Universidade de Lisboa
Faculdade de Ciências
Departamento de Matemática

Linguagens Reconhecíveis nas Variantes de Palavras, Árvores e Florestas: Caracterizações e um Estudo Comparativo

Eduardo José Caetano Branco

Dissertação
Mestrado em Matemática

2014

Universidade de Lisboa
Faculdade de Ciências
Departamento de Matemática

Linguagens Reconhecíveis nas Variantes de Palavras, Árvores e Florestas: Caracterizações e um Estudo Comparativo

Eduardo José Caetano Branco

Dissertação
Mestrado em Matemática

Orientador: Prof. Doutor Mário João de Jesus Branco

2014

Agradecimentos

Quero agradecer ao Professor Mário Branco pela sua orientação, apoio, enorme paciência e disponibilidade ao longo de todo deste trabalho. Sem ele esta tese nunca teria chegado a sê-lo.

Quero igualmente agradecer aos meus pais pelo apoio que sempre me deram e que, mais uma vez, foi o que me fez continuar.

Eduardo Branco

Resumo

O principal objetivo deste trabalho foi uniformizar a teoria do reconhecimento linguagens de árvores de aridade, de árvores sem aridade e de florestas, com ênfase especial para as duas últimas, com a teoria do reconhecimento de linguagens de palavras.

De facto, apesar de tanto as linguagens de árvores (de e sem aridade) e as linguagens de florestas serem generalizações de linguagens de palavras, o seu desenvolvimento não conhecia o mesmo tratamento sistematizado de que está munida a clássica teoria de linguagens de palavras. Nas páginas que se seguem, procurou-se formalizar e organizar alguns aspetos relativos ao reconhecimento de linguagens de árvores de aridade, de árvores sem aridade e de florestas numa teoria comparável com aquela que faz das linguagens de palavras ferramenta incontornável em inúmeras aplicações à Ciência da Computação. De notar que é também como resposta a novos problemas postos pela Ciência da Computação que outros objetos matemáticos vêm a ser estudados com maior profundidade. Esse é precisamente o caso das árvores e florestas, cujas linguagens dão resposta, por exemplo, a problemas relacionados com a representação/armazenamento e troca de dados informáticos.

O texto encontra-se organizado da seguinte forma. No primeiro capítulo, são sumariamente apresentados alguns conceitos e resultados relacionados com teoria de semigrupos; no segundo é recordada a teoria do reconhecimento de linguagens de palavras; nos dois últimos, é então descrita uma teoria de reconhecimento de linguagens de árvores de aridade e de linguagens de florestas, respetivamente, as quais, sendo inspiradas na teoria clássica, constituem uma possível uniformização.

Palavras chave

Álgebra-floresta sintática, árvore de aridade, árvore sem aridade, autómato, floresta, monoide sintático, Σ-álgebra sintática, palavra, reconhecimento de uma linguagem.

Abstract

The main aim of this study was to present a uniform theory of recognizable language of ranked trees, unranked trees, and forests as that of words.

Indeed, and despite the fact that both languages of (ranked and unranked) trees and languages of forests are generalizations of languages of words, its development had not yet achieved the systematic treatment that has long been known to languages of words. In what follows, we develop a formalization and organization of some aspects concerning language recognition for languages of ranked trees, languages of unranked trees, and languages of forests in a manner that resembles the classical — and powerful — theory of languages of words. And if the classical theory has been the major tool in various problems in computing science, new challenges have recently demanded other approaches. For instance, problems related to data representation and exchange have been solved using trees and forest, prompting renewed interest in these subjects.

The text is organized as follows. In the first chapter, we briefly present some concepts and results concerning semigroups; in the second, we recall the aspects of the classical theory of languages of words which will allow us to, in the third and fourth chapters, describe a theory of language recognition for ranked trees and for forests, respectively, that compares with the classical theory.

Keywords

Automaton, forest, language recognition, ranked tree, syntatic monoid, syntatic forest algebra, syntatic Σ-algebra, unranked tree, word.

Conteúdo

Introdução

A noção de reconhecimento de linguagens, como conjuntos de sequências de símbolos, surgiu na década de 60 do século passado e está intrinsecamente relacionada com autómatos finitos. Inicialmente foi desenvolvida sobre linguagens de palavras (finitas) mas depressa foi estendida a outros modelos. O estudo das linguagens de palavras reconhecíveis e de outras estruturas associadas tem permitido construir uma teoria rica assente em importantes teoremas que permitiram avanços significativos, dos quais destacamos resultados de Kleene, Myhill, Nerode, Elgot, Büchi e Schützenberger.

Atualmente as linguagens de palavras reconhecíveis fazem parte de qualquer curso de Ciência da Computação e são usualmente consideradas pedras basilares em muitos dos seus campos. Podem ser tratadas de diversas maneiras, entre as quais: reconhecimento por autómato, reconhecimento por monoide finito, expressões racionais e definibilidade lógica. Estes conceitos são equivalentes em termos de expressividade e levam a muitos algoritmos fundamentais no campo da compilação, processamento de texto, engenharia de *software*, etc.. Por si só estas aplicações conferem à classe das linguagens de palavras reconhecíveis uma importância inegável. As diversas maneiras de tratar as linguagens de palavras reconhecíveis que mencionámos atrás têm gerado muito conhecimento nas áreas da Combinatória, da Álgebra e da Lógica.

Objetos matemáticos que estendem as palavras têm sido, especialmente desde o virar do século, estudados com o intuito de os aplicar ao desenvolvimento de novas ideias da Computação, por exemplo, à representação/armazenamento de dados informáticos e à troca dos mesmos. Neste campo destacamos o XML (do inglês *eXtensible Markup Language*) pela sua importância. Documentos XML são produzidos comummente em bases de dados, em processamento de documentos e em aplicações *Web*. Árvores são os objetos em que o XML se baseia.

Desde há algumas décadas, árvores (finitas) e linguagens de árvores têm sido intensamente estudadas tanto como extensão dos trabalhos desenvolvidos para palavras como para dar resposta a várias questões da Computação. Embora o início do estudo das árvores de aridade (árvores etiquetadas num alfabeto finito munido de uma função de aridade) e das árvores sem aridade (árvores etiquetadas simplesmente num alfabeto finito) tenha sido contemporâneo, o desenvolvimento da teoria sobre árvores de aridade foi bem mais substancial e contínuo até à presente data. Com a criação do *XML* no final dos anos 90 no âmbito da *Web*, as árvores sem aridade voltaram a ser alvo de um interesse especial uma vez que são o modelo formal dos documentos *XML*.

Muito do trabalho que existe para linguagens de árvores de aridade e para linguagens de árvores sem aridade está feito numa perspetiva de extensão de conceitos e resultados existentes para linguagens de palavras. O mesmo sucede para florestas (finitas e sem aridade). No entanto, e contrariamente com o que se passa com o estudo das palavras, as diferentes abordagens desenvolvidas resultaram na ambiguidade de algumas definições. O objetivo principal do nosso trabalho é apresentar um estudo das linguagens de árvores sem aridade e das linguagens de florestas à semelhança daquele que se conhece para linguagens de palavras. Grande parte do que aqui se apresenta consiste na uniformização do formalismo de vários conceitos relacionados com o reconhecimento de linguagens de palavras, de árvores de aridade, de árvores sem aridade e de florestas, com o objetivo de colmatar as referidas discrepâncias.

Esta dissertação está organizada da seguinte forma. No Capítulo 1 recordamos algumas noções e resultados básicos sobre semigrupos e monoides e estabelecemos a notação considerada; cada um dos restantes capítulos é dedicado a apresentar a teoria do reconhecimento de linguagens de palavras, árvores de aridade e florestas, respetivamente. Tendo sido um dos objetivos primordiais deste trabalho apresentar um estudo comparativo destes assuntos, os capítulos 2, 3 e 4 encontram-se organizados por forma a evidenciar o facto de, ao contrário do que sucede em grande parte da literatura conhecida, ser possível apresentar a teoria relativa a cada um deles de forma uniforme.

Assim, o Capítulo 2 é dedicado a apresentar, relativamente a alfabetos finitos, definições e resultados acerca de palavras finitas e de linguagens das mesmas. Nomeadamente, recordam-se as noções de reconhecimento de linguagens de palavras finitas através de autómatos, de morfismos de semigrupos e monoides, de expressões racionais e de fórmulas da Lógica, bem como a equivalência destas noções. Além disso, é dada a caracterização de

três classes de linguagens de palavras finitas: as livres de estrela, as testáveis localmente e as testáveis por pedaços.

O Capítulo 3 sistematiza então, como já referimos, a teoria relativa às linguagens de árvores de aridade. Não sendo a teoria das linguagens destas árvores o principal tema deste trabalho, a apresentação dos rudimentos desta teoria prende-se não só com razões de completude como serve também o objetivo de permitir estabelecer pontos de contacto com as demais linguagens consideradas. Começamos por apresentar a noção de árvore sobre um alfabeto sem aridade, certas relações definidas no domínio de uma árvore sobre um alfabeto sem aridade, árvore sobre um alfabeto de aridade, que são um caso particular das primeiras, e de contexto sobre um alfabeto de aridade. Dado que as árvores sem aridade são, por sua vez, uma caso particular das florestas, o seu estudo é adiado para o último capítulo, sendo o Capítulo 3 dedicado a descrever o estudo das linguagens das árvores de aridade à luz do clássico estudo das linguagens de palavras finitas. Em particular, é recordado o conceito de Σ-álgebra para, a partir dela, poder ser desenvolvido o reconhecimento algébrico da linguagem de árvores de aridade, e o de autómato de árvores de aridade, através do qual é possível considerar o reconhecimento destas linguagens por meio de autómatos. Por fim, é enunciado o resultado que estabelece a equivalência entre os dois tipos de reconhecimento.

Por último, o Capítulo 4 diz respeito ao estudo das linguagens de florestas. Para tal, são apresentadas as noções de floresta e contexto, generalizadas as relações consideradas a propósito das árvores sem aridade e definidas certas operações. A estrutura relativamente à qual é feito o reconhecimento algébrico de uma linguagem de florestas é a de álgebra-floresta, que consiste num caso particular das chamadas "two-sorted algebras" e para a qual é desenvolvida a teoria análoga à dos monoides para as linguagens de palavras finitas. O papel desempenhado por autómatos finitos no âmbito das linguagens de palavras é agora desempenhado pelos autómatos de florestas. Mais uma vez, tem-se a desejada equivalência entre os dois tipos de reconhecimento, cuja demonstração se inclui. É também definido o conceito de árvores-reconhecimento, o qual permite reconhecer uma linguagem de árvores sem aridade com recurso a uma álgebra-floresta. Relativamente a álgebras-floresta, são ainda apresentadas outras possíveis definições, não necessariamente equivalentes, e cujo interesse reside no facto de possibilitarem estudar o reconhecimento de linguagens de florestas sob perspetivas diferentes. O trabalho é concluído com a apresentação de algumas aplicações, designadamente à caracterização de certas classes de linguagens de árvores sem aridade e de

florestas, tanto à custa de equações como à custa de fórmulas da Lógica. Para as últimas, é dado um exemplo, de forma sucinta e sem uma preocupação de muito rigor, igualmente ilustrativo de como as várias relações definidas para árvores sem aridade e florestas podem ser usadas para este efeito.

Este trabalho inclui também um apêndice sobre Lógica. Embora utilizemos conceitos da Lógica, como já mencionámos, esta não tem um papel central nesta dissertação. Por outro lado, um leitor com formação matemática não teve necessariamente um curso de Lógica. Incluímos assim em apêndice uma compilação dos vários conceitos básicos da Lógica de que necessitamos. No contexto deste trabalho, é substancialmente extensa, mas não pretende ser uma introdução a um curso de Lógica.

Capítulo 1

Preliminares

Este capítulo consiste numa breve apresentação de algumas definições e resultados, quase na totalidade sobre semigrupos, que convém ter presente nos capítulos seguintes. Estes conteúdos podem ser encontrados em variada bibliografia, entre a qual [19] que é a utilizada no que se segue.

Dados conjuntos X e Y, uma relação $\theta : X \longrightarrow Y$ e um elemento x em X, denotamos por $\theta(x)$ o conjunto $\{y \in Y \mid (x, y) \in \theta\}$. Em particular, se θ é uma função e $\theta(x) \neq \emptyset$, denotamos por $\theta(x)$ o elemento de Y que é a imagem de x por θ.

A composição da relação $\theta : X \longrightarrow Y$ com a relação $\rho : Y \longrightarrow Z$ é a relação $\rho \circ \theta : X \longrightarrow Z$ definida por

$$\rho \circ \theta = \{(x, z) \mid \exists y \in Y : \ (x, y) \in \theta \text{ e } (y, z) \in \rho\}.$$

Um *semigrupo* é um par $(S, \cdot)$ formado por um conjunto S (a que chamamos *universo*) e uma operação binária associativa $\cdot$ em S.

Um *monoide* é um terno $(M, \cdot, 1)$, onde $(M, \cdot)$ é um semigrupo e 1 é identidade da operação $\cdot$.

Vamos representar um semigrupo ou um monoide apenas pela letra que representa o seu universo.

Dado um semigrupo S, podemos construir um monoide S^1 da seguinte maneira: se S é um monoide, então S^1 é S; caso contrário, então $S^1{=}S \cup \{1\}$, onde 1 é um elemento que não pertence a S e a operação em S^1 estende a operação em S de modo que $s \cdot 1{=}1 \cdot s{=}s$, para todo o $s \in S^1$.

Dados um semigrupo S e dois subconjuntos X e Y de S, denotamos por XY o conjunto $XY = \{x \cdot y \mid x \in X,\ y \in Y\}$.

No que diz respeito a semigrupos, as noções de subsemigrupo, produto direto, quociente e morfismo são definidas como é habitual para outras estruturas algébricas. O mesmo acontece para monoides. Lembramos apenas que para um morfismo de monoides é requerida a condição extra da imagem da identidade ser a identidade.

Dados dois monoides M e N, dizemos que M *divide* N se M é imagem homomorfa de algum submonoide de N.

Seja T um semigrupo (respetivamente, monoide) e seja $X \subseteq T$.

Definimos *subsemigrupo* (respetivamente, *submonoide*) *gerado por* X, que denotamos por $\langle X \rangle$ (respetivamente, $\langle X \rangle^1$), como sendo

$$\bigcap \{Y \mid Y \textit{ subsemigrupo de } T \textit{ contendo } X\}$$

$$\big(\text{respetivamente, } \bigcap \{Y \mid Y \textit{ submonoide de } T \textit{ contendo } X\}\big).$$

Seja S um semigrupo. Um elemento e de S tal que $e^2 = e$ é chamado *idempotente*. Representamos por $E(S)$ o *conjunto de todos os elementos idempotentes de* S. Um semigrupo é *idempotente* se todos os seus elementos forem idempotentes. Se e é um idempotente de S, então eSe é um subsemigrupo de S que é um monoide com e como identidade e é chamado *submonoide local de* S.

Num semigrupo finito S todo o elemento tem uma potência idempotente, isto é, para todo o elemento $s \in S$, existe $n_s \in \mathbb{N}$ tal que s^{n_s} é idempotente. Logo existe um $n \in \mathbb{N}$ mínimo, tal que s^n é idempotente para qualquer $s \in S$, que denotamos por ω. Em particular, s^ω é a única potência de s que é idempotente.

As *relações de Green num semigrupo* S são cinco relações de equivalência em S, notadas por $\mathcal{R}$, $\mathcal{L}$, $\mathcal{J}$, $\mathcal{D}$ e $\mathcal{H}$ e que podem ser definidas da seguinte maneira:

$a \,\mathcal{R}\, b \Leftrightarrow aS^1 = bS^1$;
$a \,\mathcal{L}\, b \Leftrightarrow S^1a = S^1b$;
$a \,\mathcal{J}\, b \Leftrightarrow S^1aS^1 = S^1bS^1$, para todos os $a, b \in S$;
$\mathcal{H} = \mathcal{R} \cap \mathcal{L}$ e
$\mathcal{D}$ é a menor relação de equivalência em S que contém $\mathcal{R} \cup \mathcal{L}$.

Pode provar-se que $\mathcal{R} \circ \mathcal{L} = \mathcal{L} \circ \mathcal{R}$, pelo que, $\mathcal{D} = \mathcal{R} \circ \mathcal{L}$. Temos $\mathcal{R}, \mathcal{L} \subseteq \mathcal{J}$ e, como tal, temos também $\mathcal{D} \subseteq \mathcal{J}$.

O seguinte resultado é essencial no estudo de semigrupos finitos.

Proposição 1.0.1.
Num semigrupo finito, $\mathcal{D} = \mathcal{J}$.

Um semigrupo S diz-se *aperiódico* se existir um número natural n tal que $s^n{=}s^{n+1}$, para todo o elemento $s \in S$. Adiante veremos que os semigrupos finitos aperiódicos são importantes no estudo das linguagens. O resultado seguinte caracteriza os semigrupos finitos aperiódicos.

Proposição 1.0.2.
Seja S um semigrupo finito. As seguintes afirmações são equivalentes:

1. *S é aperiódico,*
2. *S é $\mathcal{H}$-trivial,*
3. *Qualquer subsemigrupo de S que é grupo, é trivial.*

Neste trabalho são também utilizados vários conceitos da Lógica. Em apêndice encontra-se uma compilação dos conceitos de que necessitamos.

Capítulo 2

Linguagens de Palavras

Neste capítulo vamos enunciar, relativamente a alfabetos finitos, algumas definições e resultados, que se presumem já adquiridos, acerca de palavras finitas e de linguagens das mesmas. Por isso, apenas serão feitas as demonstrações que pensamos serem mais pertinentes ou ilustrativas. Tal como já referimos, os tópicos que vamos seguir nos capítulos seguintes tiveram origem na matéria que aqui vamos abordar. Assim, aproveitaremos a maior antiguidade e consequente maior desenvolvimento do estudo da matéria presente neste capítulo para criarmos linhas orientadoras para as restantes. Com a estrutura e os conteúdos aqui presentes vamos tentar estabelecer uma organização e objetivos a perseguir nas restantes secções.

Este trabalho teve como principal inspiração um trabalho de Mikolaj Bojanczyk e Igor Walukiewicz [12], o qual é a base do capítulo 4. Em ordem a facilitar a comparação deste capítulo com o que é feito no capítulo 4, optámos por introduzir alternativas para alguns dos conceitos usuais.

A matéria aqui apresentada pode ser encontrada em variadíssima bibliografia (ver, por exemplo, [15, 19, 20, 21, 22, 16, 27, 25, 24, 23, 8, 30]).

2.1 Definições Base

Esta secção está dividida em três partes: a primeira trata essencialmente das definições de alfabeto, palavra sobre um alfabeto e de algumas relações; a segunda aborda as definições de semigrupo e monoide livres e a aplicação dos mesmas a modelos de palavras sobre um alfabeto; na última debruçamo-nos

sobre autómatos de palavras.

2.1.1 Palavras

Neste trabalho estamos apenas interessados em casos finitos e, como tal, vamos incluir essa finitude nas próprias definições de alfabeto e de palavra.

Um *alfabeto* é simplesmente um conjunto finito não vazio e *letras* ou *símbolos* são os elementos de um alfabeto.

Ao longo deste trabalho, alfabetos e letras serão habitualmente representados por letras do começo do alfabeto latino: letras maiúsculas para alfabetos $(A, B, C, \dots)$; letras minúsculas para letras $(a, b, c, \dots)$.

Seja A um alfabeto. Chamamos *palavra sobre o alfabeto* A a uma sequência ordenada finita $a_1a_2 \dots a_n$ de letras de A e à sequência vazia. Dizemos que tal palavra $a_1a_2 \dots a_n$ tem *comprimento* n e que a palavra vazia tem *comprimento* 0. Denotamos o comprimento de uma palavra u por $|u|$. A palavra vazia representa-se, sempre que não houver ambiguidade, por 1. Dada uma palavra u e uma letra a, denotamos por $|u|_a$ o número de ocorrências de a em u. Habitualmente representamos, neste capítulo, palavras sobre um alfabeto por u, v e w (possivelmente com índices).

Mais formalmente, uma palavra u sobre um alfabeto A é uma função parcial $u : \mathbb{N} \longrightarrow A$ tal que o seu domínio, que denotamos por *dom(u)*, é $\emptyset$ ou $\{1, \dots, n\}$ para certo $n \in \mathbb{N}$. Com o objetivo de comparação com definições de outras secções, notamos que isto significa que:

1. o seu domínio é finito;

2. com a ordem natural em $\mathbb{N}$, $dom(u)$ é um ideal de ordem.

Um elemento $i \in dom(u)$ é chamado *i-ésima posição na palavra* u. A função parcial atribui a uma posição $i \in \mathbb{N}$ a sua *etiqueta* $u(i) \in A$ e, como tal, dizemos que *a i-ésima posição da palavra u tem etiqueta* $u(i)$. Por esta razão também chamamos *palavras A-etiquetadas* ou *A-palavras* às palavras sobre o alfabeto A.

Com o intuito de as utilizarmos no decorrer deste capítulo, definimos agora algumas relações (estas relações são igualmente utilizadas em vários dos trabalhos que temos como base, entre os quais [27] e [21]):

- Dada uma A-palavra u, definimos uma relação binária em $dom(u)$, a que chamamos *relação sucessor* e que denotamos por S^u, como sendo o conjunto $\{(i, i+1) \in \mathbb{N} \times \mathbb{N} \mid i, i+1 \in dom(u)\}$.
- Dada uma A-palavra u, definimos uma relação binária em $dom(u)$, a que chamamos *relação ordem* e que denotamos por $\leq^u$, que é a ordem natural de $\mathbb{N}$ restringida a $dom(u)$, ou seja, como sendo o conjunto $\{(i, j) \in \mathbb{N} \times \mathbb{N} \mid i, j \in dom(u) \text{ e } i \leq j\}$.
- Dada uma A-palavra u e uma letra $a \in A$, definimos uma relação unária em $dom(u)$, que denotamos por lab_a^u, como sendo o conjunto das posições em u que têm como etiqueta a letra a, ou seja, como sendo o conjunto $\{i \in \mathbb{N} \mid i \in dom(u) \text{ e } u(i) = a\}$.

As relações S^u e $\leq^u$ são chamadas *numéricas* e as relações lab_a^u são chamadas *predicados letra*.

2.1.2 Semigrupo e Monoide Livres

Seja A um alfabeto. Denotamos o *conjunto de todas as A-palavras* por A^* e o *conjunto de todas as A-palavras não vazias* por A^+. Definimos uma multiplicação em A^*, a que chamamos *concatenação*, da seguinte maneira:

$$(a_1 a_2 \dots a_m) \cdot (b_1 b_2 \dots b_n) = a_1 a_2 \dots a_m b_1 b_2 \dots b_n,$$
$$1 \cdot u = u \cdot 1 = u,$$

para todos os $m, n \in \mathbb{N}, a_1, a_2, \dots, a_m, b_1, b_2, \dots, b_n \in A$ e $u \in A^*$. Com esta operação toda a palavra não vazia é concatenação de letras de A e, portanto, A^* (respetivamente, A^+) é gerado como monoide (respetivamente, semigrupo) por A (ver Secção 1). É chamado *monoide livre* (respetivamente, *semigrupo livre*) *gerado por A*. Esta terminologia vem do seguinte resultado:

Proposição 2.1.1.

Seja A um alfabeto e seja A^* (respetivamente, A^+) o monoide (respetivamente, semigrupo) gerado por A.

Para toda a função $f : A \longrightarrow S$, onde S é um monoide (respetivamente, semigrupo), existe um único morfismo de monoides $\alpha : A^* \longrightarrow S$ (respetivamente, morfismo de semigrupos $\alpha : A^+ \longrightarrow S$) que estende f.

Aproveitamos algumas das definições anteriores para agora definirmos um novo conceito para semigrupos. Sejam $u, v \in A^+$. Dizemos que *um semigrupo S satisfaz a equação $u = v$* se e só se $\varphi(u) = \varphi(v)$ para qualquer morfismo $\varphi : A^+ \longrightarrow S$.

Exemplo 2.1.2.
Seja S um semigrupo. As equações

$$ab = ba,\ a^2 = a \ e\ a^{n+1} = a^n$$

dizem, respetivamente, que o semigrupo S é comutativo, idempotente e aperiódico.

Se $u = u_1u_2u_3$, onde $u_1, u_2, u_3 \in A^*$, dizemos que u_1 é um *prefixo de u*, que u_3 é um *sufixo de u* e que u_2 é um *fator de u*.

Seja $u = a_1a_2 \dots a_n \in A^*$ onde $n \in \mathbb{N}$ e $a_1, a_2, \dots, a_n \in A$. Dizemos que u é uma *subpalavra de uma palavra $v \in A^*$* se $v = v_0a_1v_1a_2v_2 \dots a_nv_n$ para alguns $v_0, v_1, v_2, \dots, v_n \in A^*$.

Definimos *linguagem de A-palavras* como um subconjunto de A^*.

Consideremos duas linguagens K e L de A-palavras. Vamos agora definir algumas operações que nos vão ser úteis futuramente:

- *união*,

$$K \cup L = \{u \mid u \in K \text{ ou } u \in L\};$$

- *concatenação* ou *produto*,

$$K \cdot L = KL = \{u_1 \cdot u_2 \mid u_1 \in K \text{ e} u_2 \in L\};$$

- *estrela de Kleene* ou, simplesmente, *estrela*,

$$L^* = \{u_1 \cdots u_n \mid n \in \mathbb{N} \text{ e} u_1, \dots, u_n \in L\} \cup \{1\}.$$

É conveniente definir também a operação:

$$L^+ = LL^* = \{u_1 \cdots u_n \mid n \in \mathbb{N} \text{ e} u_1, \ldots, u_n \in L\}.$$

Notamos que L^+ é o subsemigrupo de A^* gerado por L, enquanto que L^* é o submonoide de A^* gerado por L.

É ainda importante a definição de *quociente esquerdo de L por K* (respetivamente, *quociente direito de L por K*), isto é, a linguagem

$$K^{-1}L = \{v \in A^* \mid \exists u \in K : \ uv \in L\}$$
$$\text{(respetivamente, } LK^{-1} = \{v \in A^* \mid \exists u \in K : \ vu \in L\}).$$

Os elementos de $K^{-1}L$ são os elementos de A^* que podem ser obtidos a partir de palavras de L através da eliminação dos prefixos que estão em K.

2.1.3 Autómato de Palavras

Um *autómato de palavras* é um quíntuplo $\mathcal{A} = (Q, A, E, I, F)$ onde:

- Q é um conjunto, cujos elementos são chamados *estados*;
- A é um alfabeto;
- $E \subseteq Q \times A \times Q$; os elementos de E são chamados *transições*;
- $I \subseteq Q$; os elementos de I são chamados *estados iniciais*;
- $F \subseteq Q$; os elementos de F são chamados *estados finais*.

Dizemos que um *autómato de palavras é finito* se o seu conjunto de estados for finito.

Representamos a transição (p, a, q) por uma seta cuja origem é p, a extremidade é q e a etiqueta é a, ou seja, $p \xrightarrow{a} q$. Dizemos que duas transições (p, a, q) e (p', a', q') são *consecutivas* se $q = p'$. Chamamos *caminho em* $\mathcal{A}$ a uma sequência finita de transições consecutivas:

$$(q_0, a_1, q_1), (q_1, a_2, q_2), \ldots, (q_{n-1}, a_n, q_n);$$

que também denotamos por:

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n.$$

O estado q_0 é a *origem do caminho*, o estado q_n é o *fim do caminho* e a palavra $u = a_1 a_2 \dots a_n$ é a *etiqueta do caminho*.

Dizemos que um autómato de palavras $\mathcal{A} = (Q, A, E, I, F)$ é *determinista* se verificar as seguintes condições:

1. $\mathcal{A}$ tem um único estado inicial (isto é, $I = \{q_0\}$ para certo $q_0 \in Q$);
2. para todo o $p \in Q$ e para todo o $a \in A$ existe no máximo um $q \in Q$ tal que $(p, a, q) \in E$.

Dizemos que um autómato de palavras $\mathcal{A} = (Q, A, E, I, F)$ é *completo* se para todo o $p \in Q$ e para todo o $a \in A$ existe pelo menos um $q \in Q$ tal que $(p, a, q) \in E$.

Podemos obter uma definição alternativa de *autómato de palavras determinista e completo* substituindo o conjunto E das transições por uma função, a que chamamos *função transição* e que denotamos por δ, que a cada elemento $a \in A$ faz corresponder uma função (total) $\delta_a : Q \longrightarrow Q$ onde, para qualquer $p \in Q$, $\delta_a(p)$ é o único estado q tal que $(p, a, q) \in E$. Como consequência deste autómato ser determinista vem que $I = \{q_0\}$ para certo $q_0 \in Q$. Denotamos este autómato determinista e completo por $\mathcal{A} = (Q, A, \delta, q_0, F)$.

2.2 Reconhecimento de Linguagens de Palavras

Ao longo desta secção vamos considerar um alfabeto A e uma linguagem de A-palavras L ($\subseteq A^*$). Relembramos que, mediante as definições de alfabeto e de palavra dadas anteriormente, estamos a considerar linguagens de palavras **finitas** sobre um alfabeto **finito** mas diremos, com frequência, simplesmente "linguagem de palavras" ou, ainda, "linguagem".

Existem várias alternativas de noção de linguagem reconhecível, embora se prove que todas elas são equivalentes entre si no caso de linguagens de palavras. Cada uma destas ideias de reconhecimento é interessante em si mesma

pela perspetiva que utiliza. De seguida apresentamos estas diferentes noções e enunciamos sem demonstração alguns resultados acerca das mesmas.

Os principais resultados desta secção podem ser encontrados em variada bibliografia, entre a qual os trabalhos de Eilenberg [15], Pin [19], Straubing [25], Sipser [24] e Sakarovitch [23].

Reconhecimento por Autómato de Palavras

Uma palavra $u = a_1a_2 \dots a_n$, com $n \in \mathbb{N}$ e $a_1, a_2, \dots, a_n \in A$, diz-se *reconhecida pelo autómato de palavras* $\mathcal{A} = (Q, A, E, I, F)$ se existir um caminho nesse autómato da forma:

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \cdots \xrightarrow{a_n} q_n,$$

onde $q_0 \in I$ e$q_n \in F$. A um caminho nestas condições chamamos *caminho bem sucedido em* $\mathcal{A}$. Dizemos que *a palavra vazia é reconhecida pelo autómato de palavras* $\mathcal{A} = (Q, A, E, I, F)$ se $I \cap F \neq \emptyset$.

Vejamos agora de que maneira algumas das noções anteriores podem ser adaptadas para um autómato de palavras determinista e completo $\mathcal{A} = (Q, A, \delta, q_0, F)$. Um tal autómato de palavras $\mathcal{A}$ aplica uma A-palavra u num elemento de Q, que representamos por $u^{\mathcal{A}}$, que é definido por recorrência da seguinte maneira:

- se u for a palavra vazia, então $u^{\mathcal{A}} = q_0$;
- se $u = va$, onde $v \in A^*$ e $a \in A$, então $u^{\mathcal{A}} = \delta_a(v^{\mathcal{A}})$.

Assim, se $u = a_1a_2 \dots a_n$, com $n \in \mathbb{N}$ e $a_1, a_2, \dots, a_n \in A$, temos

$$u^{\mathcal{A}} \;=\; \delta_{a_n} \;\circ\; \dots \;\delta_{a_2} \;\circ\; \delta_{a_1}.$$

Dizemos que *uma A-palavra u é reconhecida por um autómato de palavras determinista e completo* $\mathcal{A} = (Q, A, \delta, q_0, F)$ se e só se $u^{\mathcal{A}} \in F$.

Definimos *linguagem reconhecida por um autómato de palavras* $\mathcal{A}$, que denotamos por $L(\mathcal{A})$, como sendo $\{w \in A^* \mid w \text{ é reconhecida por } \mathcal{A}\}$.

Reconhecimento Algébrico

Podemos considerar linguagens de palavras reconhecidas por um monoide explorando a estrutura de monoide de A^*.

Se M é um monoide e $\varphi : A^* \longrightarrow M$ é um morfismo de monoides, dizemos que *a linguagem $L \subseteq A^*$ é reconhecida por φ* se $L = \varphi^{-1}(P)$ para algum $P \subseteq M$; equivalentemente, se $L = \varphi^{-1}(\varphi(L))$. Dizemos que *a linguagem $L \subseteq A^*$ é reconhecida por um monoide M* se existir um morfismo de A^* em M que reconhece L.

Autómato de Palavras Minimal e Monoide Sintático

Vamos aqui apresentar duas construções associadas a uma linguagem de palavras L: o *autómato de palavras minimal de L* e o *monoide sintático de L*. Estes objetos permitem estabelecer os seguintes resultados:

- O autómato de palavras minimal de L é determinista e completo, reconhece L e é minimal (num certo sentido, que não vamos definir) entre os autómatos de palavras deterministas e completos que reconhecem L; além disso, é efetivamente calculável a partir de qualquer autómato de palavras finito que reconheça L.

- O monoide sintático de L reconhece L e é minimal (num certo sentido, que vamos definir) entre os monoides que reconhecem L; além disso, é efetivamente calculável a partir de qualquer morfismo que reconheça L.

Seja $\mathcal{A} = (Q, A, E, I, F)$ um autómato de palavras. Podemos associar a cada palavra u de A^* uma relação binária em Q da seguinte maneira: se $u = a_1 a_2 \dots a_n \in A^*$, com $n \in \mathbb{N}$ e $a_1, a_2, \dots, a_n \in A$, definimos

$$\overline{u} = \{(p, q) \in Q \times Q \mid \text{existe um caminho } p \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q\};$$

definimos também $\overline{1} = \{(p, p) \mid p \in Q\}$. É claro que, para quaisquer $u, v \in A^*$, temos $\overline{uv} = \overline{u} \bullet \overline{v}$ no monoide $\mathcal{B}(Q)$ das relações binárias em Q com composição $\bullet$ definida por $R \bullet S = S \circ R$, para todas as relações binárias R e S em Q.

Seja $M(\mathcal{A}) = \{\overline{w} \mid w \in A^*\}$, que é um submonoide de $\mathcal{B}(Q)$, chamado *monoide das transições de* $\mathcal{A}$. É óbvio que o monoide $M(\mathcal{A})$ é gerado pelo conjunto $\{\overline{a} \mid a \in A\}$.

Dado um autómato de palavras finito $\mathcal{A} = (Q, A, E, I, F)$, podemos construir de maneira efetiva um autómato de palavras finito $\mathcal{B} = (Q, A, E, i, F)$ determinista e completo e que satisfaz $L(\mathcal{A}) = L(\mathcal{B})$. Assim, em $\mathcal{B}$ a relação $\overline{u}$ é uma função (total), para cada $u \in A^*$. É claro que, dada uma palavra $u \in A^*$, o autómato de palavras $\mathcal{B}_u = (Q, A, E, \overline{u}(i), F)$ reconhece a linguagem $u^{-1}L(\mathcal{A})$. Logo, se $L \subseteq A^*$ for a linguagem de palavras reconhecida por um autómato de palavras finito, então o conjunto $\{u^{-1}L \mid u \in A^*\}$ é finito.

Seja $L \subseteq A^*$. Definimos um autómato de palavras $\mathcal{A}(L)$ sobre o alfabeto A da seguinte maneira:

- estados: $u^{-1}L$, com $u \in A^*$;
- transições: $(u^{-1}L, a, (ua)^{-1}L)$, com $u \in A^*$ e $a \in A$;
- estados iniciais: L $(= 1^{-1}L)$;
- estados finais: $u^{-1}L$, com $u \in L$.

Se $u, v \in A^*$ forem tais que $u^{-1}L = v^{-1}L$ e $(u{\cdot}1 =)\ u \in L$, então $1 \in u^{-1}L$, pelo que também $1 \in v^{-1}L$, ou seja, $v \in L$.

Seja $u = a_1 a_2 \dots a_n \in A^*$, onde $n \in \mathbb{N}$ e $a_1, a_2, \dots, a_n \in A$. No autómato de palavras $\mathcal{A}(L)$ temos o caminho

$$L \xrightarrow{a_1} a_1^{-1}L \xrightarrow{a_2} (a_1a_2)^{-1}L \xrightarrow{a_3} \cdots \xrightarrow{a_n} u^{-1}L,$$

que é o único caminho em $\mathcal{A}(L)$ que começa em L e que tem etiqueta u. Logo,

$$u \in L(\mathcal{A}(L)) \Leftrightarrow u^{-1}L \text{ é um estado final} \Leftrightarrow u \in L.$$

Logo, $\mathcal{A}(L)$ reconhece L e é determinista e completo. Portanto, L é reconhecida por um autómato de palavras finito se e só se $\mathcal{A}(L)$ é finito.

O autómato de palavras $\mathcal{A}(L)$ é chamado *autómato de palavras minimal de* L. Esta terminologia deve-se ao facto de $\mathcal{A}(L)$ ser, num certo sentido, um autómato de palavras minimal dentro dos autómatos de palavras que reconhecem L e que têm uma certa propriedade. O autómato de palavras $\mathcal{A}(L)$ pode ser efetivamente construído a partir de qualquer autómato de palavras finito que reconheça L.

Consideremos agora o monoide das transições do autómato de palavras minimal de L, $M(\mathcal{A}(L))$, e o morfismo

$$\varphi : A^* \longrightarrow M(\mathcal{A}(L))$$

$$u \mapsto \overline{u}$$

Pelo que vimos atrás, este morfismo reconhece L. Definimos no monoide A^* uma relação de equivalência que denotamos por $\sim_L$:

$u \sim_L v$ se e só se, para todos os $x, y \in A^*$,

$$xuy \in L \Leftrightarrow xvy \in L.$$

Pode provar-se que a relação $\sim_L$ é uma congruência em A^*. Para quaisquer $u, v \in A^*$ temos

$$\begin{aligned} \overline{u} = \overline{v} &\Leftrightarrow \forall x \in A^*, \overline{u}(x^{-1}L) = \overline{v}(x^{-1}L) \\ &\Leftrightarrow \forall x, y \in A^*, (xuy \in L \Leftrightarrow xvy \in L) \\ &\Leftrightarrow u \sim_L v. \end{aligned}$$

Logo, $M(\mathcal{A}(L)) \simeq A^*/\sim_L$. O quociente $A^*/\sim_L$ é chamado *monoide sintático de* L e é denotado por $M(L)$. A congruência $\sim_L$ é chamada *congruência sintática de* L. O *morfismo sintático de* L é o morfismo canónico $A^* \longrightarrow M(L)$. Portanto, vimos que $M(L)$ reconhece L.

O seguinte resultado mostra que $M(L)$ é, num certo sentido, um monoide minimal entre os monoides que reconhecem L.

Proposição 2.2.1.

Se $L \subseteq A^*$ e M é um monoide finito, M reconhece L se e só se $M(L)$ divide M.

"Reconhecimento por Expressão Racional"

De uma maneira muito sucinta, vamos aqui falar em linguagens racionais e estabelecer algumas convenções a utilizar no decorrer deste trabalho. Para um estudo mais aprofundado desta matéria sugerimos a consulta de [15, 19, 20, 21, 22, 1].

Consideremos duas linguagens K e L de A-palavras (isto é, $K, L \subseteq A^*$) e as operações união, concatenação, estrela e quocientes (definidas em 2.1.2) que são usualmente chamadas de *operações racionais*. Notamos que, juntamente com as *operações booleanas* (união, interseção e complementação), as operações produto, estrela e quocientes são as operações básicas consideradas quando se estudam linguagens reconhecíveis. Notamos também que, por vezes, são utilizados os símbolos de parêntesis com o intuito evitar ambiguidades na leitura das expressões que se constroem utilizando letras que representam linguagens e os símbolos das operações anteriores. Algumas convenções foram desenvolvidas acerca da precedência das operações de maneira a, em alguns casos, conseguirmos evitar o uso de parêntesis. Uma convenção comum é:

1. L^* é considerada primeiro;
2. KL, $K^{-1}L$ e LK^{-1} são consideradas a seguir;
3. $K \cup L$ e $K \cap L$ são consideradas em último.

Quando não existir ambiguidade, utilizaremos o símbolo 0 para representar a linguagem vazia e cada palavra $u \in A^*$ para representar a linguagem $\{u\}$. O *conjunto das linguagens racionais* de A^* é o menor conjunto de linguagens de A^* que contém todas as linguagens finitas (ou, alternativamente, que tem a linguagem vazia, $\emptyset$, as linguagens $\{a\}$ para cada letra $a \in A$) e é fechado para as operações união, produto e estrela. Denotamos o conjunto das linguagens racionais de A^* por $Rat(A^*)$. Dizemos que uma linguagem $L \subseteq A^*$ é *racional* se $L \in Rat(A^*)$. Definimos *expressões racionais* como sendo expressões formais que se obtêm recursivamente através das seguintes regras:

1. 0 e 1 são expressões racionais,

2. para cada letra $a \in A$, a é uma expressão racional,

3. se e e e' são expressões racionais, então $(e+e')$, (ee') e e^* são expressões racionais.

Conseguimos evitar a utilização de alguns parêntesis recorrendo às precedências anteriormente consideradas. Pode provar-se que existe uma única função v do conjunto das expressões reacionais sobre A em $\mathcal{P}(A^*)$ que satisfaz:

$$\begin{aligned} v(0) &= 0 \\ v(1) &= 1 \\ v(a) &= a \text{ para cada letra } a \in A \\ v((e+e')) &= v(e) + v(e') \\ v((ee')) &= v(e)v(e') \\ v(e^*) &= (v(e))^* \end{aligned}$$

O *valor de uma expressão racional* e ou *linguagem representada por* e é a linguagem $v(e)$. Notamos que não se devem confundir as noções de expressão racional e de linguagem racional. Em particular, duas expressões racionais distintas podem representar a mesma linguagem.

Apresentamos sem demonstração o seguinte resultado.

Proposição 2.2.2.
L é uma linguagem racional de A^* se e só se L é o valor de pelo menos uma expressão racional.

"Reconhecimento por Definibilidade Lógica"

Mais uma vez de maneira muito sucinta, vamos agora falar em Lógica e estabelecer algumas convenções a utilizar ao longo deste trabalho. Aconselhamos a consulta do Apêndice A para a compreensão dos tópicos aqui presentes e, a leitores que pretendam um conhecimento mais aprofundado desta matéria, a leitura da bibliografia aí mencionada.

Tal como mencionado no Apêndice A, as fórmulas de Lógica de Primeira Ordem ou, como doravante as vamos chamar por questões de simplicidade, *fórmulas* FO (abreviatura do inglês *First Order*), utilizam apenas variáveis de Lógica de Primeira Ordem (intuitivamente, variáveis que são aplicadas em elementos do universo e que, por vezes, são chamadas *variáveis de elemento*), e as fórmulas de Lógica Monádica de Segunda Ordem ou, como doravante as vamos chamar por questões de simplicidade, *fórmulas* MSO (abreviatura do inglês *Monadic Second Order*), que são casos particulares de fórmulas de Lógica de Segunda Ordem onde pode apenas ser utilizado um tipo de variável de Lógica de Segunda Ordem: o das variáveis que representam relações de aridade 1 (intuitivamente, variáveis que são aplicadas em subconjuntos do universo e que, por vezes, são chamadas *variáveis de conjunto*).

Consideremos uma A-palavra u e a assinatura $\{S, \leq\} \cup \{lab_a \mid a \in A\}$ onde S e $\leq$ têm aridade 2 e, para todo o $a \in A$, lab_a tem aridade 1. O terno $(dom(u), \sigma, I)$ onde

- $\sigma = \{S, \leq\} \cup \{lab_a \mid a \in A\}$ e
- I é a função definida por

  $I(S) = S^u$, $I(\leq) = \leq^u$ e, para todo o $a \in A$, $I(lab_a) = lab_a^u$;

é uma estrutura e vamos denotá-la por $\underline{u}$.

No âmbito das palavras, vamos chamar *fórmula* $\mathtt{FO}(S, \leq)$ (respetivamente, *fórmula* $\mathtt{MSO}(S, \leq)$) ou, simplesmente, *fórmula* FO (respetivamente *fórmula* MSO) a uma fórmula de Lógica de Primeira Ordem (respetivamente, a uma fórmula de Lógica Monádica de Segunda Ordem) com a assinatura $\{S, \leq\} \cup \{lab_a \mid a \in A\}$.

Seja φ uma fórmula FO. A *linguagem de A-palavras definida por φ* é o conjunto

$$\{u \in A^* \mid \text{existe uma atribuição de variáveis } \mu \text{ tal que } (\underline{u}, \mu) \models \varphi\}$$

e denotamo-la por $L(\varphi)$. Dizemos que *a linguagem L é definida através de uma fórmula* FO se, para a assinatura $\{S, \leq\} \cup \{lab_a \mid a \in A\}$, existir uma fórmula FO, digamos φ, tal que $L = L(\varphi)$.

Chamamos *fórmula* $\mathtt{FO}(\leq)$ (respetivamente, *fórmula* $\mathtt{FO}(S)$) a uma fórmula de Lógica de Primeira Ordem com a assinatura $\{\leq\} \cup \{lab_a \mid a \in A\}$ (respetivamente, com a assinatura $\{S\} \cup \{lab_a \mid a \in A\}$). Analogamente, definimos

fórmulas $\mathtt{MSO}(\leq)$ e *fórmulas* $\mathtt{MSO}(S)$. Tal como atrás, com as adaptações naturais, estabelecemos o que significa uma *linguagem de* A^* *ser definida por uma fórmula* $\mathtt{MSO}$, $\mathtt{FO}(\leq)$, $\mathtt{FO}(S)$, $\mathtt{MSO}(\leq)$ e $\mathtt{MSO}(S)$.

Pode provar-se que a relação S pode ser expressa em $\mathtt{FO}(\leq)$ e que a relação $\leq$ pode ser expressa em $\mathtt{MSO}(S)$ (ver [21]).

Teorema de Equivalência de Reconhecimento de Linguagens de Palavras

Kleene, Nerode, Myhill e Büchi mostraram que as noções expostas atrás coincidem (ver, por exemplo, [15, 25, 16, 19, 20, 21, 22, 27, 24, 23, 30]):

Teorema 2.2.3.
Seja $L \subseteq A^*$. *Então as seguintes afirmações são equivalentes:*

i) L *é reconhecida por um autómato de palavras finito.*

ii) L *é reconhecida por um autómato de palavras determinista e completo finito.*

iii) L *é reconhecida por um monoide finito.*

iv) L *tem o monoide sintático finito.*

v) L *é uma linguagem racional.*

vi) L *é definida através de uma fórmula* $\mathtt{MSO}(S)$.

Mais ainda, existem algoritmos para passar de qualquer uma destas especificações para outra.

Com este resultado podemos, a partir de agora, falar apenas de *linguagem de palavras reconhecível* para nos referirmos a uma linguagem de palavras que satisfaça qualquer uma das condições do teorema anterior.

Do Teorema 2.2.3 conclui-se facilmente que o conjunto das linguagens de palavras racionais (ou reconhecíveis) é fechado para as operações booleanas.

A cada linguagem de palavras reconhecível L podemos associar dois objetos finitos: o autómato de palavras minimal de L e o monoide sintático de L. A relação entre os dois é estreita, pois o monoide sintático de L é exatamente o monoide das transições do seu autómato de palavras minimal.

2.3 Classificação de Linguagens de Palavras Reconhecíveis

A relação entre o autómato de palavras minimal e o monoide sintático dá uma boa maneira de classificação de linguagens de palavras reconhecíveis. Constatamos que é o monoide sintático, com a sua estrutura algébrica natural, que oferece um bom método de classificação. Nesta secção vamos dar exemplos de duas destas classificações.

Linguagens de Palavras Livres de Estrela

Um dos exemplos mais significativos de uma subclasse de linguagens de palavras reconhecíveis é o das *linguagens de palavras livres de estrela*. Estas são as linguagens de palavras que podem ser descritas por *expressões livres de estrela*, isto é, expressões racionais que utilizam apenas letras do alfabeto, as constantes $\emptyset$, 1 (a palavra vazia), as operações booleanas e a concatenação (mas não exigimos a operação estrela).

Vamos apresentar várias caracterizações de linguagens de palavras livres de estrela, sendo que uma delas requer a definição seguinte.

Um autómato de palavras determinista e completo diz-se *livre de contador* se, para qualquer palavra não vazia u, sempre que uma potência não trivial u^n, com $n \in \mathbb{N}$, for etiqueta de um caminho que tem início e fim num mesmo estado q, então o mesmo acontece com u.

O seguinte resultado combina resultados de Schützenberger, McNaughton, Pappert e Kamp (ver, por exemplo, [15, 19, 21, 27, 30]).

Teorema 2.3.1.
Seja $L \subseteq A^*$. Então as seguintes afirmações são equivalentes:

i) *L é livre de estrela.*

ii) *L é reconhecida por um autómato de palavras determinista e completo livre de contador.*

iii) *L é reconhecida por um monoide finito aperiódico.*

iv) *L tem o monoide sintático finito e aperiódico.*

v) *L é definida por uma fórmula $\mathtt{FO}(\leq)$.*

Mais ainda, existem algoritmos para passar de uma destas especificações para outra.

Podemos então concluir que a classe das linguagens de palavras livres de estrela, com a sua definição natural em termos de expressões racionais, tem também caracterizações em termos de todos os formalismos presentes no Teorema 2.2.3.

Linguagens Testáveis Localmente e Linguagens Testáveis por Pedaços

Uma linguagem $L \subseteq A^*$ diz-se *n-testável por pedaços*, onde $n \in \mathbb{N}$, se sempre que $u, v \in A^*$ tiverem as mesmas subpalavras de comprimento não superior a n e $u \in L$, então $v \in L$. Uma linguagem $L \subseteq A^*$ diz-se *testável por pedaços* se for n-testável por pedaços para algum $n \in \mathbb{N}$.

Uma linguagem $L \subseteq A^+$ diz-se *n-testável localmente*, onde $n \in \mathbb{N}$, se sempre que $u, v \in A^*$ tiverem os mesmos fatores de comprimento não superior a n, os mesmos prefixo e sufixo de comprimento $n-1$ e $u \in L$, então $v \in L$. Uma linguagem $L \subseteq A^+$ diz-se *testável localmente* se for *n-testável localmente* para algum $n \in \mathbb{N}$.

Resultados de Simon e McNaughton (ver, por exemplo, [15, 19, 27]), mostram que estas duas propriedades são caracterizadas pelas propriedades algébricas do monoide sintático.

Teorema 2.3.2.
Uma linguagem $L \subseteq A^*$ é testável por pedaços se e só se o seu monoide sintático $M(L)$ é finito e $\mathcal{J}$-trivial.

Um semigrupo diz-se *localmente idempotente e comutativo* se todos os seus submonoides locais são idempotentes e comutativos.

Teorema 2.3.3.
Uma linguagem $L \subseteq A^+$ é testável localmente se e só se o seu semigrupo sintático $M(L)$ é finito, localmente idempotente e comutativo.

Capítulo 3

Linguagens de Árvores de Aridade

Neste capítulo começamos por abordar árvores finitas sobre um alfabeto sem aridade e de seguida passamos para árvores finitas sobre um alfabeto de aridade, que são o foco deste capítulo. Os conceitos que vão ser introduzidos para árvores finitas sobre um alfabeto sem aridade vão ser fundamentais para o estudo das árvores finitas sobre um alfabeto de aridade. Existe uma extensa e rica teoria sobre árvores finitas sobre um alfabeto de aridade. No entanto, neste capítulo vamos limitar-nos às noções básicas sobre árvores finitas sobre um alfabeto de aridade para podermos fazer, mais à frente, algumas comparações com o que se passa com árvores finitas sobre um alfabeto sem aridade e com florestas, estas sim o objetivo principal deste trabalho. Para o estudo das árvores finitas sobre um alfabeto sem aridade vamos seguir variadíssima bibliografia, entre a qual [26, 27, 2, 4, 10, 11, 9, 6, 17]. No estudo das árvores finitas sobre um alfabeto de aridade vamos utilizar [7, 18, 5] e, especialmente, [14].

Tal como já referimos anteriormente, procurámos adotar neste capítulo uma estrutura semelhante à utilizada no Capítulo 2.

Em ordem a facilitar a comparação daquilo que aqui vai ser feito para árvores de aridade finitas com o que vai ser feito para florestas finitas no Capítulo 4, optámos por aqui introduzir alternativas para alguns conceitos (seguindo o princípio já utilizada no Capítulo 2).

3.1 Definições Base

Esta secção está dividida em três partes: na primeira parte tratamos essencialmente das definições de árvore sobre um alfabeto, de árvore sobre um alfabeto de aridade, de contexto sobre um alfabeto de aridade e de algumas relações; na segunda parte abordamos as definições de Σ-álgebra e Σ-álgebra livre e na terceira e última parte debruçamo-nos sobre autómatos de árvores de aridade folhas-raiz.

3.1.1 Árvores

Vamos definir árvores finitas sobre um alfabeto (também conhecidas por árvores etiquetadas) e árvores finitas sobre um alfabeto de aridade. A definição de árvore finita sobre um alfabeto de aridade é um caso particular de árvore finita sobre um alfabeto e é sobre a primeira que o nosso estudo vai incidir neste capítulo. O estudo mais aprofundado de árvores finitas sobre um alfabeto fica adiado para o Capítulo 4. A escolha desta ordem deve-se ao facto de uma árvore finita sobre um alfabeto poder ser tratada como um caso particular de floresta finita sobre um alfabeto, sendo que esta última é a definição basilar do estudo apresentado no próximo capítulo.

Vamos também estabelecer algumas relações para árvores sobre alfabetos que, sempre que fizerem sentido, ficarão assim igualmente definidas para árvores finitas sobre alfabetos de aridade. Algumas destas relações são aqui introduzidas para serem utilizadas ao longo deste trabalho e outras apenas pela importância que assumem no variado estudo que tem sido desenvolvido acerca de linguagens de árvores finitas sobre um alfabeto (onde se inclui alguma da nossa bibliografia).

Consideremos $\mathbb{N}^*$, o conjunto das palavras sobre o alfabeto $\mathbb{N}$. Observamos, para evitar ambiguidades, que o número natural 1 não é a palavra vazia de $\mathbb{N}^*$. Definimos uma relação binária em $\mathbb{N}^*$, a que chamamos *relação prefixo* e que denotamos por $\leq_{\mathbb{N}^*}$, da seguinte maneira:

$$x \leq_{\mathbb{N}^*} y \text{ se e só se } x \text{ for um prefixo de } y.$$

Árvores Sobre um Alfabeto

Seja A um alfabeto. Uma *árvore t sobre A* é uma função parcial

$$t : \mathbb{N}^+ \longrightarrow A \text{ tal que:}$$

- o seu domínio, que denotamos por $dom(t)$, é finito;

- $dom(t) \cap \mathbb{N} \subseteq \{1\}$;

- $dom(t)$ é um *conjunto fechado para prefixos não vazios*, isto é,

$$\text{se } y \in dom(t) \text{ e } x \in \mathbb{N}^+ \text{ é tal que } x \leq_{\mathbb{N}^*} y, \text{ então } x \in dom(t);$$

- *não existem saltos em* $dom(t)$, isto é,

$$\text{se } m, n \in \mathbb{N} \text{ e } x \in \mathbb{N}^+ \text{ são tais que } m < n \text{ e } xn \in dom(t),$$
$$\text{então } xm \in dom(t).$$

Chamamos *árvore vazia* à árvore de domínio vazio e denotamo-la por 0.

Exemplo 3.1.1.
Seja A um alfabeto. Qualquer função parcial de $\mathbb{N}^+$ em A que tenha como domínio qualquer dos seguintes conjuntos é uma árvore sobre A:

$$\emptyset, \{1\}, \{1, 11, 12, 13, 14\}, \{1, 11, 12, 13, 111, 121, 122, 131, 132\}.$$

Por outro lado, nenhum dos seguintes conjuntos é domínio de uma árvore sobre A:

$$\{1, 11, 12, 111, 121, 122, 131, 132\}, \{1, 11, 12, 13, 111, 122, 131, 132\}.$$

Habitualmente representamos, neste capítulo, árvores sobre um alfabeto por t e s (possivelmente com índices).

Seja t uma árvore sobre um alfabeto A. Um elemento de $dom(t)$ é chamado *nódulo*. No caso de t não ser a árvore vazia, ao elemento de $dom(t)$ que pertence a $\mathbb{N}$, ou seja, ao número natural 1, chamamos *raiz de* t.

Dadas uma árvore t sobre um alfabeto A não vazia e um nódulo x de t, chamamos *etiqueta de* x a $t(x)$. Por esta razão também chamamos *árvores A-etiquetadas* ou *A-árvores* às árvores sobre o alfabeto A. Se $y \in dom(t)$, dizemos que *o nódulo* y *é uma folha de* t se e só se o nódulo y não é o prefixo próprio de nenhum nódulo de t, isto é, se e só se não existe $n \in \mathbb{N}$ tal que $yn \in dom(t)$.

Observamos que o domínio de uma árvore t sobre um alfabeto munido da restrição da ordem parcial $\leq_{\mathbb{N}^*}$ é um inf-semi-reticulado com elemento mínimo - o número 1 - tal que, para cada $x \in dom(t)$, o ideal de ordem gerado por x é uma cadeia. Esta característica costuma ser usada na literatura para definir árvore (não etiquetada), no entanto, por uma questão de simplificação, consideramos apenas os inf-semi-reticulados com esta característica que são subconjuntos de $\mathbb{N}^+$ e que satisfazem as quatro condições anteriores que caracterizam o domínio de uma árvore sobre um alfabeto.

Como é usual, se t é uma árvore sobre um alfabeto, representamo-la pelo diagrama de Hasse do conjunto parcialmente ordenado dual de $(dom(t), \leq)$, onde $\leq$ é a restrição de $\leq_{\mathbb{N}^*}$ a $dom(t)$, mas substituindo cada elemento de $dom(t)$ pela sua etiqueta. Nesta representação nódulos com o mesmo comprimento ficarão na mesma linha.

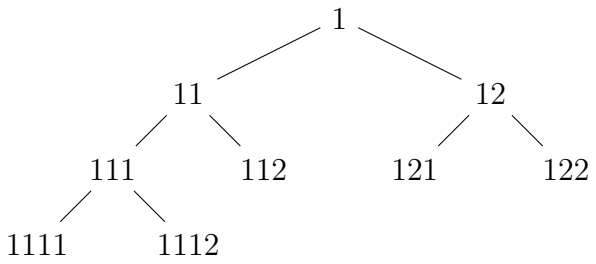
Exemplo 3.1.2.

Consideremos o alfabeto $A = \{a, b, c, d\}$. *Seja* t *a A-árvore definida da seguinte maneira:*

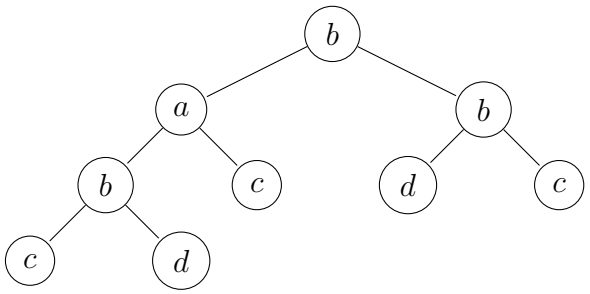
$$t : \mathbb{N}^* \longrightarrow A; \quad dom(t) = \{1, 11, 12, 111, 112, 121, 122, 1111, 1112\};$$

$t(1) = b$, $t(11) = a$, $t(12) = b$, $t(111) = b$, $t(112) = c$, $t(121) = d$, $t(122) = c$, $t(1111) = c$ *e* $t(1112) = d$.

O diagrama de Hasse do c.p.o. $dom(t)$ *é o seguinte:*



Podemos assim representar t da seguinte maneira:



Sejam A um alfabeto, t uma A-árvore e $x, y \in dom(t)$. Introduzimos as seguintes definições:

- y *é filho de* x se e só se $y = xk$ para algum $k \in \mathbb{N}$;
- y *é pai de* x se e só se $x = yk$ para algum $k \in \mathbb{N}$;
- se $x = zk$ para certos $z \in \mathbb{N}^*$ e $k \in \mathbb{N}$, y *é irmão seguinte a* x se e só se $y = z(k+1)$;
- se $x = z(k+1)$ para certos $z \in \mathbb{N}^*$ e $k \in \mathbb{N}$, y *é irmão anterior a* x se e só se $y = zk$;
- y *é descendente de* x se e só se $x <_{\mathbb{N}^*} y$;
- y *é ascendente de* x se e só se $y <_{\mathbb{N}^*} x$;
- se $x = zk$ para certos $z \in \mathbb{N}^*$ e $k \in \mathbb{N}$, y *é nódulo à direita de* x se e só se $y = zl$ para algum $l \in \mathbb{N}$ tal que $k < l$;
- se $x = zk$ para certos $z \in \mathbb{N}^*$ e $k \in \mathbb{N}$, y *é nódulo à esquerda de* x se e só se $y = zl$ para algum $l \in \mathbb{N}$ tal que $l < k$.

Definimos agora algumas relações binárias em $dom(t)$.

- A relação *filhos*, que denotamos por $child^t$:

$$child^t = \{(x, y) \mid x, y \in dom(t) \text{ e } x \text{ é filho de } y\}.$$

- A relação *pai*, que denotamos por $parent^t$:

$$parent^t = \{(x, y) \mid x, y \in dom(t) \text{ e } x \text{ é pai de } y\}.$$

- A relação *irmão seguinte*, que denotamos por ns^t (abreviatura do inglês *next sibling*):

$$ns^t = \{(x, y) \mid x, y \in dom(t) \text{ e } x \text{ é irmão seguinte de } y\}.$$

- A relação *irmão anterior*, que denotamos por ps^t (abreviatura do inglês *previous sibling*):

$$ps^t = \{(x, y) \mid x, y \in dom(t) \text{ e } x \text{ é irmão anterior de } y\}.$$

- A relação *descendente*, que denotamos por $descendant^t$:

$$descendant^t = \{(x, y) \mid x, y \in dom(t) \text{ e } x \text{ é descendente de } y\}.$$

- A relação *ascendente*, que denotamos por $ancestor^t$:

$$ancestor^t = \{(x, y) \mid x, y \in dom(t) \text{ e } x \text{ ascendente de } y\}.$$

- A relação *nódulo à direita*, que denotamos por $right^t$:

$$right^t = \{(x, y) \mid x, y \in dom(t) \text{ e } x \text{ é irmão à direita de } y\}.$$

- A relação *nódulo à esquerda*, que denotamos por $left^t$:

$$left^t = \{(x, y) \mid x, y \in dom(t) \text{ e } x \text{ é irmão à esquerda de } y\}.$$

Atendendo às definições de A-árvore e de filho de um nódulo é fácil verificar que caso um nódulo tenha filhos estes estão ordenados e, como tal, se $x, y \in dom(t)$ forem tais que $xn = y$ para algum $n \in \mathbb{N}$, podemos ser mais precisos e dizer que *y é o n-ésimo filho de x*.

Dados uma A-árvore t e $x, y \in dom(t)$, dizemos que *os nódulos x e y são irmãos* se tiverem o mesmo pai, mais precisamente, se $x = zm$ e $y = zn$ para certo $z \in dom(t)$ e certos $m, n \in \mathbb{N}$.

Sejam A um alfabeto e t uma A-árvore. Chamamos *caminho em t* a um conjunto de nódulos $\{x_1, \ldots, x_n\}$ de t, onde $n \in \mathbb{N}$ e $(x_i, x_{i+1}) \in child^t$ para todo o $i \in \{1, \ldots, n-1\}$. Dizemos que tal caminho $\{x_1, \ldots, x_n\}$ tem *comprimento n*. Chamamos *caminho maximal em t* a um tal conjunto $\{x_1, \ldots, x_n\}$ em que $x_1 = 1$ e x_n é a uma folha. Se $n \in \mathbb{N}_0$, definimos em $dom(t)$ a relação unária *profundidade n*, que denotamos por $depth^t_n$, como sendo o conjunto $\{x \mid x \in dom(t)$ e x é uma palavra sobre o alfabeto $\mathbb{N}$ de comprimento $n\}$. Se $n \in \mathbb{N}_0$, dizemos que a *árvore t tem profundidade n* se e só se n é o menor número inteiro não negativo tal que $depth^t_{n+1} = \emptyset$.

Dados um alfabeto A, uma A-árvore t e $x \in dom(t)$, definimos *subárvore sobre o alfabeto A de t com raiz em x* ou, simplesmente, *subárvore de t com raiz em x*, que denotamos por $t|_x$, como sendo a função parcial $t|_x$ tal que:

- $dom(t) = \{1y \mid y \in \mathbb{N}^*$ e $xy \in dom(t)\}$ e
- $t|_x(y) = t(xy)$, para todo o $y \in dom(t|_x)$.

Sejam A um alfabeto e t uma A-árvore. Definimos mais algumas relações.

- Dado $a \in A$, definimos em $dom(t)$ a relação unária *nódulos da árvore t de etiqueta a*, que denotamos por lab^t_a:

$$lab^t_a = \{x \mid x \in dom(t) \text{ e } t(x) = a\}.$$

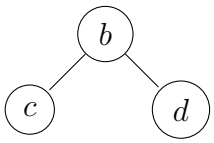
- Dado $i \in \mathbb{N}$, definimos em $dom(t)$ a relação binária *i-ésimos filhos da A-árvore t*, que denotamos por ch^t_i:

$$ch^t_i = \{(x, y) \mid x, y \in dom(t) \text{ e } y \text{ é o } i\text{-ésimo filho de } x\}.$$

  Notamos que é especialmente importante o caso em que $i = 1$, sendo recorrentemente utilizado no estudo de linguagens de árvores, onde muitas vezes é denotada por fc^t (do inglês *first child*).

Exemplo 3.1.3.
Relativamente ao Exemplo 3.1.2 temos:

- $lab_a^t = \{11\}$, $lab_b^t = \{1, 12, 111\}$, $lab_c^t = \{112, 122, 1111\}$, $lab_d^t = \{121, 1112\}$;
- $ch_i^t = \{(1, 11), (11, 111), (12, 121), (111, 1111)\}$,
  $ch_2^t = \{(1, 12), (11, 112), (12, 122), (111, 1112)\}$,
  $ch_3^t = \emptyset$.
- *A subárvore de t com raiz em* 111 *é a A-árvore*



(notamos que a raiz desta A-árvore é 1 *e não* 111*).*

Árvores Sobre um Alfabeto de Aridade

Chamamos *alfabeto de aridade* a um par ordenado (Σ, σ) formado por um conjunto finito e não vazio Σ e uma função $\sigma : \Sigma \longrightarrow \mathbb{N}_0$, ou seja, Σ é um alfabeto onde cada letra a tem um número inteiro não negativo atribuído, a que chamamos *aridade de a*. À função σ chamamos *função aridade*, a qual é frequente omitirmos, dizendo apenas, por exemplo, alfabeto de aridade Σ. Às letras de aridade zero chamamos *letras nulárias*.

Uma *árvore sobre um alfabeto de aridade* Σ é uma árvore não vazia e Σ-etiquetada, onde um nódulo de etiqueta a tem exatamente $\sigma(a)$ filhos. Em particular um nódulo é uma folha se e só se tem etiqueta de aridade zero. No resto deste trabalho, quando nos quisermos referir a uma árvore sobre um alfabeto de aridade (respetivamente, árvore sobre um alfabeto de aridade Σ)

vamos habitualmente falar em *árvore de aridade* (respetivamente, Σ-*árvore de aridade*).

Como estamos apenas a considerar árvores finitas (a finitude é consequência da definição de árvore sobre um alfabeto), vamos sempre considerar qualquer alfabeto de aridade com, pelo menos, uma letra nulária.

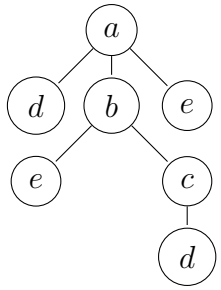
Exemplo 3.1.4.

Consideremos o alfabeto $\Sigma = \{a, b, c, d, e, f\}$ *com a função de aridade* σ *definida por* $\sigma(a) = 3$, $\sigma(b) = 2$, $\sigma(c) = 1$, $\sigma(d) = 0$, $\sigma(e) = 0$ *e* $\sigma(f) = 4$. *Seja* t *a* Σ-*árvore de aridade definida da seguinte maneira:*

$t : \mathbb{N}^+ \longrightarrow \Sigma$; $dom(t) = \{1, 11, 12, 13, 121, 122, 1221\}$;

$t(1) = a$, $t(11) = d$, $t(12) = b$, $t(13) = e$, $t(121) = e$, $t(122) = c$ *e* $t(1221) = d$.

Podemos assim representar t *da seguinte maneira:*



Nota 3.1.5.

Tal como já foi observado, a nossa definição de árvore sobre um alfabeto (e, consequentemente, a nossa definição de árvore sobre um alfabeto de aridade) implica que, sempre que um nódulo tenha filhos, esses filhos estejam ordenados. Existem definições alternativas de árvore sobre um alfabeto onde os filhos de um mesmo nódulo não têm ordem. Com a anterior opção ficamos ainda com mais hipóteses para definição de árvore [28].

Nota 3.1.6.

Da nossa definição de alfabeto de aridade decorre que cada letra tem uma aridade bem definida. Existem ideias alternativas para a definição de alfabeto

de aridade onde é permitido cada letra ter mais do que uma aridade. Por exemplo, podemos definir alfabeto de aridade n*, onde* $n \in \mathbb{N}$*, como sendo um conjunto finito e não vazio onde cada elemento tem aridade* n *ou* 0*. Uma* árvore sobre um alfabeto de aridade n *é uma árvore onde cada nódulo tem* n *ou* 0 *filhos. Esta é uma alternativa muito utilizada, por exemplo, em bibliografia relacionada com reconhecimento de linguagens de* árvores binárias*, onde é comum esta definição de alfabeto de aridade* n *com* $n = 2$*. Esta alternativa aumenta ainda as hipóteses para as definições de árvore atrás referidas.*

Contextos Sobre um Alfabeto de Aridade

Dado um alfabeto de aridade Σ, consideramos o alfabeto de aridade $\Sigma \cup \{\Box\}$, onde o símbolo $\Box$ não pertence a Σ e a função de aridade em $\Sigma \cup \{\Box\}$ estende a de Σ de maneira a que o símbolo $\Box$, a que chamamos *buraco*, tenha aridade zero.

Um *contexto sobre um alfabeto de aridade* Σ é uma árvore sobre o alfabeto de aridade $\Sigma \cup \{\Box\}$ tal que o elemento $\Box$ é etiqueta de um e um só nódulo. Notamos que o facto do elemento $\Box$ ter aridade zero implica que esse símbolo seja apenas etiqueta de folhas. Notamos que, com esta definição, admitimos um contexto sobre um alfabeto de aridade com um único nódulo e cuja etiqueta desse nódulo é $\Box$. Denotamos esse contexto, sempre que não existam ambiguidades, por 1.

No resto deste trabalho, quando nos quisermos referir a um contexto sobre um alfabeto de aridade (respetivamente, contexto sobre um alfabeto de aridade Σ) vamos habitualmente falar apenas em *contexto de aridade* (respetivamente, Σ-*contexto de aridade*). Habitualmente representaremos, nesta capítulo, Σ-contextos de aridade por p, q e r (possivelmente com índices).

Exemplo 3.1.7.

Consideremos o alfabeto $\Sigma = \{a, b, c, d, e, f\}$ *com a função de aridade* σ *definida por* $\sigma(a) = 3$*,* $\sigma(b) = 2$*,* $\sigma(c) = 1$*,* $\sigma(d) = 0$*,* $\sigma(e) = 0$ *e* $\sigma(f) = 4$*. Sejam* p *e* s *os* Σ*-contextos de aridade definidos da seguinte maneira:*

- $p : \mathbb{N}^+ \longrightarrow \Sigma$; $dom(p) = \{1, 11, 12, 13, 14, 121, 122, 131, 1221\}$;

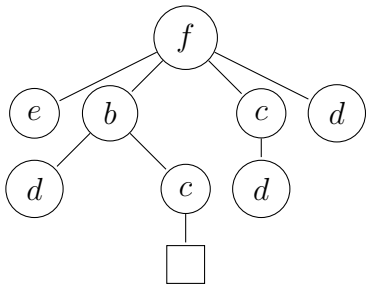
  $p(1) = f$, $p(11) = e$, $p(12) = b$, $p(13) = c$, $p(14) = d$ $p(121) = d$, $p(122) = c$, $p(131) = d$ *e* $p(1221) = \Box$;

- $s : \mathbb{N}^+ \longrightarrow \Sigma$; $dom(s) = \{1, 11, 111, 112, 113, 1111, 1131, 1132, 11321\}$;

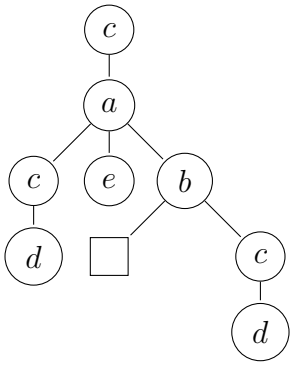
  $p(1) = c$, $p(11) = a$, $p(111) = c$, $p(112) = e$, $p(113) = b$, $p(1111) = d$, $p(1131) = \Box$, $p(1132) = c$ *e* $p(11321) = d$.

Podemos assim representar p e s da seguinte maneira:

p:



s:



Operações

Seja Σ um alfabeto de aridade. Dados um Σ-contexto de aridade p e uma Σ-árvore de aridade s, definimos a *composição de p e s*, que denotamos por $p{\cdot}s$ ou, simplesmente, ps, como sendo a Σ-árvore de aridade que, geometricamente, se obtém substituindo o buraco de p por s, isto é, a Σ-árvore de aridade definida por: sendo x o único nódulo de p com etiqueta $\Box$,

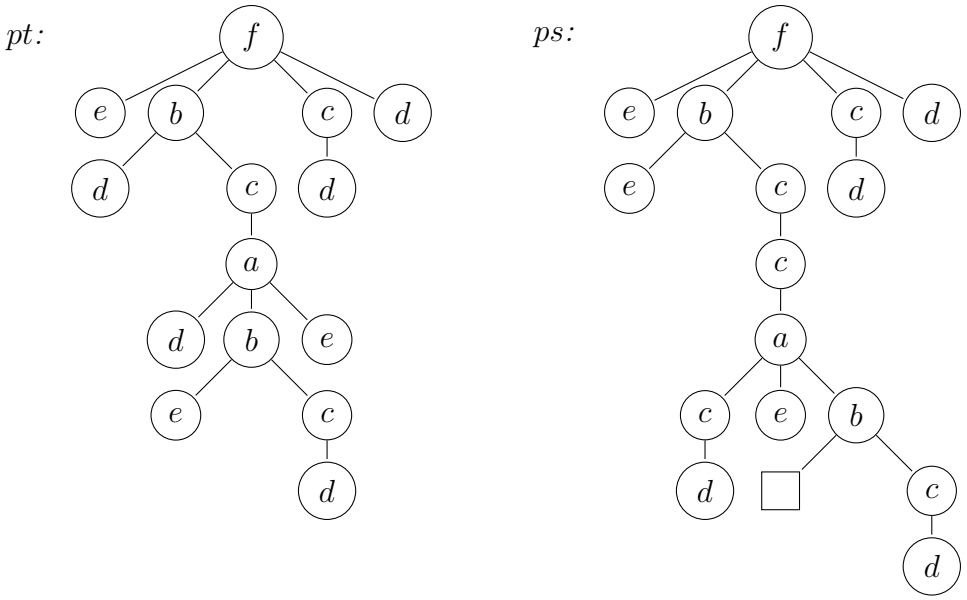
- $dom(ps) = dom(p) \cup \{xy \mid y \in \mathbb{N}^* \text{ é tal que } 1y \in dom(s)\}$;

- $(ps)|_{(dom(p)\setminus\{x\})} = p|_{(dom(p)\setminus\{x\})}$ e $(ps)(xy) = s(1y)$, para $y \in \mathbb{N}^*$ tal que $1y \in dom(s)$.

De maneira semelhante à anterior, dados Σ-contextos de aridade p e s, definimos a *composição de p e s*, que denotamos por $p{\cdot}s$ ou, simplesmente, ps. É claro que se s for uma Σ-árvore de aridade (respetivamente, Σ-contexto de aridade), então ps também o é.

Exemplo 3.1.8.

Consideremos dos Exemplos 3.1.4 e 3.1.7 o alfabeto de aridade Σ, a Σ-árvore de aridade t e os Σ-contextos de aridade p e s. Podemos assim representar, geometricamente, pt e ps da seguinte maneira:

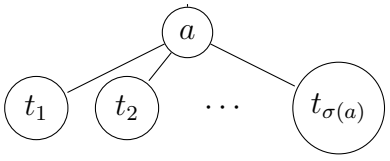


3.1.2 Σ-Álgebra e Σ-Álgebra Livre

Para uma melhor compreensão desta subsecção é conveniente ter-se alguma familiaridade com as noções elementares de Álgebra Universal. Dispensamo-nos de as apresentar aqui em geral para não sobrecarregar este trabalho, mas podem ser consultadas em qualquer livro de Álgebra Universal (ver, por exemplo, [13, 16]).

No contexto da Álgebra Universal, um alfabeto de aridade (Σ, σ) costuma ser chamado assinatura. Uma Σ-*álgebra* é um par ordenado $\mathcal{S} = \left(S, \Sigma^S\right)$, onde S é um conjunto e $\Sigma^S = \{a^S \colon a \in \Sigma\}$ é um conjunto de operações sobre S tais que, para cada $a \in \Sigma$, a operação a^S tem aridade $\sigma(a)$. Diz-se que a^S é a *interpretação* de a no conjunto S.

Consideremos o conjunto de todas as Σ-árvores de aridade, que vamos denotar por *árvores*(Σ). Neste conjunto, interpretamos cada letra a do alfabeto de aridade Σ como sendo uma operação $(\textit{árvores}(\Sigma))^{(\sigma(a))} \longrightarrow \textit{árvores}(\Sigma)$ que aplica a sequência de Σ-árvores de aridade $(t_1, t_2, \ldots, t_{\sigma(a)})$ na Σ-árvore de aridade representada por:



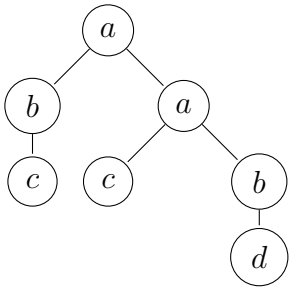
Esta Σ-árvore de aridade é denotada por $a(t_1, t_2, \ldots, t_{\sigma(a)})$. É fácil constatar que o par constituído por *árvores*(Σ) e pelo conjunto formado pelas operações atrás consideradas é uma Σ-álgebra, a qual, como habitualmente, denotamos também por *árvores*(Σ). Toda a Σ-árvore de aridade se pode obter a partir das Σ-árvores com um único nódulo e de operações deste tipo, como ilustra o exemplo seguinte. Isto é, *árvores*(Σ) é gerado pelo conjunto das Σ-árvores com um único nódulo no sentido de ser o menor subconjunto $\mathcal{T}$ de *árvores*(Σ) tal que:

- tem as Σ-árvores com um único nódulo,

- se $a \in \Sigma$ e $t_1, \ldots, t_{\sigma(a)} \in \mathcal{T}$, então $a(t_1, \ldots, t_{\sigma(a)}) \in \mathcal{T}$.

Exemplo 3.1.9.

Consideremos o alfabeto $\Sigma = \{a, b, c, d\}$ com a função de aridade σ definida por $\sigma(a) = 2$, $\sigma(b) = 1$, $\sigma(c) = 0$ e $\sigma(d) = 0$. Consideremos a seguinte Σ-árvore de aridade t:

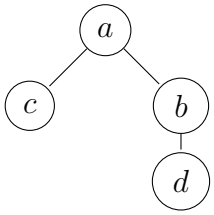


A Σ-árvore de aridade t pode ser obtida da seguinte maneira:

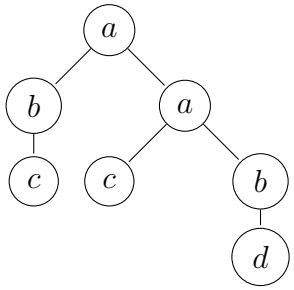
- *Primeiro, para cada etiqueta de uma folha de t, consideramos a Σ-árvore de aridade com um único nódulo e que tem essa etiqueta como etiqueta desse nódulo. Temos assim duas destas Σ-árvores de aridade, t_1 e t_2, cujos únicos nódulos são etiquetados por c e d, respetivamente:*

- *A partir das Σ-árvores de aridade construídas no passo anterior e das letras de aridade não nula de Σ, continuamos a construir as Σ-árvores de aridade que necessitamos para chegar à pretendida. Temos assim duas Σ-árvores de aridade, $b(t_1)$ e $b(t_2)$, respetivamente:*

- *A partir das Σ-árvores de aridade construidas nos passos anteriores e das letras de aridade não nula de Σ, continuamos a construção. Temos assim a Σ-árvore de aridade $a(t_1, b(t_2))$:*



- *Finalmente temos a* Σ*-árvore de aridade* t *pretendida, tendo-se* $t = a(b(t_1), a(t_1, b(t_2)))$*:*



É fácil ver que a Σ-álgebra *árvores*(Σ) é livre no seguinte sentido: para qualquer Σ-álgebra $\mathcal{S}$, existe um único morfismo da Σ-álgebra *árvores*(Σ) para $\mathcal{S}$ (recordamos que estamos a admitir que Σ tem letras nulárias). Isto significa que a Σ-álgebra *árvores*(Σ) é livre na classe das Σ-álgebras sobre o conjunto vazio. O alfabeto Σ é assim interpretado como assinatura e não como conjunto gerador. De facto, é claro que a Σ-álgebra *árvores*(Σ) é isomorfa à Σ-álgebra dos Σ-termos sobre o conjunto vazio. Recordamos, da Álgebra Universal, que os Σ-termos sobre o conjunto vazio, a que chamamos aqui, simplesmente, Σ-*termos*, podem ser definidos como expressões formais construídas recursivamente da seguinte maneira:

- as letras de Σ de aridade zero são Σ-termos;

- se $a \in \Sigma$ é tal que $\sigma(a) > 0$ e $t_1, \ldots, t_{\sigma(a)}$ são Σ-termos, então $a(t_1, \ldots, t_{\sigma(a)})$ é um Σ-termo.

Observamos que esta definição traduz o facto, que mencionámos anteriormente, de que toda a Σ-árvore se obtém das Σ-árvores com um único módulo e das operações associadas a cada letra de Σ. Denotamos o *conjunto dos* Σ*-termos* por T_Σ. A notação usada para os Σ-termos não gera, assim, ambiguidade com a notação utilizada para a construção de árvores $a(t_1, \ldots, t_{\sigma(a)})$. Identificaremos com frequência, sem qualquer menção, Σ-árvores de aridade e Σ-termos. No Exemplo 3.1.9, a Σ-árvore de aridade t é identificada com o Σ-termo $a(b(c), a(c, b(d)))$, este uma expressão formal.

Se $\mathcal{S}$ é uma Σ-álgebra e t é uma Σ-árvore de aridade (um elemento da Σ-álgebra livre), escrevemos $t^{\mathcal{S}}$ para denotar a imagem de t através do único morfismo de *árvores*(Σ) para $\mathcal{S}$.

Contrariamente ao que acontece em linguagens de palavras, em que consideramos a associatividade do monoide livre, aqui não consideramos nenhuma propriedade das operações nesta Σ-álgebra.

Definimos *linguagem de Σ-árvores de aridade* como um subconjunto de T_Σ.

3.1.3 Autómato de Árvores de Aridade Folhas-Raiz

Para as árvores de aridade existem estruturas análogas aos autómatos de palavras, a que chamamos *autómatos de árvores de aridade*. Existem vários tipos destes autómatos, entre os quais podemos encontrar: os *bottom-up* e os *top-down*, em inglês, que traduzimos por *folhas-raiz* e *raiz-folhas*, respetivamente; e os *tree walking*, que apenas abordamos a título exemplificativo nesta introdução e que por isso não lhes atribuímos outro nome. Vimos que dado um autómato de palavras finito que reconheça uma determinada linguagem de palavras, podemos sempre construir um autómato de palavras finito determinista e completo que reconheça essa mesma linguagem de palavras (ver Secção 2.2). Faz então sentido perguntarmo-nos se o mesmo acontece em autómatos de árvores de aridade. No caso de linguagens de árvores de aridade a pergunta tem de ser respondida tendo em conta cada tipo de autómato, pois não existe uma resposta que seja válida simultaneamente para todos. Por exemplo, a resposta é positiva para autómatos de árvores de aridade folhas-raiz, mas é negativa para os raiz-folhas e para os tree walking. Os autómatos de árvores de aridade folhas-raiz e raiz-folhas têm o mesmo poder expressivo, mas os autómatos de árvores de aridade folhas-raiz deterministas e completos são estritamente mais poderosos que os autómatos de árvores de aridade raiz-folhas deterministas e completos. Os autómatos de árvores de aridade tree walking são estritamente menos expressivos que os outros dois tipos de autómato de árvores de aridade. Pelas razões anteriores, pela frequência com que são utilizados no estudo de linguagens de árvores de aridade (em detrimento de outros) e pela necessidade de síntese neste trabalho, apenas vamos estudar autómatos de árvores de aridade folhas-raiz.

Um *autómato de árvores de aridade folhas-raiz* ou, simplesmente, *autómato de árvores de aridade*, é um quádruplo $\mathcal{A} = (Q, \Sigma, E, F)$ onde:

- Q é um conjunto, cujos elementos são chamados *estados*;
- Σ é um alfabeto de aridade;
- E é um conjunto, cujos elementos são chamados *transições* e são de um dos seguintes tipos:
  - (a, q), que também representamos por $\xrightarrow{a} q$, onde $a \in \Sigma$ é tal que $\sigma(a) = 0$ e $q \in Q$;
  - $((q_1, \ldots, q_{\sigma(a)}), a, q)$, que também representamos por $(q_1, \ldots, q_{\sigma(a)}) \xrightarrow{a} q$, onde $a \in \Sigma$ é tal que $\sigma(a) > 0$ e $q, q_1, \ldots, q_{\sigma(a)} \in Q$;
- $F \subseteq Q$, cujos elementos são chamados *estados finais*.

Ao contrário do que acontece nos autómatos na Secção 2.1, os autómatos de árvores de aridade folhas-raiz não têm um conjunto de estados iniciais, mas podemos interpretar qualquer estado q tal que (a, q) é uma transição para certo elemento a de aridade zero do alfabeto de aridade como sendo um estado inicial. Relembramos que assumimos a existência de pelo menos um destes elementos em Σ.

Dizemos que um autómato de árvores de aridade $\mathcal{A} = (Q, \Sigma, E, F)$ folhas-raiz é *finito* se o seu conjunto de estados for finito.

Dizemos que um autómato de árvores de aridade $\mathcal{A} = (Q, \Sigma, E, F)$ folhas-raiz é *determinista* se:

- para todo o $a \in \Sigma$ tal que $\sigma(a) = 0$, existe no máximo um $q \in Q$ tal que $(a, q) \in E$;
- para todo o $a \in \Sigma$ tal que $\sigma(a) > 0$ e para todos os $q_1, \ldots, q_{\sigma(a)} \in Q$, existe no máximo um $q \in Q$ tal que $((q_1, \ldots, q_{\sigma(a)}), a, q) \in E$.

Dizemos que um autómato de árvores de aridade $\mathcal{A} = (Q, \Sigma, E, F)$ é *completo* se:

- para todo o $a \in \Sigma$ tal que $\sigma(a) = 0$, existe pelo menos um $q \in Q$ tal que $(a, q) \in E$;
- para todo o $a \in \Sigma$ tal que $\sigma(a) > 0$ e para todos os $q_1, \ldots, q_{\sigma(a)} \in Q$, existe pelo menos um $q \in Q$ tal que $((q_1, \ldots, q_{\sigma(a)}), a, q) \in E$.

Podemos obter uma definição alternativa de *autómato de árvores de aridade folhas-raiz determinista e completo*, que denotamos por $\mathcal{A} = (Q, \Sigma, \delta, F)$, substituindo o conjunto E das transições por uma função, a que chamamos *função transição* e que denotamos por δ, que a cada elemento $a \in \Sigma$ faz corresponder uma função (total) δ_a, onde δ_a está definida da seguinte maneira:

- se $\sigma(a) = 0$, então δ_a é o único estado tal que $(a, \delta_a) \in E$;
- se $\sigma(a) > 0$, então $\delta_a : Q^{\sigma(a)} \longrightarrow Q$ é uma função (total) em que para quaisquer $q_1, \ldots, q_{\sigma(a)} \in Q$, $\delta_a(q_1, \ldots, q_{\sigma(a)})$ é o único estado q tal que $((q_1, \ldots, q_{\sigma(a)}), a, q) \in E$.

3.2 Reconhecimento de Linguagens de Árvores de Aridade

Ao longo desta secção vamos considerar um alfabeto de aridade Σ e uma linguagem de Σ-árvores de aridade L ($\subseteq T_\Sigma$).

Reconhecimento por Autómato de Árvores de Aridade Folhas-Raiz

Um *ciclo de um autómato de árvores de aridade* $\mathcal{A} = (Q, \Sigma, E, F)$ *sobre uma* Σ*-árvore de aridade* t é uma função parcial $c : dom(t) \longrightarrow Q$ que é compatível com E no seguinte sentido:

- se $x \in dom(t)$ tem etiqueta a com $\sigma(a) = 0$, então $c(x)$ é, caso exista, um elemento pertencente ao conjunto $\{q \in Q \mid (a, q) \in E\}$;
- se $x \in dom(t)$ tem etiqueta a com $\sigma(a) > 0$ e tem filhos $x_1, \ldots, x_\sigma(a)$ tais que $c(x_1) = q_1, \ldots, c(x_{\sigma(a)}) = q_{\sigma(a)}$, então $c(x)$ é, caso exista, um elemento pertencente ao conjunto $\{q \in Q \mid ((q_1, \ldots, q_\sigma(a)), a, q) \in E\}$.

Podemos obter um ciclo de um autómato de árvores de aridade sobre uma árvore de aridade de várias maneiras. Uma possibilidade é começar por aplicar folhas em estados, de seguida aplicar em estados os nódulos cujos filhos já tenham sido todos aplicados em estados e repetir este último passo enquanto for desejado e/ou possível. A seguir vamos descrever esta variante geometricamente. Neste trabalho qualquer ciclo de um autómato de árvores de aridade sobre uma árvore de aridade é obtido através da execução das seguintes ações, onde podemos ver um autómato como um mecanismo de leitura e escrita:

1. o autómato lê a etiqueta do nódulo mais à esquerda da linha mais abaixo da árvore de aridade,digamos a, e se existir pelo menos um estado q tal que $(a, q) \in E$, então atribui um desses estados ao nódulo e passa a executar 2 (a seguir) ou pára;

2. o autómato executa uma das seguintes ações:

   i) se na mesma linha existir um nódulo à direita do nódulo em que está o autómato, então o autómato anda uma posição para a direita e executa 3;

   ii) se na mesma linha não existir um nódulo à direita mas existir uma linha da árvore acima do nódulo em que está o autómato, então o autómato sobe uma linha, vai para a posição mais à esquerda dessa linha e executa 3;

   iii) se na mesma linha não existir um nódulo à direita nem existir uma linha da árvore acima do nódulo em que está o autómato, então o processo termina;

3. o autómato lê a etiqueta do nódulo em que está e executa uma das seguintes ações:

   i) se a letra lida, digamos a, tiver aridade zero e existir pelo menos um estado q tal que $(a, q) \in E$, então atribui um desses estados ao nódulo em que está e passa a executar 2 ou pára;

   ii) se a letra lida, digamos a, tiver aridade positiva n, então o autómato analisa o estado que atribuiu a cada um dos n filhos do nódulo

e, sendo eles $q_1, \ldots, q_n$ da esquerda para a direita, se existir pelo menos um estado q tal que $((q_1, \ldots, q_n), a, q) \in E$, então atribui um desses estados q ao nódulo em que está e passa a executar 2 ou pára.

Seja c um ciclo de um autómato de árvores de aridade $\mathcal{A} = (Q, \Sigma, E, F)$ sobre uma Σ-árvore de aridade t. Tal como vem sendo comum para vários conceitos presentes neste trabalho, dependendo da bibliografia que consultemos vamos encontrar diferentes alternativas para a representação de c. Neste trabalho vamos representar c da seguinte maneira:

- apresentamos geometricamente a árvore t;

- se $x \in dom(t)$ é tal que existe $c(x)$, então ao lado da posição de x na árvore t escrevemos $c(x)$ e o número que indica o número de vezes que no processo anterior foi executado o passo 1 ou 3 (isto é, $n+1$, onde n é o número de nódulos que estão em linhas inferiores à de x e na mesma linha à esquerda de x).

Exemplo 3.2.1.

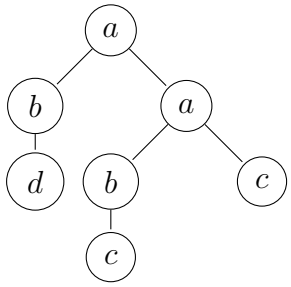
Consideremos o alfabeto $\Sigma = \{a, b, c, d\}$ com a função de aridade σ definida por $\sigma(a) = 2$, $\sigma(b) = 1$, $\sigma(c) = 0$ e $\sigma(d) = 0$, e o autómato de árvores de aridade $\mathcal{A} = (Q, \Sigma, E, F)$ onde:

- $Q = \{q_1, q_2, q_3, q_4, q_5\}$;

- $E = \{(c, q_1), (d, q_2), (q_1, b, q_2), (q_2, b, q_2), ((q_2, q_1), a, q_2), ((q_2, q_2), a, q_3), ((q_1, q_4), a, q_5)\}$;

- $F = \{q_3\}$.

Consideremos a Σ-árvore de aridade t que geometricamente se representa da seguinte maneira:

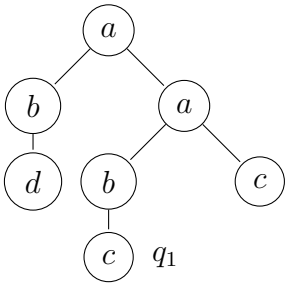


Consideremos o seguinte ciclo do autómato $\mathcal{A}$ sobre a árvore de aridade t:

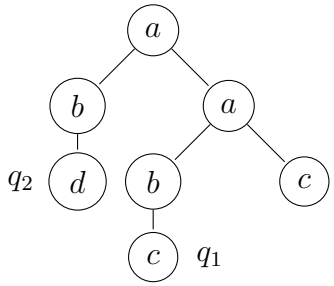
$$c : dom(t) \longrightarrow Q,$$

$c(1211) = q_1$, $c(111) = q_2$, $c(121) = q_2$, $c(122) = q_1$, $c(11) = q_2$, $c(12) = q_2$ *e* $c(1) = q_3$.

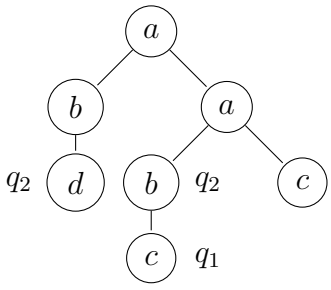
Assim podemos representar este ciclo através da seguinte sequência de grafos (notamos que a cada elemento da sequência foi acrescentada uma pequena explicação que não faz parte da representação habitual):



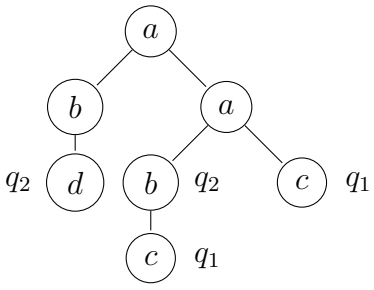
Primeiro o ciclo aplica 1211 *em* q_1.



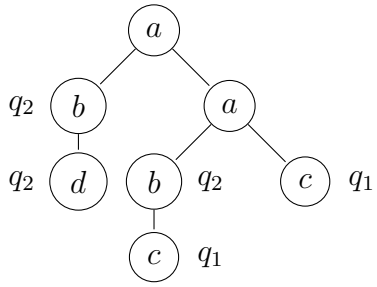
Depois aplica 111 *em* q_2.



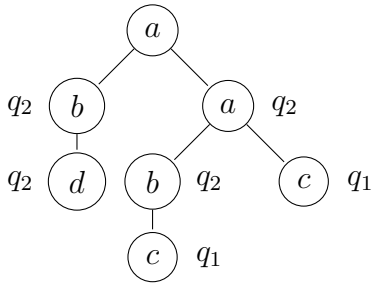
Agora o ciclo aplica 121 *em* q_2.



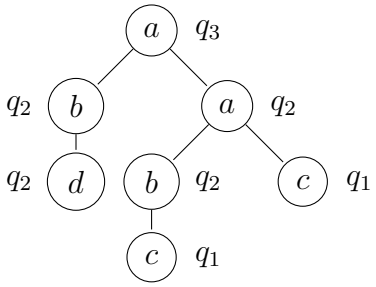
De seguida aplica 122 *em* q_1.



Aqui o ciclo aplica 11 *em* q_2.

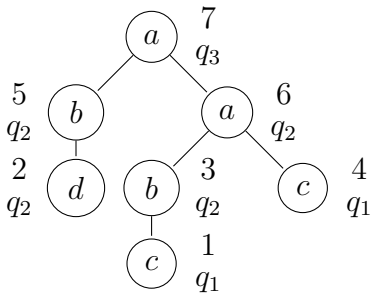


Neste passo aplica 12 *em* q_2.



Finalmente o ciclo aplica 1 *em* q_3.

Tal como referimos, neste trabalho vamos representar o ciclo do autómato $\mathcal{A}$ *sobre a árvore de aridade* t *apenas por:*



Dizemos que *um ciclo* c *de um autómato de árvores de aridade* $\mathcal{A} = (Q, \Sigma, E, F)$ *sobre uma* Σ*-árvore de aridade* t *é maximal* se não é possível ser estendido a mais nenhum nódulo, isto é, se não existir nenhum ciclo d de $\mathcal{A}$ sobre t tal que $dom(c) \subsetneqq dom(d)$ e $d|_{dom(c)} = c$.

Geometricamente, um ciclo de um autómato de árvores de aridade sobre uma árvore de aridade é maximal quando no processo descrito anteriormente para obter um ciclo, ocorre uma das seguintes condições:

- o autómato está num nódulo ao qual não é possível atribuir nenhum estado;
- 2iii)

Notamos que o ciclo do autómato sobre a árvore de aridade do Exemplo 3.2.1 é maximal.

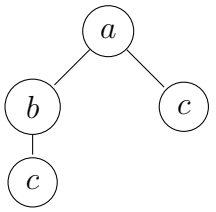
Exemplo 3.2.2.

Consideremos o alfabeto $\Sigma = \{a, b, c\}$ *com a função de aridade* σ *definida por* $\sigma(a) = 2$, $\sigma(b) = 1$ *e* $\sigma(c) = 0$ *e o autómato de árvores de aridade* $\mathcal{A} = (Q, \Sigma, E, F)$ *onde:*

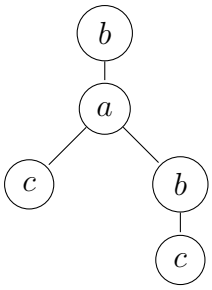
- $Q = \{q_1, q_2, q_3\}$;
- $E = \{(c, q_1), (c, q_2)(q_1, b, q_2), (q_2, b, q_2), (q_3, b, q_2), ((q_2, q_2), a, q_3)\}$;
- $F = \{q_3\}$.

Consideremos as Σ*-árvores de aridade* t_1 *e* t_2 *que geometricamente se representam da seguinte maneira:*

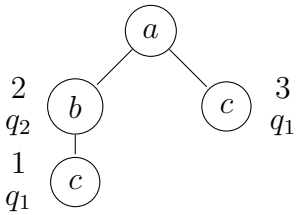
t_1:

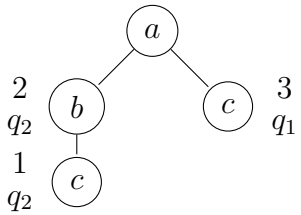


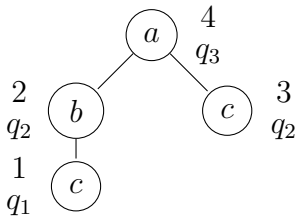
t_2:

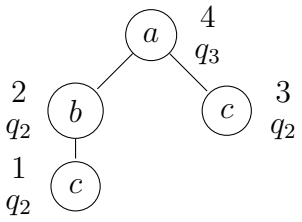


É fácil constatar que os seguintes ciclos são os únicos ciclos maximais do autómato $\mathcal{A}$ *sobre a árvore de aridade* t_1*:*

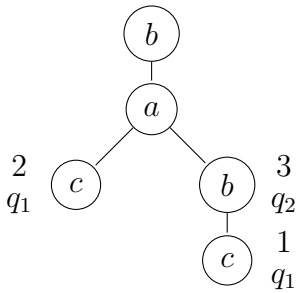


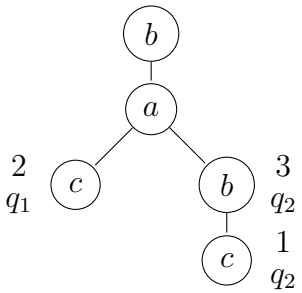


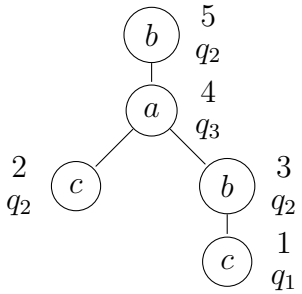


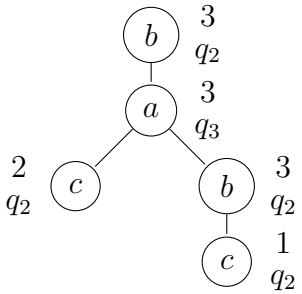


Igualmente fácil é constatar que os únicos ciclos maximais possíveis do autómato $\mathcal{A}$ sobre a árvore de aridade t_2 são:









Dizemos que *um autómato de árvores de aridade $\mathcal{A} = (Q, \Sigma, E, F)$ é bem sucedido sobre uma Σ-árvore de aridade t* se existe um ciclo de $\mathcal{A}$ sobre t que aplica a raiz num estado final.

Dizemos que *uma Σ-árvore de aridade t é reconhecida por um autómato de árvores de aridade $\mathcal{A} = (Q, \Sigma, E, F)$* se $\mathcal{A}$ é bem sucedido sobre t.

Dizemos que um estado q de um autómato de árvores de aridade folhas-raiz $\mathcal{A} = (Q, \Sigma, E, F)$ é *acessível* se existe alguma Σ-árvore de aridade t e um ciclo de $\mathcal{A}$ sobre t que aplica a raiz em q.

Dizemos que um autómato de árvores de aridade folhas-raiz $\mathcal{A} = (Q, \Sigma, E, F)$ é *reduzido* se todos os seus estados são acessíveis.

Exemplo 3.2.3.

Consideremos o Exemplo 3.2.1. Constatamos que o autómato $\mathcal{A}$ é bem sucedido sobre a árvore t e, como tal, a árvore t é reconhecida pelo autómato $\mathcal{A}$. É fácil concluir que os estados q_4 e q_5 não são acessíveis e, como tal, o autómato A não é reduzido.

Exemplo 3.2.4.

Consideremos agora o Exemplo 3.2.2. Constatamos que:

- *o autómato $\mathcal{A}$ é bem sucedido sobre a árvore t_1, logo, a árvore t_1 é reconhecida pelo autómato $\mathcal{A}$;*
- *o autómato $\mathcal{A}$ não é bem sucedido sobre a árvore t_2, logo, a árvore t_2 não é reconhecida pelo autómato $\mathcal{A}$.*

É fácil concluir que todos os estados do autómato $\mathcal{A}$ são acessíveis e, como tal, o autómato A é reduzido.

Vejamos agora de que maneira algumas das noções anteriores podem ser adaptadas para um autómato de árvores de aridade determinista e completo $\mathcal{A} = (Q, \Sigma, \delta, F)$.

Consideremos um autómato de árvores de aridade $\mathcal{A} = (Q, \Sigma, \delta, F)$ determinista e completo. O autómato de árvores de aridade $\mathcal{A}$ aplica uma Σ-árvore de aridade t num elemento de Q, que representamos por $t^{\mathcal{A}}$, que é definido por recorrência da seguinte maneira:

- se $t = a$ (neste caso temos de ter $\sigma(a) = 0$), então $t^{\mathcal{A}} = \delta_a$;
- se $t = a(t_1, \ldots, t_{\sigma(a)})$, onde $\sigma(a) \in \mathbb{N}$, então $t^{\mathcal{A}} = \delta_a(t_1^{\mathcal{A}}, \ldots, t_{\sigma(a)}^{\mathcal{A}})$.

Assim, *uma Σ-árvore de aridade t é reconhecida por $\mathcal{A} = (Q, \Sigma, \delta, F)$* se e só se $t^{\mathcal{A}} \in F$.

Seja $\mathcal{A}$ um autómato de árvores de aridade. Definimos

$$L(\mathcal{A}) = \{t \in T_\Sigma \mid t \text{ é reconhecida por } \mathcal{A}\},$$

a que chamamos *linguagem de Σ-árvores de aridade reconhecida por $\mathcal{A}$*.

Exemplo 3.2.5.

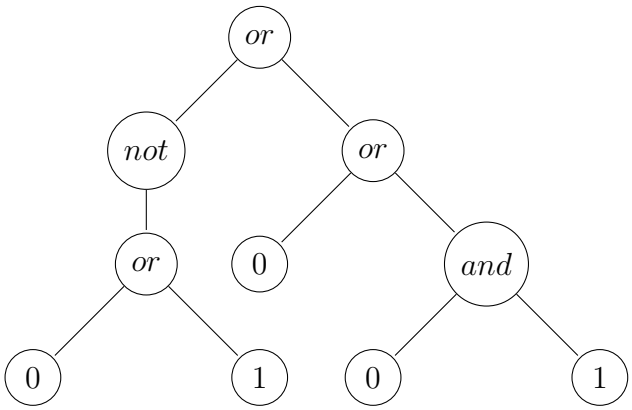
Consideremos o alfabeto $\Sigma = \{0, 1, not, and, or\}$ *com a função de aridade* σ *definida por* $\sigma(0) = 0$, $\sigma(1) = 0$, $\sigma(not) = 1$, $\sigma(and) = 2$ *e* $\sigma(or) = 2$ *e o autómato de árvores de aridade* $\mathcal{A} = (Q, \Sigma, E, F)$ *definido por:*

- $Q = \{q_0, q_1\}$;

- $E = \big\{(0, q_0), (1, q_1), (q_0, not, q_1), (q_1, not, q_0), ((q_0, q_0), and, q_0),$
  $((q_0, q_1), and, q_0), ((q_1, q_0), and, q_0), ((q_1, q_1), and, q_1),$
  $((q_0, q_0), or, q_0), ((q_0, q_1), or, q_1), ((q_1, q_0), or, q_1), ((q_1, q_1), or, q_1)\big\}$;

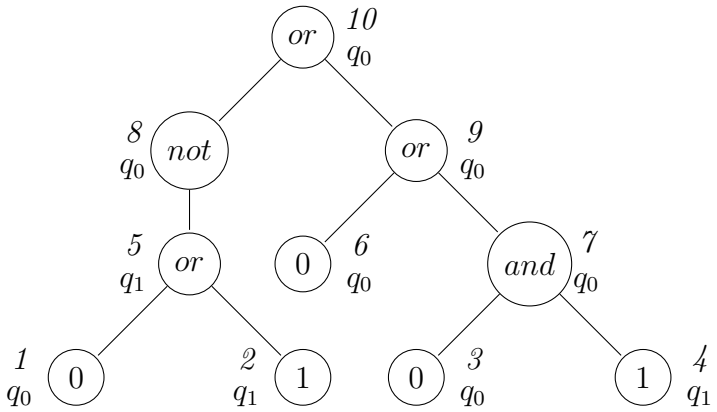
- $F = \{q_1\}$.

Consideremos as Σ*-árvores de aridade* t_1 *e* t_2 *que geometricamente se representam da seguinte maneira:*



Notamos que $\mathcal{A}$ *é um autómato de árvores de aridade determinista e completo.*

O seguinte ciclo é o único ciclo do autómato $\mathcal{A}$ *sobre a árvore de aridade* t *que é maximal:*



Podemos então afirmar que $\mathcal{A}$ não reconhece t.

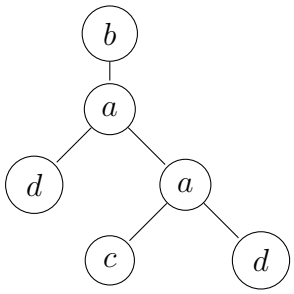
É fácil concluir que a linguagem de Σ-árvores de aridade reconhecida pelo autómato $\mathcal{A}$ é formada por e apenas por expressões booleanas verdadeiras sobre o alfabeto Σ.

Exemplo 3.2.6.

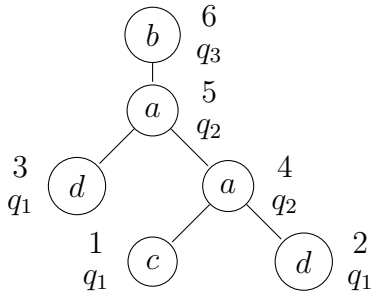
Consideremos o alfabeto $\Sigma = \{a, b, c, d\}$ com a função de aridade σ definida por $\sigma(a) = 2$, $\sigma(b) = 1$, $\sigma(c) = 0$ e $\sigma(d) = 0$, e o autómato de árvores de aridade $\mathcal{A} = (Q, \Sigma, E, F)$ definido por:

- $Q = \{q_1, q_2, q_3\}$;
- $E = \big\{(c, q_1), (d, q_1), (q_1, b, q_1), (q_2, b, q_3), (q_3, b, q_1)\big\}$
  $\bigcup \big((Q \times Q) \times \{a\} \times \{q_2\}\big)$;
- $F = \{q_3\}$.

Consideremos a Σ-árvore de aridade t que geometricamente se representa da seguinte maneira:



É fácil constatar que o seguinte ciclo é o único ciclo do autómato $\mathcal{A}$ sobre a árvore de aridade t que é maximal:



Através deste ciclo podemos concluir que $\mathcal{A}$ reconhece t.

É fácil constatar que a linguagem de Σ-árvores de aridade reconhecida pelo autómato $\mathcal{A}$ é constituída por e apenas por Σ-árvores de aridade cuja raiz tem etiqueta b e o filho da raiz tem etiqueta a, isto é,
$L(\mathcal{A}) = \{t \in T_\Sigma \mid t(1) = b,\ t(11) = a\}$.

Reconhecimento Algébrico

Se M é uma Σ-álgebra e $\varphi : T_\Sigma \longrightarrow M$ é o único morfismo de Σ-álgebras de T_Σ em M, dizemos que L *é reconhecida pelo morfismo* φ ou que L *é reconhecida por* M se $L = \varphi^{-1}(P)$ para algum $P \subseteq M$; equivalentemente, se $L = \varphi^{-1}(\varphi(L))$.

Σ-Álgebra Sintática

Dada uma Σ-álgebra $\mathcal{S} = \left(S, \Sigma^S\right)$ e dado $s \in S$, existe um único morfismo de Σ-álgebras $\varphi\colon T_{\Sigma\cup\{\Box\}} \to S$ tal que $\varphi(\Box) = s\colon$ basta considerar a $\Sigma \cup \{\Box\}$-álgebra $\mathcal{S}'$ obtida de $\mathcal{S}$ acrescentando-lhe a operação nulária s como interpretação de $\Box$ e tomar para φ o único morfismo de $\Sigma \cup \{\Box\}$-álgebras de $T_\Sigma \cup \{\Box\}$ em $\mathcal{S}'$. Dado $s \in T_\Sigma$, denotemos por ξ_s o único morfismo de Σ-álgebras de $T_{\Sigma\cup\{\Box\}}$ em T_Σ tal que $\xi_s(\Box) = s$. Então, dado um Σ-contexto

de aridade p e uma Σ-árvore de aridade s, a Σ-árvore de aridade ps é a que corresponde precisamente ao Σ-termo $\xi_s(p)$.

Dada $L \subseteq T_\Sigma$, definimos na Σ-álgebra T_Σ uma relação de equivalência que denotamos por $\sim_L$:

> $s \sim_L t$ se e só se para todo o $a \in \Sigma$ de aridade n e todo o $i \in \{1, \ldots, n\}$ e $r_1, \ldots, r_{i-1}, r_{i+1}, \ldots, r_n \in T_\Sigma$,
>
> $$(a(r_1, \ldots, r_{i-1}, s, r_{i+1}, \ldots, r_n)) \in L \Leftrightarrow (a(r_1, \ldots, r_{i-1}, t, r_{i+1}, \ldots, r_n)) \in L.$$

Este facto é equivalente a

$$\xi_s(p) \in L \text{ se e só se } \xi_t(p) \in L, \quad \text{para todo o } \Sigma\text{-contexto de aridade } p,$$

ou seja,

$$s \sim_L t \text{ se e só se, para todo o } \Sigma\text{-contexto de aridade } p, \quad ps \in L \Leftrightarrow pt \in L.$$

Proposição 3.2.7.
A relação $\sim_L$ é uma congruência da Σ-álgebra T_Σ.

Demonstração. Dados um Σ-contexto p, uma letra $a \in \Sigma$ com aridade n e $s_1, \ldots, s_n \in T_\Sigma$, é fácil perceber que

$$p\big(a(s_1, \ldots, s_n)\big) = \big(p\big(a(s_1, \ldots, s_{i-1}, \Box, s_{i+1}, \ldots, s_n)\big)\big)s_i,$$

para todo o $i \in \{1, \ldots, n\}$. Tomemos agora $t_1, \ldots, t_n \in T_\Sigma$. Suponhamos que $s_i \sim_L t_i$, para todo o $i \in \{1, \ldots, n\}$. Então

$$\begin{aligned}
p\big(a(s_1, \ldots, s_n)\big) \in L &\iff \big(p\big(a(s_1, \ldots, s_{n-1}, \Box)\big)\big)s_n \in L \\
&\iff \big(p\big(a(s_1, \ldots, s_{n-1}, \Box)\big)\big)t_n \in L \\
&\iff p\big(a(s_1, \ldots, s_{n-1}, t_n)\big) \in L \\
&\iff \big(p\big(a(s_1, \ldots, s_{n-2}, \Box, t_n)\big)\big)s_{n-1} \in L \\
&\iff \big(p\big(a(s_1, \ldots, s_{n-2}, \Box, t_n)\big)\big)t_{n-1} \in L \\
&\iff p\big(a(s_1, \ldots, s_{n-2}, t_{n-1}, t_n)\big) \in L \\
&\;\;\vdots \\
&\iff p\big(a(t_1, \ldots, t_n)\big) \in L.
\end{aligned}$$

Portanto $\sim_L$ é uma congruência da Σ-álgebra T_Σ. $\square$

O quociente de T_Σ por $\sim_L$ diz-se a Σ-*álgebra sintática de* L e será representada por $M(L)$. É claro que $M(L)$ reconhece L, pois $\sim_L$ satura L uma vez que $\xi_s(\Box) = s$, para todo o $s \in T_\Sigma$.

Os dois resultados que se seguem são análogos ao que se passa para linguagens de palavras.

Proposição 3.2.8.
Uma Σ-álgebra M reconhece L se e só se $M(L)$ divide M.

Demonstração. Seja M uma Σ-álgebra e seja $\varphi\colon T_\Sigma \to M$ o único morfismo de Σ-álgebras de T_Σ em M.

Suponhamos que M reconhece L. Então $L = \varphi^{-1}(P)$, para um certo $P \subseteq M$. Sejam $s, t \in T_\Sigma$ tais que $\varphi(s) = \varphi(t)$. A composição $\varphi\xi_s\colon T_{\Sigma\cup\{\Box\}} \to M$ é o único morfismo de Σ-álgebras $\psi\colon T_{\Sigma\cup\{\Box\}} \to M$ tal que $\psi(\Box) = \varphi(s)$ e a composição $\varphi\xi_t\colon T_{\Sigma\cup\{\Box\}} \to M$ é o único morfismo de Σ-álgebras $\psi\colon T_{\Sigma\cup\{\Box\}} \to M$ tal que $\psi(\Box) = \varphi(t)$. Logo $\varphi\xi_s = \varphi\xi_t$. Então, para qualquer Σ-contexto de aridade p,

$$\begin{aligned} \xi_s(p) \in L &\iff \varphi\xi_s(p) \in P \\ &\iff \varphi\xi_t(p) \in P \\ &\iff \xi_t(p) \in L. \end{aligned}$$

Logo $s \sim_L t$. Podemos agora concluir que existe um morfismo de Σ-álgebras $\mu\colon \varphi(T_\Sigma) \to M(L)$ tal que $\mu(\varphi(s)) = [s]_{\sim_L}$ para todo o $s \in T_\Sigma$. Este morfismo é sobrejetivo e, consequentemente, $M(L)$ é imagem homomorfa de uma Σ-álgebra de M.

Agora suponhamos que $M(L)$ é imagem homomorfa de uma subálgebra de M. Sejam então M' uma subálgebra de M e $\varphi\colon M' \to M(L)$ um morfismo sobrejectivo. Seja $\psi\colon T_\Sigma \to M'$ o único morfismo de T_Σ em M'. Então $\varphi\psi\colon T_\Sigma \to M(L)$ é o único morfismo de T_Σ em $M(L)$, logo o morfismo canónico associado à congruência $\sim_L$. Assim, $L = (\varphi\psi)^{-1}(\varphi\psi(L)) = \psi^{-1}(\varphi^{-1}\varphi\psi(L))$, pelo que L é reconhecida por M' e, portanto, por M. $\square$

Uma consequência imediata da Proposição 3.2.8 é é a seguinte.

Corolário 3.2.9.
A linguagem de Σ-árvores de aridade L é reconhecida por uma Σ-álgebra finita se e só se a Σ-álgebra $M(L)$ é finita.

"Reconhecimento por Definibilidade Lógica"

Consideremos uma Σ-árvore de aridade t e a assinatura

$$\big\{ch_i \mid i \in \mathbb{N} \text{ é tal que existe } a \in \Sigma \text{ com aridade superior ou igual a } i\big\} \bigcup \big\{lab_a \mid a \in \Sigma\big\},$$

onde, para cada $i \in \mathbb{N}$ tal que existe $a \in \Sigma$ com aridade superior ou igual a i, ch_i tem aridade 2 e, para cada $a \in A$, lab_a tem aridade 1. O terno $(dom(t), \sigma, I)$ onde

- $\sigma = \big\{ch_i \mid i \in \mathbb{N} \text{ é tal que existe } a \in \Sigma \text{ com aridade superior ou igual a } i\big\} \cup \big\{lab_a \mid a \in \Sigma\big\}$ e

- I é a função definida por: para cada $i \in \mathbb{N}$ tal que existe $a \in \Sigma$ com aridade superior ou igual a i, $I(ch_i) = ch_i^t$ e, para cada $a \in \Sigma$, $I(lab_a) = lab_a^t$;

é uma estrutura e vamos denotá-la por $\underline{t}$.

No âmbito das Σ-árvores de aridade, vamos chamar *fórmula* $\texttt{MSO}(ch_i)$ a uma fórmula da Lógica Monádica de Segunda Ordem com a assinatura anterior.

Seja φ uma fórmula $\texttt{MSO}(ch_i)$. A *linguagem de palavras definida por* φ é o conjunto

$$\{t \in T_\Sigma \mid \text{existe uma atribuição de variáveis } \mu \text{ tal que } (\underline{t}, \mu) \models \varphi\}$$

e denotamo-la por $L(\varphi)$. Dizemos que *a linguagem L é definida através de uma fórmula* $\texttt{MSO}(ch_i)$ se, para aquela assinatura, existir uma fórmula $\texttt{MSO}(ch_i)$, digamos φ, tal que $L = L(\varphi)$.

Teorema de Equivalência de Reconhecimento de Linguagens de Árvores de Aridade

Teorema 3.2.10.
Seja $L \subseteq T_\Sigma$. Então as seguintes afirmações são equivalentes:

i) L é reconhecida por um autómato de árvores de aridade folhas-raiz finito.

ii) L é reconhecida por um autómato de árvores de aridade folhas-raiz determinista e completo finito.

iii) L é reconhecida por uma Σ-álgebra finita.

iv) L tem a Σ-álgebra sintática finita.

v) L é definida por uma fórmula $\mathtt{MSO}(ch_i)$.

Mais ainda, existem algoritmos para passar de qualquer uma destas especificações para outra.

Com este resultado podemos, a partir de agora, falar apenas de *linguagem de árvores de aridade reconhecível* para nos referirmos a uma linguagem de de árvores de aridade que satisfaça qualquer uma das condições do Teorema 3.2.10.

Observamos que autómatos de árvores de aridade deterministas e completos folhas-raiz são Σ-álgebras, logo, as noções de reconhecimento por autómato e de reconhecimento algébrico não são verdadeiramente distintas. Isto faz com que este teorema seja menos satisfatório que o seu correspondente para linguagens de palavras.

3.3 Limitações das Σ-Álgebras

Vamos aqui seguir [2] e descrever alguns dos benefícios decorrentes de utilizar Álgebra (semigrupos e monoides) no caso das linguagens de palavras

que não correm igualmente bem para a utilização de Álgebra (Σ-álgebras) no caso das Σ-árvores de aridade.

Ao longo desta secção vamos considerar um alfabeto de aridade Σ.

Como vimos, uma das consequências importantes de utilizar a abordagem algébrica nas linguagens de palavras é que propriedades de linguagens de palavras do tipo "a linguagem de palavras pode ser definida por uma fórmula $\mathtt{FO}(S)$" corresponde a propriedades do monoide sintático, tal como "o monoide sintático é finito" (ver Teorema 2.2.3). Por outro lado, propriedades importantes dos semigrupos podem ser dadas através de equações, por exemplo, como vimos na Subsecção 2.1.3, a equação

$$s^{\omega} = s^{\omega+1},$$

que diz que um semigrupo é aperiódico, ou as equações

$$st = ts \quad \text{e} \quad ss = s,$$

que dizem, respetivamente, que um semigrupo é comutativo e idempotente. Infelizmente, deparamo-nos com problemas quando tentamos fazer o mesmo para Σ-álgebras. Por exemplo, suponhamos que queremos estudar a classe das linguagens de Σ-árvores de aridade que são invariantes para a reordenação dos filhos. Esta propriedade pode ser expressa usando equações, mas de uma maneira não natural: para cada letra a do alfabeto Σ, com aridade $\sigma(a)$, necessitamos de equações que impliquem que os filhos possam ser reordenados, isto é,

$$a(x_1, \ldots, x_{\sigma(a)}) = a(x_j, x_2, \ldots, x_{j-1}, x_1, x_{j+1}, \ldots, x_{n-1}, x_n) \quad \text{para todo o } j \in \{2, \ldots, n\}.$$

Outro problema é o conjunto dos objetos da Σ-álgebra não ser "suficientemente rico" no sentido que descrevemos de seguida. Consideremos a equação de aperiodicidade num monoide livre. O que faz com que esta equação seja tão poderosa é ela dizer que numa expressão podemos substituir o lado esquerdo pelo lado direito. Quando estamos a lidar com palavras, isto significa que para um n suficientemente grande, um fator u^n pode ser substituído por um fator u^{n+1}. Em Σ-álgebras, elementos da Σ-álgebra livre correspondem a Σ-árvores de aridade e qualquer equação na Σ-álgebra livre semelhante à anterior (com as naturais adaptações), quando aplicada a Σ-árvores, irá apenas permitir substituir uma Σ-subárvore de aridade por outra Σ-subárvore de

aridade. O resultado semelhante a este para o caso das palavras é podermos apenas substituir sufixos em vez de fatores. Isto significa que muito poucas propriedades importantes das Σ-árvores possam ser descritas utilizando equações nas Σ-álgebras livres.

Capítulo 4

Linguagens de Florestas

Neste capítulo temos como principais objetivos definir florestas finitas sobre um alfabeto e apresentar alguns conceitos e resultados relacionados com elas e com linguagens das mesmas. Os alfabetos que vamos considerar neste capítulo não são alfabetos de aridade.

Já referimos que a definição de árvore finita sobre um alfabeto é um caso particular de floresta finita sobre um alfabeto. Vamos aqui ver alguns casos em que as noções estudadas neste capítulo para florestas finitas sobre uma alfabeto podem ser aplicadas no estudo de linguagens de árvores finitas sobre um alfabeto.

No estudo aqui apresentado seguimos [2, 3, 10, 4, 11, 27, 29] e, especialmente, [12].

Também neste capítulo adotámos uma estrutura semelhante à utilizada no Capítulo 2.

4.1 Definições Base

Esta secção está dividida em quatro partes: na primeira parte tratamos inicialmente das definições de floresta finita sobre um alfabeto, contexto finito sobre um alfabeto, de algumas relações e operações; na segunda parte defi-

nimos álgebra-floresta; na terceira parte definimos álgebra-floresta livre; na quarta, e última, parte debruçamo-nos sobre autómatos de florestas. Vamos basear-nos em [12, 2, 4, 11, 27].

4.1.1 Florestas e Contextos Sobre um Alfabeto

Florestas Sobre um Alfabeto

Definimos *floresta sobre um alfabeto* adaptando de uma maneira natural a definição de árvore sobre um alfabeto do Capítulo 3. Permitimos agora múltiplas raízes.

Seja A um alfabeto. Uma *floresta sobre o alfabeto* A é uma função parcial

$$t : \mathbb{N}^+ \longrightarrow A \text{ tal que:}$$

- o seu domínio, que denotamos por $dom(t)$, é finito;
- $dom(t)$ é um *conjunto fechado para prefixos não vazios*, isto é,

  se $y \in dom(t)$ e $x \in \mathbb{N}^+$ é tal que $x \leq_{\mathbb{N}^*} y$, então $x \in dom(t)$;

- *não existem saltos em* $dom(t)$, isto é,

  se $m, n \in \mathbb{N}$ e $x \in \mathbb{N}^*$ são tais que $m < n$ e $xn \in dom(t)$,
  então $xm \in dom(t)$.

Chamamos *floresta vazia* à floresta de domínio vazio e denotamo-la por 0.

Habitualmente representamos, neste capítulo, florestas sobre um alfabeto por t e s (possivelmente com índices).

Seja t uma floresta sobre um alfabeto A. Um elemento de $dom(t)$ chama-se *nódulo*. Um elemento de $dom(t)$ que pertença a $\mathbb{N}$ chama-se *raiz*.

Dadas uma floresta t sobre um alfabeto A não vazia e um nódulo x de t, chamamos *etiqueta de* x a $t(x)$. Por esta razão também chamamos *florestas A-etiquetadas* ou *A-florestas* às florestas sobre o alfabeto A.

Uma *folha* de uma A-floresta t é um nódulo de t que não é prefixo próprio de nenhum nódulo de t. Também as definições que demos na Subsecção 3.1.1 para árvores sobre um alfabeto de *filho de*, ..., *nódulo à esquerda de*, e das relações associadas podem ser dadas para florestas sobre um alfabeto de forma análoga, pelo que nos dispensamos de as apresentar. Define-se ainda de forma similar as relações $depth_n^t$, lab_a^t e ch_i^t. Tal como para árvores, a *profundidade* de uma A-floresta t é o menor número inteiro não negativo n tal que $depth_{n+1}^t = \emptyset$.

Tal como foi observado no início do Capítulo 3, notamos que uma árvore sobre um alfabeto é um caso particular de floresta sobre um alfabeto (logo, também uma árvore sobre um alfabeto de aridade é um caso particular de uma floresta sobre um alfabeto).

O domínio de uma floresta t sobre um alfabeto A, se não é vazio, satisfaz $dom(t) \cap \mathbb{N} = \{1, \ldots, n\}$, onde $n \in \mathbb{N}$. Para cada $i \in \{1, \ldots, n\}$, seja X_i o conjunto formado pelas palavras de dom(t) cuja primeira letra (letra mais à esquerda) é i e seja $t_i = t_{|_{X_i}}$. É claro que $t = t_1 \cup \cdots \cup t_n$ e que t_1 é uma árvore. Embora se $i \in \{2, \ldots, n\}$, t_i não seja uma árvore sobre A (segundo a definição de árvore sobre A que fixámos), apenas difere de uma árvore na primeira letra de cada palavra do seu domínio, que é i e não o número 1; isto é, sendo $t_i' \colon \mathbb{N}^+ \to A$ a função parcial em que $dom(t_i') = \{1x \colon x \in \mathbb{N}^+ \text{ e } ix \in dom(t_i)\}$ e $t_i'(1x) = t_i(ix)$, para todo o $x \in \mathbb{N}^*$ tal que $ix \in dom(t_i)$, t_i' é uma árvore sobre A. Assim, a floresta t pode ser representada geometricamente de forma natural, representando a árvore t_1, ao seu lado direito t_2 como se fosse a árvore t_2', etc.. Notamos que, nessas representações, nódulos de iguais comprimentos se encontram na mesma linha. Também no caso das florestas daremos algumas definições usando a representação geométrica para evitar demasiado formalismo.

Contextos Sobre um Alfabeto

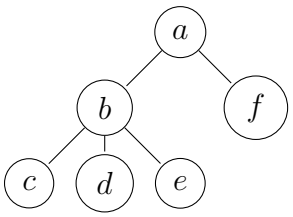
Definimos *contexto sem aridade* adaptando de uma maneira natural a definição de contexto de aridade do Capítulo 3. Permitimos agora múltiplas raízes.

Dado um alfabeto A, consideramos o alfabeto $A \cup \{\Box\}$, onde o símbolo $\Box$, a que chamamos *buraco*, não pertence a A. Um *contexto sem aridade sobre*

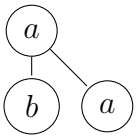
um alfabeto A é uma floresta sobre $A \cup \{\Box\}$, onde o elemento $\Box$ é etiqueta de um e um só nódulo, o qual é uma folha. É claro que o símbolo $\Box$ pode ser etiqueta de uma raiz. O contexto sem aridade com um único nódulo, será denotado, sempre que não houver ambiguidade, por 1.

No resto deste trabalho, quando nos quisermos referir a um contexto sem aridade sobre um alfabeto (respetivamente, contexto sem aridade sobre um alfabeto A) vamos habitualmente falar apenas em *contexto* (respetivamente, A-*contexto*). Habitualmente representamos, neste capítulo, contextos sem aridade por p, q e r (possivelmente com índices).

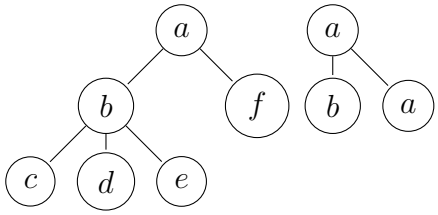
Exemplo 4.1.1.



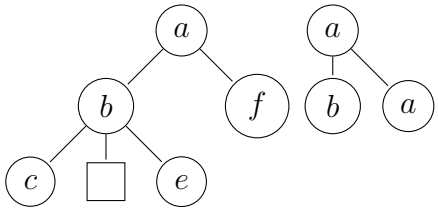
Uma árvore



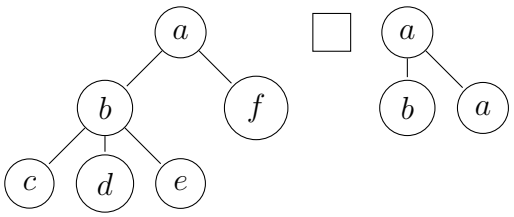
Uma árvore



Uma floresta

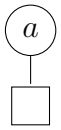


Um contexto



Um contexto

Para cada letra $a \in A$, consideramos o seguinte A-contexto, que vamos denotar por $a\Box$:

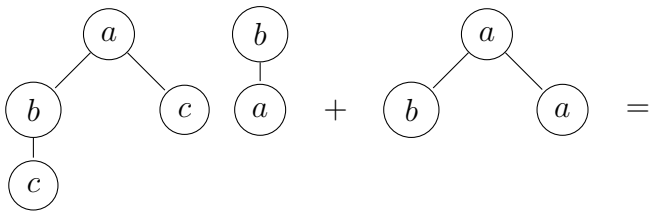


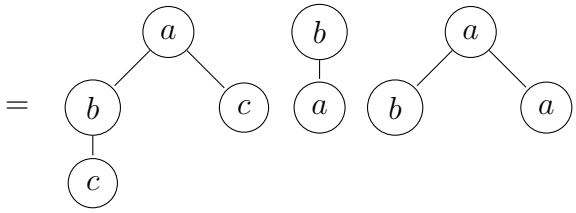
Operações

Estabelecemos algumas "operações":

- Duas *constantes*, isto é, duas operações nulárias: a *floresta vazia*, 0, e o *contexto sem aridade identidade*, 1.
- Uma operação binária no conjunto das A-florestas, que denotamos por $+$ e a que chamamos *concatenação*: dadas A-florestas s e t, a concatenação de s com t, denotada $s+t$, é a A-floresta que, geometricamente, se obtém colocando t à direita de s (daí o nome *concatenação*).

  Exemplo 4.1.2.



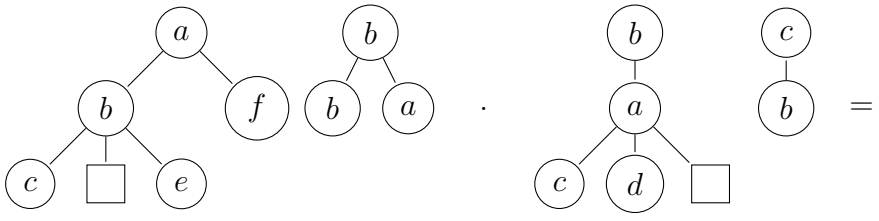


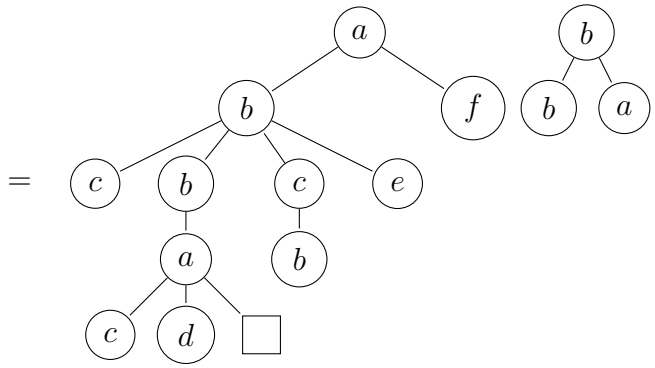
Esta operação é, claramente, associativa e tem elemento neutro, que é a floresta vazia.

- Dados um A-contexto p e uma A-floresta t, definimos a *concatenação de t com p*, que representamos por $t+p$, e a *concatenação de p com t*, que representamos por $p+t$, como sendo o A-contexto que, geometricamente, se obtém colocando, no primeiro caso p à direita de t, e no segundo caso t à direita de p (tal como a concatenação de duas A-florestas). Se t for a floresta vazia, então $t + p = p + t = p$.

- Uma operação binária no conjunto dos A-contextos *composição*: se p e q são A-contextos, a *composição de p com q*, que denotamos por $p \cdot q$, é o contexto q se $p = 1$, e, no caso de $p \neq 1$, é o contexto que, geometricamente, se obtém de p substituindo o seu buraco por q de modo que todas as raízes de q sejam filhas do pai do nódulo de p que tem etiqueta $\Box$.

Exemplo 4.1.3.





Esta operação é, obviamente, associativa e tem elemento neutro, que é o contexto 1.

- Uma operação entre um A-contexto p e uma A-floresta t chamada *composição* e que denotamos por $p{\cdot}t$: se p é o contexto 1, $p{\cdot}t$ é a A-floresta t; se p é um A-contexto diferente de 1, $p \cdot t$ é a A-floresta que, geometricamente, se obtém de p substituindo o seu buraco por t de modo a que todas as raízes de t sejam filhas do pai do nódulo de p que tem etiqueta $\Box$ (tal como a composição de dois A-contextos). No caso de p ser diferente de 1 e t ser a floresta vazia, queremos dizer que $p \cdot t$ é a A-floresta que, geometricamente, se obtém de p suprimindo o buraco.

Assumimos que $\cdot$ tem precedência sobre $+$, logo, por exemplo $p.q + s$ representa $(p.q) + s$ e não $p.(q + s)$.

4.1.2 Álgebra-Floresta

No Capítulo 2 definimos monoide sintático e, dado um alfabeto de aridade Σ, no Capítulo 3 definimos Σ-álgebra sintática. Estas definições tiveram como motivação a sua aplicação, respetivamente, ao reconhecimento de linguagens de palavras e ao reconhecimento de linguagens de árvores de aridade. Agora, vamos construir uma nova estrutura algébrica, a que chamaremos *álgebra-floresta*, com intuito de a utilizarmos no reconhecimento de linguagens de florestas. Pela sua maior complexidade mas também pela

importância que damos a linguagens de florestas neste trabalho e, em particular, a álgebras-floresta, vamos dedicar toda esta subsecção à construção da mesma. Notamos que a definição de álgebra-floresta que apresentamos pode ser vista numa perspetiva da Álgebra Universal, onde é um caso particular de "many-sorted algebra", mais precisamente, de "two-sorted algebra" (ver, por exemplo, [29]).

Baseados nas operações de concatenação e composição definidas anteriormente para florestas e contextos, definimos agora álgebra-floresta.

Uma *álgebra-floresta* é uma sequência ordenada

$$\big((H, +_{HH}, 0), (V, \cdot_{VV}, 1), +_{HV}, +_{VH}, \cdot_{VH}\big),$$

que representamos também, simplesmente, por (H, V), onde

$$+_{HH}\colon \begin{array}{rcl} H \times H & \longrightarrow & H\ , \\ (g,h) & \longmapsto & g +_{HH} h \end{array} \qquad \cdot_{VV}\colon \begin{array}{rcl} V \times V & \longrightarrow & V \\ (g,h) & \longmapsto & g \cdot_{VV} h \end{array}$$

e

$$+_{HV}\colon \begin{array}{rcl} H \times V & \longrightarrow & V\ , \\ (g,h) & \longmapsto & g +_{HV} h \end{array}$$

$$+_{VH}\colon \begin{array}{rcl} V \times H & \longrightarrow & V \\ (g,h) & \longmapsto & g +_{VH} h \end{array}$$

e

$$\cdot_{VH}\colon \begin{array}{rcl} V \times H & \longrightarrow & H \\ (g,h) & \longmapsto & g \cdot_{VH} h \end{array}$$

são funções tais que

1. $(H, +_{HH}, 0)$ é um monoide, chamado *monoide horizontal*;
2. $(V, \cdot_{VV}, 1)$ é um monoide, chamado *monoide vertical*;
3. para quaisquer $g, h \in H$ e $v, w \in V$,
   - 3.1. $(g +_{HH} h) +_{HV} v = g +_{HV} (h +_{HV} v)$,
   - 3.2. $0 +_{HV} v = v$,
   - 3.3. $v +_{VH} (g +_{HH} h) = (v +_{VH} g) +_{VH} h$,
   - 3.4. $v +_{VH} 0 = v$,

3.5. $(g +_{HV} v) +_{VH} h = g +_{HV} (v +_{VH} h)$,

3.6. $(v \cdot_{VV} w) \cdot_{VH} h = v \cdot_{VH} (w \cdot_{VH} h)$,

3.7. $1 \cdot_{VH} h = h$,

3.8. $(1 +_{VH} h) \cdot_{VH} g = g +_{HH} h$ e $(h +_{HV} 1) \cdot_{VH} g = h +_{HH} g$.

3.9. $(1 +_{VH} h) \cdot_{VV} v = v +_{VH} h$ e $(h +_{HV} 1) \cdot_{VV} v = h +_{HV} v$.

Notamos que as propriedades 3.3 e 3.6 permitem escrever sem ambiguidade $v +_{VH} g +_{HH} h$ e $v \cdot_{VV} w \cdot_{VH} h$, respectivamente. Deduz-se da lista de propriedades anterior que, dados $g, h_1, h_2 \in H$ e $v, w \in V$,

3.8' $(h_1 +_{HV} w +_{VH} h_2) \cdot_{VH} g = h_1 +_{HV} (w \cdot_{VH} g) +_{VH} h_2$;

3.9' $(h_1 +_{HV} w +_{VH} h_2) \cdot_{VV} v = h_1 +_{HV} (w \cdot_{VV} v) +_{VH} h_2$.

A propriedade 3.8' conclui-se de 3.9 e 3.8 através do seguinte raciocínio:

$$\begin{aligned}(h_1 +_{HV} w +_{VH} h_2) \cdot_{VH} g &= (1 +_{VH} h_2) \cdot_{VV} (h_1 +_{HV} w) \cdot_{VH} g \\ &= (1 +_{VH} h_2) \cdot_{VV} (h_1 +_{HV} 1) \cdot_{VV} w \cdot_{VH} g \\ &= (1 +_{VH} h_2) \cdot_{VH} (h_1 +_{HH} (w \cdot_{VH} g)) \\ &= h_1 +_{HV} (w \cdot_{VH} g) +_{VH} h_2.\end{aligned}$$

A propriedade 3.9' conclui-se analogamente.

Na perspetiva da Álgebra Universal, uma álgebra floresta

$$\big((H, +_{HH}, 0), (V, \cdot_{VV}, 1), +_{HV}, +_{VH}, \cdot_{VH}\big)$$

é uma "two-sorted algebra" onde os "sorts" são H e V (ver [29]).

Consideremos duas álgebras-floresta

$$F_1 = \big((H, +_{HH}, 0), (V, \cdot_{VV}, 1), +_{HV}, +_{VH}, \cdot_{VH}\big)$$

e

$$F_2 = \big((G, +_{GG}, 0), (W, \cdot_{WW}, 1), +_{GW}, +_{WG}, \cdot_{WG}\big)$$

Da Álgebra Universal extraímos várias definições.

- Dizemos que F_1 *é finita* se e só se H e V são finitos.

- Se $\alpha : H \longrightarrow G$ e $\beta : V \longrightarrow W$ são funções, dizemos que (α, β) *é um morfismo de álgebras-floresta de* F_1 para F_2 se e só se (α, β) preserva as operações de álgebra-floresta, isto é:

  - α e β são morfismos de monoides,
  - $\alpha(v \cdot_{VH} h) = \beta(v) \cdot_{WG} \alpha(h)$,
  - $\beta(h +_{HV} v) = \alpha(h) +_{GW} \beta(v)$ e
  - $\beta(v +_{VH} h) = \beta(v) +_{WG} \alpha(h)$,

  para quaisquer $h \in H$ e $v \in V$. Nesse caso, denotamo-lo por

$$(\alpha, \beta) : (H, V) \longrightarrow (G, W).$$

Proposição 4.1.4.
Sejam

$$F_1 = \big((H, +_{HH}, 0), (V, \cdot_{VV}, 1), +_{HV}, +_{VH}, \cdot_{VH}\big)$$

e

$$F_2 = \big((G, +_{GG}, 0), (W, \cdot_{WW}, 1), +_{GW}, +_{WG}, \cdot_{WG}\big)$$

duas álgebras-floresta e sejam $\alpha : H \longrightarrow G$ *e* $\beta : V \longrightarrow W$ *dois morfismos de monoides.*

Então (α, β) *é um morfismo de álgebras-floresta se e só se as seguintes condições são satisfeitas:*

- $\alpha(v \cdot_{VH} h) = \beta(v) \cdot_{WG} \alpha(h)$,
- $\beta(h +_{HV} 1) = \alpha(h) +_{GW} 1$ *e*
- $\beta(1 +_{VH} h) = 1 +_{WG} \alpha(h)$.

Demonstração. A implicação direta é óbvia. Agora suponhamos que as três condições do enunciado são satisfeitas. Sejam $h \in H$ e $v \in V$. Então

$$\begin{aligned} \alpha(h) +_{GW} \beta(v) &= (\alpha(h) +_{GW} 1) \cdot_{WW} \beta(v) \\ &= \beta(h +_{HV} 1) \cdot_{WW} \beta(v) \\ &= \beta((h +_{HV} 1) \cdot_{VV} v) \\ &= \beta(h +_{HV} v). \end{aligned}$$

A prova de que $\beta(v +_{VH} h) = \beta(v) +_{WG} \alpha(h)$ faz-se de forma análoga. □

Nota 4.1.5.
Se $(\alpha, \beta)\colon F_1 \to F_2$ for um morfismo, então o morfismo α fica determinado pelo morfismo β:

$$\alpha(h) = \alpha(h +_{HH} 0) = \alpha((h +_{HV} 1) \cdot_{VH} 0) = \beta(h +_{HV} 1) \cdot_{WG} 0.$$

Logo, para obter um morfismo de F_1 em F_2 é suficiente dar um morfismo $\beta\colon V \to W$ e verificar que (α, β) preserva as operações, onde $\alpha\colon H \to G$ é o morfismo obtido de β como atrás.

- Dizemos que F_2 *é imagem homomorfa de* F_1 se existir um morfismo $(\alpha, \beta)\colon F_1 \to F_2$ sobrejetivo, isto é, se α e β são sobrejetivos.
- Dizemos que F_1 *é uma subálgebra-floresta de* F_2 se e só se
  - $H \subseteq G$ e $V \subseteq W$,
  - $(\iota_{F_1}, \iota_{F_2})$, onde $\iota_{F_1} : H \longrightarrow G$ e $\iota_{F_2} : V \longrightarrow W$ são as funções inclusão, é um morfismo.
- Dizemos que F_1 *divide* F_2 se e só se F_1 é imagem homomorfa de alguma subálgebra-floresta de F_2.
- Dizemos que (R_H, R_V) *é uma relação em* F_1 se e só se R_H e R_V são relações em H e V respetivamente. Se R_H e R_V são relações de equivalência em H e V respetivamente, dizemos que (R_H, R_V) *é uma relação de equivalência em* F_1.
- Dizemos que (R_H, R_V) *é uma relação de congruência em* F_1 se e só se
  - (R_H,R_V) é uma relação de equivalência em F_1,
  - (R_H, R_V) é compatível com as operações da álgebra-floresta, isto é, para quaisquer $h_1, h_2, h'_1, h'_2 \in H$ tais que $(h_1, h_2), (h'_1, h'_2) \in R_H$, e para quaisquer $v_1, v_2, v'_1, v'_2 \in V$ tais que $(v_1, v_2), (v'_1, v'_2) \in R_V$, temos:
    - $(h_1 +_{HH} h'_1, h_2 +_{HH} h'_2) \in R_H$,
    - $(v_1 \cdot_{VV} v'_1, v_2 \cdot_{VV} v'_2) \in R_V$,
    - $(v_1 \cdot_{VH} h_1, v_2 \cdot_{VH} h_2) \in R_H$,
    - $(h_1 +_{HV} v_1, h_2 +_{HV} v_2) \in R_V$ e
    - $(v_1 +_{VH} h_1, v_2 +_{VH} h_2) \in R_V$.

- Se (R_H, R_V) é uma relação de congruência em F_1, sendo $H' = H/R_H$ e $V' = V/R_V$, definimos o *quociente de* F_1 *por* (R_H, R_V) como sendo a álgebra-floresta $\big((H', +_{H'H'}, 0), (V', \cdot_{V'V'}, 1), +_{H'V'}, +_{V'H'}, \cdot_{V'H'}\big)$ com as operações induzidas pelas operações de F_1.

4.1.3 Álgebra-Floresta Livre

Seja A um alfabeto. Atendendo à forma como, geometricamente, se efetuam as operações definidas na subsecção anterior, representamos o conjunto das A-florestas por H_A, da palavra *horizontal*, e o conjunto dos A-contextos por V_A, da palavra *vertical*.

Consideremos agora a sequência ordenada

$$((H_A, +_{H_AH_A}, 0), (V_A, \cdot_{V_AV_A}, 1), +_{H_AV_A}, +_{V_AH_A}, \cdot_{V_AH_A})$$

onde

$$\begin{array}{rrcl} +_{H_AH_A}\colon & H_A \times H_A & \longrightarrow & H_A \\ & (s,t) & \longmapsto & s +_{H_AH_A} t = s + t \end{array}$$

e

$$\begin{array}{rrcl} \cdot_{V_AV_A}\colon & V_A \times V_A & \longrightarrow & V_A \\ & (s,t) & \longmapsto & s \cdot_{V_AV_A} t = s \cdot t \end{array}$$

são operações binárias e

$$\begin{array}{rrcl} +_{H_AV_A}\colon & H_A \times V_A & \longrightarrow & V_A\ , \\ & (s,t) & \longmapsto & s +_{H_AV_A} t = s + t \end{array}$$

$$\begin{array}{rrcl} +_{V_AH_A}\colon & V_A \times H_A & \longrightarrow & V_A \\ & (s,t) & \longmapsto & s +_{V_AH_A} t = s + t \end{array}$$

e

$$\begin{array}{rrcl} \cdot_{V_AH_A}\colon & V_A \times H_A & \longrightarrow & H_A \\ & (s,t) & \longmapsto & s \cdot_{V_AH_A} t = s \cdot t \end{array}$$

são funções a que podemos chamar *ação esquerda de* H_A *sobre* V_A, *ação direita de* H_A *sobre* V_A e *ação esquerda de* V_A *sobre* H_A, respetivamente. É fácil ver que são válidas as seguintes propriedades (onde suprimimos o índice das operações e das ações, uma vez que os argumentos as identificam):

1. $(H_A, +, 0)$ é um monoide;
2. $(V_A, \cdot, 1)$ é um monoide;
3. para quaisquer $s, t \in H_A$ e $p, q \in V_A$,
   - 3.1. $(s + t) + p = s + (t + p)$,
   - 3.2. $0 + p = p$
   - 3.3. $p + (s + t) = (p + s) + t$,
   - 3.4. $p + 0 = p$,
   - 3.5. $(s + p) + t = s + (p + t)$,
   - 3.6. $(p \cdot q) \cdot t = p \cdot (q \cdot t)$,
   - 3.7. $1 \cdot t = t$;
   - 3.8. $(1 + s) \cdot t = t + s$ e $(s + 1) \cdot t = s + t$;
   - 3.9. $(1 + s) \cdot p = p + s$ e $(s + 1) \cdot p = s + p$.

Como tal, $((H_A, +_{H_AH_A}, 0), (V_A, \cdot_{V_AV_A}, 1), +_{H_AV_A}, +_{V_AH_A}, \cdot_{V_AH_A})$ é uma álgebra-floresta. Notamos que este era um resultado esperado: a definição de álgebra-floresta (em particular, a escolha dos axiomas) teve como motivação o facto de

$$((H_A, +_{H_AH_A}, 0), (V_A, \cdot_{V_AV_A}, 1), +_{H_AV_A}, +_{V_AH_A}, \cdot_{V_AH_A})$$

possuir as propriedades anteriores.

Não é difícil constatar que é possível construir qualquer A-floresta ou A-contexto aplicando as operações de concatenação e composição, atrás definidas, sobre a floresta vazia, 0, e os A-contextos 1 e $a\Box$, com $a \in A$. Isto é, a álgebra-floresta (H_A, V_A) é gerada pelo par $(\emptyset, \{a\Box \mid a \in A\})$ no sentido de ser o menor par (F, C) (para a relação de inclusão em cada coordenada), com $F \subseteq H_A$ e $C \subseteq V_A$, tal que:

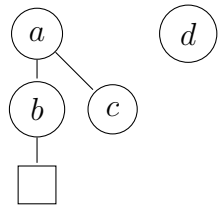
- $\{a\Box \mid a \in A\} \subseteq C$;
- F é um submonoide de H_A e C é um submonoide de V_A.
- para quaisquer $h \in F$, $v \in C$,

$$h +_{H_AV_A} v \in C,\ v +_{V_AH_A} h \in C \quad \text{e} \quad v \cdot_{V_AH_A} h \in F.$$

A construção pode ser por indução no número de nódulos da A-floresta ou do A-contexto e corresponde a iterações das operações a partir das folhas até à raiz.

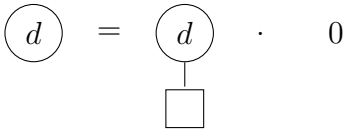
Exemplo 4.1.6.

Consideremos o alfabeto $A = \{a, b, c, d\}$ e o A-contexto p:

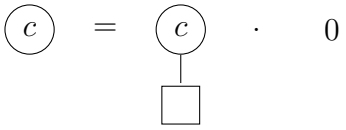
p = 

Podemos obter o A-contexto p da seguinte maneira:

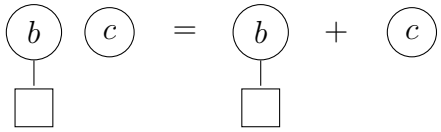
- *Primeiro necessitamos ter as A-floresta $t_1 = d\Box.0$ e $t_2 = c\Box.0$:*



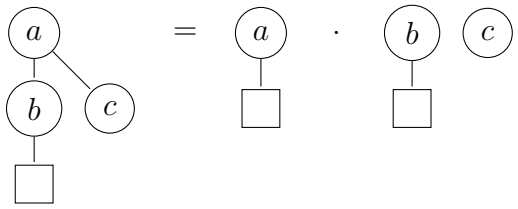
e



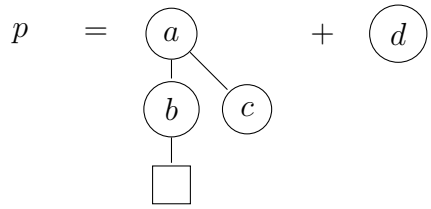
- *Agora já podemos construir o A-contexto $p_1 = b\Box + t_2$:*



- *Neste momento já é possível obter o A-contexto $p_2 = a\Box.p_1$:*



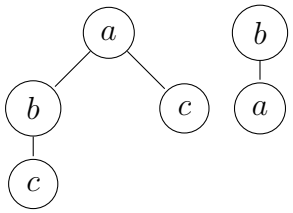
- *Podemos finalmente chegar ao A-contexto* $p = p_2 + t_1$*:*



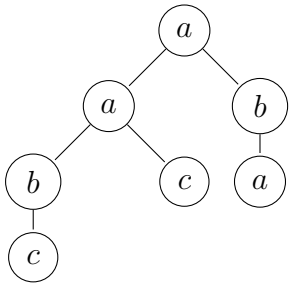
Seja $a \in A$. Consideremos a A-árvore com um único nódulo e cuja etiqueta desse nódulo é a. Quando não existir ambiguidade, denotamos esta A-árvore pela própria letra a. Se $a \in A$ e t é uma A-floresta ou um A-contexto, denotamos $a\Box \cdot t$ por at.

Exemplo 4.1.7.

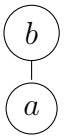
Se s é a A-floresta



então as é a A-árvore

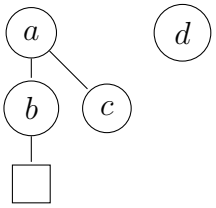


A A-árvore



é $b\Box \cdot a$ *e pode ser denotada por* ba*.*

Assim, se t é o A-contexto (A-contexto considerado anteriormente),



temos $t = a\Box \cdot (b\Box + c) + d = a(b\Box + c) + d$.

Qualquer A-floresta diferente de 0 se escreve de modo único como concatenação de A-árvores diferentes de 0 e é claro que qualquer A-árvore t diferente de 0 é da forma $a\Box \cdot s$, onde s é uma A-floresta de profundidade inferior em uma unidade à profundidade de t. Também um A-contexto diferente de 1 se escreve de modo único como concatenação de A-árvores e de A-contextos, onde, necessariamente, apenas uma das parcelas é um A-contexto com uma única raiz e as restantes são A-árvores; é ainda óbvio que qualquer A-contexto p diferente de 1 com uma única raiz é da forma $a\Box \cdot q$, onde q é um A-contexto de profundidade inferior em uma unidade à profundidade de p. Assim, é imediato concluir que, por um lado, qualquer A-floresta diferente de 0 se escreve de modo único como uma expressão que utiliza (para além de eventuais parêntesis) apenas as operações de concatenação e composição e as A-árvores a e os A-contextos $a\Box$, com $a \in A$; por outro lado, também qualquer A-contexto admite uma única expressão que utiliza (para além de eventuais parêntesis) somente as operações de concatenação e composição e as A-árvores a e os A-contextos $a\Box$, com $a \in A$, e, eventualmente, o A-contexto 1, mas uma única vez. Pode provar-se ainda que todas as expressões destes tipos representam uma A-floresta ou um A-contexto. A álgebra-floresta (H_A, V_A) é um objeto livre na categoria das álgebras-floresta sobre o alfabeto A, tal como é mostrado na seguinte proposição (ver, por exemplo, [12]).

Proposição 4.1.8.

Seja (H, V) *uma álgebra-floresta e seja* $f : A \longrightarrow V$ *uma função.*
Então existe um único morfismo de álgebras-floresta

$$(\alpha, \beta) : (H_A, V_A) \longrightarrow (H, V)$$

que estende a função f *no sentido de* $\beta(a\Box) = f(a)$ *para todo o* $a \in A$*.*

Demonstração. Definimos funções $\alpha\colon H_A \to H$ e $\beta\colon V_A \to V$ por indução no número de nódulos da A-floresta ou do A-contexto da seguinte forma: para quaisquer $a \in A$, $s, t \in H_A$ e $p \in V_A$,

- $\alpha(0) = 0 \quad \text{e} \quad \beta(1) = 1$;
- $\beta(a\Box) = f(a)$ para todo o $a \in A$;
- $\alpha(a\Box \cdot_{V_AH_A} t) = f(a) \cdot_{VH} \alpha(t)$;
- $\beta(a\Box \cdot_{V_AV_A} p) = f(a) \cdot_{VV} \beta(p)$;
- $\alpha(s +_{H_AH_A} t) = \alpha(s) +_{HH} \alpha(t)$;
- $\beta(s +_{H_AV_A} p +_{V_AH_A} t) = \alpha(s) +_{HV} \beta(p) +_{VH} \alpha(t)$.

Atendendo à unicidade da decomposição de A-florestas e A-contextos que descrevemos antes do enunciado da proposição, estas aplicações estão, de facto, bem definidas. Além disso, esta é a única extensão possível de f a um morfismo de (H_A, V_A) a (H, V). Vejamos agora que (α, β) é um morfismo. É claro que α preserva a operação $+_{H_AH_A}$. Vejamos por indução no número de nódulos de p que $\beta(p \cdot_{V_AV_A} q) = \beta(p) \cdot_{VV} \beta(q)$, para quaisquer $p, q \in V_A$. O caso de $p = 1$ é óbvio. Suponhamos que $p \neq 1$. Então $p = s +_{H_AV_A} p' +_{V_AH_A} t$, onde $s, t \in H_A$ e $p' \in V_A$ tal que p' tem uma única raiz. Logo

$$\begin{aligned}\beta(p \cdot_{V_AV_A} q) &= \beta(s +_{H_AV_A} (p' \cdot_{V_AV_A} q) +_{V_AH_A} t)\\ &= \alpha(s) +_{HV} \beta(p' \cdot_{V_AV_A} q) +_{VH} \alpha(t)\\ &= \alpha(s) +_{HV} \beta(p' \cdot_{V_AV_A} q) +_{VH} \alpha(t).\end{aligned}$$

Se $p' = 1$, então $\beta(p' \cdot_{V_AV_A} q) = \beta(q) = \beta(q') \cdot_{VV} \beta(q)$. Caso contrário, $p' = a\Box \cdot_{V_AV_A} r$, para certo $a \in A$ e certo $r \in V_A$, pelo que o número de nódulos de r é inferior ao número de nódulos de p e, por indução, obtemos

$$\begin{aligned}\beta(p' \cdot_{V_AV_A} q) &= f(a) \cdot_{VV} \beta(r \cdot_{V_AV_A} q)\\ &= f(a) \cdot_{VV} \beta(r) \cdot_{VV} \beta(q)\\ &= \beta(a\Box \cdot_{V_AV_A} r) \cdot_{VV} \beta(q)\\ &= \beta(p') \cdot_{VV} \beta(q).\end{aligned}$$

Logo

$$\begin{aligned}\beta(p \cdot_{V_A V_A} q) &= \alpha(s) +_{HV} (\beta(p') \cdot_{VV} \beta(q)) +_{VH} \alpha(t) \\ &= (\alpha(s) +_{HV} \beta(p') +_{VH} \alpha(t)) \cdot_{VV} \beta(q) \\ &= \beta(p) \cdot_{VV} \beta(q).\end{aligned}$$

Analogamente se prova por indução no número de nódulos de p que $\alpha(p \cdot_{V_A H_A} t) = \beta(p) \cdot_{VH} \alpha(t)$, para todos os $p \in V_A$ e $t \in H_A$.

Da definição de β resulta que, para qualquer $t \in H_A$,

$$\begin{aligned}\beta(t +_{H_A V_A} 1) &= \beta(t +_{H_A V_A} 1 +_{V_A H_A} 0) \\ &= \alpha(t) +_{HV} \beta(1) +_{VH} \alpha(0) \\ &= \alpha(t) +_{HV} 1 +_{VH} 0 \\ &= \alpha(t) +_{HV} 1\end{aligned}$$

e, analogamente, $\beta(1 +_{V_A H_A} t) = 1 +_{VH} \alpha(t)$.

Pela Proposição 4.1.4 podemos concluir agora que (α, β) é um morfismo. □

Chamamos *álgebra-floresta livre sobre o alfabeto* A à álgebra-floresta

$$((H_A, +_{H_A H_A}, 0), (V_A, \cdot_{V_A V_A}, 1), +_{H_A V_A}, +_{V_A H_A}, \cdot_{V_A H_A})$$

e vamos usualmente denotá-la, simplesmente, por (H_A, V_A).

Utilizando esta noção algébrica, definimos *linguagem de A-florestas* como um sendo um subconjunto de H_A.

4.1.4 Autómato de Florestas

Um *autómato de florestas determinista e completo* ou, simplesmente, *autómato de florestas* é um quádruplo $\mathcal{A} = ((Q, +, q_0), A, \delta, F)$ onde:

- $(Q, +, q_0)$ é um monoide, aos elementos de Q chamamos estados e à operação $+$ chamamos composição de estados;
- A é um alfabeto;

- δ é uma função, a que chamamos *função transição*, que a cada elemento $a \in A$ faz corresponder uma função $\delta_a : Q \longrightarrow Q$;

- $F \subseteq Q$, cujos elementos são chamados *estados finais*.

Dizemos que um *autómato de florestas é finito* se o conjunto dos seus estados for finito.

4.2 Reconhecimento de Linguagens de Floresta

Ao longo desta secção vamos considerar um alfabeto A e uma linguagem de A-florestas L ($\subseteq H_A$). Vamos continuar a seguir [12] e [2].

Reconhecimento por Autómato de Florestas

Sejam t uma A-floresta e $\mathcal{A} = ((Q, +, q_0), A, \delta, F)$ um autómato de florestas. Definimos *ciclo do autómato de florestas $\mathcal{A}$ sobre a floresta t* como sendo o elemento de Q, que denotamos por $t^{\mathcal{A}}$, definido recursivamente da seguinte maneira:

- se t for a floresta vazia, então $t^{\mathcal{A}} = q_0$;

- se $t = a\Box \cdot s$, com $a \in A$ e $s \in H_A$, então $t^{\mathcal{A}}$ é definido como sendo $\delta_a(s^{\mathcal{A}})$; em particular, se t for uma floresta com um único nódulo e esse nódulo tiver etiqueta a, então $t^{\mathcal{A}}$ é $\delta_a(q_0)$;

- se $t = t_1 + \ldots + t_n$, com $t_1 \ldots, t_n \in H_A$, então $t^{\mathcal{A}}$ é definido como sendo $t_1^{\mathcal{A}} + \ldots + t_n^{\mathcal{A}}$; observe-se que a operação $+$ na última expressão é feita em $(Q, +, q_0)$.

Como as operações + em H_A e + em Q são associativas, $t^{\mathcal{A}}$ está bem definido.

Seja t uma A-floresta e $\mathcal{A}$ um autómato de A-florestas. A maneira como neste trabalho obtemos o ciclo do autómato $\mathcal{A}$ sobre a floresta t é semelhante à que foi descrita para ciclos de autómatos de árvores de aridade na Secção 3.2. Temos apenas de considerar uma ação adicional: quando todas as raízes da floresta estiverem aplicadas em estados, obtemos $t^{\mathcal{A}}$ como sendo o resultado da composição de todos esses estados pela operação + do respetivo monoide. Também a representação geométrica que escolhemos para o ciclo do autómato $\mathcal{A}$ sobre a floresta t é semelhante à que adotámos para representar geometricamente um ciclo de um autómato de árvores de aridade sobre uma árvore de aridade. Existe mais uma vez uma diferença: é obviamente necessário indicar adicionalmente o estado correspondente a $t^{\mathcal{A}}$.

Exemplo 4.2.1.
Consideremos o alfabeto $A = \{a, b, c\}$ e o autómato de florestas

$$\mathcal{A} = ((Q, +, q_0), A, \delta, F)$$

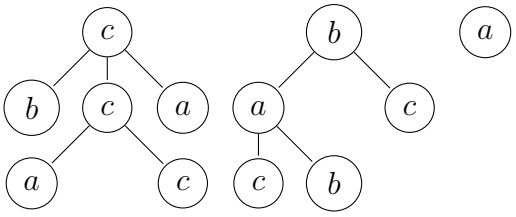
onde:

- *$Q = \{q_0, q_1, q_2, q_3\}$ e a operação + é definida por $q_i + q_j = q_j + q_i = q_{max\{i,j\}}$ onde i e j são elementos arbitrários de $\{0, 1, 2, 3\}$;*

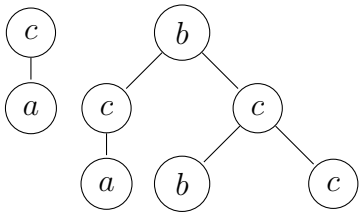
- *a função δ é a definida por $\delta_a(q_0) = q_0$, $\delta_b(q_0) = q_0$, $\delta_c(q_0) = q_1$, $\delta_a(q_1) = q_1$, $\delta_b(q_1) = q_1$, $\delta_c(q_1) = q_2$, $\delta_a(q_2) = q_2$, $\delta_b(q_2) = q_2$, $\delta_c(q_2) = q_3$, $\delta_a(q_3) = q_3$, $\delta_b(q_3) = q_3$ e $\delta_c(q_3) = q_3$;*

- *$F = \{q_0, q_1, q_2\}$.*

Consideremos as A-florestas t_1 e t_2 que geometricamente se representam, respetivamente, da seguinte maneira:

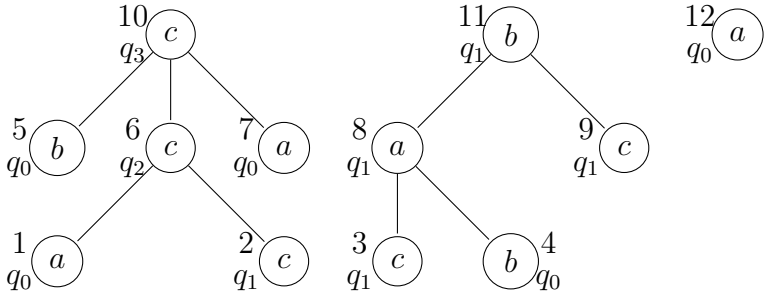


e

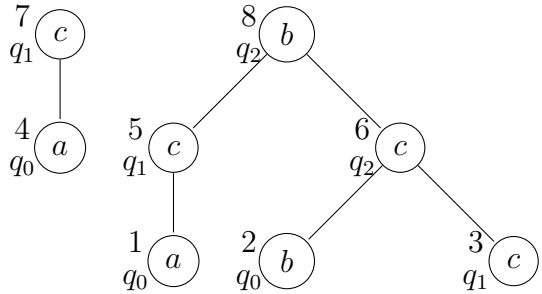


.

Temos $t_1^{\mathcal{A}} = q_3$ *pois*



e $q_3 + q_1 + q_0 = q_3$. *Temos* $t_2^{\mathcal{A}} = q_2$ *pois*



e $q_1 + q_2 = q_2$.

Sejam t uma A-floresta e $\mathcal{A} = ((Q, +, q_0), A, \delta, F)$ um autómato de A-florestas. Dizemos que *o autómato* $\mathcal{A}$ *é bem sucedido sobre a floresta* t ou que *a floresta* t *é reconhecida pelo autómato* $\mathcal{A}$ se e só se $t^{\mathcal{A}} \in F$.

Exemplo 4.2.2.

Consideremos o Exemplo 4.2.1. Constatamos que:

- *o autómato $\mathcal{A}$ não é bem sucedido sobre a floresta t_1, logo, a floresta t_1 não é reconhecida pelo autómato $\mathcal{A}$;*
- *o autómato $\mathcal{A}$ é bem sucedido sobre a floresta t_2, logo, a floresta t_2 é reconhecida pelo autómato $\mathcal{A}$.*

Seja $\mathcal{A}$ um autómato de A-florestas. Definimos

$$L(\mathcal{A}) = \{t \in H_A \mid t \text{ é reconhecida por } \mathcal{A}\},$$

a que chamamos *linguagem de A-florestas reconhecida por $\mathcal{A}$*.

Reconhecimento Algébrico

Se (H, V) for uma álgebra-floresta e $(\alpha, \beta) : (H_A, V_A) \longrightarrow (H, V)$ for um morfismo de álgebras-floresta, dizemos que a linguagem de A-florestas L é *reconhecida pelo morfismo* (α, β) se $L = \alpha^{-1}(G)$, para algum $G \subseteq H$; equivalentemente, se $L = \alpha^{-1}(\alpha(L))$. Dizemos que (H, V) *reconhece* L se existir algum morfismo de álgebras-floresta $(\alpha, \beta) : (H_A, V_A) \longrightarrow (H, V)$ que reconhece L.

Álgebra-Floresta Sintática

Definimos de seguida duas relações de equivalência associadas a L, uma em H_A e outra em V_A, ambas representadas por $\sim_L$ (ver [12, 2]).

- Em H_A: $s \sim_L t$ se e só se, para todo o $p \in V_A$, $ps \in L \Leftrightarrow pt \in L$.
- Em V_A: $p \sim_L q$ se e só se, para todo o $s \in H_A$, $ps \in L \Leftrightarrow qs \in L$.

Proposição 4.2.3.
Ambas as relações de equivalência $\sim_L$ são compatíveis com as operações da álgebra-floresta livre (H_A, V_A), isto é, o par ordenado constituído pela relação $\sim_L$ em H_A e pela relação $\sim_L$ em V_A é uma relação de congruência na álgebra-floresta livre (H_A, V_A).

Demonstração. Primeiro vamos mostrar que a relação de equivalência $\sim_L$ em H_A é compatível para a concatenação de duas A-florestas, isto é, que respeita a operação $+_{H_A,H_A}$. Apenas vamos demonstrar para a concatenação à direita. Vamos mostrar que, se $s \sim_L s'$ para $s, s' \in H_A$, então $s+t \sim_L s'+t$, para qualquer $t \in H_A$. Utilizando a definição de $\sim_L$, temos de mostrar que para qualquer A-contexto p se verifica: $p \cdot (s+t) \in L$ se e só se $p \cdot (s'+t) \in L$. Seja $q = p \cdot (\Box + t)$. Temos $q \cdot s = p \cdot (s + t)$ e $q \cdot s' = p \cdot (s' + t)$. Como consequência:

$$p \cdot (s+t) \in L \text{ se e só se } q \cdot s \in L \text{ se e só se } q \cdot s' \in L \text{ se e só se } p \cdot (s'+t) \in L,$$

A prova para a concatenação à esquerda é semelhante.

Vamos agora mostrar que a relação de equivalência $\sim_L$ em V_A é compatível com a composição de dois A-contextos, isto é, que respeita a operação $\cdot_{V_A,V_A}$. Apenas vamos demonstrar para a composição à direita. Vamos mostrar que, se $p \sim_L p'$, para $p, p' \in V_A$, então $p \cdot q \sim_L p' \cdot q$, para qualquer $q \in V_A$. Utilizando a definição de $\sim_l$, temos de mostrar que para qualquer A-floresta s se verifica: $(p \cdot q) \cdot s \in L$ se e só se $(p' \cdot q) \cdot s \in L$, o que é equivalente a ver que $p \cdot q \cdot s \in L$ se e só se $p' \cdot q \cdot s \in L$. A equivalência vem imediatamente da definição de $\sim_L$ e de considerarmos $q \cdot s$ como uma A-floresta aplicada nos A-contextos p e p'. A prova para a composição à esquerda é semelhante.

De maneira análoga pode mostrar-se que as relações $\sim_L$ são compatíveis com as operações $+_{H_A,V_A}$, $+_{V_A,H_A}$, $\cdot_{V_A,H_A}$, isto é, respetivamente:

- para quaisquer $s, s' \in H_A$ e $p, p' \in V_A$ tais que $s \sim_L s'$ e $p \sim_L p'$, então $s + q \sim_L s' + q$ e $t + p \sim_L t + p'$, para quaisquer $t \in H_A$ e $q \in V_A$;
- para quaisquer $s, s' \in H_A$ e $p, p' \in V_A$ tais que $s \sim_L s'$ e $p \sim_L p'$, então $q + s \sim_L q + s'$ e $p + t \sim_L p' + t$, para quaisquer $t \in H_A$ e $q \in V_A$;
- para quaisquer $s, s' \in H_A$ e $p, p' \in V_A$ tais que $s \sim_L s'$ e $p \sim_L p'$, então $s \cdot q \sim s' \cdot q$ e $t \cdot p \sim_L t \cdot p'$, para quaisquer $t \in H_A$ e $q \in V_A$;

□

Com o objetivo de sermos coerentes com a notação utilizada na bibliografia seguida, se não existir ambiguidade, denotamos a relação de congruência referida no Lema 4.2.3 simplesmente por $\sim_L$ e chamamos-lhe *congruência sintática de* L.

Definimos *álgebra-floresta sintática de L* como sendo o quociente da álgebra-floresta livre pela relação de congruência $\sim_L$ e denotamo-la por $(H_A, V_A)/\sim_L$. O morfismo que aplica cada A-floresta ou A-contexto na sua classe de equivalência é chamado *morfismo de álgebras-floresta sintático* e denotamo-lo por (α_L, β_L).

Proposição 4.2.4.
A linguagem de A-florestas L é reconhecida pelo seu morfismo de álgebras-floresta sintático (α_L, β_L). Mais ainda, uma álgebra-floresta (H, V) reconhece L se e só se (H_L, V_L) divide (H, V).

Demonstração. Temos $L = \alpha_L^{-1}(\alpha_L(L))$. Logo (α_L, β_L) reconhece L.

Suponhamos que L é reconhecida por uma álgebra-floresta finita (H, V). Então existe um morfismo $(\alpha, \beta) : (H_A, V_A) \to (H, V)$ tal que $L = \alpha^{-1}(G)$ para certo $G \subseteq H$. Tomemos $H' = \alpha(H_A)$ e $V' = \beta(V_A)$. Claramente, (H', V') é uma subálgebra-floresta de (H, V). Consideremos o morfismo $(\alpha', \beta') : (H_A, V_A) \to (H', V')$ definido por $\alpha'(t) = \alpha(t)$ e $\beta'(p) = \beta(p)$, o qual também reconhece L. Mostra-se facilmente que, para quaisquer $s, t \in H_A$ e $p, q \in V_A$,

- $\alpha(s) = \alpha(t) \Rightarrow \alpha_L(s) = \alpha_L(t)$.
- $\beta(p) = \beta(q) \Rightarrow \beta_L(p) = \beta_L(q)$.

Então ficam bem definidas funções

$$\alpha'' : H' \to H_L \qquad \text{e} \qquad \beta'' : V' \to V_L$$

por $\alpha''(\alpha'(s)) = \alpha_L(s)$ e $\beta''(\beta'(p)) = \beta_L(p)$. É claro que

$$(\alpha'', \beta'') : (H', V') \to (H_L, V_L)$$

é um morfismo sobrejetivo. Logo (H_L, V_L) divide (H, V).

Agora suponhamos que (H_L, V_L) divide (H, V). Existe então uma subálgebra-floresta (H', V') de (H, V) e um morfismo sobrejetivo

$$(\alpha, \beta) : (H', V') \to (H_L, V_L).$$

Para cada $a \in A$, existe $p_a \in V'$ tal que $\beta(p_a) = \beta_L(a\Box)$ porque β é sobrejetiva. Pela Proposição 4.1.8, existe um morfismo

$$(\alpha', \beta') : (H_A, V_A) \to (H', V')$$

tal que $\beta'(a\Box) = p_a$. Para cada $a \in A$, temos $(\beta \circ \beta')(a\Box) = \beta(\beta'(a\Box)) = \beta(p_a) = \beta_L(a\Box)$. Logo $(\alpha, \beta) \circ (\alpha', \beta') = (\alpha_L, \beta_L)$, pela Proposição 4.1.8. Consequentemente, $L = \alpha_L^{-1}(\alpha_L(L)) = (\alpha \circ \alpha')^{-1}(\alpha_L(L)) = \alpha'^{-1}(\alpha^{-1}(\alpha_L(L))$, pelo que (H', V'), e portanto (H, V), reconhece L. □

Notamos que em geral uma álgebra-floresta sintática pode ser infinita. No entanto, pela proposição anterior, se uma linguagem de A-florestas é reconhecida por alguma álgebra-floresta finita sobre o alfabeto A, então a sua álgebra-floresta sintática é finita. Mostra-se que a álgebra-floresta sintática de uma linguagem de A-florestas L reconhecível pode ser construída a partir de um morfismo sobrejetivo de álgebras-floresta que reconheça L.

Teorema de Equivalência de Reconhecimentos de Linguagens de A-Florestas

Teorema 4.2.5.
Seja $L \subseteq H_A$. Então as seguintes afirmações são equivalentes:

i) L é reconhecida por um autómato de A-florestas finito.

ii) L é reconhecida por uma álgebra-floresta finita.

iii) L tem a álgebra-floresta sintática finita.

Demonstração. A equivalência entre *ii)* e *iii)* vem da Proposição 4.2.4.

Suponhamos que L é uma linguagem de A-florestas reconhecida por um morfismo de álgebras-floresta $(\alpha, \beta) : (H_A, V_A) \longrightarrow (H, V)$, com (H, V) álgebra-floresta finita. Logo $L = \alpha^{-1}(F)$ para algum $F \subseteq H$. Vejamos que L é reconhecida por um autómato de florestas finito. Seja $\mathcal{A} = (H, A, \delta, F)$, onde δ é a função que a cada elemento $a \in A$ faz corresponder uma função $\delta_a : H \longrightarrow H$ onde $\delta_a(h) = \beta(a\Box) \cdot_{VH} h$. Pode mostrar-se por indução no número de nódulos da floresta que $t^{\mathcal{A}} = \alpha(t)$. Então $\mathcal{A}$ reconhece a linguagem de A-florestas L.

Para a outra implicação, suponhamos que temos um autómato de florestas $\mathcal{A} = ((Q, +, q_0), A, \delta, F)$ finito que reconhece L. É fácil ver que obtemos uma álgebra-floresta (H, V) definindo:

- H é o monoide $(Q, +, q_0)$;
- $V = \{f \mid f : Q \longrightarrow Q$ é uma função$\}$ munido da operação de composição;
- para quaisquer $q \in Q$ e $f \in V$,
  - $q +_{HV} f : Q \to Q$ definida por $(q +_{HV} f)(p) = q +_{HH} f(p)$;
  - $f +_{VH} q : Q \to Q$ definida por $(f +_{VH} q)(p) = f(p) +_{HH} q$;
  - $f \cdot_{VH} q = f(q)$.

Consideremos agora o único morfismo $(\alpha, \beta) : (H_A, V_A) \longrightarrow (H, V)$ tal que, para $a \in A$, $\beta(a\Box) = \delta_a$. Observamos que cada δ_a é uma função de Q em Q. Mostra-se facilmente, por indução no número de nódulos de t, que $t^{\mathcal{A}} = \alpha(t)$, para todo o $t \in H_A$. Por conseguinte, $L = \alpha^{-1}(F)$ e, portanto, (α, β) reconhece L. □

Com este resultado podemos, a partir de agora, falar apenas de *linguagem de A-florestas reconhecível* para nos referirmos a uma linguagem de A-florestas que satisfaça qualquer uma das condições do teorema.

4.2.1 Álgebras-Floresta e Linguagens de Árvores Sem Aridade

Ao longo desta subsecção vamos considerar um alfabeto A e uma linguagem de A-árvores sem aridade L. Notamos que $L \subseteq H_A$, pois uma A-árvore é um caso particular de A-floresta.

Seja $a \in A$. Consideremos a A-árvore constituída por um único nódulo de etiqueta a. Neste trabalho, tal como já foi afirmado, denotamos usualmente esta A-árvore da mesma maneira que a letra a, mas no que se segue, para evitar ambiguidades, vamos representá-la por a'. O a'-quociente, denotado por $a'^{-1}L$, é o conjunto das A-florestas t que satisfazem $at \in L$. Seja

$$(\alpha, \beta) : (H_A, V_A) \longrightarrow (H, V)$$

um morfismo. Dizemos que *a linguagem de A-árvores L é árvores-reconhecida por* (α, β) se e só se, para toda a A-árvore constituída por um único nódulo de etiqueta a, a linguagem de A-florestas $a'^{-1}L$ é reconhecida por (α, β).

Exemplo 4.2.6.

Seja $a \in A$. Consideremos a linguagem L como sendo a linguagem de A-florestas constituída unicamente por A-árvores cuja etiqueta do nódulo raiz é a. A linguagem L é árvores-reconhecida por qualquer álgebra-floresta. Isto acontece porque, para todo o $b \in A$, o quociente $b'^{-1}L$ é vazio (quando $b \neq a$) ou é L (quando $b = a$).

Existe uma alternativa para a definição de árvores-reconhecimento. Na definição alternativa, dizemos que *a linguagem de A-árvores L é árvores-reconhecida pela álgebra-floresta* (H, V) se existe uma linguagem de A-florestas K reconhecida pela álgebra-floresta (H, V) tal que L é a interseção de K com o conjunto das A-árvores. No entanto verifica-se que esta definição é menos vantajosa que a anterior.

4.3 Outras Possíveis Variantes para Álgebra-Floresta

Para palavras sobre um alfabeto A, podemos utilizar monoides ou semigrupos para reconhecer as linguagens de palavras sobre o alfabeto A. No primeiro caso, as linguagens apropriadas são da forma $L \subseteq A^*$, enquanto no segundo caso, que não permite a palavra vazia, apenas linguagens $L \subseteq A^+$ são permitidas.

Para florestas existe um número de possibilidades muito maior. Para além de trabalharmos com objetos de dois tipos (florestas e contextos) em vez de apenas um (palavras), esses tipos são mais complexos. Podemos considerar que a floresta vazia não é uma floresta e que o contexto vazio não é um

contexto. Também podemos considerar outras restrições sobre a posição do buraco nos contextos, por exemplo, não permitir que o buraco tenha irmãos ou não permitir que apareça num nódulo raiz. Cada combinação de opções para as hipóteses anteriores, desde que os axiomas corretos sejam formulados, dá origem a uma respetiva definição de álgebra-floresta.

As escolhas seguidas neste trabalho não são reivindicadas como sendo superiores. As outras opções são igualmente viáveis mas isto não significa que sejam todas equivalentes. As diferenças ficam visíveis quando tentamos caracterizar álgebras-floresta através de equações na álgebra-floresta livre. Por exemplo, a equação $v \cdot h = v \cdot g$ com as nossas considerações implica $h = g$ porque esta equação deve ser válida para todas as atribuições de elementos do monoide V a v e, em particular, podemos atribuir o contexto identidade à variável v. Mas nesse caso obtemos $h = g$, que diz que o monoide horizontal é trivial. Se não permitirmos contextos com buraco em nódulos raiz, esta equação irá descrever linguagens floresta onde a pertença de determinada floresta nessa linguagem depende apenas das etiquetas dos seus nódulos raiz.

Podemos também perguntar o que aconteceria se não considerássemos a estrutura vertical. Poderíamos trabalhar com pares (H, Z) onde Z é apenas um conjunto (e não um semigrupo), mas onde ainda teríamos uma operação de substituição de florestas em contextos semelhante à considerada na nossa conjetura. Tais pares corresponderiam a autómatos onde o alfabeto não está fixo. Para tais objetos pode-se dispensar os axiomas presentes na definição de álgebra-floresta que estabelecem a estrutura vertical de semigrupo (que aqui não estaria presente). Toda a teoria podia ser desenvolvida com estas hipóteses mas mais uma vez as equações teriam aqui significados diferentes. Em particular não teríamos maneira de nos referirmos explicitamente à composição vertical. Estas considerações não foram tomadas por se acreditar importante a estrutura vertical de semigrupo.

4.4 Aplicações

Vamos apresentar, de forma resumida, várias caracterizações de linguagens de florestas sobre um alfabeto (algumas das quais são, em particular,

linguagens de árvores sobre um alfabeto) reconhecíveis, privilegiando, por isso, as ideias intuitivas dos conceitos e das demonstrações (quando apresentadas) em detrimento do rigor.

Ao longo desta secção vamos considerar um alfabeto A, a álgebra-floresta livre (H_A, V_A), uma linguagem $L \subseteq H_A$ (notamos que L pode ser exclusivamente constituída por A-árvores) e uma álgebra-floresta (H, V). Quando utilizarmos parâmetros $a, b, s, t, p, q, h, g, v$ e w (eventualmente com índices) e não especificarmos explicitamente as suas naturezas, estamos a assumir que:

- $a, b \in A$,
- $s, t \in H_A$,
- $p, q \in V_A$,
- $h, g \in H$ e
- $v, w \in V$.

As caracterizações de linguagens de A-florestas que vamos apresentar são todos elas *efetivas*, no sentido em que as condições na álgebras-floresta livre podem ser efetivamente testadas. Nelas vamos habitualmente utilizar equações na álgebra-floresta livre (que, intuitivamente, vão exprimir propriedades da álgebra-floresta). As equações são responsáveis pela efetividade das caracterizações apresentadas no sentido que ilustramos de seguida. Tomemos por exemplo a equação $pq = qp$. Para sabermos se esta equação é satisfeita por uma linguagem de A-florestas reconhecível não podemos testar todos os A-contextos p e q, pois existem em número infinito. A solução é observarmos que apenas temos de testar um A-contexto em cada classe de equivalência da congruência sintática de L.

4.4.1 Aplicações Simples

Nesta subsecção vamos seguir o trabalho feito em [4] e apresentar três exemplos de linguagens reconhecíveis de A-árvores cujas caracterizações são expressas por equações na álgebra-floresta livre. Com estes três exemplos

vamos também introduzir conceitos que irão aparecer posteriormente em caracterizações efetivas de linguagens reconhecíveis de florestas. O primeiro exemplo é o da classe das linguagens que apenas depende do conjunto das etiquetas que ocorre em cada árvore; este exemplo é caracterizado por duas equações simples. No segundo exemplo introduzimos um expoente ω nas equações. No terceiro exemplo utilizamos uma classe de linguagens que não é fechada para as operações booleanas.

Nos exemplos seguintes consideramos sempre que a linguagem L é exclusivamente constituída por A-árvores e é reconhecível.

Linguagens de A-Árvores Testável Por Etiquetas

Dizemos que L *é testável por etiquetas* se e só se a pertença de uma determinada A-árvore a L depender apenas do conjunto constituído pelas suas etiquetas, isto é, se e só se L for uma combinação booleana de linguagens da forma "A-árvores que têm algum nódulo a-etiquetado".

Teorema 4.4.1.
Seja L uma linguagem de A-árvores reconhecível. Então L é testável por etiquetas se e só se satisfaz as equações:

- $p \cdot q = q \cdot p \quad e$

- $p \cdot p = p.$

Ideia da demonstração. A parte "só se" é clara uma vez que, para quaisquer A-contextos, é óbvio que temos o mesmo conjunto de etiquetas em ambos os lados de cada equação.

Para a parte "se" iremos mostrar que ambas as equações presentes no teorema implicam que cada A-árvore t é equivalente a uma A-árvore na *forma normal*, no sentido em que apenas depende do conjunto das suas etiquetas (exemplificamos de seguida o queremos dizer por isto). O primeiro passo para chegarmos à forma normal é mostrarmos que toda a A-floresta é equivalente a uma A-floresta da forma $a_1 + \ldots + a_n$, onde, como usualmente, cada a_i é a A-árvore que contém apenas um nódulo e que tem etiqueta a_i. A prova é por indução na profundidade da floresta utilizando

$$at = (a\Box) \cdot (\Box + t) \cdot 0 = (\Box + t) \cdot (a\Box) \cdot 0 = a + t$$

De uma maneira semelhante, pode mostrar-se que a raiz de uma árvore pode ser trocada por qualquer outro nódulo. Uma vez que $p \cdot q = q \cdot p$ implica $s + t = t + s$ (tomando $p = \Box + s$ e $q = \Box + t$), podemos agora concluir que as etiquetas podem ser ordenadas de qualquer maneira. Finalmente podemos remover as etiquetas que estejam em duplicado utilizando $p \cdot p = p$. □

Linguagens de A-Árvores Definite

Este exemplo é o das linguagens de A-árvores *definite*. Dizemos que L *é definite* se existir algum $k \in \mathbb{N}$ tal que a pertença de uma determinada A-árvore a L dependa apenas dos nódulos que tenham no máximo profundidade k. Notamos que o facto das árvores que consideramos serem sobre um alfabeto sem aridade implica que uma linguagem que satisfaça a anterior definição possa, para cada profundidade, possuir uma infinidade de árvores. Por exemplo, a linguagem constituída pelas A-árvores cuja raiz tem um número par de filhos é *definite*.

Teorema 4.4.2.
Seja L uma linguagem de A-árvores reconhecível. Então L é definite *se e só se satisfaz a equação*

$$p^\omega \cdot 0 = p^\omega \cdot t \quad \textit{onde } p \neq 1.$$

A caracterização anterior apresenta um novo interveniente nas equações, o expoente ω. Informalmente, o expoente ω deveria ser substituído por "um número suficientemente grande". Por exemplo, a equação presente no teorema anterior diz que se um contexto for repetido (multiplicado por ele próprio em V_A) um número suficientemente grande de vezes, então o buraco está a uma grande profundidade dentro da A-árvore e, como tal, é irrelevante.

Notamos que ω também funciona para a concatenação de A-florestas, pois se $t \in H_A$, podemos definir t^ω como sendo $(t + 1)^\omega \cdot 0$.

Ideia da demonstração. Se a linguagem de A-árvores é *definite*, então a equação do teorema tem de ser satisfeita. Caso contrário, para algum A-contexto q e para qualquer $k \in \mathbb{N}$, apenas uma das A-árvores $q \cdot p^{\omega k} \cdot 0$ e $q \cdot p^{\omega k} \cdot t$, que são semelhantes até à profundidade k, irá pertencer à linguagem de A-árvores.

Para a outra implicação, tomamos uma A-árvore que satisfaça a equação do teorema. Vamos mostrar que para uma profundidade suficientemente grande, digamos $k \in \mathbb{N}$, qualquer modificação com profundidade maior que k nessa A-árvore não afeta a sua inclusão na linguagem. Para isso necessitamos mostrar que se p é um A-contexto que tem o buraco a uma profundidade pelo menos k, então $p \cdot t = p \cdot 0$ vai verificar-se para qualquer A-floresta t. Seja p um A-contexto com o buraco a uma profundidade pelo menos $k \in \mathbb{N}$. Se conseguirmos decompor este A-contexto na forma $p = q_1 \cdot q^{\omega} \cdot q_2$, então o resultado pretendido virá da equação do teorema. Como p tem o buraco a uma profundidade pelo menos k, existe uma decomposição $p = p_1 \cdots p_k$ onde nenhum dos A-contextos p_i tem o buraco na raiz. Uma vez que existe um número finito de classes de equivalência para a congruência sintática, $\sim_L$, pelo teorema de Ramsey pode provar-se que se k é suficientemente grande, então existem alguns $i < j < l$ em $1, \ldots, k$ com

$$p_1 \cdots p_{j-1} \sim_L p_j \cdots p_{l-1} \sim_L p_1 \cdots p_{l-1}$$

Seja $q = p_1 \cdots p_{j-1}$. Pelo visto em cima, temos

$$q \sim_L p_1 \ldots p_{l-1} = q.p_{j-1} \ldots p_{l-1} \sim_L q \cdot q$$

Em particular, temos $q = q^{\omega}$. □

Linguagens de A-Árvores Finita

O último exemplo envolve linguagens de A-árvores com um número finito de elementos. Neste caso não podemos esperar caracterizações dadas por equações, pois a classe destas linguagens não é fechada para a complementação. A razão porque tal acontece é o facto de qualquer linguagem reconhecível de A-árvores ter a mesma congruência sintática que a sua linguagem complementar e, como tal, também satisfazer as mesmas equações.

Teorema 4.4.3.
Seja L uma linguagem de A-árvores reconhecível. Então L é finita se e só se não possuir nenhum elemento da forma $q \cdot p^\omega \cdot s$, onde $p \neq 1$.

A demonstração deste teorema segue as mesmas linhas do Teorema 4.4.2.

4.4.2 Procurando Padrões em A-Árvores

Nesta subsecção continuamos a seguir [4] e damos caracterizações efetivas para mais três classes de linguagens de A-árvores. Em cada caso, a classe da linguagem é dada por combinações booleanas de sentenças que testam se uma árvore sobre um alfabeto contém um dado padrão. Os padrões discutidos são: uma subárvore de uma árvore sobre um alfabeto, um pedaço de uma árvore sobre um alfabeto e um caminho maximal de uma árvore sobre um alfabeto.

Linguagens de A-Árvores Testáveis Pela Fronteira

Definimos *n-fronteira de uma A-árvore* (onde $n \in \mathbb{N}$) como sendo o conjunto constituído pelas suas subárvores que têm no máximo n nódulos. Uma linguagem de A-árvores reconhecível diz-se *testável pela fronteira* se, para algum $n \in \mathbb{N}$, a pertença de uma A-árvore à linguagem dependa apenas da sua n-fronteira. Notamos que não contamos as subárvores que estão na numa n-fronteira, apenas testamos se pertencem ou não a essa n-fronteira. Uma definição equivalente é: uma linguagem reconhecível de A-árvore é testável pela fronteira se e só se for uma combinação booleana finita de linguagens da forma "A-árvores que têm s como uma subárvore".

Teorema 4.4.4.
Seja L uma linguagem de A-árvores reconhecível. Então L é testável pela fronteira se e só se satisfaz as equações:

i) $a(s + p^\omega \cdot t) = s + p^\omega \cdot t$;

ii) $p^\omega \cdot t + s + u = u + s + p^\omega \cdot t \quad e$

iii) $p^\omega \cdot t + s = p^\omega \cdot t + s + s;$

onde $a \in A,\ s, t, u \in H_A$ *e* $p \in V_A \setminus \{1\}$.

Ideia da demonstração. A primeira equação diz que um nódulo de etiqueta a com muitos descendentes (uma floresta $p^\omega \cdot t$ tem muitos nódulos) pode ser removido sem que as n-fronteiras sejam afetadas. A segunda equação diz que a ordem das A-árvores numa A-floresta com muitos nódulos não é importante. A terceira equação diz que cópias de A-árvores podem ser adicionadas ou retiradas a qualquer A-floresta com muitos nódulos. A chave da demonstração é mostrar que, para qualquer $n \in \mathbb{N}$ fixado, quaisquer duas A-florestas com a mesma fronteira podem ser transformadas uma na outra aplicando sucessivamente as equações do teorema. □

Linguagens de A-Árvores Testáveis por Pedaços

Dizemos que *uma A-floresta s é um pedaço de uma A-floresta t*, denotando esta relação por $s \preceq t$, se s puder ser obtida a partir de t através da remoção de nódulos, isto é, se existir uma função injetiva dos nódulos de s para os nódulos de t que preserve as relações $descendant^s$ e, para todo o $a \in A$, lab^s_a. Uma linguagem de A-árvores reconhecível diz-se *testável por pedaços* se for uma combinação booleana finita de linguagens da forma "A-árvores que contêm a A-floresta s como um pedaço".

Teorema 4.4.5.
Seja L uma linguagem de A-árvores reconhecível. Então L é testável por pedaços se e só se satisfaz a equação:

$$q \cdot p^\omega = p^\omega = p^\omega \cdot q \quad \textit{para qualquer} \quad q \preceq p.$$

Ideia da demonstração. A ideia na implicação "só se" é que, para qualquer $n \in \mathbb{N}$ fixado, os A-contextos $q \cdot p^\omega$, p^ω e $p^\omega \cdot q$ tenham todos os

mesmos pedaços de tamanho n e possam assim ser trocados entre eles. A demonstração da implicação "se" requer um argumento combinatório complicado. □

Linguagens de A-Árvores Testáveis Por Caminhos

Neste exemplo a ideia é a de, a cada caminho maximal de uma A-árvore, associar a sequência de etiquetas desse caminho ordenada da raiz para a folha.

Uma linguagem de A-árvores reconhecível diz-se *testável por caminhos* se é uma combinação booleana finita de linguagens da forma "A-árvores onde a sequência de etiquetas que ocorre nalgum caminho maximal da A-árvore pertence à linguagem reconhecível de palavras $K \subseteq A^*$".

Por exemplo, A linguagem de A-árvores constituída pelas A-árvores que têm algum nódulo com etiqueta a é testável por caminhos uma vez que uma A-árvore pertence a esta linguagem se e só se a sequência de etiquetas que ocorre nalgum caminho maximal da A-árvore pertence à linguagem A^*aA^*. No entanto, a linguagem constituída pelas A-florestas que têm pelo menos dois nódulos com etiqueta a não é testável por caminhos: as A-florestas a e $a + a$ têm ambas os mesmos caminhos maximais mas apenas uma pertence à linguagem considerada.

Teorema 4.4.6.
Seja L uma linguagem de A-árvores reconhecível. Então L é testável por caminhos se e só se satisfaz as equações:

i) $s + t = t + s$;

ii) $s + s = s$;

iii) $p \cdot (s + t) = p \cdot s + p \cdot t$.

Ideia da demonstração. A implicação "só se" é óbvia, pois ambos os lados de cada equação têm obviamente os mesmos caminhos maximais. Para

a implicação "se", basta mostrar que, utilizando as equações do teorema, quaisquer duas A-árvores que tenham os mesmos caminhos maximais podem ser transformadas uma na outra. □

4.4.3 Linguagens de A-Florestas Testáveis por Etiqueta e Linguagens de A-florestas Definíveis por uma Fórmula Σ_1

Nesta subsecção vamos voltar a seguir [12] e apresentar dois exemplos de caracterizações de linguagens de florestas sobre um alfabeto. A primeira caracterização, de linguagens floresta sobre um alfabeto testáveis por etiquetas, é semelhante à de linguagens de árvores sobre um alfabeto já falada e ilustra de que maneira uma propriedade no monoide dos contextos pode ter implicações importantes no monoide das florestas. A segunda caracterização, de linguagens floresta sobre um alfabeto definíveis por uma fórmula Σ_1, mostra que também podemos considerar linguagens floresta que não são fechadas para operações booleanas.

Linguagens de A-Florestas Testáveis por Etiquetas

Dizemos que uma linguagem de A-florestas reconhecível é *testável por etiquetas* se e só se a pertença de uma determinada floresta a essa linguagem depender apenas do conjunto constituído pelas suas etiquetas, isto é, se e só se for uma combinação booleana de linguagens da forma "A-florestas que têm algum nódulo a-etiquetado".

Teorema 4.4.7.
Seja L uma linguagem de A-florestas reconhecível. Então L é testável por etiquetas se e só se satisfaz as equações:

i) $p \cdot q = q \cdot p \quad e$

- $p \cdot p = p$.

Ideia da demonstração. A parte "só se" é clara uma vez que, para quaisquer A-contextos, é óbvio que obtemos o mesmo conjunto de etiquetas em ambos os lados de cada equação. Demonstremos agora a parte "se". Seja L uma linguagem de A-florestas reconhecida por um morfismo $(\alpha, \beta) : (H_A, V_A) \longrightarrow (H, V)$, com a álgebra-floresta do conjunto de chegada a satisfazer ambas as equações do teorema. Vamos mostrar que, para cada a A-floresta t da linguagem L, a sua imagem $\alpha(t)$ depende apenas do conjunto de etiquetas de t.

Começamos por mostrar que as duas equações do teorema implicam outras três.

A primeira é a idempotência do monoide horizontal:

$$h + h = h$$

Esta equação tem de se verificar em qualquer álgebra-floresta que satisfaça as nossas premissas por causa do seguinte argumento que utiliza a idempotência do monoide vertical:

$$h + h = (t + 1)(h + 1) \cdot 0 = (h + 1) \cdot 0 = h.$$

A segunda é a comutatividade do monoide horizontal:

$$h + g = g + h$$

O argumento utiliza a comutatividade do monoide vertical:

$$h + g = (h + 1) \cdot (g + 1)0 = (g + 1) \cdot (h + 1) \cdot 0 = g + h.$$

Finalmente, temos uma equação que nos permite "achatar" as árvores:

$$vh = h + v0.$$

A demonstração utiliza outra vez a comutatividade do monoide vertical:

$$vh = v \cdot (h + 1) \cdot 0 = (h + 1) \cdot v \cdot 0 = h + v0.$$

Vamos mostrar que utilizando as equações anteriores, toda a A-floresta t tem a mesma imagem através de α como uma A-floresta na forma normal $a_1 + \cdots + a_n$, onde, como usualmente, cada a_i é a A-árvore que contém apenas um nódulo e que tem etiqueta a_i. Para além disso, as etiquetas $a_1, \ldots, a_n$

são exatamente as etiquetas utilizadas em t, distribuídas sem repetição e por uma ordem arbitrária no conjunto A. Começando pela forma normal, podemos inicialmente utilizar a idempotência para "criar" tantas cópias de cada etiqueta quantas as que apareçam na A-árvore t. Então, utilizando a última equação e a comutatividade do monoide horizontal, podemos reconstruir a A-árvore começando pelas folhas e avançando até à raiz. □

Se omitirmos a equação $p \cdot p = p$, obtemos linguagens que podem ser definidas por combinações booleanas de linguagens da forma "a etiqueta a ocorre pelo menos k vezes", ou "o número de ocorrências da etiqueta a é $k \mod n$".

Linguagens de A-Florestas Definíveis através de uma Fórmula Σ_1

Uma *fórmula* Σ_1 é uma fórmula de Lógica de Primeira Ordem com assinatura $\{descendant\} \cup \{lab_a \mid a \in A\}$, onde *descendant* tem aridade 2 e, para todo $a \in A$, lab_a tem aridade 1, e onde apenas podem aparecer quantificadores existenciais na quantificação de uma fórmula na forma normal prenexa que lhe seja equivalente.

Consideremos uma A-floresta t e a assinatura

$$\{descendant\} \cup \{lab_a \mid a \in A\}$$

onde *descendant* tem aridade 2 e, para todo o $a \in A$, lab_a tem aridade 1. O terno $(dom(t), \sigma, I)$ onde

- $\sigma = \{descendant\} \cup \{lab_a \mid a \in A\}$ e
- I é a função definida por

  $I(descendant) = descendant^t$ e, para todo o $a \in A$, $I(lab_a) = lab_a^t$;

é uma estrutura e vamos denotá-la por $\underline{t}$.

Seja φ uma fórmula Σ_1. A *linguagem de A-florestas definida por φ* é o conjunto

$$\{t \in H_A \mid \text{existe uma atribuição de variáveis } \mu \text{ tal que } (\underline{t}, \mu) \models \varphi\}$$

e denotamo-la por $L(\varphi)$. Dizemos que *a linguagem L é definida através de uma fórmula* Σ_1 se, para a assinatura $\{descendant\} \cup \{lab_a \mid a \in A\}$, existir uma fórmula Σ_1, digamos φ, tal que $L = L(\varphi)$.

O seguinte resultado, que apresentamos sem qualquer ideia da demonstração, mostra que linguagens de A-florestas podem ser definidas através de uma fórmula Σ_1:

Teorema 4.4.8.
Seja L uma linguagem de A-florestas reconhecível e seja (α, β) o seu morfismo de álgebras-floresta sintático.

A linguagem L é definível por uma fórmula Σ_1 se e só se $vh \in \alpha(L)$ implica $vwh \in \alpha(L)$.

As várias relações que definimos para árvores e florestas, *filho de*, ..., *nódulo à esquerda de*, $depth_n^t$, lab_a^t e ch_i^t, são muito úteis para caracterizar classes importantes de linguagens de florestas através de fórmulas da Lógica, como ilustra o teorema anterior.

Apêndice A

Lógica

A.1 Lógica de Primeira Ordem

Existem duas partes essenciais em Lógica de Primeira Ordem: a parte sintática e a parte semântica. Intuitivamente, a sintaxe determina que sequências de símbolos são expressões aceites em Lógica de Primeira Ordem enquanto que a semântica determina o significado dessas expressões.

Sintaxe de Lógica Primeira Ordem

Alfabeto de Lógica Primeira Ordem

Um *alfabeto de Lógica de Primeira Ordem* é um conjunto $A = A_1 \dot{\cup} A_2$, onde A_1 é o *conjunto dos símbolos lógicos de Lógica Primeira Ordem* (intuitivamente, podemos considerá-los como sendo os símbolos que têm sempre o mesmo significado, ou seja, os seus significados não dependem de nenhuma

interpretação) e A_2 é o *conjunto dos símbolos não-lógicos de Lógica de Primeira Ordem* (intuitivamente, podemos considerá-los como sendo os símbolos cujos significados dependem de uma interpretação).

Omitiremos neste Apêndice A.1 os nomes "Lógica de Primeira Ordem" sempre que mencionarmos elementos que a ela se refiram. Por exemplo, no que se segue, vamos designar símbolos lógicos de Lógica de Primeira Ordem apenas por símbolos lógicos.

- **Símbolos Lógicos**

  O *conjunto dos símbolos lógicos* é constituído por elementos que pertençam a um dos seguintes conjuntos:

  1. Um conjunto infinito de *símbolos de variáveis* ou, simplesmente, *variáveis*, habitualmente representadas por $x, y, z, \ldots$ (possivelmente com índices).
  2. Um conjunto infinito de *símbolos de parâmetros* ou, simplesmente, *parâmetros*, habitualmente representados por $\alpha, \beta, \gamma, \ldots$ (possivelmente com índices).
  3. Dois símbolos: o *quantificador universal* $\forall$ e o *quantificador existencial* $\exists$.
  4. Os *símbolos das conexões lógicas*: $\wedge$ para a *conjunção*, $\vee$ para a *disjunção*, $\neg$ para a *negação*, $\Rightarrow$ para a *implicação* e $\Leftrightarrow$ para a *equivalência*.
  5. Um *símbolo de igualdade* $=$.
  6. Os *símbolos de parêntesis* () e, eventualmente, outros símbolos de pontuação.

Por vezes são também utilizados adicionalmente os símbolos V ou 1 (para *verdade*) e F ou 0 (para *não verdadeiro* ou *falso*) e, nesse caso, são também símbolos lógicos.

- **Símbolos Não-Lógicos**

  O *conjunto dos símbolos não-lógicos* é constituído por elementos que pertençam a um dos seguintes conjuntos:

1. Um conjunto de *símbolos de predicados* ou *símbolos de relações*, em que cada símbolo tem associado um número inteiro não negativo chamado *aridade*. Os símbolos de predicados são habitualmente representados por $P, Q, R, \ldots$ (possivelmente com índices).
2. Um conjunto de *símbolos de funções*, em que cada símbolo tem associado um número inteiro não negativo chamado *aridade*. Os símbolos de funções são habitualmente representados por $f, g, h, \ldots$ (possivelmente com índices).
3. Um conjunto de *símbolos de constantes* ou, simplesmente, *constantes*, habitualmente representados por $a, b, c, \ldots$ (possivelmente com índices).

Notamos que os símbolos de constantes podem ser interpretados como sendo símbolos de funções com aridade zero. Notamos também que o símbolo lógico de igualdade pode ser interpretado como sendo um símbolo não-lógico de predicado com aridade 2 mas, atendendo ao tratamento especial que normalmente lhe é atribuído, é aqui considerado como sendo um símbolo lógico.

Assinatura de Lógica de Primeira Ordem

Dado um conjunto não vazio D, podemos considerar os símbolos não-lógicos como sendo relações, operações e constantes em D com a mesma aridade (por exemplo, um símbolo de predicado de aridade 2 pode ser considerado em D como sendo uma relação binária em D e um símbolo de função de aridade 3 pode ser considerado como sendo uma função D^3 em D). Consoante o conjunto D e as relações, as operações e as constantes que tivermos em mente, assim são escolhidos os símbolos não-lógicos. A *assinatura de uma Lógica de Primeira Ordem* é o conjunto formado por esses símbolos não-lógicos e as respetivas aridades. A assinatura pode ser vazia, finita ou infinita.

Gramática de Lógica de Primeira Ordem

Há duas espécies de expressões *bem formadas* que nos interessa considerar e que vão constituir a nossa *gramática de Lógica de Primeira Ordem*:

os *termos* (intuitivamente, designações) e as *fórmulas* (intuitivamente, proposições).

- **Termos**

  O conjunto dos *termos bem formados* ou, simplesmente, *termos* é definido por recorrência da seguinte maneira:

  1. **Variáveis**:
     Qualquer variável é um termo.
  2. **Funções**:
     Se f é um símbolo de função de aridade n e $t_1, \ldots, t_n$ são termos, então $f(t_1, \ldots, t_n)$ é um termo.

Apenas expressões que se possam obter através de um número finito de aplicações das regras 1 e 2 são termos. Em particular, uma expressão que envolva um símbolo de predicado não é um termo.

- **Fórmulas**

  O conjunto das *fórmulas bem formadas* ou, simplesmente, *fórmulas* é definido por recorrência da seguinte maneira:

  1. **Símbolos de predicados**:
     Se P é um símbolo de predicado de aridade n e $t_1, \ldots, t_n$ são termos, então $P(t_1, \ldots, t_n)$ é uma fórmula.
  2. **Igualdade**:
     Se t_1 e t_2 são termos, então $t_1 = t_2$ é uma fórmula.
  3. **Negação**:
     Se φ é uma fórmula, então $\neg\varphi$ é uma fórmula.
  4. **Conexões binárias**:
     Se φ e ψ são fórmulas, então $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \Rightarrow \psi$ e $\varphi \Leftrightarrow \psi$ são fórmulas.
  5. **Quantificadores**:
     Se φ é uma fórmula e x é uma variável, então $\forall x\varphi$ e $\exists x\varphi$ são fórmulas.

Apenas expressões que se possam obter através de um número finito de aplicações das regras 1, 2, 3, 4 e 5 são fórmulas. As fórmulas obtidas através de um número finito de aplicações das regras 1 e 2 dizem-se *fórmulas atómicas*.

Numa fórmula da forma $\forall x\varphi$ ou da forma $\exists x\varphi$, φ é o *alcance* ou *alçada* do quantificador em x respetivo.

Unicidade de Representação e Convenções Notacionais

Notamos que o papel dos símbolos de parêntesis em Lógica de Primeira Ordem é o de evitar ambiguidades na leitura das fórmulas, as quais se escrevem de uma e uma só maneira como sequência de símbolos lógicos e não-lógicos. A esta propriedade chamamos *unicidade de representação*. Algumas convenções foram desenvolvidas acerca da precedência dos operadores lógicos de maneira a, em alguns casos, conseguirmos evitar o uso de parêntesis. Uma convenção comum é:

1. $\neg$ é considerado primeiro;
2. $\wedge$ e $\vee$ são considerados a seguir;
3. quantificadores são considerados a seguir;
4. $\Rightarrow$ e $\Leftrightarrow$ são considerados em último.

Variáveis Livres e Mudas

Numa fórmula, uma variável pode ocorrer *livre* ou *muda*. Intuitivamente, uma variável tem uma ocorrência livre se essa ocorrência não for quantificada e tem uma ocorrência muda caso contrário.

As *ocorrências livres* de uma variável x são definidas por recorrência da seguinte maneira:

1. **Fórmulas Atómicas**:

   Se φ é uma fórmula atómica, então todas as ocorrências de x em φ são livres.

2. **Negação**:

   As ocorrências livres de x em $\neg\varphi$ são as ocorrências livres de x em φ.

3. **Conexões Binárias**:

   As ocorrências livres de x em $\varphi \wedge \psi$ (respetivamente, $\varphi \vee \psi$, $\varphi \Rightarrow \psi$, $\varphi \Leftrightarrow \psi$) são as ocorrências livres de x em φ e as ocorrências livres de x em ψ.

4. **Quantificadores**:

   As ocorrências livres de x em $\forall y\varphi$ (respetivamente, $\exists y\varphi$), onde $y \neq x$, são as ocorrências livres de x em φ.

   Em $\forall x\varphi$ (respetivamente, $\exists x\varphi$) não existem ocorrências livres de x.

As ocorrências de x numa fórmula que não sejam livres são chamadas *mudas*. Uma variável diz-se *livre numa fórmula* se tiver alguma ocorrência livre nessa fórmula. As fórmulas sem variáveis livres dizem-se *sentenças*. São as sentenças que vão ter um valor lógico de verdade bem definido. Notamos que a notação $\varphi(x_1, \ldots, x_n)$ representa uma fórmula onde, no máximo, ocorrem as variáveis livres $x_1, \ldots, x_n$.

Semântica de Lógica de Primeira Ordem

Tal como já foi dito anteriormente, queremos agora com a parte semântica determinar o valor de verdade de sequências de símbolos construídas segundo os critérios estabelecidos na parte sintática, ou seja, fórmulas. Intuitivamente, dada uma fórmula e dado um conjunto não vazio a que vamos chamar universo, atribuindo elementos desse universo a variáveis, relações definidas no universo a predicados com a mesma aridade e funções a símbolos de função com a mesma aridade, queremos chegar a um valor de verdade.

Estrutura

Uma *estrutura* é um terno $\mathcal{D} = (D, \sigma, I)$ onde:

1. D é um conjunto não vazio a que chamamos *universo* ou *domínio do discurso*;
2. σ é uma assinatura;
3. I é uma função que indica a maneira como a assinatura é considerada no universo em questão e a que chamamos *função interpretação*. A função interpretação I atribui predicados e funções aos símbolos não-lógicos da assinatura da seguinte maneira:
   - A cada símbolo de predicado P de aridade n é atribuído uma relação n-ária em $I(P)$.
   - A cada símbolo de função f de aridade n é atribuído uma função $I(f)$ de D^n em D. Em particular, a cada símbolo de constante a da assinatura de primeira ordem é atribuído um elemento do universo, $I(a)$.

Atribuição de Variáveis

Uma *atribuição de variáveis* é uma função que a cada variável atribui um elemento do universo. A razão pela qual é necessária uma atribuição de variáveis é a de dar sentido a fórmulas com variáveis livres.

Uma atribuição de variáveis μ pode ser estendida a todos os termos da linguagem com o resultado de cada termo ser aplicado um único elemento do universo. As seguintes regras são utilizadas para fazer esta extensão:

1. **Variáveis**:

   Cada variável x é aplicada em $\mu(x)$ e dizemos que $\mu(x)$ *é atribuído a* x.

2. **Funções**:

   Dados termos $t_1, \ldots, t_n$ que tenham sido aplicados em elementos $d_1, \ldots, d_n$ do universo e um símbolo de função f de aridade n, o termo

$f(t_1, \ldots, t_n)$ é aplicado em $(I(f))(d_1, \ldots, d_n)$ e dizemos que $(I(f))(d_1, \ldots, d_n)$ *é atribuído a* $f(t_1, \ldots, t_n)$.

Avaliação

Para uma estrutura $\mathcal{D} = (D, \sigma, I)$ e uma atribuição de variáveis μ, uma sentença é *avaliada* através das seguintes *regras de satisfação*:

1. **Fórmulas atómicas (1)**:

   Se for do tipo $P(t_1, \ldots, t_n)$, onde P é um símbolo de predicado de primeira ordem de aridade n e $t_1, \ldots, t_n$ são termos, é avaliada verdadeira caso $(\mu(t_1), \ldots, \mu(t_n))$ esteja em $I(P)$.

2. **Fórmulas atómicas (2)**:

   Se for do tipo $t_1 = t_2$, onde t_1 e t_2 são termos de primeira ordem, é avaliada verdadeira caso $\mu(t_1)$ e $\mu(t_2)$ sejam o mesmo objeto do universo.

3. **Conexões lógicas**:

   Se for do tipo $\varphi \wedge \psi$, é avaliada verdadeira caso φ e ψ sejam ambas verdadeiras.

   Se for do tipo $\varphi \vee \psi$, é avaliada verdadeira caso φ seja verdadeira ou caso ψ seja verdadeira.

   Se for do tipo $\neg\varphi$, é avaliada verdadeira caso φ seja não verdadeira.

   Se for do tipo $\varphi \Rightarrow \psi$, é avaliada verdadeira caso φ seja não verdadeira ou caso ψ seja verdadeira.

   Se for do tipo $\varphi \Leftrightarrow \psi$, é avaliada verdadeira caso φ e ψ sejam ambas verdadeiras ou sejam ambas não verdadeiras.

4. **Quantificadores universais**:

   Se for do tipo $\forall x \varphi(x)$, é verdadeira caso qualquer atribuição de variáveis μ' que apenas difira de μ no máximo no valor atribuído a x faça com que φ seja verdadeira para a estrutura $\mathcal{D}$ e a atribuição de variáveis μ'.

Esta definição formal expressa a ideia de que $\forall x\varphi(x)$ é verdadeira se qualquer valor atribuído a x faz com que $\varphi(x)$ seja verdadeira.

5. **Quantificadores existenciais**:

   Se for do tipo $\exists x\varphi(x)$, é verdadeira caso exista uma atribuição de variáveis μ' que apenas difira de μ no máximo no valor atribuído a x tal que φ seja verdadeira para a estrutura $\mathcal{D}$ e a atribuição de variáveis μ'. Esta definição formal expressa a ideia de que $\exists x\varphi(x)$ é verdadeira se e só se existe um valor do universo que atribuído a x faz com que $\varphi(x)$ seja verdadeira.

Se nenhum destes casos se verificar, a sentença de primeira ordem é avaliada *não verdadeira* ou *falsa*.

Validação, Satisfação e Consequência Lógica

Sejam $\mathcal{D} = (D, \sigma, I)$ uma estrutura e μ uma atribuição de variáveis.

A um par ordenado da forma $(\mathcal{D}, \mu)$ damos o nome de *interpretação de Lógica de Primeira Ordem*.

Se uma sentença φ é avaliada verdadeira numa interpretação $(\mathcal{D}, \mu)$, dizemos que $(\mathcal{D}, \mu)$ *satisfaz* φ, escrevendo $(\mathcal{D}, \mu) \models \varphi$. Uma *sentença é satisfazível* se existir alguma interpretação para a qual seja verdadeira.

Dar uma definição de fórmula com variáveis livres satisfazível é mais complicado, porque uma interpretação por si só não determina o valor de verdade de tal fórmula. A convenção mais comum é que uma *fórmula com variáveis livres é satisfazível por uma interpretação* se a fórmula for verdadeira independentemente dos elementos do universo atribuídos às variáveis livres.

Uma *fórmula é válida logicamente* ou, simplesmente, *válida* se for verdadeira dada qualquer interpretação.

Uma *fórmula φ é uma consequência lógica da fórmula ψ* se toda a interpretação que torna ψ verdadeira também torna φ verdadeira. Nesse caso dizemos que *φ é implicada logicamente por ψ*.

A.2 Lógica Monádica de Segunda Ordem

Neste trabalho, a utilização que vamos fazer de Lógica de Segunda Ordem resume-se a um seu caso particular, a Lógica Monádica de Segunda Ordem. Notamos que, intuitivamente, Lógica de Segunda Ordem é uma extensão de Lógica de Primeira Ordem onde, para cada $n \in \mathbb{N}$, são adicionalmente permitidas variáveis que representam predicados de aridade n e Lógica Monádica de Segunda Ordem é uma extensão de Lógica de Primeira Ordem onde são apenas permitidas variáveis que representam predicados de aridade 1.

Vamos apenas referir aqui os conteúdos de Lógica Monádica de Segunda Ordem que não foram enunciados em Lógica de Primeira Ordem, assumindo que tudo o resto decorre de igual maneira ou que será introduzido, conforme for sendo necessário, no decorrer do trabalho.

Tal como em Lógica de Primeira Ordem, existem duas partes essenciais em Lógica de Monádica Segunda Ordem: a parte sintática e a parte semântica.

Sintaxe de Lógica Monádica de Segunda Ordem

Alfabeto de Lógica Monádica de Segunda Ordem

Estendemos o alfabeto de Lógica de Primeira Ordem A introduzindo-lhe:

- **Símbolos Lógicos**

  1. O *símbolo de pertence* $\in$.
     Notamos que a inclusão deste símbolo é opcional.
  2. Um conjunto de *símbolos de variáveis relacionais de Lógica Monádica de Segunda Ordem* ou, simplesmente, *variáveis relacionais de Lógica de Monádica Segunda Ordem*, em que cada variável representa um predicado de aridade 1. Notamos que uma variável deste tipo é usualmente interpretada como representante de um conjunto de elementos do universo.

- **Símbolos Não-Lógicos**

  Tal como acontece em Lógica de Primeira Ordem, a Lógica Monádica de Segunda Ordem pode incluir no seu alfabeto símbolos não-lógicos. No entanto, estes símbolos estão restringidos: todos os termos formados por símbolos não-lógicos têm de ser termos de Lógica de Primeira Ordem (que podem ser substituídos por uma variável de Lógica de Primeira Ordem) ou termos de Lógica Monádica de Segunda Ordem (que podem ser substituídos por uma variável de Lógica Monádica de Segunda Ordem).

Assinatura de Lógica Monádica de Segunda Ordem

A assinatura de Lógica Monádica de Segunda Ordem é idêntica à da Lógica de Primeira Ordem.

Gramática de Lógica Monádica de Segunda Ordem

Estendemos a gramática de Lógica de Primeira Ordem introduzindo as regras seguintes:

1. Se X é uma variável de Lógica Monádica de Segunda Ordem e t é um termo, então $t \in X$ é uma fórmula.

2. Se X é uma variável de Lógica Monádica de Segunda Ordem e φ é uma fórmula, então $\forall X \varphi$ e $\exists X \varphi$ são fórmulas.

Apenas expressões que se possam obter através de um número finito de aplicações das regras presentes em gramática de Lógica de Primeira Ordem e das duas regras anteriores são fórmulas de Lógica Monádica de Segunda Ordem.

Semântica de Lógica Monádica de Segunda Ordem

Estrutura e Atribuição de Variáveis de Lógica Monádica de Segunda Ordem

A semântica de Lógica Monádica de Segunda Ordem pode ser facilmente obtida através da extensão da semântica de Lógica de Primeira Ordem. Ela é definida através de uma estrutura $\mathcal{D} = (D, \sigma, I)$ de Lógica de Primeira Ordem e de uma *atribuição de variáveis de Lógica Monádica de Segunda Ordem* μ'. Esta última não é mais que uma atribuição de variáveis de Lógica de Primeira Ordem estendida de maneira a abranger as variáveis de Lógica Monádica de Segunda Ordem: a cada termo faz corresponder um elemento do universo D e a cada variável de Lógica Monádica de Segunda Ordem faz corresponder um subconjunto do universo D.

Avaliação em Lógica Monádica de Segunda Ordem

Para uma estrutura $\mathcal{D} = (D, \sigma, I)$ e uma atribuição de variáveis de Lógica Monádica de Segunda Ordem μ, uma sentença de Lógica Monádica de Segunda Ordem é avaliada por um conjunto de regras de satisfação semelhante ao de Lógica de Primeira Ordem mas que inclui adicionalmente as seguintes regras:

1. **Quantificadores universais em variáveis de segunda ordem**.

   Se for do tipo $\forall X\varphi$, é verdadeira caso qualquer atribuição de variáveis de Lógica Monádica de Segunda Ordem μ' que apenas difira de μ no máximo no conjunto atribuído a X faça com que φ seja verdadeira para a estrutura $\mathcal{D}$ e a atribuição de variáveis de Lógica Monádica de Segunda Ordem μ'. Esta definição formal expressa a ideia de que $\forall X\varphi$ é verdadeira se qualquer subconjunto do universo atribuído a X faz com que φ seja verdadeira.

2. **Quantificadores existenciais em variáveis de segunda ordem**.

   Se for do tipo $\exists X\varphi$, é verdadeira caso exista uma atribuição de variáveis de Lógica Monádica de Segunda Ordem μ' que apenas difira de μ no máximo no conjunto atribuído a X tal que φ seja verdadeira para a estrutura $\mathcal{D}$ e a atribuição de variáveis de Lógica Monádica de Segunda

Ordem μ'. Esta definição formal expressa a ideia de que $\exists X\varphi$ é verdadeira se e só se existe um subconjunto do universo que atribuído a X faz com que φ seja verdadeira.

Se nenhum destes casos ou dos casos descritos nas regras de satisfação de Lógica de Primeira Ordem se verificar, a sentença de Lógica Monádica de Segunda Ordem é avaliada *não verdadeira* ou *falsa*.

Validação, Satisfação e Consequência Lógica de Lógica Monádica de Segunda Ordem

Estes conceitos definem-se analogamente aos da Lógica de Primeira Ordem.

Bibliografia

[1] Nicolas Bedon, Alexis Bès, Olivier Carton, and Chloe Rispal. Logic and rational languages of words indexed by linear orderings. *Theory Comput. Syst.*, 46(4):737–760, 2010.

[2] Mikolaj Bojanczyk. Algebra for trees. *AutomathA Handbook*, 2001.

[3] Mikolaj Bojanczyk. Forest expressions. In *CSL*, pages 146–160, 2007.

[4] Mikolaj Bojanczyk. Effective characterizations of tree logics. In *PODS*, pages 53–66, 2008.

[5] Mikolaj Bojanczyk. Tree-walking automata. In Carlos Martín-Vide, Friedrich Otto, and Henning Fernau, editors, *Language and Automata Theory and Applications, Second International Conference, LATA 2008, Tarragona, Spain, March 13-19, 2008. Revised Papers*, volume 5196 of *Lecture Notes in Computer Science*, pages 1–2. Springer, 2008.

[6] Mikolaj Bojanczyk. Algebra for tree languages. In *CSL*, page 1, 2009.

[7] Mikolaj Bojanczyk and Thomas Colcombet. Tree-walking automata cannot be determinized. In *ICALP*, pages 246–256, 2004.

[8] Mikolaj Bojanczyk, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-variable logic on data trees and xml reasoning. *J. ACM*, 56(3), 2009.

[9] Mikolaj Bojanczyk and Luc Segoufin. Tree languages defined in first-order logic with one quantifier alternation. In *ICALP (2)*, pages 233–245, 2008.

[10] Mikolaj Bojanczyk, Luc Segoufin, and Howard Straubing. Piecewise testable tree languages. In *LICS*, pages 442–451, 2008.

[11] Mikolaj Bojanczyk, Howard Straubing, and Igor Walukiewicz. Wreath products of forest algebras, with applications to tree logics. In *LICS*, pages 255–263, 2009.

[12] Mikolaj Bojanczyk and Igor Walukiewicz. Forest algebras. In *Logic and Automata*, pages 107–132, 2008.

[13] S. Burris and H.P. Sankappanavar. *A course in universal algebra*. Graduate texts in mathematics. Springer-Verlag, 1981.

[14] Hubert Comon, Max Dauchet, Rémi Gilleron, Christof Löding, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. Tree automata techniques and applications, 2007.

[15] Samuel Eilenberg. *Automata, languages and machines*, volume A. Academic Press, 1974.

[16] John Mackintosh Howie. *Automata and languages*. Oxford science publications. Clarendon Press, 1991.

[17] Leonid Libkin. Logics for unranked trees: An overview. *Logical Methods in Computer Science*, 2(3), 2006.

[18] Bojanczyk Mikolaj and Thomas Colcombet. Tree-walking automata do not recognize all regular languages. In *STOC*, pages 234–243, 2005.

[19] Jean-Éric Pin. *Variétés de langages formels*. Masson, Paris, 1984.

[20] Jean-Éric Pin. Finite semigroups and recognizable languages: an introduction. In *NATO Advanced Study Institute Semigroups, Formal Languages and Groups*, pages 1–32. Kluwer academic publishers, 1995.

[21] Jean-Éric Pin. Logic, semigroups and automata on words. *Ann. Math. Artif. Intell.*, 16:343–384, 1996.

[22] Jean-Éric Pin. Syntactic semigroups. In G. Rozenberg and A. Salomaa, editors, *Handbook of formal languages*, volume 1, chapter 10, pages 679–746. Springer, 1997.

[23] Jacques Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.

[24] Michael Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997.

[25] Howard Straubing. *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhäuser, 1994.

[26] Wolfgang Thomas. Handbook of theoretical computer science (vol. b). chapter Automata on Infinite Objects, pages 133–191. MIT Press, Cambridge, MA, USA, 1990.

[27] Wolfgang Thomas. *Languages, Automata, and Logic*, volume 3, pages 389–455. Springer, 1996.

[28] Igor Walukiewicz. Finding your way in a forest: On different types of trees and their properties. In *FoSSaCS*, pages 1–4, 2008.

[29] Wolfgang Wechler. *Universal Algebra for Computer Scientists*, volume 25 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1992.

[30] Pascal Weil. Algebraic recognizability of languages. In *Mathematical Foundations of Computer Science 2004, 29th International Symposium, MFCS 2004, Prague, Czech Republic, August 22-27, 2004, Proceedings*, pages 149–175, 2004.