiste de um alfabeto Σ e um conjunto finito de pares ordenados $[u_i, v_i]$, $i = 1, 2, ..., n$, onde $u_i, v_i \in \Sigma^+$. Uma* **solução** *para um SCP consiste de uma seqüência $i_1, ..., i_k$ tal que $u_{i_1} u_{i_2} ... u_{i_k} = v_{i_1} v_{i_2} ... v_{i_k}$.*

Definição 1.2. *O* **Problema da Correspondência de Post** *(PCP) consiste em decidir se um determinado SCP tem solução.*

Exemplo 1.1. Dado o conjunto $\{\frac{a}{ba}, \frac{b}{bb}, \frac{bab}{a}, \frac{ba}{a}\}$ as seqüências $\langle \frac{b}{bb}, \frac{ba}{a} \rangle$ e $\langle \frac{b}{bb}, \frac{bab}{a}, \frac{a}{ba}, \frac{b}{bb}, \frac{ba}{a} \rangle$ são soluções para o Problema da Correspondência de Post determinado pelo conjunto de dominós de entrada. ◇

Exemplo 1.2. O SCP $\{\frac{ab}{a}, \frac{b}{ba}\}$ não tem solução. Se iniciamos com o dominó $\frac{ab}{a}$ devemos necessariamente colocar o dominó $\frac{b}{ba}$ pois é o único que inicia com b embaixo. As seqüências obtidas não casam. Se iniciamos com o dominó $\frac{b}{ba}$ devemos pelo mesmo motivo colocar o dominó $\frac{ab}{a}$ em seguida, mas novamente as seqüências não casam. Logo este SCP não tem solução. ◇

EXEMPLO 1.3. Considere um SCP cujo alfabeto é unário, i.e. $|\Sigma| = 1$. Os dominós podem ser vistos então como pares de números $\{\frac{i_1}{j_1}, ..., \frac{i_n}{j_n}\}$ tais que $i_1, ..., i_n, j_1, ..., j_n \in$ $^+$. Assim o PCP se reduz a achar uma solução para a equação $a_1 i_1 + \cdots + a_n i_n = a_1 j_1 + \cdots + a_n j_n$ que equivale a $a_1(i_1 - j_1) + \cdots + a_n(i_n - j_n) = 0$ com $a_1, ..., a_n \in$ e algum $a_i > 0, 1 < i < n$. ◇

A indecidibilidade do PCP será demonstrada realizando uma redução ao problema da palavra para sistemas de reescrita de palavras. Cada sistema de reescrita de palavra está associado a uma estrutura algébrica que chamamos de *monóide* e solubilidade do problema da palavra do sistema de reescrita de palavras está estritamente associada à solubilidade do problema da palavra do respectivo monóide.

DEFINIÇÃO 1.3. *Um* **monóide** *é um par* $\langle M, \cdot \rangle$ *onde* M *é um conjunto e* $\cdot$ *é uma operação binária associativa e é tal que existe um elemento identidade* $e \in M$ *tal que para qualquer* $x \in M$ *temos* $x \cdot e = x$ *e* $e \cdot x = x$.

O problema da palavra de um monóide M definido por geradores e relações, i.e. $M = \Sigma^* / \sim$ para algum alfabeto Σ e para alguma relação de congruência $\sim$, consiste de decidir se dadas duas palavras $v, w \in \Sigma^*$ se v e w representam o mesmo elemento, i.e. se $v \sim w$.

O problema da palavra de um sistema de reescrita de palavras $\langle \Sigma, \to \rangle$ consiste de decidir, dadas duas palavras $v, w \in \Sigma^*$ se $v \leftrightarrow^* w$, onde $\leftrightarrow^*$ é o fecho reflexivo, simétrico e transitivo da relação $\to$. É um problema indecidível e isto pode ser demonstrado por redução ao problema da parada para máquinas de Turing.

Observação. O problema da palavra para monóides é indecidível.

De fato se o problema da palavra para monóides fosse decidível o monóide $\Sigma^* / \leftrightarrow$ teria o problema da palavra decidível para qualquer relação de redução $\to$. Desta maneira poderiamos decidir se dadas duas palavras $u, v \in \Sigma^*$ se $u \leftrightarrow^* v$. Assim o problema da palavra do sistema de reescrita $\langle \Sigma, \to \rangle$ seria decidível, uma contradição.

TEOREMA 1.1. *Não existe algoritmo que determine se um SCP arbitrário tem solução.*

DEMONSTRAÇÃO. Seja $S = \langle \Sigma, P \rangle$ um sistema de reescrita de palavras cujo problema da palavra é indecidível. Vamos construir um SCP $C_{u,v}$ para cada par de palavras $u, v \in \Sigma^+$ que terá solução se e somente se $u \to_S^* v$.

Inicialmente acrescentemos as regras $0 \to 0$ e $1 \to 1$ que não alteram as derivações possíveis de S e que nos permitem supor que qualquer derivação tem comprimento par.

Sejam $u, v \in \Sigma^*$ e $C_{u,v}$ um SCP tal que o alfabeto é $0, \overline{0}, 1, \overline{1}, [,], *, \overline{*}$. Uma palavra w que consiste somente de símbolos marcados com barra é denotada por $\overline{w}$.

Para cada produção $x_i \to y_i$, $i = 1, ..., n$, de S temos dois dominós: $\frac{\overline{y}_i}{x_i}$ e $\frac{y_i}{\overline{x}_i}$.

Adicionalmente temos os dominós

$$\frac{[u*}{[} \quad \frac{*}{\overline{*}} \quad \frac{\overline{*}}{*} \quad \frac{]}{\overline{*}v]}$$

Observe que podemos combinar os dominós

$$\frac{0}{\overline{0}} \quad \frac{\overline{0}}{0} \quad \frac{1}{\overline{1}} \quad \frac{\overline{1}}{1}$$

para formar as seguintes palavras nas partes de cima e de baixo dos dominós:

$$\frac{w}{\overline{w}} \quad \frac{\overline{w}}{w}$$

para qualquer palavra $w \in \Sigma^*$.

Mostremos que se $u \to^* v$ então $C_{u,v}$ tem solução.

Suponha que $u = u_0 \to u_1 \to \cdots \to u_k \to v$ é uma derivação de comprimento ímpar. O i-ésimo passo da derivação pode ser visto como

$$u_{i-1} = p_{i-1} x_{j_{i-1}} q_{i-1} \to_R p_{i-1} y_{j_{i-1}} q_{i-1} = u_i$$

usando a regra $x_{j_{i-1}} \to y_{j_{i-1}}$.

Afirmação: A palavra $[u_0 * \overline{u}_1 \overline{*} u_2 * \cdots * \overline{u}_{k-1} \overline{*} u_k]$ é uma solução para o SCP $C_{u,v}$.

De fato a solução pode ser construída da seguinte maneira:

1. Inicialmente colocamos o dominó

$$\frac{[u*}{[}$$

2. Para obter um casamento colocamos dominós que formem u na parte de baixo.

$$\frac{[u*}{[}\ \frac{\overline{p}_0}{p_0}\ \frac{\overline{y}_{j_0}}{x_{j_0}}\ \frac{\overline{q}_0}{q_0}\ \frac{\overline{*}}{*}$$

já que $u = p_0 x_{j_0} q_0$ e $p_0 x_{j_0} q_0 \to_R p_0 y_{j_0} q_0 = u_1$. Formando em cima a palavra $[u * \overline{u}_1 \overline{*}$ e embaixo a palavra $[u*$.

3. Colocamos $\overline{u}_1 \overline{*}$ na parte de baixo:

$$\frac{[u*}{[}\ \frac{\overline{p}_0}{p_0}\ \frac{\overline{y}_{j_0}}{x_{j_0}}\ \frac{\overline{q}_0}{q_0}\ \frac{\overline{*}}{*}\ \frac{p_1}{\overline{p}_1}\ \frac{y_{j_1}}{\overline{x}_{j_1}}\ \frac{q_1}{\overline{q}_1}\ \frac{*}{\overline{*}}$$

Formando em cima a palavra $[u * \overline{u}_1 \overline{*} u_2 *$ e embaixo a palavra $[u * \overline{u}_1 \overline{*}$.

4. Continuamos com este processo até que tenhamos na parte de cima $[u * \overline{u}_1 \overline{*} u_2 * \cdots * \overline{u}_{k-1} \overline{*} u_k$ e embaixo $[u * \overline{u}_1 \overline{*} u_2 * \cdots * \overline{u}_{k-1}$.

5. Colocamos então o dominó $\frac{]}{\overline{*}v]}$ obtendo uma solução do SCP $C_{u,v}$.

Dada uma solução do SCP $C_{u,v}$ mostremos que existe uma derivação $u \to_R^* v$.

Tal solução deve necessariamente iniciar pelo dominó $\frac{[u*}{[}$ uma vez que este é o único que têm um casamento à esquerda. Pela mesma razão a solução deve terminar com o dominó $\frac{]}{\overline{*}v]}$. Desta maneira a solução tem a forma $[u * w \overline{*} v]$. Se w contém $]$ então $[u * w \overline{*} v] = [u * x \overline{*} v] y \overline{*} v]$ e assim $[u * x \overline{*} v]$ é também uma solução de $C_{u,v}$. Logo podemos supor que a solução é da forma $[u * x \overline{*} v]$ onde $]$ aparece apenas na extremidade direita.

Como a solução começa por $\frac{[u*}{[}$ devemos fazer o casamento na parte de baixo colocando u, se $u = x_{i_1} x_{i_2} ... x_{i_k}$ teremos

$$\frac{[u*}{[}\ \frac{\overline{y}_{i_1}}{x_{i_1}}\ \cdots \frac{\overline{y}_{i_k}}{x_{i_k}}\ \frac{\overline{*}}{*}$$

Como cada dominó representa uma derivação $x_{i_j} \to_R y_{i_j}$ combinamos as derivações para obter $u \to_R^* u_1 = y_{i_1} ... y_{i_k}$. Desta maneira ficamos com $[u * \overline{u}_1 \overline{*}$ na parte de cima e $[u*$ na parte de baixo.

Seguindo desta maneira obtemos $[u * \overline{u}_1 \overline{*} \cdots * \overline{u}_{k-1} \overline{*} v]$ que resulta em $u \to_R^* v$.

Assim reduzimos o problema da palavra de um sistema de reescrita de palavras ao Problema da Correspondência de Post. Logo, como o problema da palavra para sistemas de reescrita de palavras é indecidível, o Problema da Correspondência de Post também é. □

Observe que, segundo a demonstração acima, o PCP é equivalente ao problema da palavra para sistemas de reescrita de palavras. E como este último problema está fortemente relacionado com o problema da palavra de monóides podemos perceber a grande importância do conhecimento de propriedades de tais estruturas algébricas na Ciência da Computação em geral e em particular em Teoria da Computação.

2. Objetivos

Estudaremos o procedimento completador para grupos finitamente apresentados proposto por Friedrich Otto e Robert Cremanns em [**CO98**], uma alternativa à completação usual com o procedimento de Knuth-Bendix. Pretendemos descrever de maneira mais detalhada as estratégias utilizadas por Otto e Cremans na completação assim como a estratégia sendo proposta neste trabalho. Aplicaremos as duas versões do procedimento, implementadas de acordo com as estratégias propostas, a algumas apresentações clássicas de grupos de maneira a obter evidência empírica da superioridade da estratégia sendo aqui proposta. A dissertação está organizada como segue:

No segundo capítulo apresentamos as noções preliminares necessárias ao entendimento do restante da dissertação. Em particular serão apresentadas noções de álgebra como estruturas algébricas (semigrupos, monóides e grupos), apresentações de monóides e apresentações de grupos e noções de teoria da reescrita como sistemas abstratos de reescrita, sistemas de reescrita de palavras.

No terceiro capítulo será mostrado em detalhe o procedimento de Knuth-Bendix para palavras e serão exibidos alguns exemplos de como usar o procedimento na solução do problema da palavra de grupos e de monóides.

No quarto capítulo será exibido o procedimento completador de Otto/Cremanns para grupos finitamente apresentados de maneira detalhada e serão exibidas algumas de suas propriedades. Será apresentada também uma estratégia mais eficiente que utiliza a aplicação ordenada das regras. As apresentações de grupos do terceiro capítulo serão submetidas ao procedimento para fins de comparação com o procedimento de Knuth-Bendix. São comparados também os tempos de execução das duas estratégias para alguns exemplos, a originalmente proposta por Otto e Cremanns e a proposta nesta dissertação.

CAPÍTULO 2

Conceitos Preliminares

Nesta seção serão apresentados pré-requisitos ao entendimento dos capítulos subseqüentes. Inicialmente serão introduzidos conceitos de teoria dos conjuntos e álgebra. Em seguida são apresentadas noções de teoria da reescrita, incluindo-se aí sistemas de reescrita de palavras. Os teoremas deste capítulo são incluidos por razões de completeza e podem ser encontrados nas referências-padrão de cada tópico abordado como por exemplo [**Sim94**], [**BN98**], [**AR98**] e [**BO93**].

1. Álgebra e Teoria dos Conjuntos

Será assumido conhecimento dos conceitos de conjunto, pertinência e inclusão assim como das operações de união, interseção e diferença de conjuntos. O conjunto vazio será denotado por $\emptyset$. Se A for um subconjunto de B isto será denotado por $A \subseteq B$ ou $B \supseteq A$. Se A for um subconjunto próprio de B isto será denotado por $A \subset B$ ou $A \supset B$. A cardinalidade de um conjunto A será denotada por $|A|$.

Definição 1.1. *Sejam A e B conjuntos. O* **produto cartesiano** *$A \times B$ é o conjunto dos pares ordenados (a, b) tais que $a \in A$ e $b \in B$. A* **diagonal** *de $X \times X$ é o conjunto $D = \{(x, x) \mid x \in X\}$.*

Definição 1.2. *Uma* **relação** *de X em Y é um subconjunto R de $X \times Y$. Se $X = Y$ dizemos que R é uma relação em X. $(x, y) \in R$ será denotado por xRy.*

Definição 1.3. *Seja R uma relação de X em Y. Se $A \subseteq R$ então definimos o conjunto AR como sendo $\{y \in Y \mid aRy$ para algum $a \in A\}$.*

Definição 1.4. *A relação* **inversa** *de uma relação R de X em Y é o conjunto $R^{-1} = \{(y, x) \mid (x, y) \in R\}$.*

DEFINIÇÃO 1.5. *Seja R uma relação de X em Y e S uma relação de Y em Z. A* **composição** *de R e S, denotada por $R \circ S$, consiste dos pares (x, z) tais que existe um $y \in Y$ onde xRy e ySz.*

DEFINIÇÃO 1.6. *Uma* **relação de equivalência** *em um conjunto X é uma relação E sobre X tal que*

1. *para todo $x \in X$ se tem xEx (reflexividade),*
2. *se para x, $y \in X$ xEy então yEx (simetria),*
3. *se para x, y, $z \in X$ xEy e yEz então xEz (transitividade).*

Observação. Se E é uma relação de equivalência em X então para x, $y \in X$ os conjuntos $\{x\}E$ e $\{y\}E$ são iguais ou disjuntos. Desta maneira $\Pi = \{\{x\}E \mid x \in X\}$ é uma partição de X cujos elementos são chamados de **classes de equivalência** de E ($\{x\}E$ é dito classe de equivalência de x em E).

DEFINIÇÃO 1.7. *Uma* **função** *(ou* **aplicação***) de X em Y é uma relação f de X em Y tal que para todo $x \in X$ o conjunto $\{x\}f$ é unitário. A função é denotada por $f : X \to Y$ e dizemos que X é o* **domínio** *de f e que Y é o* **contra-domínio** *de f. Se y é o único elemento de $\{x\}f$ então dizemos que $f(x) = y$ ou $y = x^f$ ou ainda $x \longmapsto^f y$*

DEFINIÇÃO 1.8. *Seja $f : X \to Y$ uma função de X em Y.*

- *f é* **sobrejetiva** *se $Xf = Y$.*
- *f é* **injetiva** *se, para todo, x_1, $x_2 \in X$, se $f(x_1) = f(x_2)$ então $x_1 = x_2$.*
- *f é* **bijetiva** *se é simultaneamente injetiva e sobrejetiva.*

DEFINIÇÃO 1.9. *Uma* **permutação** *de um conjunto X é uma função bijetiva de X em X.*

DEFINIÇÃO 1.10. *Um alfabeto é um conjunto qualquer. Seja Σ um alfabeto, uma seqüência finita $w = w_1 w_2 ... w_n$, onde $w_i \in \Sigma$ para $1 \leq i \leq n$, é dita uma* **palavra** *sobre Σ. A* **palavra vazia** *é a palavra dada pela seqüência vazia e será denotada por ϵ.*

O conjunto de todas as palavras sobre Σ é denotado por Σ^*.

DEFINIÇÃO 1.11. *A* **concatenação** *das palavras* $u = u_1u_2...u_n$ *e* $v = v_1v_2...v_m$ *de* Σ^* *é dada por* $uv = u_1u_2...u_nv_1v_2...v_m$.

DEFINIÇÃO 1.12. *O* **comprimento** *de uma palavra* w*, denotado por* $|w|$*, é o comprimento da seqüência* w.

DEFINIÇÃO 1.13. *Se* u*,* v *e* $w \in \Sigma^*$ *e* $z = uvw$ *dizemos que* u *é* **prefixo** *de* z*,* v *é* **subpalavra** *de* z *e que* w *é* **sufixo** *de* z.

DEFINIÇÃO 1.14. *Se* $u = u_1u_2...u_m \in \Sigma^*$ *então cada palavra* $u_iu_{i+1}...u_mu_1...u_{i-1}$ *para* $1 \leq i \leq m$ *é dita uma* **permutação cíclica** *de* u*. A palavra* $u_mu_{m-1}...u_1$ *é dita o* **reverso** *de* u.

DEFINIÇÃO 1.15. *Uma* **operação binária** *em um conjunto* M *é uma função* $\cdot : M \times M \to M$ *que leva* $(x, y) \in M \times M$ *em* $\cdot(x, y)$

Notação. Será utilizado $x \cdot y$ para denotar $\cdot(x, y)$. A operação binária é dita **associativa** se para quaisquer $x, y, z \in M$ se tem $(x \cdot y) \cdot z = x \cdot (y \cdot z)$.

DEFINIÇÃO 1.16. *Um* **semigrupo** *consiste de um conjunto* M *munido de uma operação binária* $\cdot$*, associativa.*

DEFINIÇÃO 1.17. *Um semigrupo* $(M, \cdot)$ *com um elemento* $e \in M$ *tal que para qualquer* $x \in M$ *se tem* $x \cdot e = x$ *e* $e \cdot x = e$ *é chamado de* **monóide**. *O elemento* e *é dito* **identidade** *do monóide* M.

EXEMPLO 1.1. Seja Σ um alfabeto qualquer, Σ^+ é um semigrupo e Σ^* é um monóide para a operação de concatenação. A identidade de Σ^* é ϵ. ◇

DEFINIÇÃO 1.18. *Um monóide* $(M, \cdot)$ *tal que para qualquer* $x \in M$ *existe um elemento* x^{-1} *tal que* $x \cdot x^{-1} = e$ *é chamado de* **grupo**. *O elemento* x^{-1} *é dito* **inverso** *do elemento* x *no grupo* G.

Será utilizado apenas o símbolo G para denotar o grupo $(G, \cdot)$.

DEFINIÇÃO 1.19. *Dado um monóide* M *com identidade* e *o conjunto* $U(M) = \{x \in M \mid$ *existe* $y \in M$ *tal que* $xy = e\}$ *é chamado de* **grupo de unidades** *de* M.

Observação. $U(M)$ é de fato um grupo pois $e \in U(M)$ já que $ee = e$ e se $x \in U(M)$ então existe um $y \in M$ tal que $xy = e$ e desta maneira $xyx = x$ e $yx = e$. Logo $y \in U(M)$.

DEFINIÇÃO 1.20. *Seja M um monóide com identidade e. Um* **submonóide** *de M é um subconjunto N de M tal que*

1. $e \in N$,
2. *se $x, y \in N$ então $xy \in N$.*

Além disso se todos os elementos de N são inversíveis e N contêm o inverso de cada um de seus elementos então dizemos que N é um **subgrupo** *de M.*

TEOREMA 1.1. *A interseção de um conjunto de submonóides de um monóide M é um submonóide de M. Da mesma maneira a interseção de um conjunto de subgrupos de M é também um subgrupo de M.*

DEMONSTRAÇÃO. Seja um conjunto de submonóides de M (e é a identidade de M). Logo se $N \in$, i.e. N é um submonóide de M, então $e \in N$. Desta maneira e pertence a todos os elementos de e portanto pertence também à interseção de todos os elementos de N. Se x e y pertencem à interseção de todos os elementos de N então x e y pertencem a cada elemento de N. Logo xy pertence a cada elemento de N. Assim xy pertence à interseção de todos os membros de N. Portanto a interseção de submonóides de M é um monóide. □

DEFINIÇÃO 1.21. *Seja M um monóide, $Y \subseteq M$ e o conjunto de submonóides de M que contém Y, que é não vazio pois $M \in$. Se K é a interseção dos elementos de dizemos que K é o submonóide* **gerado** *por Y que será denotado por $Mon\langle Y\rangle$.*

Observação. Na definição anterior K será o menor submonóide de M que contém Y e consistirá de todos os produtos da forma $y_1y_2...y_n$ onde $y_i \in Y$ para $0 \leq i \leq n$ (no caso em que $n = 0$ temos e).

DEFINIÇÃO 1.22. *Seja Y um subconjunto de $U(M)$ para algum monóide M. O conjunto de subgrupos de M que contém Y é não-vazio pois $U(M)$ é grupo e $Y \subseteq U(M)$. A interseção H destes subgrupos é chamada de subgrupo* **gerado** *por Y e será denotada por $Grp\langle Y\rangle$.*

DEFINIÇÃO 1.23. *Sejam M e N monóides com identidades 1_M e 1_N respectivamente. Um homomorfismo de M em N é uma função $h : M \to N$ tal que:*

1. $h(1_M) = 1_N$.
2. $h(xy) = h(x)h(y)$.

DEFINIÇÃO 1.24. *Um homomorfismo de M em M é dito um* **endomorfismo**. *Um homomorfismo bijetivo $h : M \to N$ é um* **isomorfismo**.

DEFINIÇÃO 1.25. *Para qualquer conjunto X o monóide X^* é chamado de* **monóide livre** *gerado por X.*

TEOREMA 1.2. *Seja $f : M \to N$ um homomorfismo entre monóides. Se H é um submonóide de M e K é um submonóide de N então $f(H)$ é um submonóide de N e $f^{-1}(K)$ é um submonóide de M. Se M e K são grupos, então $f(H)$ é um subgrupo de N e $f^{-1}(K)$ é um subgrupo de M.*

DEFINIÇÃO 1.26. *A* **ordem** *de um elemento x de um grupo, é o menor inteiro positivo i tal que $x^i = e$.*

TEOREMA 1.3. *Seja x um elemento de um grupo G tal que a ordem de x é finita e igual a n. Então $x^n = 1$ e $x^m = 1$ se e somente se $n \mid m$.*

DEFINIÇÃO 1.27. *Se todo os elemento de um grupo G tem ordem finita e estas ordens são limitadas superiormente então o mínimo múltiplo comum destes números é dito o* **expoente** *de G. Equivalentemente dizemos que o expoente de G é o menor inteiro positivo exp tal que $g^{exp} = 1$ para todo elemento g de G.*

DEFINIÇÃO 1.28. *Seja G um grupo e H um subgrupo de G. O conjunto $R = \{Hx | x \in G\}$ é uma partição de G e é chamado de conjunto das* **classes laterais à direita** *de H. Similarmente, o conjunto $L = \{xH | x \in G\}$ é chamado de conjunto das* **classes laterais à esquerda** *de H.*

Observação. Quando R é finito sua cardinalidade é denotada por $|G : H|$ e é chamada de **índice** de H em G. Se G é finito então $|G| = |G : H||H|$ (Teorema de Lagrange).

DEFINIÇÃO 1.29. *Se $R = L$ então H é dito* **subgrupo normal** *de G.*

Observação. Se H é subgrupo normal de G então R é um grupo para a operação $(Hx)(Hy) = Hxy$. Este grupo é chamado de **grupo quociente** de G por H.

TEOREMA 1.4. *Seja $H \leq G$, as seguintes condições são equivalentes:*

- *H é subgrupo normal de G.*
- *$Hx = xH$ para todo $x \in G$.*
- *$x^{-1}Hx \subseteq H$ para todo $x \in G$.*

DEFINIÇÃO 1.30. *Seja $f : G \to H$ um homomorfismo de grupos. O* **kernel** *(ou núcleo) de f é $ker(f) = \{x \in G | f(x) = 1\}$.*

DEFINIÇÃO 1.31. *Sejam x, y elementos de um grupo G. Se existe $z \in G$ tal que $x = z^{-1}yz$ dizemos que x é dito um* **conjugado** *de y.*

Conjugação é uma relação de equivalência e suas classes de equivalência são chamadas de **classes de conjugação**.

DEFINIÇÃO 1.32. *Seja $X \subseteq G$. O subgrupo N gerado por todos os conjugados de elementos de X é normal em G. N é o* **fecho normal** *de X em G.*

DEFINIÇÃO 1.33. *Seja M um monóide e $\sim$ uma relação de congruência sobre M. O* **monóide quociente** *de M módulo $\sim$ é o conjunto das classes de congruência de $\sim$ com a operação binária $[x][y] = [x \cdot y]$, onde $[x]$, p.ex., é a classe de congruência de x.*

TEOREMA 1.5. *Seja M um monóide e seja S um subconjunto de $M \times M$. A interseção de todas as congruências que contém S é uma congruência.*

DEFINIÇÃO 1.34. *No teorema 1.5 dizemos que a interseção de todas as congruências que contém S é a* **congruência gerada** *por S.*

DEFINIÇÃO 1.35. *Seja X um conjunto e R um subconjunto de $X^* \times X^*$. O monóide $Mon\langle X|R\rangle$ é definido como sendo o monóide quociente Q de X^* módulo a congruência gerada por R. O par (X, R) é dito uma* **apresentação de monóide** *para Q e de qualquer monóide isomorfo a Q. Uma apresentação (X, R) é dita* **finita** *se X e R são finitos. Um monóide M é dito* **finitamente apresentado** *se tem uma apresentação finita.*

Notação. Na definição 1.35 se $X = \{x_1, ..., x_m\}$ e $R = \{(u_i, v_i) | u_i, v_i \in X^* \text{ e } 1 \leq i \leq n\}$ denotaremos Q por $Mon\langle x_1, ..., x_m | u_1 = v_1, ..., u_n = v_n \rangle$. As equações $u_i = v_i$ são chamadas de relações.

DEFINIÇÃO 1.36. *Seja X um conjunto e $X^{\pm} = X \times \{-1, 1\}$. Denotaremos $(x, \alpha) \in X^{\pm}$ por x^{α}. Seja $R = \{(x^{\alpha}x^{-\alpha}, \epsilon) | \alpha \in \{-1, 1\}, x \in X\}$ e $F = Mon\langle X^{\pm} | R \rangle$. F é dito o* **grupo livre** *gerado por X.*

Notação. Na definição 1.36 o conjunto R será denotado por $RelGL(X)$.

DEFINIÇÃO 1.37. *A congruência sobre $(X^{\pm})^*$ gerada por $RelGL(X)$ é chamada de* **equivalência livre**. *Uma palavra $w \in (X^{\pm})^*$ é dita* **livremente reduzida** *se não contém subpalavras da forma $x^{\alpha}x^{-\alpha}$, $\alpha \in \{-1, 1\}$.*

DEFINIÇÃO 1.38. *Seja $u = x_1^{\alpha_1}...x_n^{\alpha_n} \in (X^{\pm})^*$. u^{-1} é definida como sendo $x_n^{-\alpha_n}...x_1^{-\alpha_1}$.*

DEFINIÇÃO 1.39. *Seja S um subconjunto de $(X^{\pm})^* \times (X^{\pm})^*$. $Grp\langle X | S \rangle$ denotará o monóide $G = Mon\langle X^{\pm} | RelGL(X) \cup S \rangle$. Dizemos que (X, S) é uma* **apresentação de grupo** *para G.*

Notação. Na definição 1.39 se $X = \{x_1, ..., x_m\}$ e $S = \{(u_i, v_i) | u_i, v_i \in (X^{\pm})^* \text{ e } 1 \leq i \leq n\}$ denotaremos G por $Grp\langle x_1, ..., x_m | u_1 = v_1, ..., u_n = v_n \rangle$. As equações $u_i = v_i$ são chamadas de relações.

2. Sistemas de Reescrita

DEFINIÇÃO 2.1. *Seja M um conjunto e $\rightarrow$ uma relação binária sobre M. O par $(M, \rightarrow)$ é um* **sistema de reescrita**. *$\rightarrow$ é denominada* **relação de reescrita**.

Dado um sistema de reescrita $(M, \rightarrow)$ denotamos a classe de equivalência de um elemento $x \in M$ para a relação de equivalência $\leftrightarrow^*$ por $[x]$, i.e. $[x] = \{y \in M \mid y \leftrightarrow^* x\}$. x é dito **irredutível** se não existe $y \in M$ tal que $x \rightarrow y$. Se para $u, v \in M$ temos que $u \rightarrow^* v$ e v é irredutível então v é uma **forma normal** de u. u e v são **juntáveis** (notação: $u \downarrow v$) se existe $w \in M$ tal que $u \rightarrow^* w$ e $v \rightarrow^* w$.

DEFINIÇÃO 2.2. *$(M, \rightarrow)$ é* **terminante** *se não existem cadeias de redução infinitas da forma $u_0 \rightarrow u_1 \rightarrow \cdots$.*

Notação	Significado
$\leftarrow$	inversa de $\rightarrow$
$\rightarrow^+$	fecho transitivo de $\rightarrow$
$\rightarrow^*$	fecho reflexivo e transitivo de $\rightarrow$
$\leftrightarrow$	fecho simétrico de $\rightarrow$
$\leftrightarrow^*$	fecho reflexivo, simétrico e transitivo de $\rightarrow$
$\rightarrow^0$	$=$
$\rightarrow^{n+1}$	$\rightarrow^n \cup \rightarrow$

TABELA 1. Notação padrão da teoria da reescrita.

DEFINIÇÃO 2.3. $(M, \rightarrow)$ *é dito* **confluente** *sempre que para quaisquer* $u, v \in M$ $u \rightarrow^* v$ *e* $u \rightarrow^* w$ *implicar* $v \downarrow w$. $(M, \rightarrow)$ *é* **local-confluente** *sempre para quaisquer* $u, v \in M$ $u \rightarrow v$ *e* $u \rightarrow w$ *implicar* $v \downarrow w$. $(M, \rightarrow)$ *tem a* **propriedade de Church-Rosser** *se para* $u, v \in m \leftrightarrow^*$ *implicar que* $u \downarrow v$.

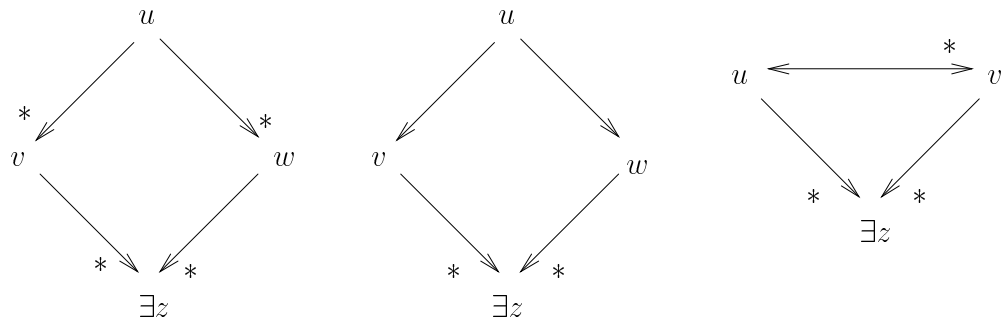

FIGURA 1. Confluência, Confluência Local e Propriedade de Church-Rosser

A seguir mostramos que confluência e a propriedade de Church-Rosser são equivalentes.

TEOREMA 2.1. $(M, \rightarrow)$ *tem a propriedade de Church-Rosser se e somente se é confluente.*

DEMONSTRAÇÃO. Suponha que $(M, \rightarrow)$ tem a propriedade de Church-Rosser. Se para u, v e $w \in M$ temos $u \rightarrow^* v$ e $u \rightarrow^* w$ então $v \leftrightarrow^* u$ e $u \leftrightarrow^* w$. Por transitividade temos $v \leftrightarrow^* w$ e pela Propriedade de Church-Rosser $v \downarrow w$. Logo $(M, \rightarrow)$ é confluente.

Se $(M, \rightarrow)$ é confluente e temos para u e $v \in M$ que $u \leftrightarrow^* v$ mostremos por indução que se $u \leftrightarrow^n v$ então $u \downarrow v$.

Base indutiva: se $n = 0$ então $u = v$ e desta maneira $u \downarrow v$.

Passo indutivo: Suponha que se $u \leftrightarrow^n v$ então $u \downarrow v$. Se $u \leftrightarrow^{n+1} v$ então para algum $v' \in M$ $u \leftrightarrow^n v' \leftrightarrow v$. São dois os casos possíveis:

1. $u \leftrightarrow^n v' \rightarrow v$. Pela hipótese de indução $u \downarrow v'$ logo existe $z' \in M$ tal que $u \rightarrow^* z'$ e $v' \rightarrow^* z'$. Mas como $(M, \rightarrow)$ é confluente existe $z \in M$ tal que $z' \rightarrow^* z$ e $v \rightarrow^* z$. Desta maneira $u \rightarrow^* z' \rightarrow^* z$ e $v \rightarrow^* z$ logo $u \downarrow z$.
2. $u \leftrightarrow^n v' \leftarrow v$. Novamente pela hipótese de indução $u \downarrow v'$ logo existe $z \in M$ tal que $u \rightarrow^* z$ e $v' \rightarrow^* z$. Como $u \rightarrow^* z$ e $v \rightarrow v' \rightarrow^* z$ temos que $u \downarrow v$.

Logo $(M, \rightarrow)$ tem a propriedade de Church-Rosser. □

LEMA 2.1. *Se $(M, \rightarrow)$ é confluente e $u \in M$ tem forma normal então ela é única.*

DEMONSTRAÇÃO. Suponha que $u \in M$ tenha formas normais u' e u''. Logo $u \rightarrow^* u'$ e $u \rightarrow^* u''$. Como $(M, \rightarrow)$ é confluente existe $z \in M$ tal que $u' \rightarrow^* z$ e $u'' \rightarrow^* z$ mas u' e u'' são formas normais logo $u' \rightarrow^0 z$ e $u'' \rightarrow^0 z$. Portanto $u' = u''$ pois $u' = z = u''$. □

DEFINIÇÃO 2.4. *Um sistema de reescrita terminante e confluente é dito* **convergente**.

TEOREMA 2.2. *Se $(M, \rightarrow)$ é convergente então cada $u \in M$ tem exatamente uma forma normal.*

DEMONSTRAÇÃO. Tome x um elemento de M. Se x é irredutível então x tem forma normal. Caso contrário existe um $y \in M$ tal que $x \rightarrow y$, logo existe uma seqüência y_0, y_1, ... tal que $x = y_0$ e $y_i \rightarrow y_{i+1}$. Como $(M, \rightarrow)$ é terminante esta seqüencia é finita e para algum $k \geq 0$ y_k é irredutível. Assim y_k é forma normal de x. Portanto todo $x \in M$ tem ao menos uma forma normal e como $(M, \rightarrow)$ é confluente esta formal normal é única. □

O **problema da palavra** para um sistema de reescrita $(M, \rightarrow)$ consiste em saber, dados $u, v \in M$, se $u \leftrightarrow^* v$. Observe que se $(M, \rightarrow)$ for convergente é possível reduzir u e v às suas formas normais, $\overline{u}$ e $\overline{v}$ respectivamente. Desta maneira $u \leftrightarrow^* v$ se e somente se $\overline{u} = \overline{v}$. De fato se $u \leftrightarrow^* v$ então existe z tal que $u \rightarrow^* z$ e $v \rightarrow^* z$, logo $\overline{u} = \overline{z} = \overline{v}$ onde $\overline{z}$ é a forma normal de z. Se $\overline{u} = \overline{v}$ então $u \rightarrow^* \overline{u} = \overline{v} \ ^*\leftarrow v$ e portanto $u \leftrightarrow^* v$. Logo o problema da palavra de um sistema de reescrita convergente é decidível.

Com o intuito de mostrar que sob hipótese de terminação as propriedades de confluência e confluência local são equivalentes será introduzido o conceito de indução *noetheriana.*

Definição 2.5. *Seja $(M, \to)$ um sistema de reescrita. Um predicado P sobre elementos de M é* $\to$**-completo** *se para todo $x \in M$ tivermos: se $P(y)$ é verdadeiro para todo $y \in \Delta(x)$, onde $\Delta(x) = \{a \in M \mid x \to a\}$, então $P(x)$ é verdadeiro.*

O seguinte teorema é chamado de princípio de indução noetheriana.

Teorema 2.3. *Seja $(M, \to)$ um sistema de reescrita tal que $\to$ é terminante. Se P é um predicado $\to$-completo então para todo $x \in M$, $P(x)$ é verdadeiro.*

Demonstração. Seja $A = \{x \in M \mid P(x)$ é verdadeiro$\}$. Suponha por contradição que $A \neq M$. Nenhum dos elementos de $M - A$ é forma normal pois se z é forma normal $P(z)$ é verdadeiro por vacuidade já que $\Delta(z)$ é vazio. Logo deve existir um $y \in M - A$ de maneira que não exista $z \in M - A$ tal que $y \to z$ pois senão existiria uma seqüencia infinita $y = y_0 \to y_1 \to y_2 \to \cdots$ com $y_i \in M - A$, o que contradiz a terminalidade de $(M, \to)$. Como $\Delta(y)$ é não-vazio e $\Delta(y) \subseteq A$ temos que para todo $x \in \Delta(y)$ $P(x)$ é verdadeiro. Como P é $\to$-completo $P(y)$ também é verdadeiro o que contradiz a suposição de que $y \in M - A$. Logo $A = M$. □

Teorema 2.4. *Seja $(M, \to)$ um sistema de reescrita terminante. $(M, \to)$ é confluente se e somente se for localmente confluente.*

Demonstração. Se $(M, \to)$ é confluente e temos $z \leftarrow x \to y$ com $x, y, z \in M$ então, por definição, $z \,{}^*\!\leftarrow x \to^* y$. Pela confluência temos que $z \downarrow y$. Logo $(M, \to)$ é localmente confluente.

Suponha que $(M, \to)$ seja localmente confluente. Seja $P(x)$ um predicado sobre M que é verdadeiro se e somente se para quaisquer $x, y, z \in M$, $x \to^* y$ e $x \to^* z$ implicar que existe $w \in M$ tal que $y \to^* w$ e $z \to^* w$. Tome um $x \in M$ e suponha que $P(y)$ é verdadeiro para todo $y \in \Delta(x)$. Se $u, v \in M$ são tais que $x \to^* u$ e $x \to^* v$. Se $x = u$, $x = v$ ou $u = v$ então temos que $u \downarrow v$. Senão para $m, n > 0$ temos $x \to u' \to^{m-1} u$ e $x \to v' \to^{n-1} v$. Pela confluência local existe w tal que $u' \to^* w$ e $v' \to^* w$. $u' \in \Delta(x)$

logo $P(u')$ é verdadeiro e existe $y \in M$ tal que $u \to^* y$ e $w \to^* y$. $v' \in \Delta(x)$ logo $P(v')$ é verdadeiro. Temos que $v' \to^* y$ ($v' \to^* w \to^* y$) e $v' \to v$ logo existe $z \in M$ tal que $y \to^* z$ e $v \to^* z$. Desta maneira $u \to^* z$ ($u \to^* y \to^* z$) e $v \to^* z$ e portanto $P(x)$ é verdadeiro. Logo $(M, \to)$ é confluente. □

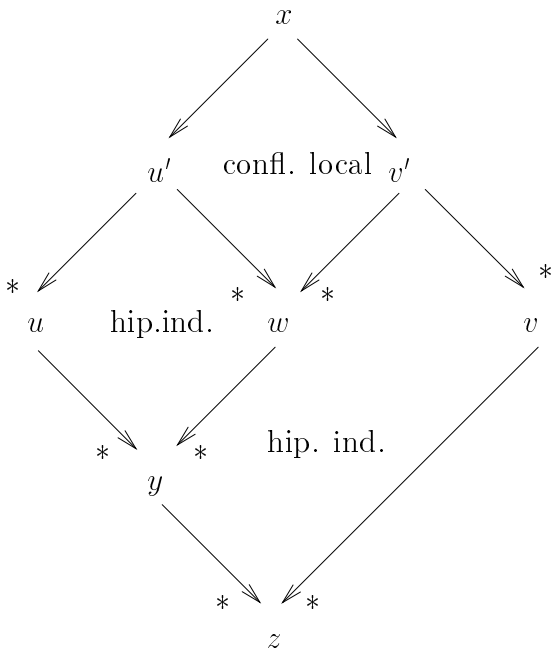

FIGURA 2. Confluência local implica confluência.

3. Sistemas de Reescrita de Palavras

Sistemas de reescrita de palavras [**BO93**] são um caso particular de sistemas de reescrita onde os conjuntos considerados são palavras sobre um determinado alfabeto.

DEFINIÇÃO 3.1. *Um* **sistema de reescrita de palavras** R *em* Σ *é um subconjunto de* $\Sigma^* \times \Sigma^*$. *Cada elemento* $(l, r) \in R$ *é chamado de* **regra de reescrita**. *A* **relação de reescrita** $\to_R$ *induzida por* R *é definida como: para quaiquer* $x, y \in \Sigma^*$, $x \to_R y$ *se e somente se existe* $(l, r) \in R$ *e* $u, v \in \Sigma^*$ *tais que* $x = ulv$ *e* $y = urv$.

Observe que se R é um sistema de reescrita de palavras então $\langle \Sigma^*, \to_R \rangle$ é um sistema de reescrita conforme a definição da seção anterior.

DEFINIÇÃO 3.2. *A relação de equivalência* $\leftrightarrow_R^*$ *é chamada de* **congruência de Thue**.

$\leftrightarrow_R^*$ é chamada de congruência pois é compatível com a concatenação (toda relação de equivalência compatível com a concatenação é chamada de congruência).

DEFINIÇÃO 3.3. *As palavras* $u, v \in \Sigma^*$ *são* **congruentes** *se* $u \leftrightarrow_R^* v$. *Para cada* $w \in \Sigma^*$ $[w]_R$ *é a* **classe de congruência** *de* w.

CAPÍTULO 3

O Procedimento de Knuth-Bendix para Palavras

1. Cálculo de Formas Normais

Já foi mencionado anteriormente que se um sistema de reescrita é convergente o seu problema da palavra é facilmente decidível pois se queremos saber se $u \leftrightarrow^* v$ basta que calculemos as respectivas formas normais $\overline{u}$ e $\overline{v}$. Nesta seção será mostrado como calcular as formas normais em questão.

DEFINIÇÃO 1.1. *Seja $>$ uma relação binária sobre Σ^*.*

1. $>$ *é uma* **ordem parcial estrita** *se é irreflexiva, antisimétrica e transitiva.*
2. $>$ *é uma* **ordem linear** *se é uma ordem parcial estrita e para quaisquer $x, y \in \Sigma^*$ se tem $x > y$ ou $x = y$ ou $y > x$.*
3. $>$ *é* **admissível** *se para quaisquer $u, v, x, y \in \Sigma^*$ se $u > v$ então $xuy > xvy$.*

DEFINIÇÃO 1.2. *Seja $\Sigma = \{a_1, ..., a_n\}$.*

1. *Seja $>_l$ a ordem dada por $x >_l y$ se $|x| > |y|$. $>_l$ é dita a* **ordem por comprimento** *em Σ^*.*
2. *Seja $w : \Sigma \to \mathbf{N}^+$ uma aplicação que associa cada símbolo de Σ a um natural não-nulo. A* **ordem por peso $>_w$ induzida por** *w é definida dada por $x >_w y$ se $w(x) > w(y)$. w pode ser extendida a uma aplicação de $\Sigma^* em \mathbf{N}$ fazendo $w(\epsilon) = 0$ e $w(xa) = w(x) + w(a)$ para $x \in \Sigma^*$ e $a \in \Sigma$.*
3. *A* **ordem lexicográfica** *$>_{lex}$ em Σ^* é definida como segue: $x >_{lex} y$ se existe uma palavra não-vazia z tal que $x = yz$, ou $x = ua_iv$ e $y = ua_jz$ para $u, v, z \in \Sigma^*$ e $i, j \in \{1, ..., n\}$ satisfazendo $i > j$.*
4. *A* **ordem lexicográfica e por comprimento** *$>_{ll}$ é dada por: $x >_{ll} y$ se $|x| > |y|$, ou se $|x| = |y|$ e $x >_{lex} y$.*

Observação. $>_l$, $>_w$, $>_{lex}$ e $>_{ll}$ são ordens parciais estritas admissíveis. $>_{lex}$ e $>_{ll}$ são lineares e $>_l$ e $>_w$ não.

DEFINIÇÃO 1.3. *Seja $>$ uma ordem parcial estrita em Σ^*. $>$ é* **bem-fundada** *se não existe cadeia infinita da forma $x_0 > x_1 > \cdots$. Se $>$ é linear e bem-fundada dizemos que $>$ é uma* **boa-ordem**.

Observação. $>_l$ e $>_w$ são bem-fundadas pois existe um número finito de palavras de um dado comprimento ou de um dado peso. Se $|\Sigma| > 1$, $>_{lex}$ não é bem-fundada pois neste caso temos a seguinte seqüência: $a_2 >_{lex} a_1a_2 >_{lex} a_1a_1a_2 >_{lex} \cdots >_{lex} a_1^n a_2 >_{lex} \cdots$.

TEOREMA 1.1. *Seja R um sistema de reescrita de palavras em Σ. As duas afirmações seguintes são equivalentes:*

1. *a relação de redução $\to_R$ é terminante.*
2. *existe uma ordem parcial estrita admissível e bem-fundada $>$ em Σ^* tal que $l > r$ vale para cada regra $(l, r) \in R$.*

DEMONSTRAÇÃO. Seja $>_R$ uma relação binária sobre Σ^* dada por: $x >_R y$ se e somente se $x \to_R^+ y$.

Suponha que $\to_R$ seja terminante.

1. $>_R$ é ordem parcial estrita.
   (a) Transitividade: $>_R$ é transitiva pois $\to_R^+$ também o é.
   (b) Antisimetria: suponha por contradição que existem $a, b \in \Sigma^*$ tais que $a >_R b$ e $b >_R a$. Logo existe uma cadeia infinita $a >_R b >_R a >_R b >_R \cdots$. Pela definição de $>_R$ existe uma cadeia infinita $a \to_R^+ b \to_R^+ a \to_R^+ b \to_R^+ \cdots$ e isto contradiz a terminalidade de $\to_R$.
   (c) Irreflexividade: suponha por contradição que existe $a \in \Sigma^*$ tal que $a >_R a$. Assim existem cadeias infinita $a >_R a >_R \cdots$ e $a \to_R^+ a \to_R^+ \cdots$ sendo que a última contradiz a terminalidade de $\to_R$.
2. Para quaisquer $(x, y) \in R$ e $u, v \in \Sigma^*$ se tem $uxv \to uyv$. $(x, y) \in R$ implica que $x \to_R y$ e $x >_R y$. De $uxv \to uyv$ se tem $uxv >_R uyv$. Logo $>_R$ é admissível.
3. $>_R$ é bem-fundada é consequüência direta da terminação de $\to_R$.
4. Se $(l, r) \in R$ então $l \to_R r$ e $l >_R r$.

Reciprocamente suponha por contradição que $\to_R$ não seja noetheriana. Logo existe uma seqüencia infinita de redução da forma $u_1 \to_R u_2 \to_R \cdots$. Para cada $u_i \to_R u_{i+1}$ temos que existem $x, y \in \Sigma^*$ e $(l, r) \in R$ tais que $u_i = xly$ e $u_{i+1} = xry$. R é compatível com $>$ logo $l > r$ e como $>$ é admissível temos $xly > xry$, ou seja $u_i > u_{i+1}$. Assim existiria uma seqüência infinita da forma $u_1 > u_2 > \cdots$, o que contradiz o fato de $>$ ser bem-fundada. □

Logo para verificar que um sistema de reescrita de palavras R em Σ é terminante é suficiente construir uma ordem parcial estrita $>$ que seja admissível e bem-fundada de maneira que R seja compatível com $>$, i.e. se $(l, r) \in R$ então $l > r$.

Observação. Se R é um sistema de reescrita de palavras finito então para cada $x \in \Sigma^*$, $\Delta(x)$ é finito, pois existe um número finito de regras que são aplicáveis. Assim se $\to_R$ é terminante temos que $\Delta^*(x)$ é finito pelo lema de König.

Definição 1.4. *Seja R um sistema de reescrita de palavras sobre Σ, uma redução $w \to_R z$ é* **mais à esquerda**, *denotada por $w \to^L z$, se a seguinte condição é satisfeita: se $w = x_1u_1y_1$, $z = x_1v_1y_1$ e $(u_1, v_1) \in R$ e também $w = x_2u_2y_2$ e $(u_2, v_2) \in R$ então x_1u_1 é um prefixo próprio de x_2u_2, ou $x_1u_1 = x_2u_2$ e x_1 é um prefixo próprio de x_2 ou $x_1 = x_2$ e $u_1 = u_2$.*

Notação. O fecho reflexivo e transitivo de $\to^L$ é denotado por $\to^{*L}$.

Definição 1.5. *Uma seqüência de reduções onde cada redução é mais à esquerda é dita uma* **redução mais à esquerda**.

Teorema 1.2. *Seja R um sistema de reescrita de palavras sobre Σ tal que $\to_R$ é terminante. Para cada $x \in \Sigma^*$ existe um $y \in \Sigma^*$ irredutível tal que $x \to^{*L} y$.*

Demonstração. Seja $x = x_1x_2 \cdots x_n \in \Sigma^*$. Se x é redutível seja S o conjunto de subpalavras de x tais que se $s \in S$ se e somente se $s \in dom(R)$. S é finito pois é subconjunto do conjunto de subpalavras de x. Seja $1 \leq i_k \leq n$ tal que existe $x_{i_1}x_{i_2} \cdots x_{i_k} \in S$ mínimo, i.e. qualquer elemento $x_{j_1}x_{j_2} \cdots x_{j_l} \in S$ é tal que $i_k \leq j_l$. Seja $S' = \{x_{m_1} \cdots x_{m_s} \in S \mid m_s = i_k\}$ e $l = x_{m_1} \cdots x_{m_s}$ um elemento de S' tal que m_1 seja mínimo. Usando a regra $(l, r) \in R$, para algum r, obtemos uma redução

$x \to_R y$, que é uma redução mais à esquerda. Seguindo desta maneira eventualmente atingiremos uma forma normal, pois caso contrário a cadeia de redução seria infinita e contradiria a hipótese de terminalidade. □

TEOREMA 1.3. *Seja R um sistema de reescrita de palavras sobre Σ. Seja R' um subconjunto qualquer de R tal que $dom(R') = dom(R)$. Então*

1. $IRR(R') = IRR(R)$
2. *para quaisquer $x, y \in \Sigma^*$, se $x \to_{R'}^{*L} y$ então $x \to_R^{*L} y$.*

DEMONSTRAÇÃO. $w \in IRR(R')$ se e somente se não existe $v \in \Sigma^*$ tal que $w \to_{R'} v$. Mas isto ocorre se e somente se não existe $l \in \Sigma^*$ tal que $w = xly$ e $(l, r) \in R'$ para $w, y, r \in \Sigma^*$. Equivalentemente não existe $l \in \Sigma^*$ tal que $w = xly$ e $(l, r) \in R$ para $w, y, r \in \Sigma^*$. Assim não existe $v \in \Sigma^*$ tal que $w \to_R v$ e $w \in IRR(R)$.

Se $x \to_{R'}^{*L} y$ então existe uma seqüência de reduções mais à esquerda que começam com x e terminam com y. Seja $x_i \to_{R'}^{L} x_{i+1}$ um passo de redução. Logo, como $R' \subseteq R$, $x_i \to_R^L x_{i+1}$. Desta maneira $x \to_R^{*L} y$. □

TEOREMA 1.4. *Seja R um sistema de reescrita de palavras finito sobre Σ tal que $\to$ é terminante. Existe um algoritmo para resolver o seguinte problema.*

Entrada: *uma palavra $w \in \Sigma^*$.*

Saída: *uma palavra irredutível $\overline{w} \in \Sigma^*$ tal que $w \to_R^{*L} \overline{w}$.*

DEMONSTRAÇÃO. Seja $R' \subseteq R$ tal que $dom(R') = dom(R)$ e para cada $u \in dom(R)$ existe um único $v \in im(R)$ tal que $(u, v) \in R'$. Assim, pelo teorema anterior, $IRR(R') = IRR(R)$ e para todo $x \in \Sigma^*$ existe um único x' irredutível tal que $x \to_L^* x'$ Considere o seguinte procedimento:

Algoritmo FNORMAL.

Entrada: $w \in \Sigma^*$ e R' s.r.p. sobre Σ.

$S := \{l \sqsubseteq w \mid \exists (l, r) \in R'\}$;

enquanto $S \neq \emptyset$

Seja $l \in S$ tal que $\forall s \in S$, l está mais à esquerda que s em w;

Seja w' tal que $w \to_{R'} w'$ usando a regra $(l, r) \in R'$;

$w := w'$;

$S := \{l \sqsubseteq w \mid \exists (l, r) \in R'\}$;

retorne w;

Fim.

Como cada execução do laço **enquanto** corresponde a um passo de redução mais à esquerda e R é terminante o algoritmo sempre pára e $w \to^{*L} w'$, quando w' é a palavra dada como saída pelo algoritmo. w' deve ser irredutível pois esta é a condição de saída do procedimento. □

Definição 1.6. *Um sistema de reescrita de palavras em Σ é* **normalizado** *se as seguintes condições valem para cada regra $(l, r) \in R$:*

1. $l \in IRR(R - (l, r))$
2. $r \in IRR(R)$

Definição 1.7. *Dois sistemas de reescrita de palavras R e S sobre o mesmo alfabeto Σ são equivalentes se geram a mesma congruência de Thue, i.e. $\leftrightarrow_R^* = \leftrightarrow_S^*$.*

Teorema 1.5. *Seja $>$ uma boa-ordem admissível em Σ^* e R um sistema de reescrita de palavras sobre Σ compatível com $>$. Para cada $w \in \Sigma^*$ seja $\overline{w}$ um descendente irredutível de w. Seja $R_0 = \{l \to \overline{r} | (l, r) \in R\}$. Então R_0 é equivalente a R, é compatível com $>$ e $im(R_0) \subseteq IRR(R_0)$. Adicionalmente se R é confluente então R_0 é confluente.*

Demonstração. Se $(l, r') \in R_0$ então existe $r \in \Sigma^*$ tal que $r \to_R^* r'$ e $(l, r) \in R$. Logo $l > r$, $r > r'$ e assim $l > r'$. Portanto R_0 é compatível com $>$.

$im(R_0)$ está, pela definição de R_0, contida em $IRR(R)$. Pelo teorema 1.3 sabemos que $IRR(R) = IRR(R_0)$. Assim $im(R_0) \subseteq IRR(R_0)$.

Mostremos agora que R e R_0 são equivalentes, ou equivalentemente que $\leftrightarrow_R^* = \leftrightarrow_{R_0}^*$:

1. $\to_R \subseteq \leftrightarrow_{R_0}^*$ (e conseqüentemente $\leftrightarrow_R^* \subseteq \leftrightarrow_{R_0}^*$): suponha por contradição que $(l, r) \in R$ e que $l \not\leftrightarrow_{R_0}^* r$. Como $>$ é uma boa-ordem podemos supor que (l, r) é tal que r é mínimo para esta propriedade por $>$. Como $l \not\leftrightarrow_{R_0}^* r$ temos que $(l, r) \notin R_0$. Adicionalmente temos que $r \to_R^+ r'$ e $(l, r') \in R_0$ para algum $r' \in \Sigma^*$. No entanto

para cada regra $(l_1, r_1) \in R$ usada na redução $r \rightarrow_R^+ r'$ temos que $r > r_1$. Logo pela escolha de (l, r) temos que $l_1 \leftrightarrow_{R_0}^* r_1$ e $r \leftrightarrow_{R_0}^* r' \leftrightarrow_{R_0}^* l$ (contradição). Portanto $\rightarrow \subseteq \leftrightarrow_{R_0}^*$.

2. $\leftrightarrow_{R_0}^* \subseteq \leftrightarrow_R^*$: Se $x = ulv \rightarrow_{R_0}^* y = urv$ com $(l, r) \in R_0$ então existe $r' \in \Sigma^*$ tal que $ulv \rightarrow_R ur'v \rightarrow urv$. Logo $x \rightarrow_R^* y$ e $\rightarrow_{R_0}^* \subseteq \rightarrow_R^*$. Com o fecho simétrico temos $\leftrightarrow_{R_0}^* \subseteq \leftrightarrow_R^*$.

Se R é confluente então cada classe de congruência $[w]_R$ para $w \in \Sigma^*$ contém uma única palavra irredutível. Como $IRR(R) = IRR(R_0)$ e R e R_0 são equivalentes então R_0 também é confluente. □

Embora R_0 do teorema anterior tenha os lados direitos das regras reduzidos ele ainda não é necessáriamente normalizado. O seguinte algoritmo realiza o processo de normalização reduzindo também os lados esquerdos das regras.

Algoritmo NORMALIZA.

Entrada: Uma boa-ordem admissível $>$ sobre Σ^* e um sistema de reescrita de palavras R sobre Σ compatível com $>$.

$R_1 := R$;

Reduza os lados direitos de R_1 com **FNORMAL**;

enquanto $\exists (l_1, r_1), (l_2, r_2) \in R_1$ $\exists x, y \in \Sigma^*$ tais que $l_2 = xl_1y$ **e** $(xy \neq \epsilon$ **ou** $r_2 > r_1)$

$R_1 := R_1 - \{l_2 \rightarrow r_2\}$;

se $r_2 \notin \Delta_{R_1}^*(xr_1y)$

se $xr_1y > r_2$

$R_1 := R_1 \cup \{xr_1y \rightarrow r_2\}$;

senão

$R_1 := R_1 \cup \{r_2 \rightarrow xr_1y\}$;

Reduza os lados direitos de R_1 com **FNORMAL**;

retorne R_1;

Fim.

Dada $>$ uma boa-ordem admissível sobre Σ^* seja $>_2$ a ordem sobre $\Sigma^* \times \Sigma^*$ definida a seguir: dados $x_1, x_2, y_1, y_2 \in \Sigma^*$, $(x_1, x_2) >_2 (y_1, y_2)$ se $x_1 > x_2$ ou se $x_1 = y_1$ e $x_2 > y_2$. Seja $S_1, S_2 \subseteq \Sigma^* \times \Sigma^*$ $\gg_2$ a ordem sobre multi-conjuntos induzida por $>_2$, ou seja

$S_1 \gg_2 S_2$ se e somente se existe um subconjunto não-vazio $T_1 \subseteq S_1$ e um subconjunto $T_2 \subseteq S_1$ tal que $S_2 = (S_1 - T_1) \cup T_2$ e para cada par $(y_1, y_2) \in T_2$ existe um par $(x_1, x_2) \in T_1$ tal que $(x_1, x_2) >_2 (y_1, y_2)$. Em [**DM79**] é demonstrado que uma ordem sobre multi-conjuntos induzida por uma boa-ordem é também uma boa-ordem. Assim podemos assumir que $\gg_2$ é uma boa-ordem.

TEOREMA 1.6. *Dada como entrada uma ordem admissível $>$ sobre Σ^* e um sistema de reescrita de palavras finito R sobre Σ compatível com $>$ o algoritmo* **NORMALIZA** *retorna um sistema de reescrita de palavras normalizado e finito R_1 tal que R_1 é equivalente a R e compatível com $>$.*

DEMONSTRAÇÃO. Sempre que, no algoritmo **NORMALIZA**, uma regra $l_2 \to r_2$ é removida de R_1 então existe uma outra regra $(l_1, r_1) \in R_1$ tal que $l_2 = xl_1y$, ou seja $r_2 \leftrightarrow^*_{R_1} xr_1y$. Se $xr_1y \to^*_{R_1} r_2$ então a regra removida não é usada na redução e portanto $R_1 - \{l_2 \to r_2\}$ é equivalente a R_1. Se $xr_1y \not\to^*_{R_1} r_2$ então ou a regra $xr_1y \to r_2$ ou a regra $r_2 \to xr_1y$ são adicionadas ao sistema. Logo o sistema resultante é equivalente a R_1. Pela mesma observação o sistema resultante também é compatível com $>$ e a sua imagem está contida em $IRR(R)$.

Falta ainda demonstrar que o algoritmo sempre pára. Inicialmente, para uma entrada $(>, R)$, é definido um sistema R_0 que resulta de R substituindo todo lado direito das regras por um descendente irredutível. Logo ou $R_0 = R$ ou $R \gg_2 R_0$. Para todo $i \geq 1$ seja R_{i-1} o sistema de reescrita de palavras obtido na i-ésima iteração do laço **enquanto**. Mostremos que para todo $i \geq 1$ temos $R_{i-1} \gg_2 R_i$. O laço **enquanto** é realizado pela i-ésima vez se e somente se existem regras (l_1, r_1), $(l_2, r_2) \in R_{i-1}$ e $x, y \in \Sigma^*$ tais que $l_2 = xl_1y$ com $xy \neq \epsilon$ ou $r_2 > 1$. Neste caso $(l_2 \to r_2)$ é removida de R_{i-1}. Se $r_2 \in \Delta^*_{R_{i-1}-\{l_2 \to r_2\}}(xr_1y)$ então $R_i = R_{i-1} - \{l_2 \to r_2\}$ e assim temos que $R_{i-1} \gg_2 R_i$. Caso $r_2 \notin \Delta^*_{R_{i-1}-\{l_2 \to r_2\}}(xr_1y)$ obtemos um conjunto R'_i de R_{i-1} acrescentando a regra $xr_1y \to r_2$ ou a regra $r_2 \to xr_1y$. Como $l_2 > r_2$, $l_1 > r_1$ e $>$ é uma ordem admissível temos que $l_2 = xl_1y > xr_1y$ e em ambos os casos $R_{i-1} \gg_2 R'_i$. Mas como R_i é obtido de R'_i através da redução do lado direito das regras temos que $R'_i \gg_2 R_i$. Logo $R_{i-1} \gg_2 R_i$.

Desta maneira, como $\gg_2$ é uma boa ordem, não pode existir uma cadeia infinita da forma $R_0 \gg_2 R_1 \cdots$ e o laço **enquanto** deve ser realizado um número finito de vezes e conseqüentemente o algoritmo **NORMALIZA** deve parar sempre. □

TEOREMA 1.7. *Seja $>$ uma ordem parcial bem-fundada e admissível sobre Σ^* e sejam R_1 e R_2 dois sistemas de reescrita de palavras sobre Σ que sejam confluentes, normalizados e compatíveis com $>$. Se R_1 e R_2 são equivalentes então eles são idênticos.*

DEMONSTRAÇÃO. Suponha por contradição que R_1 e R_2 sejam equivalentes mas que não sejam idênticos e seja $(l, r) \in (R_1 - R_2) \cup (R_2 - R_1)$, ou seja um elemento da diferença simétrica de R_1 e R_2. De todas as regras na diferença simétrica de R_1 e R_2 seja (l, r) tal que $|l|$ é mínimo. Suponha que $(l, r) \in R_1$ (o caso $(l, r) \in R_2$ é simétrico). Logo, como R_1 é normalizado, $r \in IRR(R_1)$ e para todo $w \in [r]_{R_1}$ (distinto de r) $w > r$ pela confluência e compatibilidade com $>$. R_1 e R_2 são equivalentes e assim $[r]_{R_1} = [r]_{R_2}$. Como R_2 também é confluente e compatível com $>$ temos que $w \to^*_{R_2} r$ para todo $w \in [r]_{R_1}$ e em particular $l \to^*_{R_2} r$. Como $(l, r) \notin R_2$ temos que ou $(l, r') \in R_2$ e $r' \to_2 r$, o que contradiz a normalidade de R_2, ou $(l', r') \in R_2$ tal que $l = xl'y$ para $x, y \in \Sigma^*$, $xy \neq \epsilon$ e $xr'y \to r$. No entanto temos que $|l'| < |l|$ e $(l', r') \in R_1$ (pela minimalidade de l temos que (l', r') deve estar na interseção de R_1 e R_2). Isso contradiz a normalidade de R_1. □

2. Teste de Confluência

Suponha que dado um sistema de reescrita de palavras R sobre Σ tenhamos para $w, x, y \in \Sigma^*$ que $w \to_R x$, $w \to_R y$, $w = w_1u_1w_2u_2w_3$, $(u_1, v_1), (u_2, v_2) \in R$, $x = w_1v_1w_2u_2w_3$ e $y = w_1u_1w_2v_2w_3$ então x e y têm $w_1v_1w_2v_2w_3$ como descendente comum e portanto são juntáveis. Logo é necessário considerar apenas pares de regras de reescrita de uma forma especial definida a seguir.

DEFINIÇÃO 2.1. *Existem dois tipos de* **pares críticos** *em um sistema de reescrita de palavras R:*

1. *Sejam $xyz \to u$ e $y \to v$ duas regras distintas de R, onde $x, y, z, u, v \in \Sigma^*$, então $u \leftarrow_R xyz \to_R xvz$ e (u, xvz) é um* par crítico.

2. *Sejam* $xy \to u$ *e* $yz \to v$ *duas regras distintas de* R*, onde* $x, y, z, u, v \in \Sigma^*$ *e* $x, y, z \neq \epsilon$*, então* $uz \leftarrow_R xyz \rightarrow_R xv$ *e* (uz, xv) *é um* par crítico.

TEOREMA 2.1. *Existe um algoritmo que dado um sistema de reescrita de palavras* R *tal que* $\rightarrow_R$ *seja terminante decida se* R *é localmente confluente.*

DEMONSTRAÇÃO. O conjunto de pares críticos de um sistema de reescrita R é facilmente determinado com algoritmos de reconhecimentos de padrões. Basta então para cada par crítico $\langle z_1, z_2 \rangle$ calcular as respectivas formas normais z_1' e z_2', se para algum par crítico estas formas normais forem distintas então R não é localmente confluente. □

Foi visto que, sob hipótese de terminação, confluência e confluência local são equivalentes. Assim o teorema anterior tem como consequência imediata o seguinte corolário.

COROLÁRIO 2.1. *Existe um algoritmo que dado um sistema de reescrita de palavras* R *tal que* $\rightarrow_R$ *seja terminante decida se* R *é confluente.*

3. Procedimento de Knuth-Bendix

O seguinte procedimento toma como entrada um sistema de reescrita de palavras e uma boa-ordem admissível e tenta gerar um sistema de reescrita confluente, compatível com a ordem de entrada e que seja equivalente ao sistema de reescrita original. Para tal, o procedimento itera orientando pares críticos não-juntáveis, incluindo estes pares no sistema de reescrita. O procedimento pára quando todos os pares críticos são juntáveis, é possível que isto nunca aconteça, i.e. sempre existam pares críticos não juntáveis.

Procedimento KNUTH-BENDIX.

Entrada: Um sistema de reescrita finito R sobre Σ e uma boa-ordem admissível $>$ sobre Σ^*.

$R_0 := \{(l, r) \mid l > r \text{ e } (l, r) \in R \text{ ou } (r, l) \in R\}$;

$i := -1$;

repita

$i := i + 1$;

$R_{i+1} := \emptyset$;

$CP :=$ pares críticos de R_i;
enquanto $CP \neq \emptyset$
seja $\langle z_1, z_2 \rangle \in CP$;
sejam z_1' e z_2' formas normais por R_i de z_1 e z_2 respectivamente;
se $z_1' > z_2'$
$R_{i+1} := R_{i+1} \cup \{(z_1', z_2')\}$;
senão
$R_{i+1} := R_{i+1} \cup \{(z_2', z_1')\}$;
$CP := CP - \{(z_1, z_2)\}$
fim enquanto
se $R_{i+1} \neq \emptyset$
$R_{i+1} := R_i \cup R_{i+1}$;
até que $R_{i+1} = \emptyset$;
$R^* := \bigcup_{i \geq 0} R_i$;
retorne R^*;
Fim.

TEOREMA 3.1. *Seja R um sistema de reescrita de palavras finito sobre Σ e seja $>$ uma boa-ordem admissível sobre Σ^*. Além disso seja R^* um sistema de reescrita de palavras enumerado por* **KNUTH-BENDIX** *para a entrada $(R, >)$. Então as seguintes afirmações são verdadeiras:*

1. *R^* é convergente.*
2. *R e R^* são equivalentes.*
3. *$IRR(R^*) = \{x \in \Sigma^* \mid \forall y \in \Sigma^* : se\ x \leftrightarrow_R^* y\ então\ y = x\ ou\ y > x\}$.*

DEMONSTRAÇÃO. O procedimento **KNUTH-BENDIX** calcula uma seqüência de sistemas de reescrita de palavras $R_0, R_1, R_2, ...$ sobre Σ. Se $\langle z_1, z_2 \rangle$ é um par crítico de R_i então $z_1' \leftrightarrow_{R_i}^* z_2'$. Assim é possível mostrar por indução que $R_0, R_1, ...$ e R^* são equivalentes a R. No laço **enquanto** os pares críticos não juntáveis são orientados de acordo com $>$, logo R_i é compatível com $>$ e conseqüentemente terminante, o mesmo valendo para R^*. Seja $\langle z_1, z_2 \rangle$ um par crítico de R^* resultante das regras

$(l_1, r_1), (l_2, r_2) \in R^*$. Logo existe um índice $i \geq 0$ tal que $(l_1, r_1), (l_2, r_2) \in R_i$. O procedimento resolve o par crítico $\langle z_1, z_2 \rangle$ de maneira que em R_{i+1} z_1 e z_2 sejam juntáveis. Conseqüentemente z_1 e z_2 são juntáveis em R^* também implicando que R^* é confluente. Foi visto que em sistemas convergentes as formas normais são únicas o que implica $IRR(R^*) = \{x \in \Sigma^* \mid \forall y \in \Sigma^* : \text{se } x \leftrightarrow_R^* y \text{ então } y = x \text{ ou } y > x\}$. □

Corolário 3.1. *Para cada sistema de reescrita de palavras R sobre Σ e cada boa-ordem admissível $>$ sobre Σ^* exite um sistema de reescrita R^* sobre Σ possívelmente confluente tal que:*

1. *R e R^* são equivalentes*
2. *$l > r$ para cada regra $(l, r) \in R^*$*

Teorema 3.2. *O procedimento* **KNUTH-BENDIX** *pára para a entrada $(R, >)$ se e somente se existe um sistema de reescrita de palavras confluente e finito R' sobre Σ tal que R e R' sejam equivalentes e $l > r$ para cada regra $(l, r) \in R'$.*

Observação. Nos exemplos subseqüentes da dissertação o inverso de um gerador a será representado por A ao invés de a^{-1}.

Exemplo 3.1. Consideremos o grupo D_8 (grupo de simetrias do quadrado) apresentado pelo sistema de reescrita de palavras abaixo compatível com a ordem por comprimento e lexicográfica induzida por $\{A > a > B > b\}$.

[1] $aA \to \epsilon$
[2] $Aa \to \epsilon$
[3] $bB \to \epsilon$
[4] $Bb \to \epsilon$
[5] $a^4 \to \epsilon$
[6] $b^2 \to \epsilon$
[7] $Baba \to \epsilon$

1ª Iteração:

Pares Críticos:

[1, 2] $A \leftarrow AaA \to A$

[1, 2] $a \leftarrow aAa \to a$

[1, 5] $a^3 \leftarrow a^4A \to A$, nova regra: $a^3 \to A$

[1, 7] $Bab \leftarrow BabaA \to A$, nova regra $Bab \to A$

[2, 5] $a^3 \leftarrow Aa^4 \to A$

$[3,4]$ $b \leftarrow bBb \rightarrow b$

$[3,4]$ $B \leftarrow BbB \rightarrow B$

$[3,6]$ $B \leftarrow b^2B \rightarrow b$, nova regra $B \rightarrow b$

$[4,6]$ $b \leftarrow Bb^2 \rightarrow B$

$[5,7]$ $A \leftarrow a^3 \leftarrow Baba^4 \rightarrow Bab$

Simplificação:

Conjunto de regras atual:

[1] $aA \rightarrow \epsilon$

[2] $Aa \rightarrow \epsilon$

[3] $bB \rightarrow \epsilon$, elimina-se por [10]

[4] $Bb \rightarrow \epsilon$, elimina-se por [10]

[5] $a^4 \rightarrow \epsilon$, elimina-se por [8]

[6] $b^2 \rightarrow \epsilon$

[7] $Baba \rightarrow \epsilon$, elimina-se por [9]

[8] $a^3 \rightarrow A$

[9] $Bab \rightarrow A$, substitui-se por $bab \rightarrow A$ [11]

[10] $B \rightarrow b$

Conjunto de regras resultante:

[1] $aA \rightarrow \epsilon$	[8] $a^3 \rightarrow A$
[2] $Aa \rightarrow \epsilon$	[10] $B \rightarrow b$
[6] $b^2 \rightarrow \epsilon$	[11] $bab \rightarrow A$

2^a Iteração:

Pares Críticos:

$[1,8]$ $A^2 \leftarrow a^3A \rightarrow a^2$, nova regra $A^2 \rightarrow a^2$

$[2,8]$ $A^2 \leftarrow Aa^3 \rightarrow a^2$

$[6,11]$ $bA \leftarrow b^2ab \rightarrow ab$, nova regra $bA \rightarrow ab$

$[6,11]$ $Ab \leftarrow bab^2 \rightarrow ba$, nova regra $ba \rightarrow Ab$

Simplificação:

[1] $aA \to \epsilon$
[2] $Aa \to \epsilon$
[6] $b^2 \to \epsilon$
[8] $a^3 \to A$
[10] $B \to b$
[11] $bab \to A$, elimina-se por [14]
[12] $A^2 \to a^2$
[13] $bA \to ab$
[14] $ba \to Ab$

3^a Iteração:

Pares Críticos:

$[1, 12]$ $A \leftarrow aA^2 \to a^3 \to A$

$[1, 14]$ $b \leftarrow baA \to AbA \to Aab \to b$

$[2, 12]$ $A \leftarrow A^2a \to a^3 \to A$

$[2, 13]$ $b \leftarrow bAa \to aba \to aAb \to b$

$[6, 13]$ $A \leftarrow Ab^2 \leftarrow bab \leftarrow b^2A \to A$

$[6, 14]$ $a \leftarrow ab^2 \leftarrow bAb \leftarrow b^2a \to a$

Todos os pares críticos são juntáveis, logo o procedimento pára com sistema de reescrita de palavras:

[1] $aA \to \epsilon$
[2] $Aa \to \epsilon$
[6] $b^2 \to \epsilon$
[8] $a^3 \to A$
[10] $B \to b$
[12] $A^2 \to a^2$
[13] $bA \to ab$
[14] $ba \to Ab$

O sistema de reescrita de palavras acima é convergente portanto nos fornece uma maneira simples para resolver o problema da palavra do grupo D_8. Por exemplo a palavra $aababA$ representa a identidade já que sua forma normal é ϵ:

$$aa\underline{ba}bA \to a\underline{aA}bbA \to a\underline{bb}A \to \underline{aA} \to \epsilon$$

Observe que as classes de congruência de $\leftrightarrow^*$ correspondem aos elementos do grupo D_8. Estas classes podem ser determinadas calculando-se as formas normais de Σ^* pelo sistema de reescrita e são dadas por $[\epsilon]$, $[a]$, $[a^2]$, $[A]$, $[b]$, $[ab]$, $[a^2b]$, $[Ab]$. A forma normal de um elemento $w \in \Sigma^*$ determina em que classe de congruência está w ou equivalentemente que elemento do grupo w representa. Assim podemos verificar que Aba^3Ba e $Baba^5B$ representam elementos distintos do grupo:

$$Ab\underline{a^3}Ba \to A\underline{bA}Ba \to \underline{Aa}bBa \to \underline{bB}a \to a$$

$$\underline{B}aba^5B \to \underline{ba}ba^5B \to A\underline{bb}a^5B \to \underline{Aa}a^4B \to \underline{a^3}aB \to \underline{Aa}B \to \underline{B} \to b$$

Logo $Aba^3Ba \in [a]$ e $Baba^5B \in [b]$ representam elementos distintos de D_8. $\diamond$

CAPÍTULO 4

Completação Baseada em Ciclos de Palavras

1. Introdução

Neste capítulo será apresentado um procedimento completador para grupos finitamente apresentados proposto por Friedrich Otto e Robert Cremanns [**CO98**] que se utiliza de uma estrutura chamada de *ciclo de palavras* para representar conjuntos de regras de reescrita.

DEFINIÇÃO 1.1. *Seja Σ um alfabeto finito e $^{-1} : \Sigma \to \Sigma$ uma função tal que $(a^{-1})^{-1} = a$ para todo $a \in \Sigma$. Esta função pode ser estendida para todo Σ^* fazendo $wa \mapsto a^{-1}w^{-1}$ para todo $a \in \Sigma$ e $w \in \Sigma^*$.*

DEFINIÇÃO 1.2. *Seja $\sim$ uma congruência sobre Σ^* satisfazendo $aa^{-1} \sim \epsilon$.*

Observação. Para todo $u, v \in \Sigma^*$, se $u \sim \epsilon$ então $u^{-1} \sim \epsilon$ e se $uv \sim \epsilon$ então $vu \sim \epsilon$. Assim a classe de congruência de ϵ, denotada por $[\epsilon]_\sim$ é fechada para inversos e permutações cíclicas.

DEFINIÇÃO 1.3. *Seja $\approx$ a menor relação de equivalência sobre Σ^* satisfazendo $u \approx u^{-1}$ e $uv \approx vu$. As classes de equivalência de $\approx$ são ditas* **ciclos de palavras**.

Observação. Pela minimalidade de $\approx$ a classe de equivalência de um elemento $w \in \Sigma^*$, denotada por $[w]_\approx$, é dada por $[w]_\approx = \{x \mid x$ é permutação cíclica ou inverso de permutação cíclica de $w\}$. Assim $|[w]_\approx|$ será no máximo $2|w|$.

Exemplo. Se $\Sigma = \{a, b\}$ então

$$[\epsilon]_\approx = \{\epsilon\},$$

$$[a^3]_\approx = \{a^3, a^{-3}\},$$

$$[abab^{-1}]_\approx = \{abab^{-1}, b^{-1}aba, ab^{-1}ab, bab^{-1}a, ba^{-1}b^{-1}a^{-1},$$

$$a^{-1}ba^{-1}b^{-1}, b^{-1}a^{-1}ba^{-1}, a^{-1}b^{-1}a^{-1}b\}.$$

DEFINIÇÃO 1.4. *O ciclo de palavras* $[\epsilon]_\approx$ *é dito* **ciclo de palavras vazio**. *Um ciclo de palavras* $[w]_\approx$ *é dito* **livremente reduzido** *se toda palavra* $z \in [w]_\approx$ *é livremente reduzida.*

Seja Z um conjunto de ciclos de palavras, a congruência $\sim_Z$ gerada por Z é definida como a congruência gerada pelo conjunto $L(Z) = \{x \in \Sigma^* \mid [w]_\approx\}$, isto é, a congruência gerada pelo sistema de reescrita de palavras $R_{L(Z)} = \{(w \to \epsilon) \mid w \in \Sigma^*, [w]_\approx \in Z\} \cup \{aa^{-1} \to \epsilon \mid a \in \Sigma\}$.

DEFINIÇÃO 1.5. *Seja* $\sim$ *uma congruência sobre* Σ^* *tal que* $aa^{-1} \sim \epsilon$ *para todo* $a \in \Sigma$. *O* **problema da palavra** $WP_\sim$ *é o conjunto de todos os ciclos de palavras* $[w]_\approx$ *tais que* $w \sim \epsilon$. *O* **problema da palavra reduzido** $RWP_\sim$ *é o conjunto de todos os ciclos de palavras* $[w]_\approx$ *para os quais* w *é uma palavra livremente reduzida tal que* $w \sim \epsilon$ *e nenhuma subpalavra* w' *de* w *é tal que* $w' \sim \epsilon$.

2. Procedimento

O procedimento recebe como entrada um conjunto de ciclo de palavras consistindo exatamente das relações do grupo finitamente apresentado e aplica as regras de inferência descritas em seguida no processo de completação. Seja Z um conjunto de ciclos de palavras.

Sistema de inferência P:

- P.1 Se $[\epsilon] \in Z$ remova-o de Z.
- P.2 Se $u \in \Sigma^*$ e $a \in \Sigma$ são tais que $[uaa^{-1}]_\approx \in Z$, então substitua $[uaa^{-1}]_\approx$ por $[u]_\approx$ em Z.
- P.3 Se $u, v \in \Sigma^+$ são tais que $[u]_\approx \in Z$ e $[uv]_\approx \in Z$ então substitua $[uv]_\approx$ por $[v]_\approx$ em Z.
- P.4 Se $u, x, y \in \Sigma^+$ são tais que $[ux]_\approx \in Z$ e $[uy]_\approx \in Z$, onde x e y não começam e não terminam na mesma letra então adicione $[xy^{-1}]_\approx$ em Z.

Sejam Z e Z' ciclos de palavras, $Z \vdash_P Z'$ denota que Z' é obtido de Z pela aplicação de uma regra de inferência de P.

DEFINIÇÃO 2.1. *Uma P-***derivação** *é uma seqüência finita* $(Z_i)_{i\in I}$ *de conjuntos de ciclos de palavras (onde* $I = \{n \in \quad | \ n \leq k\}$ *para algum* $k \in \quad$ *ou* $I = \quad$*) tal que para todo* $i \in I - \{0\}$, $Z_{i-1} \vdash_P Z_i$.

DEFINIÇÃO 2.2. *Seja* $(Z_i)_{i\in I}$ *uma P-derivação. Um ciclo de palavras é dito* **persistente** *se para algum* $i \in I$ *ele pertence a* Z_j *para todo* $j \in I$ *tal que* $j \geq i$. $Z = \bigcup_{i\in I} Z_i$ *e* Z^∞ *é o conjunto de todos os ciclos de palavras persistentes de* $(Z_i)_{i\in I}$.

LEMA 2.1. *Seja* $(Z_i)_{i\in I}$ *uma P-derivação e* w *uma palavra livremente reduzida e não-vazia tal que* $[w]_\approx \in Z$. *Então* w *tem uma subpalavra não-vazia* w' *tal que* $[w']_\approx \in Z^\infty$.

DEMONSTRAÇÃO. A demonstração é realizada por indução em $|w|$.

Se $|w| = 1$ e $w \in Z$ então $w = a$ para algum $a \in \Sigma$ e claramente $[a]_\approx \in Z^\infty$.

Seja w uma palavra não-vazia livremente reduzida tal que $[w]_\approx \in Z - Z^\infty$. Logo existe um índice $i \in I$ tal que $[w]_\approx \in Z_i$ mas $[w]_\approx \notin Z_{i+1}$. Logo Z_{i+1} é obtido de Z_i por uma aplicação da regra de inferência P.2 ou da regra de inferência P.3. Se a regra de inferência P.2 foi aplicada então $[w]_\approx = [uaa^{-1}]_\approx$ para algum $u \in \Sigma^*$ e $a \in \Sigma$. Assim $[u]_\approx \in Z_{i+1}$. Como w é livremente reduzido temos que $w = a^{-1}ua$ ou $w = au^{-1}a^{-1}$ e que $u \neq \epsilon$. Se $w = a^{-1}ua$ aplicamos a hipótese de indução a u e se $w = au^{-1}a^{-1}$ aplicamos a hipótese de indução a u^{-1}. Logo w' é uma subpalavra não-vazia de u ou u^{-1} que é também subpalavra de w e é tal que $[w']_\approx \in Z^\infty$. Se a regra de inferência P.3 foi aplicada então $[w]_\approx = [uv]_\approx$ para $u, v \in \Sigma^+$ tais que $[u]_\approx$ e $[v]_\approx \in Z_{i+1}$. Então u, u^{-1}, v, v^{-1} é uma subpalavra própria de w. Aplicando a hipótese de indução a u, u^{-1}, v ou v^{-1} obtemos uma subpalavra não vazia w' de w tal que $[w']_\approx \in Z^\infty$. □

DEFINIÇÃO 2.3. *Um conjunto de ciclos de palavras é* **inter-reduzido** *se as regras de inferência P.1, P.2 e P.3 não são aplicáveis.*

DEFINIÇÃO 2.4. *Uma P-***derivação** $(Z_i)_{i\in I}$ *é* **honesta**[1] *se as seguintes propriedades são satisfeitas:*

1. Z^∞ *é inter-reduzido.*

[1] regras de inferência não deixaram de ser aplicadas

2. *Para quaisquer* $u, x, y \in \Sigma^+$, *se* $[ux]_\approx, [uy]_\approx \in Z^\infty$, *e* x, y *não começam e nem terminam com o mesmo símbolo então* $[xy^{-1}]_\approx \in Z$.

Definição 2.5. *Um* **procedimento completador** *toma como entrada um conjunto de ciclos de palavras* Z_0 *e gera uma* P*-derivação honesta* $(Z_i)_{i\in I}$ *tal que se o conjunto de ciclos de palavras persistentes* Z^∞ *é finito a* P*-derivação gerada é finita também e o procedimento pára dando como saída* Z^∞.

Procedimento COMPLETA-CICLOS.

Entrada: um conjunto Z_0 de ciclos de palavras.

$A := Z_0$;

Inter-reduza(A)[2]

repita

$B := A$;

$C :=$ conjunto dos ciclos de palavra obtidos de A com P_4;

$A := A \cup C$;

Inter-reduza(A)

até que $A = B$;

retorne A;

Fim.

Observe que as regras P.1 - P.3 do sistema de inferência P são utilizadas para simplificar o conjunto de ciclos de palavras e que a regra P.4 é uma regra para dedução de novos ciclos de palavras. Foi demonstrado em [**CO98**] que o procedimento acima termina se e somente se o *problema da palavra reduzido* do grupo que se deseja completar é um conjunto finito de ciclos de palavras. Neste caso o conjunto de ciclos de palavras gerado pode ser transformado num sistema de reescrita de palavras **especial** (o lado direito das regras consiste da palavra vazia) confluente obtendo-se assim uma solução simples para o problema da palavra do grupo que se reduz agora ao cálculo e comparação de formas normais.

Teorema 2.1. *O procedimento* **COMPLETA-CICLOS** *é um procedimento completador.*

[2]Aplique as regras P.1 - P.3 do sistema de inferência P enquanto possível.

i	Ciclo 1	Ciclo 2	Resultado
1	$[a^2]$	$[b^2]$	$[a^{-1}bab^{-1}]$
	$[b^2]$	$[b^{-1}aba]$	$[baba]$
	$[b^2]$	$[b^{-1}aba]$	$[b^{-1}ab^{-1}a]$
2	$[b^2]$	$[a^{-1}bab^{-1}]$	$[b^{-1}ab^{-1}a^{-1}]$

TABELA 1. Completação do grupo D_4.

DEMONSTRAÇÃO. Dado um conjunto Z_0 de ciclos de palavras como entrada o procedimento gera uma P-derivação $(Z_i)_{i\in I}$. Ao final do laço **repita** o conjunto A será sempre inter-reduzido logo Z^∞ também é inter-reduzido. Sejam u, x e $y \in \Sigma^+$ tais que $[ux]_\approx$ e $[uy]_\approx \in Z^\infty$ e x, y não começam e nem terminam com o mesmo símbolo. Como $[ux]_\approx$ e $[uy]_\approx$ são ciclos de palavras persistentes a partir de um determinado momento ambos serão elementos de A. Assim o ciclo $[xy^{-1}]_\approx$ é gerado por uma aplicação da regra de inferência P.4 e portanto a P-derivação é honesta.

Suponha que Z^∞ seja finito, logo a partir de um determinado momento Z^∞ passa a ser subconjunto de A. Deste momento em diante a forma inter-reduzida de A contém Z cada vez que o final do laço **repita** atingido. Suponha que Z^∞ seja um subconjunto próprio de A, ou seja existe um $[w]_\approx \in A$ tal que $[w]_\approx \notin Z^\infty$. Como A é inter-reduzido w é não-vazia e livremente reduzida. Pelo lema anterior existe uma subpalavra própria não-vazia w' de w tal que $[w']_\approx$ está em Z^∞ e conseqüentemente em A também. O que é uma contradição pois A é inter-reduzido. Portanto $A = Z^\infty$. Se o laço **repita** for realizado mais uma vez obtemos de maneira análoga que $A = Z^\infty$ e assim o procedimento pára. Logo o procedimento **COMPLETA-CICLOS** é um procedimento completador. □

Exemplo. Considere os grupos $D_{2n} = \langle a, b | a^n, b^2, b^{-1}aba\rangle$ (grupos diedrais). Neste caso o conjunto dado como entrada para o procedimento é $Z_0 = \{[a^n], [b^2], [b^{-1}aba]\}$. As tabelas 1, 2 e 3 apresentam as completações para os casos $n = 2, 3, 4$.

3. Transformando Ciclos de Palavras em Regras

Seja $>$ uma ordem de redução em Σ^*. A regra $(u \to v)$ está **associada** a um ciclo de palavra z se $[uv^{-1}]_\approx = z$ com $u > v$ e para todas as palavras u_1, u_2, u_3 satisfazendo

i	Ciclo 1	Ciclo 2	Resultado
1	$[a^3]$	$[b^{-1}aba]$	$[a^{-2}bab^{-1}]$
	$[a^3]$	$[b^{-1}aba]$	$[a^{-2}b^{-1}ab]$
	$[b^2]$	$[b^{-1}aba]$	$[baba]$
	$[b^2]$	$[b^{-1}aba]$	$[b^{-1}ab^{-1}a]$
	$[b^{-1}aba]$	$[b^{-1}aba]$	$[a^2ba^2b^{-1}]$
2	$[b^2]$	$[a^{-2}bab^{-1}]$	$[a^{-2}b^{-1}ab^{-1}]$
	$[b^2]$	$[a^{-2}bab^{-1}]$	$[a^{-2}bab]$
	$[baba]$	$[b^2]$	$[a^2ba^2b]$
	$[b^{-1}ab^{-1}a]$	$[b^2]$	$[a^2b^{-1}a^2b^{-1}]$

TABELA 2. Completação do grupo D_6.

i	Ciclo 1	Ciclo 2	Resultado
1	$[a^4]$	$[b^{-1}aba]$	$[a^{-3}bab^{-1}]$
	$[a^4]$	$[b^{-1}aba]$	$[a^{-3}b^{-1}ab]$
	$[b^2]$	$[b^{-1}aba]$	$[baba]$
	$[b^2]$	$[b^{-1}aba]$	$[b^{-1}ab^{-1}a]$
	$[b^{-1}aba]$	$[b^{-1}aba]$	$[a^2ba^2b^{-1}]$
2	$[a^4]$	$[a^{-3}bab^{-1}]$	$[a^{-3}ba^{-3}b^{-1}]$
	$[b^2]$	$[a^{-3}bab^{-1}]$	$[a^{-3}b^{-1}ab^{-1}]$
	$[b^2]$	$[a^{-3}bab^{-1}]$	$[a^{-3}bab]$
	$[b^{-1}aba]$	$[a^{-3}bab^{-1}]$	$[a^2ba^{-2}b]$
	$[b^{-1}aba]$	$[a^{-3}bab^{-1}]$	$[bab^{-1}a^{-1}]$
	$[b^{-1}aba]$	$[a^{-3}bab^{-1}]$	$[b^{-1}aba^{-1}bab^{-1}a^{-1}]$
	$[b^{-1}aba]$	$[a^{-3}bab^{-1}]$	$[ba^2b^{-1}a^{-2}]$
	$[a^{-3}bab^{-1}]$	$[a^{-3}bab^{-1}]$	$[ba^{-2}ba^{-2}]$
	$[a^{-3}bab^{-1}]$	$[a^{-3}bab^{-1}]$	$[b^{-1}a^{-2}b{-}1a^{-2}]$
	$[baba]$	$[a^{-3}bab^{-1}]$	$[baba^{-1}bab^{-1}a^{-1}]$
	$[b^{-1}ab^{-1}a]$	$[a^{-3}bab^{-1}]$	$[b^{-1}ab^{-1}a^{-1}bab^{-1}a^{-1}]$
	$[baba]$	$[a^2ba^2b^{-1}]$	$[a^3ba^3b]$
	$[b^{-1}ab^{-1}a]$	$[a^2ba^2b^{-1}]$	$[a^3b^{-1}a^3b^{-1}]$
3	$[b^{-1}ab^{-1}a]$	$[a^{-3}b^{-1}ab^{-1}]$	$[(b^{-1}ab^{-1}a^{-1})^2]$

TABELA 3. Completação do grupo D_8.

$u = u_1u_2u_3$ e $u_1u_3 \neq \epsilon$ temos $u_2 \leq u_1^{-1}vu_3^{-1}$. Para um conjunto Z de ciclos de palavras seja $R(Z, >)$ o conjunto de regras associadas aos ciclos de palavras de Z segundo $>$.

LEMA 3.1. *Seja $\sim$ uma congruência sobre Σ^* tal que $aa^{-1} \sim \epsilon$ para todo $a \in \Sigma$, seja $>$ uma ordem de redução sobre Σ^* e R o sistema de reescrita de palavras canônico que gera $\sim$ e é compatível com $>$. Se $(u \to v)$ é uma regra de R então $(u \to v) = (aa^{-1} \to \epsilon)$ para algum $a \in \Sigma$ ou $(u \to v)$ é uma regra de $R(RWP_\sim, >)$.*

DEMONSTRAÇÃO. Seja $(u \to v)$ uma regra de R. Se $u = aa^{-1}$ então $v = \epsilon$ pois R é normalizado. Se $u \neq aa^{-1}$ para todo $a \in \Sigma$. Pela compatibilidade de R com $>$ temos que $u \neq v$ e $u \neq \epsilon$ e como $u \sim v$ temos que $uv^{-1} \sim \epsilon$. Como R é canônico nenhuma subpalavra própria de u e nenhuma subpalavra de v é redutível. Em particular nenhuma subpalavra própria de u e nenhuma subpalavra de v é congruente à palavra vazia. Assim u e v são livremente reduzidas. Se $v = \epsilon$ então $u \sim \epsilon$ e $[u]_{\approx} \in RWP_{\sim}$. Se $v \neq \epsilon$ então $u \not\sim \epsilon$ e $v \not\sim \epsilon$. Suponha que $[uv^{-1}]_{\approx} \notin RWP_{\sim}$. Se $uv^{-1} = aa^{-1}$ para algum $a \in \Sigma$ então $(u \to v) = (a \to a)$ o que contradiz o fato de R ser compatível com $>$. Portanto uv^{-1} contém uma subpalavra própria não-vazia congruente à palavra vazia. Como nem u e nem v contém tal palavra existem palavras $x, y_1, y_2, z \in \Sigma^*$ tais que $u = xy_2$, $v^{-1} = y_2z$, $y_1y_2 \sim \epsilon$ e y_1, y_2 e xz são não-vazias. Como $y_1y_2 \sim \epsilon$ então $zx \sim \epsilon$ também. Se $x = \epsilon$ então $z \neq \epsilon$ e $z \sim \epsilon$ o que contradiz o fato de que v não contém subpalavras congruentes à palavra vazia. Então $x \neq \epsilon$. Analogamente $z \neq \epsilon$. Como $y_1y_2 \sim \epsilon$ temos que $y_1 \sim y_1^{-1}$. Se $y_1 = y_2^{-1}$ então $u = xy_1$, $v = z^{-1}y_2^{-1}$ e $u \neq v$ implicam $x \neq z^{-1}$, e $u \sim v$ implica $x \sim z^{-1}$. Logo x ou z^{-1} é redutível. Porém isto contradiz a normalidade de R. Se $y_1 \neq y_2^{-1}$ então $y_1 \sim y_2^{-1}$ implica que y_1 ou y_2^{-1} é redutível o que também contradiz a normalidade de R. Assim $[uv^{-1}]_{\approx} \in RWP_{\sim}$.

Como R é compatível com $>$ temos que $u > v$. Sejam $u_1, u_2, u_3 \in \Sigma^*$ tais que $u = u_1u_2u_3$ e $u_1u_3 \neq \epsilon$. Então $u_2 \sim u_1^{-1}vu_3^{-1}$. Como cada subpalavra própria de u é irredutível temos que u_2 é irredutível. Logo u_2 é mínima para $>$ na sua classe de congruência. Assim $u_2 \leq u_1^{-1}vu_3^{-1}$. □

O seguinte teorema trata da terminação do procedimento de Otto-Cremanns e sua demonstração é baseada em propriedades de diagramas de grupo.

TEOREMA 3.1. *Para cada P-derivação honesta temos que $RWP_{\sim} = Z^{\infty}$. Em particular $\sim = \sim_{Z^{\infty}}$.*

Observe que o teorema tem como conseqüencia que o procedimento, que gera uma P-derivação honesta, pára se e somente se o problema da palavra reduzido do grupo finitamente apresentado dado como entrada é finito.

TEOREMA 3.2. *Se $\langle \Sigma, L \rangle$ é uma apresentação finita de grupo tal que o problema reduzido da palavra $RWP_{\sim_L}$ é um conjunto finito de palavras então o procedimento*

Algoritmo REDUZ-INVERSOS
Entrada: $u \in \Sigma^*$.
  enquanto $[u]_{\approx} = [aa^{-1}x]_{\approx}$ para algum $x \in \Sigma^*$ e $a \in \Sigma$.
    $u := x$;
  retorne u.
Fim.

ALGORITMO 4. Redução exaustiva de inversos cíclicos adjacentes.

Algoritmo REGRA-1
Entrada: um conjunto Z de ciclos de palavras.
  para cada $[u]_{\approx} \in Z$
    se $u = \epsilon$
      $Z := Z - \{[u]_{\approx}\}$;
  retorne Z.
Fim.

ALGORITMO 5. Eliminação de ciclos vazios.

COMPLETA-CICLOS *pára quando recebe como entrada* $Z_0 = \{[w]_{\approx} \mid w \in L\}$. *Neste caso um sistema de reescrita de palavras finito e convergente é obtido para cada ordem de redução sobre* Σ^*.

DEMONSTRAÇÃO. Se $RWP_{\sim}$ é finito então $R(RWP_{\sim}, >)$ também é finito para qualquer ordem de redução $>$. □

4. Estratégia Modificada

Na apresentação original do procedimento [**CO98**] a aplicação das regras, tanto na inter-redução como na geração de novos ciclos, é feita de forma não-deterministica. Nesta seção descrevemos uma estratégia de aplicação das regras de inferência que foi implementada computacionalmente (veja o código no apêndice A) e que se mostrou eficiente que a descrição genérica do algoritmo.

O algoritmo **REDUZ-INVERSOS** recebe uma palavra armazenada em Z eliminando desta todos os inversos ciclicamente consecutivos. É obtido então um representante de um ciclo de palavras livremente reduzido.

O algoritmo **REGRA-1** analisa todos os ciclos de um conjunto de ciclos de palavras Z eliminando aqueles que sejam vazios.

Algoritmo REGRA-2
Entrada: um conjunto Z de ciclos de palavras.
  para cada $[u]_{\approx} \in Z$
    $u :=$ **REDUZ-INVERSOS**(u);
    se $u = \epsilon$
      $Z := Z - \{[u]_{\approx}\}$;
  retorne Z.
Fim.

ALGORITMO 6. Redução de inversos adjacentes.

No algoritmo **REGRA-2** é aplicado, para cada elemento do vetor Z, o algoritmo **REDUZ-INVERSOS**. Se a palavra resultante é vazia então a mesma é eliminada imediatamente. Desta maneira, como será mostrado em um lema a seguir, elimina-se a necessidade de aplicar novamente o algoritmo **REGRA-1** neste caso.

REGRA-3 realiza a aplicação exaustiva em pares da regra de inferência P.3, ou seja, são considerados todos os pares distintos de ciclos de palavras e para cada par a regra é aplicada de maneira exaustiva. Logo, após uma aplicação do algoritmo a um conjunto de ciclos Z, o conjunto não será necessariamente reduzido para a regra de inferência P.3. A exemplo de **REGRA-2** o algoritmo trata de simplificar os ciclos obtidos para as regras de inferência anteriores também. Assim quando um ciclo de palavras é simplificado com a regra de inferência P.3 trata-se logo de aplicar o algoritmo **REDUZ-INVERSOS** para tornar o ciclo livremente reduzido e de testar em seguida se o ciclo obtido é vazio e eliminá-lo em caso afirmativo.

O algoritmo **REGRA-4** faz a aplicação exaustiva da regra de inferência P.4, após a sua execução com um conjunto de ciclos de palavras Z são gerados todos os ciclos de palavras possíveis de se obter com P.4. Como já foi comentado anteriormente a regra de inferência P.4 é a única que gera ciclos uma vez que as outras são regras de simplificação. No procedimento **COMPLETA-CICLOS** original, apresentado em [**CO98**], a estratégia usada na completação consiste de gerar de uma vez só, a cada iteração do laço **repita**, todas as regras possíveis com a regra de inferência P.4. Em seguida é realizado um passo de inter-redução sobre o conjunto obtido (ciclos antigos acrescidos dos ciclos gerados com P.4). No entanto o conjunto de regras geradas com P.4 pode ser muito grande o que pode tornar o passo seguinte de inter-redução caro.

Algoritmo REGRA-3
Entrada: um conjunto Z de ciclos de palavras.
  para cada par $[u]_\approx, [v]_\approx \in Z$ com $[u]_\approx \neq [v]_\approx$
  ainda não considerado por **REGRA-3**
    seja u_M o maior dentre u e v e
    seja u_m o menor dentre u e v, em comprimento;
    enquanto $[u_M]_\approx = [u_m x]_\approx$ para algum $x \in \Sigma^+$
      $u_M :=$ **REDUZ-INVERSOS**(x);
      sejam u_M e u_m os representantes de maior e
      de menor comprimento respectivamente;
      se $u_M = \epsilon$
        $Z := Z - \{[u_M]_\approx\}$;
  retorne Z.
Fim.

ALGORITMO 7. Eliminação de subciclos.

A estratégia utilizada aqui, e sugerida nas conclusões de [**CO98**] como um possível aprimoramento consiste de verificar se os ciclos já presentes em Z aparecem como subciclos de cada regra que for gerada com P.4 e em caso afirmativo eliminar este subciclo. Em seguida pode-se reduzir cada ciclo para inversos adjacentes consecutivos. Realizando este processo de maneira exaustiva cada ciclo gerado não terá como subciclo ciclos já presentes anteriormente em Z e será livremente reduzido. Finalmente se o ciclo obtido não for vazio ele poderá ser incluido em Z. Assim o número de ciclos de palavras novos incluidos em Z a cada iteração é potencialmente bem menor.

Agora descrevemos o procedimento completador **COMPLETA-CICLOS-2** que utiliza as estratégias mencionadas anteriormente. O procedimento recebe como entrada um conjunto de ciclos de palavras Z_0 que não é necessariamente inter-reduzido. Inicialmente são aplicados os algoritmos **REGRA-1** e **REGRA-2** e é realizada a aplicação consecutiva de **REGRA-3** até que não seja mais possível e que resultará em um conjunto de ciclos inter-reduzidos. Em seguida, no laço **repita** é aplicado o algoritmo **REGRA-4** que gera um conjunto de ciclos de palavras livremente reduzidos sem ocorrência de subciclos do conjunto original. No entanto é possível que ciclos gerados com **REGRA-4** sejam subciclos de ciclos do conjunto original, assim é realizado em seguida a aplicação exaustiva do algoritmo **REGRA-3**. Observe que como os ciclos gerados com **REGRA-4** são livremente reduzidos e não-vazios e o conjunto original era

Algoritmo REGRA-4
Entrada: um conjunto Z de ciclos de palavras.
  $Z' := Z$;
  para cada par $[u]_\approx, [v]_\approx \in Z$ ainda não considerado por **REGRA-4**
    para cada $w, x, y \in \Sigma^+$ tal que $[u]_\approx = [wx]_\approx$ e $[v]_\approx = [wy]_\approx$
    com $x = x_1...x_k, y = y_1...y_l$ e $x_1 \neq y_1, x_k \neq y_l$
    $temp := xy^{-1}$;
    $Z'' := Z'$;
    $Z' := Z' \cup \{[xy^{-1}]_\approx\}$;
    se $[temp]_\approx \notin Z''$ e $temp \neq \epsilon$;
      para cada $s \in Z' \backslash \{[xy^{-1}]_\approx\}$
        se $|temp| \geq |s|$
          enquanto $[temp]_\approx = [xs]_\approx$ para algum $x \in \Sigma^*$
            $temp :=$**REDUZ-INVERSOS**(x);
    se $temp \neq \epsilon$
      $Z' := (Z' \backslash \{[xy^{-1}]_\approx\}) \cup \{[temp]_\approx\}$;
    senão
      $Z' := Z' \backslash \{[xy^{-1}]_\approx\}$;
  retorne Z'.
Fim.

ALGORITMO 8. Geração de novas regras.

inter-reduzido o conjunto atual é inter-reduzido para as regras de inferência P.1 e P.2. Assim é suficiente aplicar novamente o algoritmo **REGRA-3** de maneira exaustiva. É possível que durante a aplicação de **REGRA-3** surjam regras que sejam redutíveis para as regras de inferência P.1 e P.2 mas, como foi visto anteriormente, o próprio algoritmo **REGRA-3** trata destes casos. Assim, neste contexto, a aplicação exaustiva de **REGRA-3** é equivalente ao procedimento de inter-redução. O procedimento pára quando nenhuma regra foi alterada de uma iteração anterior para a atual.

LEMA 4.1. *Seja Z um conjunto de ciclos de palavras inter-reduzido para a regra de inferência P.1. O resultado de se aplicar o procedimento* **REGRA-2** *a Z é inter-reduzido para as regras de inferência P.1 e P.2.*

DEMONSTRAÇÃO. Se Z é inter-reduzido para a regra de inferência P.1 então não existem ciclos vazios em Z. Seja $Z[i]$ um ciclo presente em Z, se a regra de inferência P.2 não é aplicável então $Z[i]$ é inter-reduzido para a regra de inferência P.2 não sendo alterado por **REGRA-2** e continuando portanto sendo não-vazio. Se a regra de inferência P.2 é aplicável então todos os inversos adjacentes são removidos exaustivamente e se a

Procedimento COMPLETA-CICLOS-2.
Entrada: um conjunto Z de ciclos de palavras.
  $Z :=$ **REGRA-1**(Z);
  $Z :=$ **REGRA-2**(Z);
  aplique ($Z :=$ **REGRA-3**(Z)) até que não seja mais possível;
  $Z' := Z$;
  repita
    $Z := Z'$;
    $Z' :=$ **REGRA-4**(Z');
    aplique ($Z' :=$ **REGRA-3**(Z')) até que não seja mais possível;
  até que $Z' = Z$;
  retorne Z;
Fim.

PROCEDIMENTO 9. Procedimento completador.

palavra resultante, que é reduzida para a regra de inferência P.2, for vazia a mesma é removida por **REGRA-2**. Assim após a aplicação nenhum ciclo de Z é vazio sendo este então inter-reduzido para as regras de inferência P.1 e P.2. □

LEMA 4.2. *Seja Z um conjunto finito de ciclos de palavras inter-reduzido para as regras de inferência P.1 e P.2. A aplicação exaustiva do algoritmo* **REGRA-3** *a Z é um processo finito que resulta num conjunto de ciclos de palavras inter-reduzido (para as regras de inferência P.1, P.2 e P.3).*

DEMONSTRAÇÃO. Seja Z um conjunto inter-reduzido para as regras de inferência P.1 e P.2. Se a regra de inferência $P.3$ não é aplicável a Z então **REGRA-3** pára retornando Z que claramente é inter-reduzido. Caso contrário, se a regra de inferência $P.3$ é aplicável a Z então existe ao menos um par de ciclos de palavras da forma $[u]_{\approx}$ e $[uv]_{\approx} \in Z$ com $u, v \in \Sigma^*$. O ciclo de palavras $[uv]_{\approx}$ será substituido por $[v]_{\approx}$ e $[v]$ será então simplificado para as regras inferência P.1 e P.2, portanto o par de ciclos de palavras continuará sendo inter-reduzido para as regras inferência P.1 e P.2. Seja $||Z||$ o somatório dos comprimentos de todos os representantes de ciclos de palavras distintos de Z. Seja Z_i o conjunto de ciclos de palavras Z e Z_{i+1} o conjunto obtido por uma aplicação de **REGRA-3** a Z. Logo $||Z_{i+1}|| \leq ||Z_i|| - |u| < ||Z_i||$ e assim a cada aplicação bem sucedida de **REGRA-3** a Z, $||Z||$ decresce. Como $||Z||$ é limitado inferiormente por 0 o processo de aplicações sucessivas de **REGRA-3** a Z deve parar,

e quando isto ocorrer a regra de inferência P.3 não é aplicável. Neste caso a aplicação exaustiva de **REGRA-3** também pára finitamente retornando um conjunto Z inter-reduzido. □

LEMA 4.3. *Seja Z um conjunto de ciclos de palavras inter-reduzido. A aplicação de* **REGRA-4** *a Z é um processo finito que gera um conjunto de ciclos de palavras Z' irredutível para as regras de inferência P.1 e P.2 e todos os ciclos possíveis de se gerar com P.4 foram gerados e reduzidos com as regras P.1 e P.2. Além disto se existem ciclos de palavras $[uv]_{\approx}, [u]_{\approx} \in Z'$ tais que ao se aplicar a regra de inferência P.3 pode-se simplificar $[uv]_{\approx}$ como $[v]_{\approx}$, então $[uv]_{\approx} \in Z$ e $[v]_{\approx} \in Z' \backslash Z$ ou $[uv]_{\approx}$ e $[v]_{\approx} \in Z' \backslash Z$.*

DEMONSTRAÇÃO. **REGRA-4** pára num número finito de passos pois:

1. tanto o laço **enquanto** quanto a chamada a **REDUZ-INVERSOS** interna ao **enquanto** são realizados um número finito de vezes pois em ambos o comprimento da palavra diminui (quando os mesmos são aplicados) sendo o mesmo limitado inferiormente por 0.
2. no segundo laço **para** dado um par de ciclos de palavras existe um número também finito de palavras w da forma descrita, isto é, subpalavras comuns aos dois ciclos e de comprimento máximo.
3. o terceiro laço **para** também é realizado um número finito de vezes pois o número de ciclos de palavras armazenados em Z é sempre finito.
4. no primeiro laço **para** existe um número finito de pares de ciclos de palavras possíveis em Z.

Observe que, em $(Z' := Z' \cup \{[temp]_{\approx}\};)$, os ciclos incluidos no conjunto de ciclos de palavras são não-vazios e foram previamente inter-reduzidos com **REDUZ-INVERSOS**. Logo todas os ciclos de palavras gerados são inter-reduzidos para as regras de inferência P.1 e P.2. Como as regras dadas como entrada tamém eram irredutíveis para estas regras o conjunto Z' dado como saída pelo algoritmo é inter-reduzido para as regras de inferência P.1 e P.2.

Claramente no primeiro e segundo laços **para** são considerados todos os ciclos possíveis de serem gerados com a regra de inferência P.4. Estes ciclos são armazenados

temporariamente em *temp* e em seguida busca-se simplificar *temp* para ocorrência de subciclos já presentes em Z e inversos ciclicamente adjacentes. Ou seja, o ciclo é reduzido com aplicações das regras de inferência P.1 e P.2 até que não seja mais possível.

Agora suponha que $[u]_\approx$ e $[uv]_\approx$ pertençam a Z'. Não é possível termos ambos em Z pois Z é inter-reduzido para a regra de inferência P.3. Não é possível que tenhamos $[u]_\approx \in Z$ e $[uv]_\approx$ em $Z'\backslash Z$ pois neste caso $[uv]_\approx$ obriatoriamente seria gerado por **REGRA-4** e armazenado em *temp* que por sua vez seria simplificado no laço **enquanto** por $[v]_\approx$. Restam então apenas duas possibilidades, ou seja ambos $[u]_\approx \in Z$ e $[uv]_\approx$ em $Z'\backslash Z$ ou $[u]_\approx \in Z'\backslash Z$ e $[uv]_\approx \in Z$. □

LEMA 4.4. *O procedimento* **COMPLETA-CICLOS-2** *é um procedimento completador.*

DEMONSTRAÇÃO. O procedimento **COMPLETA-CICLOS-2** gera uma P-derivação $(Z_i)_{i\in I}$ para um conjunto de ciclos de palavras de entrada Z_0. Observe que antes do laço **repita** são aplicados os algoritmos **REGRA-1** e **REGRA-2** fazendo com que o conjunto de ciclos de palavras seja irredutível para as regras de inferência P.1 e P.2. Logo a aplicação exaustiva de **REGRA-3** a este conjunto gera um conjunto de ciclos de palavras inter-reduzido. Assim o laço **repita** é iniciado com um conjunto inter-reduzido de ciclos de palavras. Foi visto que, neste caso, a aplicação do algoritmo **REGRA-4** gera um conjunto de ciclos de palavras irredutível para as regras de inferência P.1 e P.2. Novamente então a aplicação exaustiva de **REGRA-3** a este conjunto gera um conjunto de ciclos de palavras inter-reduzido. Desta maneira ao final do laço **repita** o conjunto de ciclos de palavras é sempre inter-reduzido. Logo o conjunto de ciclos de palavras persistente Z^∞ é inter-reduzido. Suponha agora que $[ux]_\approx$ e $[uy]_\approx$ estejam em Z^∞ para $u, x, y \in \Sigma^*$ e não comecem e nem terminem com o mesmo símbolo. Assim a partir de um determinado momento ambos sempre pertencerão ao conjunto de ciclos de palavras. Neste momento a aplicação de **REGRA-4** gerará o ciclo de palavra $[xy^{-1}]_\approx$, logo $[xy^{-1}]_\approx \in Z$ e a P-derivação é honesta.

Suponha agora que Z^∞ é um conjunto finito. Logo a partir de um determinado momento Z^∞ estará sempre contido no conjunto de ciclos de palavras sendo manipulado

Grupo	Estratégia original	Estratégia modificada
D_6	8 seg.	< 1 seg.
D_8	35 min.	10 seg.
D_{10}	–	2 min.
D_{12}	–	34 min.

TABELA 10. Tempos de execução.

por **COMPLETA-CICLOS-2**. Portanto sempre que o conjunto Z' for inter-reduzido Z^∞ estará contido no conjunto resultante. Suponha por contradição que Z^∞ esteja contido propriamente no conjunto resultante. Portanto existe um ciclo de palavras $[w]_\approx$ tal que $[w]_\approx \notin Z^\infty$ e $[w]_\approx \in Z'$. Foi visto que existe uma subpalavra w' de w tal que $[w']_\approx \in Z^\infty$, o que contradiz o fato de Z' ser inter-reduzido. Assim $Z' = Z^\infty$ e usando o mesmo argumento isto ocorrerá na iteração seguinte de **COMPLETA-CICLOS-2** fazendo com que o procedimento pare. Portanto **COMPLETA-CICLOS-2** é um procedimento completador. □

O procedimento foi implementado em linguagem C em uma plataforma AMD K6 de 266MHz com 64MB de RAM sob o sistema operacional Linux. Inicialmente o procedimento foi implementado utilizando a mesma estratégia descrita em **COMPLETA-CICLOS**, que não simplifica imediatamente os ciclos de palavras gerados com a regra de inferência P.4. Em seguida foi utilizada a estratégia modificada descrita no procedimento **COMPLETA-CICLOS-2**. A execução de alguns exemplos de grupos diedrais ($D_{2n} = \langle a, b | a^n = e, b^2 = e, abab^{-1} = e \rangle$) com a estratégia modificada mostrou-se bem mais rápida do que com a estratégia original como pode ser verificado na tabela 10.

CAPÍTULO 5

Conclusões

As aplicações de técnicas de reescrita à computação com grupos já são bem conhecidas e vêm se tornando cada vez mais relevantes na área de computação simbólica. Logo o estudo de procedimentos completadores é de extrema importância uma vez que grande parte destas aplicações requer que os sistemas de rescrita usados para apresentar os grupos sejam convergentes. Por ser um procedimento recente [**CO98**] que se utiliza de ciclos de palavras, uma estrutura potencialmente econômica em espaço, e pelo fato de ser especializado para apresentações de grupos julgamos relevante o estudo realizado sobre o procedimento de Otto-Cremanns.

Foi proposta uma estratégia de aplicação das regras de inferência P.1 - P.4 que na prática mostrou-se bem mais eficiente que a original. Esta estratégia é utilizada pelo procedimento **COMPLETA-CICLOS-2** que é uma das contribuições deste trabalho. Foi demonstrado que este procedimento, a exemplo do procedimento de Otto-Cremanns, é também um procedimento completador e portanto as propriedades de terminação vistas são também aplicáveis. Assim o procedimento **COMPLETA-CICLOS-2** pára sempre que o problema da palavra reduzido de um grupo finitamente apresentado é finito. A implementação do procedimento foi testada com algumas apresentações de grupos diedrais e de grupos de Fibonacci. No caso dos grupos diedrais a performance da implementação resultou comparável à performance de implementações do procedimento de Knuth-Bendix.

Como trabalho futuro existe uma série de melhoramentos que poderiam ainda ser realizados no procedimento. No algoritmo **REGRA-4** a simplificação dos ciclos que estão sendo gerados é unidirecional, ou seja, são usadas somente as regras já presentes no conjunto de ciclos de palavras para simplificar as regras novas. Uma possível otimização adicional seria utilizar a regra recém gerada para simplificar as regras já

presentes no conjunto de ciclos de palavras. No caso em que **REGRA-4** consegue simplificar um ciclo gerado para a regra de inferência P.3 o ciclo resultante será redutível para a regra de inferência P.2 somente nos símbolos que eram adjacentes à subpalavra eliminada. Reduzir o teste de aplicabilidade da regra de inferência P.2 a estes símbolos seria outra otimização que poderia ser realizada na implementação.

Outro ponto que pode ser aprimorado na implementação das regras de inferência é a realização de casamento de ciclos de palavras de maneira mais eficiente. Atualmente o casamento é feito utilizando-se *força bruta*, isto é o ciclo menor é deslocado sobre o ciclo maior posição a posição. Algoritmos eficientes para casamento de palavras, como o algoritmo de Knuth-Morris-Pratt, poderiam ser adaptados para ciclos de palavras tornando o processo mais rápido. Como na maioria dos grupos testados o comprimento dos ciclos de palavras era pequeno esta ineficiência no casamento de ciclos não foi sentida.

Uma das contribuições importantes desta dissertação é a implementação do procedimento de Otto-Cremanns que segundo o próprio Otto [**Ott99b**] é a primeira de que se tem conhecimento. A implementação foi bastante importante para que se pudesse experimentar com diferentes estratégias de completação. Em particular verificou-se que a aplicação desordenada ou não-determinística das regras de inferência, como utilizada na descrição original do procedimento, torna o processo de completação extremamente ineficiente pois muitos ciclos triviais que poderiam ser simplificadas até o ciclo vazio na aplicação da regra de inferência P.4 são incluidas desnecessariamente no conjunto de ciclos de palavras. Em seguida o passo de Inter-redução receberá um conjunto de ciclos de palavras potencialmente muito maior do que o que seria gerado com estratégia proposta nesta dissertação.

Ainda com relação à implementação será interessante incluir futuramente a conversão do conjunto de ciclos de palavras resultante do procedimento de Otto-Cremanns num sistema de reescrita de palavras. A conversão poderia inclusive ser realizada para que o sistema de reescrita já ficasse no formato padrão de entrada de programas como o KBMAG que implementam o procedimento de Knuth-Bendix para palavras.

É esperado que, pela utilização dos ciclos de palavras no armazenamento dos ciclos de palavras, o procedimento de Otto-Cremanns seja mais eficiente em espaço em alguns

casos do que o procedimento de Knuth-Bendix. Seria fundamental então determinar grupos para os quais a completação com o procedimento de Knuth-Bendix é ineficiente em espaço para que estes mesmos grupos fossem testados com a completação baseada em ciclos de palavras.

Referências

[Ani71] A. V. Anisimov. Group languages. *Kibernetika*, 4:18–24, 1971.

[AR98] M. Ayala-Rincón. *Fundamentos de Programação Lógica e Funcional - O Princípio de Resolução e a Teoria da Reescrita*. Departamento de Matemática, Universidade de Brasília, 1998.

[BN98] F. Baader and T. Nipkow. *Term-Rewriting and All That*. Cambridge University Press, 1998.

[BO93] R. Book and F. Otto. *String-Rewriting Systems*. Springer-Verlag, 1993.

[CO98] R. Cremanns and F. Otto. A completion procedure for finitely presented groups that is based on word cycles. Technical report, Universität Gesamthochschule Kassel, 1998.

[Deh11] M. Dehn. Über unendliche diskontinuierliche Gruppen. *Math. Ann.*, 71:44–116, 1911.

[DM79] N. Dershowitz and Z. Manna. Proving termination with multiset orderings. *Communications of the ACM*, 22(8):465–476, 1979.

[ECH+92] D. B. A. Epstein, J. W. Cannon, D. F. Holt, S. V. F. Levy, M. S. Patterson, and W. P. Thurston. *Word Processing in Groups*. Jones and Bartlett, 1992.

[EHR91] D. B. A. Epstein, D. F. Holt, and S. E. Rees. The Use of Knuth-Bendix Methods to Solve the Word Problem in Automatic Groups. *Journal of Symbolic Computation*, 12:397–414, 1991.

[HU79] J. Hopcroft and J. Ullman. *An Introduction to Automata, Languages and Computation*. Addison-Wesley, 1979.

[Joh97] D. Johnson. *Presentations of Groups*, volume 15 of *London Mathematical Society Student Texts*. Cambridge University Press, 2nd edition, 1997.

[KB70] D. Knuth and P. Bendix. Simple word problems in universal algebra. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.

[Ott99a] F. Otto. On the connections between rewriting and formal language theory. In *10th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science, Trento, Italy, 1999. Springer-Verlag.

[Ott99b] F. Otto. On the connections between rewriting and formal language theory. In *10th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science. Springer-Verlag, 1999.

[Ott00] F. Otto. Comunicação pessoal, 2000.

[Sim94] C. Sims. *Computation with Finitely Presented Groups*, volume 48 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 1994.

[ST99] I. Stewart and R. Thomas. Formal languages and the word problem for groups. In *Groups St. Andrews 1997 in Bath, II*, volume 2 of *London Mathematical Society Lecture Note Series*, pages 689–700. Cambridge University Press, 1999.

[Sud97] T. Sudkamp. *Languages and Machines: An Introduction to the Theory of Computer Science*. Addison-Wesley, 1997.

APÊNDICE A

Implementação

A implementação do procedimento de Cremmans-Otto foi realizada em linguagem C e foi utilizado o compilador GCC (GNU C Compiler) 2.95.2 em plataforma Linux 2.2.17 (Debian) nas arquiteturas AMD K6 266MHz e Intel Pentium 133MHz. Foram realizados testes bem sucedidos nas seguintes plataformas também: Digital Alpha/Digital UNIX, Sun SPARC/Linux e Intel Pentium/FreeBSD. A implementação segue o padrão ANSI e deve ser de fácil compilação em qualquer plataforma e arquitetura com um compilador apropriado e pode ser obtida via *web* no endereço:

`http://www.cic.unb.br/~gadelha/ccp.tar.gz`.

Para armazenar os ciclos de palavras foi utilizado um vetor de estruturas do tipo `ciclo_de_palavras` que consistem de um apontador para um elemento de tipo `char` e de um elemento do tipo `int` que indica o estado do ciclo de palavras armazenado. Todo ciclo de palavras pode ter três estados: 0, 1 ou 2. Inicialmente todo ciclo de palavras tem estado 1 e isto indica que deve ser considerado para inter-redução. Em seguida, após a inter-redução todo ciclo muda para o estado 0 para que não se realiza mais a inter-redução para pares com este estado. No laço principal os ciclos gerados com **REGRA-4** mudam para o estado 1 para que possam em seguida ser inter-reduzidos. Se dentro deste laço e após a inter-redução um ciclo continua com estado 0 isto significa que o mesmo é irredutível para as regras de inferência P.1 a P.3 e que já foram gerados todos ciclos possíveis da sobreposição com regras que têm estado 0 com a regra de inferência P.4, este fato é indicado com o estado 2 que este ciclo passa a assumir. Pares de ciclos com estado 2 são desconsiderados por **REGRA-4**. Como se pode ver o controle de estados é utilizado para que algumas aplicações desnecessárias das regras de inferência não sejam realizadas.

Os demais detalhes da implementação podem ser verificados nos comentários do código-fonte que é apresentado a seguir:

APÊNDICE B

Execução de Exemplos

Neste apêndice exibimos a execução de exemplos de apresentações de grupos num Intel Pentium de 133MHz com 48MB de memória rodando o sistema operacional Linux 2.2.17 (Debian). Considere o caso dos grupos diedrais $D_{2n} = \langle a, b | a^n = e, b^2 = e, abab^{-1} = e \rangle$.

Exemplo 1.
$D_4 = \langle a, b | a^2 = e, b^2 = e, abab^{-1} = e \rangle$.

Exemplo 2.
$D_6 = \langle a, b | a^3 = e, b^2 = e, abab^{-1} = e \rangle$.

Exemplo 3.
$D_8 = \langle a, b | a^4 = e, b^2 = e, abab^{-1} = e \rangle$.

Exemplo 4.
$D_{10} = \langle a, b | a^5 = e, b^2 = e, abab^{-1} = e \rangle$.

Exemplo 5.
$D_{12} = \langle a, b | a^6 = e, b^2 = e, abab^{-1} = e \rangle$.

Exemplo 6.
$D_{14} = \langle a, b | a^7 = e, b^2 = e, abab^{-1} = e \rangle$.

Exemplo 7[1].
$A_4 = \langle a, b | a^3 = e, b^3 = e, abab = e \rangle$.

[1]Exemplo executado num
de 200MHz com o sistema operacional

Exemplo 8.
$F(2,3) = \langle a, b, c | ab = c, bc = a, ca = b \rangle$.