Andréia Borges Avelar

Formalização da Automação da Terminação Através de Grafos com Matrizes de Medida

Tese de Doutorado

Instituto de Ciências Exatas
Departamento de Matemática
Programa de Pós Graduação em Matemática

Brasília, agosto de 2014

Universidade de Brasília

Andréia Borges Avelar[1]

Formalização da Automação da Terminação Através de Grafos com Matrizes de Medida

Tese de Doutorado

Tese apresentada ao Programa de Pós-Graduação em Matemática do Departamento de Matemática, Instituto de Ciências Exatas da Universidade de Brasília, como requisito parcial para obtenção do título de Doutor em Matemática. Área de concentração: Ciências Exatas.

Orientador: Prof. Dr. Mauricio Ayala Rincón

Brasília, agosto de 2014

[1]A autora contou com o apoio financeiro do CNPq

Ficha catalográfica elaborada pela Biblioteca Central da Universidade de Brasília. Acervo 1018933.

Avelar, Andréia Borges.
A948f Formalização da automação da terminação através de grafos com matrizes de medida / Andréia Borges Avelar. -- 2014.
124 f. : il. ; 30 cm.

Tese (doutorado) - Universidade de Brasília, Instituto de Ciências Exatas, Departamento de Matemática, Programa de Pós-Graduação em Matemática, 2014.
Inclui bibliografia.
Orientação: Mauricio Ayala Rincón.

1. Teoria dos grafos. 2. Matrizes (Matemática). I. Rincón, Mauricio Ayala. II. Título.

CDU 519.17

Ao meu esposo, Eduardo Antonio.

Agradecimentos

Agradeço:

Ao meu orientador, Professor Mauricio Ayala Rincon pela paciência, pelos conselhos e pelo estímulo para a realização deste trabalho.

Aos professores que participaram da comissão examinadora.

Ao professor Cesar Muñoz, pelas valiosas contribuições e pelo apoio.

Ao CNPq e a UnB, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

As minhas amigas Thaynara e Kaliana, pelo companheirismo e apoio.

Aos meus pais, pela educação, atenção, apoio e carinho em todas as horas.

Ao meu esposo, pela sua compreensão e amor incondicional.

Aos meus colegas da UnB.

A todos os professores e funcionários do Departamento de Matemática pelos ensinamentos e pela ajuda.

A todos os amigos que de uma forma ou de outra me estimularam ou me ajudaram.

Formalização da Automação da Terminação Através de Grafos com Matrizes de Medida

RESUMO

Uma nova estrutura baseada em digrafos e denominada Grafos com Matrizes de Medida (MWGs) é apresentada. Esta estrutura se baseia na estrutura conhecida como Grafos de Contextos de Chamado (CCGs) e é aplicada na verificação de terminação de programas funcionais de primeira-ordem especificados no estilo da linguagem de especificação do assistente de prova PVS. Similarmente a CCGs, os vértices em um MWG correspondem aos chamados recursivos da função modelada. Caminhos em um MWG, assim como em um CCG, representam o fluxo de execução dos chamados recursivos do programa em questão. De acordo com o princípio de mudança de tamanho, MWGs são aplicados na verificação de terminação através da especificação de critérios para garantir que todos os circuitos no MWG, que corresponderiam a possíveis execuções infinitas de chamados recursivos, não podem ser repetidos infinitamente. Em CCGs, isto é realizado através da construção de combinação de medidas, sobre o domínio de uma relação bem-fundada, definidas nos parâmetros da função recursiva que está sendo modelada. Assim, a fim de verificar terminação da função, deve-se verificar que a combinação de medidas decresce estritamente em todos os possíveis circuitos. Ao invés de procurar uma combinação de medidas para cada circuito, em MWGs as arestas são rotuladas com matrizes quadradas cujas entradas expressam relações entre diferentes medidas aplicadas aos parâmetros formais e atuais. Desta forma, o comportamento das medidas após executar uma sequência de chamados recursivos associada a um caminho no MWG é expresso através da multiplicação especializada das matrizes que rotulam as arestas do caminho. O critério de terminação em MWG corresponde a uma simples noção de positividade da matrix associada a um determinado circuito. As matrizes de medida modelam o resultado das combinações de medidas da tecnologia CCG de uma forma muito elegante possibilitando a formulação de critérios simples de terminação em MWGs. Tais critérios baseiam-se na positividade dos ciclos (circuitos simples) do MWG, levando a positividade de qualquer circuito. Dois critérios de terminação para MWGs são formalizados. A correspondência entre terminação em CCGs e MWGs é formalizada. A especificação de uma simples linguagem funcional de primeira-ordem com sua respectiva semântica para terminação é apresentada, e são discutidos os elementos necessários para a formalização da correspondência entre a semântica de terminação para esta linguagem e terminação em MWGs.

Palavras-chave: Terminação; Digrafos com Peso; Linguagem Funcional de Primeira-Ordem; Grafos de Contextos de Chamado; Grafos com Matrizes de Medida.

Formalization of Automation of Termination Through Matrix-Weigthed Graphs

ABSTRACT

A new digraph structure called Matrix-Weigthed Graphs (MWGs) is presented. This structure is based on Calling Context Graphs (CCGs) and is applied to verify termination of first-order functional programs written in the style of the specification language of the PVS proof assistant restricted to its first-order fragment. Similarly to CCGs, a MWG has nodes that correspond to the recursive calls of the modeled program. Paths in a MWG, as well as in a GCC, model the execution flow of recursive calls in the associated program. According to the size-change termination principle, MWGs are used to verify termination of the specification through criteria that guarantee that all circuits in the MWG, would correspond to possible infinite executions of recursive calls, can not be repeated infinitely. In CCG, this is done by building well-founded measures for the parameters of the recursive functions that should be proved to strictly decreasing in all possible circuits. Instead searching a combination of measures for any circuit that guarantees the impossibility of infinite recursive calls as done in CCGs, MWGs edges are labelled with square matrices whose components express relations between different measures applied to the formal and actuals parameters. In this way, the behavior of measures after executing the recursive calls associated to a path in the MWG is expressed by an specialized multiplication of the matrices labeling the edges in the path. The termination criterion in the MWG corresponds to a simple notion of positivity of the matrix associated to a given circuit. The measure matrices model the results of combination of measures of the CCG's technology in a very elegant manner making possible the formulation of simple termination criteria in MWGs. Such criteria is based on the positivity of the cycles (simple circuits) of the MWG, leading to positivity of any circuit. Two termination criteria for MWG are formalized.The correspondence between the termination in CCGs and MWGs is formalized. The specification of a simple first-order language with its semantics for termination is presented and the necessary elements to formalize the correspondence between the semantics of termination of this language and termination in MWGs.

Keywords: Termination; Weighted-digraphs; First-Order Functional Language; Calling Context Graphs; Matrix-Weigthed Graphs.

Sumário

Capítulo 1

Introdução

Neste trabalho, é abordado o problema de terminação para programas funcionais de primeira-ordem. Análise de terminação é um tópico fundamental em ciência da computação, uma vez que correção total de um algoritmo depende de terminação. Mas o problema de terminação é indecidível e portanto a análise de terminação não pode funcionar corretamente em todos os casos.

O objetivo durante uma análise de terminação desenvolvida por um algoritmo (ou uma pessoa) consiste em verificar se a avaliação de um dado programa termina definitivamente, sempre que for possível. Desta forma, se o sucesso não é alcançado durante a análise, a resposta pode ser "talvez" ou "não sei" ou continua-se trabalhando infinitamente sem nunca dar uma resposta. Devido à importância do problema, novas técnicas para verificar terminação têm surgido. Essas técnicas têm se tornado muito úteis na prática, pois respondem "talvez" ou "não sei" em uma quantidade cada vez menor de casos, e são aplicadas na verificação de terminação (ou de não terminação) de problemas complexos, por exemplo na indústria em *drivers* para sistemas operacionais, em sistemas de reescrita de termos de primeira ordem e de ordem superior, em programas imperativos, etc. Novas técnicas para verificação automática de terminação têm sido disseminadas anualmente na *Competição Anual de Terminação*[1].

Um método simples, que tem sido aplicado por muito tempo para construir provas de terminação, envolve associar uma medida a cada passo de um algoritmo. Esta medida é também conhecida na literatura como *ranking function* e é definida sobre o domínio de uma relação bem fundada. Se a medida *decresce* de acordo com uma relação bem fundada ao longo de cada possível passo do algoritmo, então o algoritmo deve terminar, pois não podem existir sequências infinitas decrescentes com respeito a uma relação bem fundada.

[1] http://termination-portal.org/wiki/Termination_Competition

Estão presentes na literatura sobre terminação vários *surveys* discutindo técnicas para verificação de terminação. Por exemplo, em [DSD94] os autores abordam e comparam algumas técnicas para análise de terminação de programas lógicos; em [Der85] e [Der87], métodos para verificar terminação de sistemas de reescrita de termos são descritos; em [CPR11] o autor faz um apanhado dos avanços recentes em verificação de terminação.

Por ser um dos problemas mais antigos em ciência da computação, a análise de terminação tem recebido muita atenção da comunidade científica. Assim, muito trabalho no sentido de obter técnicas para verificação de terminação de programas tem sido desenvolvido, em especial para sistemas de reescrita de termos e programas lógicos. Por exemplo em [AG00] a abordagem desenvolvida para verificação de terminação é similar a que será descrita neste trabalho. Os autores apresentam um técnica para verificação de terminação de sistemas de reescrita de termos (TRS) baseada em digrafos com uma quase-ordem bem fundada. É definida a noção de *dependency-pairs*, que são pares de termos conectados por uma regra de redução do TRS. Em seguida é definida uma $\mathcal{R}$-*chain*, que são sequências de *dependency-pairs* conectados, via regra de redução para uma dada substituição. Desta forma um resultado onde diz que *"Um TRS é terminante se, e somente se, não existem* $\mathcal{R}$-*chains infinitas"* é estabelecido. Então os autores automatizam a técnica e desenvolvem um refinamento através de *dependency graphs*, que são digrafos cujos vértices são *dependency-pairs* e as arestas são $\mathcal{R}$-*chains*. E assim, terminação através de *dependency graphs* é estabelecida da seguinte maneira: *"Um TRS é terminante se, e somente se, existe uma quase-ordem* $\geqslant$ *bem fundada e fechada para substituições tal que:* i) $l \geqslant r$ *para toda regra* $l \to r$ *do TRS;* ii) $s \geqslant t$ *para todo dependency-pair* $\langle s, t\rangle$ *em um ciclo do dependency graph; e* iii) $s > t$ *para ao menos um dependency-pair em cada ciclo do dependency graph"*. Em [GTSKF06], a técnica descrita em [AG00] é implementada, gerando um ambiente automatizado para verificação de terminação de programas, denominado `AProVE`, que inicialmente era aplicado para verificação automática de terminação e provas de complexidade de sistemas de reescrita de termos, porém atualmente este sistemas suporta outros formalismos, como por exemplo programas imperativos, programas funcionais e programas lógicos.

A ferramenta acima mencionada, `AProVE`, dentre outras, por exemplo `KITTeL` [FKS11, KSZM09] $\texttt{T}_\texttt{T}\texttt{T}_\texttt{2}$, ambas para sistemas de reescrita de termos, têm se destacado e frequentemente estão presentes na competição de terminação, trazendo inovações e aperfeiçoamentos das técnicas implementadas que promovem maior poder de verificação para as ferramentas propostas, e onde novas ferramentas para verificação automática de terminação são apresentadas. A competição anual de terminação engloba três categorias

principais: i) Terminação de sistemas de reescrita de termos incluindo sistemas de reescrita de primeira ordem e de ordem superior; ii) Análise de complexidade de sistemas de reescrita; iii) Terminação de linguagens de programação.

A abordagem deste trabalho é baseada em uma técnica para verificação de terminação de programas funcionais denominada *o princípio de mudança de tamanho* (SCP), introduzida em [LJBA01]. Neste princípio é desenvolvida uma análise sobre *grafos de mudança de tamanho*, que representam as sequências de passos da execução de um programa. Uma estrutura de dados desenvolvida para implementar o SCP são *grafos de mudança de tamanho* (CCG), introduzida em [MV06], e aplicada para incorporar o SCP ao assistente de prova ACL2. Um CCG associado a uma função recursiva é um digrafo, onde cada vértice representa um chamado recursivo na definição da função, denominado contexto de chamado, e cada aresta é uma sequência bem formada de dois chamados. A noção de sequência de chamados bem formada é equivalente à possibilidade de que esta sequência ocorra durante uma computação da função recursiva, ou seja, existe uma designação de valores para os parâmetros formais da função, a partir da qual cada passo de computação corresponde a um elemento da sequência de chamados. Então uma família de medidas, definidas no conjunto de parâmetros formais sobre o domínio de uma relação bem fundada, é associada ao CCG. Terminação da função é condicionada à existência de uma combinação de medidas para cada circuito no digrafo, que leve a uma sequência decrescente relativamente à relação bem fundada.

Parte do desenvolvimento deste trabalho consistiu em especificar e formalizar no assistente de prova PVS [ORS92] as contribuições que serão descritas. Isto implica no desenvolvimento de bibliotecas em PVS. Assim, ao longo do texto sempre que se mencionar uma biblioteca em PVS será usada a palavra enfatizada "*teoria*" ou "*sub-teoria*". E o nome de uma *teoria* será escrito em destaque, como na expressão `nome-da-teoria`.

As principais contribuições desta tese são descritas a seguir, e também ilustradas na Figura 1.1, onde uma das propostas de trabalhos futuros também é destacada.

1. Desenvolvimento de uma nova estrutura de dados baseada em digrafos com pesos, denominada *Grafos com Matrizes de Medida* (MWG), para análise de terminação baseada no SCP.

   MWG's são digrafos com pesos que abstraem o comportamento de funções recursivas, e onde:

   - Os vértices representam chamados recursivos na definição da função e carregam informações sobre este chamado, como os parâmetros formais da função,

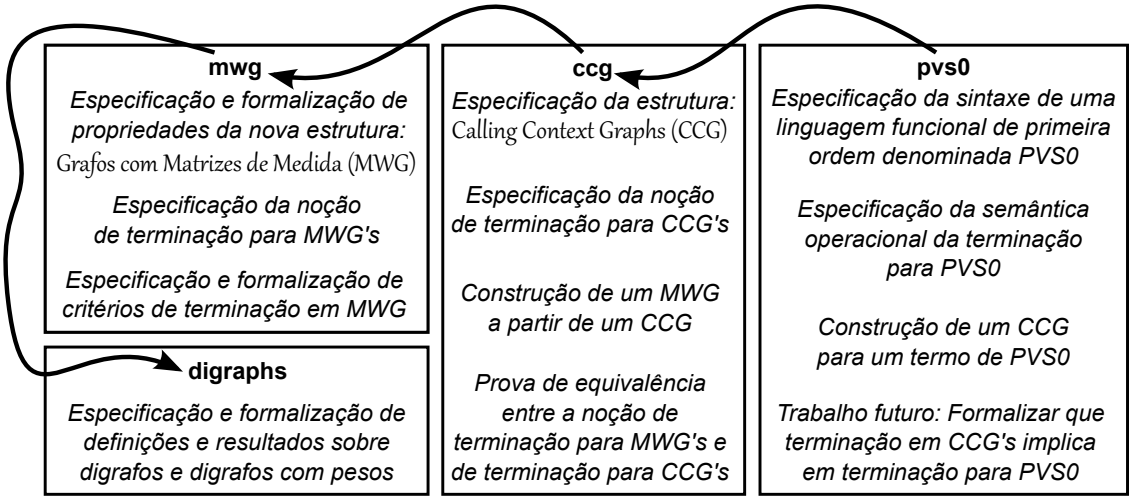


Figura 1.1: *Teorias* envolvidas na formalização descrita neste trabalho.

as condições (que são expressões booleanas que dependem dos parâmetros formais) necessárias para que o chamado ocorra, e os parâmetros atuais do chamado (que também são expressões que dependem dos parâmetros formais).

- As arestas representam a possibilidade de executar um chamado após o outro, dado que as condições de chamado são satisfeitas pelos seus parâmetros formais e que os parâmetros atuais deste chamado satisfazem as condições do chamado subsequente.
- Os rótulos das arestas são matrizes quadradas. Tais matrizes organizam os resultados de comparações de medidas, sobre o domínio de uma relação bem fundada, aplicadas aos parâmetros em cada chamado recursivo.

2. Especificação de CCG's com sua definição de terminação e formalização da equivalência entre terminação em CCG's e terminação em MWG's. Com este objetivo, especificou-se e formalizou-se definições e propriedades sobre digrafos e digrafos com pesos.

   A noção de terminação para MWG's equivale à noção de positividade das matrizes rotulando os vértices do digrafo. Além disso, neste trabalho é desenvolvida uma especificação de CCG's como digrafos com uma família de medidas associada, onde os vértices são contextos de chamados e as arestas uma sequência bem formada de dois contextos. A definição de terminação para CCG's é especificada e corresponde à existência de uma combinação de medidas, para cada circuito do CCG, que leve a uma sequência decrescente no domínio da relação bem fundada sobre o qual as medidas estão definidas. A equivalência entre a definição de terminação para MWG's

e terminação para CCG's é formalizada. Para se ter a base necessária ao desenvolvimento de MWG's e CCG's foram realizadas contribuições para a *teoria* sobre digrafos, existente em PVS.

3. Formalização de dois critérios de terminação para MWG's.

   Os critérios consistem em verificar que sob certas condições (baseadas na positividade das matrizes rotulando os ciclos do MWG) todos os circuitos do MWG tem uma matriz de medida positiva. As condições propostas são definidas sobre os ciclos e arestas do digrafo.

Assim, este trabalho está organizado da seguinte maneira: no Capítulo 2 o problema de terminação é abordado e apresenta-se uma prova da indecidibilidade do problema; no Capítulo 3, na Seção 3.1 é estabelecida a notação necessária ao restante do trabalho, assim como definições acerca de digrafos e digrafos com peso, na Seção 3.2 é apresentada a tecnologia CCG e, na Seção 3.3 é apresentado o assistente de prova PVS utilizado na formalização destacando alguns pontos de sua linguagem de especificação e poder dedutivo; nos capítulos 4 e 5, as contribuições deste trabalho são apresentadas; no Capítulo 4 a especificação da nova estrutura proposta, MWG's, é apresentada assim como a formalização dos critérios de terminação para MWG's; no Capítulo 5 é apresentada a especificação de CCG's e de terminação para CCG's, para então, ao final do capítulo, ser apresentada a formalização de equivalência entre terminação para CCG's e MWG's; no Capítulo 6, é apresentada a semântica de terminação para uma linguagem de primeira ordem, cuja sintaxe é brevemente apresentada na Seção 5.1, do capítulo anterior, e também são apresentadas a especificação de funções aplicadas para construir um CCG para termos desta linguagem e uma possível formalização de que terminação em CCG implica em semântica da terminação para esta linguagem é proposta.

Capítulo 2

O Problema de Terminação

Neste capítulo o problema de terminação é abordado. Uma prova de indecidiblidade do mesmo é apresentada na Seção 2.1 e uma breve exposição sobre a importância do problema é descrita na Seção 2.2.

2.1 Indecidibilidade do Problema de Terminação

O Problema de Terminação para programas, também conhecido como Problema da Parada, é um problema de decisão que pode ser definido da seguinte maneira:

Usando uma quantidade finita de tempo, decidir se um dado programa termina uma execução sobre uma entrada finita ou nunca termina.

A importância do Problema da Parada vem do fato de que este foi um dos primeiros problemas a ser provado indecidível. Em 1936, Alonzo Church apresenta uma prova [Chu36] de que não existe uma função computável que, dadas duas expressões em cálculo λ, decide se tais expressões são equivalentes ou não. No mesmo ano e de maneira independente, Alan Turing apresenta um trabalho [Tur36] onde ele define Máquinas de Turing, formula o Problema da Parada e mostra que este não tem solução.

Nesta seção apresenta-se uma prova da indecidibilidade do Problema da Parada baseada em máquinas de registro que também pode ser vista em [EFT94]. Assim como Turing em seu trabalho, primeiramente introduz-se o conceito de procedimento. Para tanto, começa-se com uma definição formal de programa.

No que segue o alfabeto $\mathcal{A} = \{a_0, \ldots, a_r\}$ é fixado. O conjunto de todas as palavras sobre $\mathcal{A}$ é denotado por $\mathcal{A}^*$ e $\square$ representa a palavra vazia. Programas são executados por computadores com um número arbitrário de unidades de memória, denominadas *registros*.

As unidades de memória são denotadas por $R_0, \ldots, R_m$ e contêm uma palavra de $\mathcal{A}^*$. Tais computadores são conhecidos como *máquinas de registros*.

Um programa sobre $\mathcal{A}$ consiste de instruções rotuladas por um número natural n que é único para cada instrução e serve como uma referência para que o programa possa executar uma determinada instrução a qualquer momento da execução em que o rótulo seja chamado. Tais instruções podem ser como segue:

(1) Instrução de adição.
Para $n, i, j \in \mathbb{N}$ com $j \leqslant r$: $\boxed{n \texttt{ LET } R_i = R_i + a_j}$
Significado: "adicionar a letra a_j ao final da palavra no registro R_i".

(2) Instrução de subtração.
Para $n, i, j \in \mathbb{N}$ com $j \leqslant r$: $\boxed{n \texttt{ LET } R_i = R_i - a_j}$
Significado: "Se a palavra no registro R_i termina com a letra a_j, apague a_j. Caso contrário não faça nada".

(3) Instrução de salto.
Para $n, i, n', n_0, \ldots, n_r \in \mathbb{N}$: $\boxed{n \texttt{ IF } R_i = \square \texttt{ THEN } n' \texttt{ ELSE } n_0 \texttt{ OR } \ldots \texttt{ OR } n_r}$
Significado: "Se o registro R_i contém a palavra vazia vá para a instrução n'; caso contrário, se a palavra no registro R_i termina com a letra a_0 (respectivamente $a_1, \ldots, a_r$) vá para a instrução n_0 (respectivamente $n_1, \ldots, n_r$)".

(4) Instrução de impressão.
Para $n \in \mathbb{N}$: $\boxed{n \texttt{ IMPRIMA}}$
Significado: "Imprima como resposta a palavra armazenada no registro R_0".

(5) Instrução de parada.
Para $n \in \mathbb{N}$: $\boxed{n \texttt{ PARA}}$
Significado: "Pare de executar".

Definição 2.1.1 (Programa de Registro)**:** *Um* programa de registro *(ou* programa*) é uma sequência finita $\alpha_0 \ldots \alpha_k$ de instruções da forma* (1) *a* (5) *tais que:*

i) α_i tem rótulo $i \in \{0, \ldots, k\}$.

ii) Cada instrução de salto refere-se a rótulos menores ou iguais a k.

iii) Somente a última linha α_k é uma instrução de parada.

Tendo em mente que um procedimento deve ser de tal forma que seja possível escrevê-lo em uma linguagem que possa ser lida por um computador, tem-se que um programa dá origem a um procedimento.

Diz-se que um programa P começa uma computação com uma palavra $\zeta \in \mathcal{A}^*$, se no início do procedimento a palavra gravada no registro R_0 é ζ e nos outros registros tem-se a palavra vazia $\square$. Se P começa com ζ e eventualmente atinge a instrução de parada, escreve-se $P : \zeta \to \texttt{PARA}$, caso contrário escreve-se $P : \zeta \to \infty$. Para $\zeta, \eta \in \mathcal{A}^*$, $P : \zeta \to \eta$ significa que P começa com ζ e eventualmente para dando uma resposta, denominada η no registro R_0.

No que segue apresenta-se um significado formal para a expressão "um subconjunto W de palavras de $\mathcal{A}^*$ é decidível via programas de registro".

Definição 2.1.2: *Seja* $W \subset \mathcal{A}^*$.

a) Diz-se que um programa P decide W, *se* $\forall \zeta \in \mathcal{A}^*$
$P : \zeta \to \square$, *se* $\zeta \in W$;
$P : \zeta \to \eta$ *com* $\eta \neq \square$, *se* $\zeta \notin W$.

b) Diz-se que W *é* registro-decidível *(ou* R-decidível*) se existe um programa que decide* W.

No que segue, o objetivo é apresentar um subconjunto de $\mathcal{A}^*$ que não seja R-decidível. Este conjunto será formado por programas sobre $\mathcal{A}$. Inicialmente estende-se o alfabeto $\mathcal{A}$ da seguinte forma:

$$\mathcal{B} := \mathcal{A} \cup \{A, B, \ldots, X, Y, Z\} \cup \{0, 1, \ldots, 9\} \cup \{=, +, -, \square, \S\} \tag{2.1}$$

Assim, um programa P é uma palavra sobre $\mathcal{B}$. A cada programa P associa-se uma palavra $\xi_P \in \mathcal{A}^*$ da seguinte forma: considere a ordem lexicográfica das palavras de $\mathcal{B}^*$ induzida pela ordem das letras apresentada em 2.1; se a palavra de $\mathcal{B}^*$ que representa um programa P é a n-ésima palavra na ordem lexicográfica de $\mathcal{B}^*$, então defina a palavra $\xi_P := \underbrace{a_0 \ldots a_0}_{n-vezes}$. Agora, defina o conjunto $\Pi := \{\xi_P \mid P \text{ é um programa sobre } \mathcal{A}\}$.

Lema 2.1.3: *O conjunto* Π *é R-decidível.*

Prova: Deve-se verificar que existe um programa P que decide Π. Mas ao invés de escrever explicitamente o programa, apresenta-se uma descrição intuitiva. Claramente, se $\zeta \in \mathcal{A}^*$, é possível decidir se $\zeta \in \{a\}^*$ ou não, no caso positivo se o comprimento de ζ é igual a n, então é possível determinar a palavra sobre $\mathcal{B}$ na n-ésima posição segundo a ordem lexicográfica determinada por 2.1. Denote tal palavra por P e verifique se P é

um programa sobre $\mathcal{A}$ segundo a definição 2.1.1. Portanto, tem-se um procedimento para verificar que: se $\zeta \in \Pi$, então é possível determinar o programa P com $\xi_P = \zeta$. Este procedimento pode ser programado em uma máquina de registro. Logo Π é R-decidível. ■

No próximo teorema apresentam-se exemplos de conjuntos que não são R-decidíveis, caracterizando assim a indecidibilidade do Problema da Parada.

Teorema 2.1.4 (Indecidibilidade do Problema da Parada)**:**

a) O conjunto: $\Pi'_{\texttt{PARA}} = \{\xi_P \mid P$ *é um programa sobre* $\mathcal{A}$ *e* $P : \xi_P \to \texttt{PARA}\}$ *não é R-decidível.*

b) O conjunto $\Pi_{\texttt{PARA}} = \{\xi_P \mid P$ *é um programa sobre* $\mathcal{A}$ *e* $P : \square \to \texttt{PARA}\}$ *não é R-decidível.*

Prova:

a) Por contradição, suponha que existe um programa P_0 tal que P_0 decide $\Pi'_{\texttt{PARA}}$. Assim, para todo P tem-se que:

$$P_0 : \xi_P \to \square, \text{ se } P : \xi_P \to \texttt{PARA},$$
$$P_0 : \xi_P \to \eta \text{ para algum } \eta \neq \square, \text{ se } P : \xi_P \to \infty.$$

A partir de P_0, obtem-se um novo programa P_1 da seguinte forma: altere P_0 de tal forma que se P_0 imprime a palavra vazia e para, o novo programa P_1 não alcança a instrução de parada. Isto é obtido substituindo-se a linha k `PARA` em P_0 pelas seguintes instruções:

$$\begin{array}{ll} k & \texttt{IF } R_0 = \square \texttt{ THEN } k \texttt{ ELSE } k+1 \texttt{ OR } \ldots \texttt{ OR } k+1 \\ k+1 & \texttt{PARA} \end{array}$$

E todas as instruções da forma n `IMPRIMA` em P_0 por instruções da forma n `IF` $R_0 = \square$ `THEN` k `ELSE` k `OR` $\ldots$ `OR` k, que é simplesmente substituir a instrução n por uma instrução de salto que manda para a instrução k. Desta forma P_1 é tal que para todo P:

$$P_1 : \xi_P \to \infty, \text{ se } P : \xi_P \to \texttt{PARA},$$
$$P_1 : \xi_P \to \texttt{PARA}, \text{ se } P : \xi_P \to \infty.$$

Assim, o seguinte vale para todos os programas P:

$$P_1 : \xi_P \to \infty \text{ se e somente se } P : \xi_P \to \texttt{PARA}.$$

Em particular, tem-se que:

$$P_1 : \xi_{P_1} \to \infty \text{ se e somente se } P_1 : \xi_{P_1} \to \texttt{PARA},$$

o que é uma contradição.

b) A fim de verificar que $\Pi_{\texttt{PARA}}$ não é R-decidível, a cada programa P designa-se um programa P^+ da seguinte maneira: Se $\xi_P = \underbrace{a_0 \ldots a_0}_{n-vezes}$, defina P^+ como sendo o programa que começa com as instruções

$$\begin{aligned} 0 \quad & \texttt{LET } R_0 = R_0 + a_0 \\ & \vdots \\ n-1 \quad & \texttt{LET } R_0 = R_0 + a_0 \end{aligned}$$

seguidas pelas linhas de P com todos os rótulos acrescidos de n. Quando P^+ é iniciado com $\square$ como entrada, ele primeiro constrói a palavra ξ_P em R_0 e então procede da mesma maneira que P aplicado a ξ_P. Desta forma, P^+ é tal que:

$$\begin{aligned} & P : \xi_P \to \texttt{PARA} \text{ se e somente se } P^+ : \square \to \texttt{PARA}, \text{ isto é} \\ & \xi_P \in \Pi'_{\texttt{PARA}} \text{ se e somente se } \xi_{P^+} \in \Pi_{\texttt{PARA}} \end{aligned} \qquad (\star)$$

Assim, suponha que $\Pi_{\texttt{PARA}}$ é R-decidível, por exemplo por um programa P_0. Então, obtem-se o seguinte procedimento de decisão para $\Pi'_{\texttt{PARA}}$ em contradição a (a): para um dado $\zeta \in \mathcal{A}^*$ arbitrário, primeiro verifique se $\zeta \in \Pi$, conforme o Lema 2.1.3; se $\zeta \notin \Pi$ então $\zeta \notin \Pi'_{\texttt{PARA}}$; se $\zeta \in \Pi$, então tome o programa P tal que $\xi_P = \zeta$ e construa P^+, em seguida, usando P_0, decida se $\xi_{P^+} \in \Pi_{\texttt{PARA}}$. Portanto, considerando $(\star)$, o resultado é um procedimento de decisão para responder se $\xi_P \in \Pi'_{\texttt{PARA}}$, ou seja, se $\zeta \in \Pi'_{\texttt{PARA}}$.

O que conclui a prova. ■

O item (b) do teorema diz que não existe um programa de registro que possa decidir o conjunto $\Pi_{\texttt{PARA}}$ e pode ser reformulado da seguinte maneira:

Não existe um procedimento de decisão que, para um dado programa P arbitrário, decide se $P : \square \to \texttt{PARA}$.

Em suma, o Teorema 2.1.4 diz que não existe um procedimento que responda de maneira uniforme se $P : \square \to \texttt{PARA}$ para um P arbitrário, embora talvez para um P particular esta verificação seja fácil.

2.2 Automação da Verificação da Terminação

O fato de o problema de terminação ser indecidível significa que: *não existe um procedimento geral que, dado um programa arbitrário finito, responde "sim" no caso em que o programa termina e "não" caso contrário.* Contudo, terminação é fundamental tanto em matemática quanto em ciência da computação, pois quando se tem uma prova de terminação de um programa ou definição recursiva, isto significa que sempre um resultado é obtido dada qualquer entrada. Além disso, provas de terminação são aplicadas na prática na verificação de correção de *drivers* de sistemas em geral. Mas como não existe um procedimento para verificação de terminação e, sendo esta propriedade tão importante, é necessário verificar terminação em casos particulares, uma vez que a correção de um algoritmo depende de terminação, e se faz necessário o desenvolvimento de métodos para verificação de terminação. Isto vem acontecendo desde a década de oitenta, veja por exemplo os famosos *surveys* de Dershowitz [Der85, Der87].

Além disso, muita pesquisa tem sido realizada visando desenvolver métodos para verificação de terminação, no sentido de obter provas de terminação de maneira automática e certificada, isto envolve formalização (em algum assistente de provas) de critérios conhecidos para verificação de terminação e desenvolvimento de ferramentas que sejam capazes de certificar provas de terminacação. Devido à complexidade de tais provas de terminação, vários projetos envolvendo o desenvolvimento de certificadores para tais provas, que aplicam bibliotecas desenvolvidas em alguma provador automático tem surgido e estado presentes, mostrando a sua eficiência, na *Competição Anual de Terminação*[1]. Por exemplo: em [CPR06b] e [CSZ13] são apresentados métodos e são brevemente descritas ferramentas (a saber o TERMINATOR e T2) que automatizam tais métodos para verificação de terminação, e que têm sido largamente aplicados em sistemas reativos [CPR06a, CPR07], como por exemplo sistemas operacionais, servidores de internet, servidores de e-mail, etc., para verificar que *drivers* de dispositivos sempre retornam ao sistema operacional quando chamados; em [FKS11, FKS12] é apresentada a ferramenta KITTeL que é aplicada para verificação de terminação de programas em linguagem C, contudo esta ferramenta é baseada em obter um sitema de reescrita de termos a partir do código C e aplicar técnicas de sistemas de reescrita para verificação de terminação; em [GSKT06] é apresentada a ferramenta AProVE, que fornece um sistema para certificação de provas automatizadas de terminação para sistemas de reescrita de termos através da técnica de pares dependentes; em [CGBA+11] é apresentado o Size-change/MCNP, que é uma implementação de provas

[1] http://termination-portal.org/wiki/Termination_Competition

de terminação via *monotonicity constraints* (uma generalização do princípio de mudança de tamanho); em [KSZM09] é apresentada a ferramenta $\mathsf{T_TT_2}$, que é um analizador automático de terminação para sistemas de reescrita de termos de primeira ordem e que também é baseada na estratégia de pares dependentes, etc.

Técnicas para verificação de terminação também têm sido implementadas com sucesso em provadores de teorema, com o objetivo de automatizar cada vez mais as provas, dando origem a bibliotecas completas, no sentido de que compilam diversas técnicas para verificação de terminação, principalmente sobre sistemas de reescrita de termos, e que tem sido largamente utilizadas por alguns dos certificadores já mencionados. Por exemplo:

- em Coq, a biblioteca CoLoR [BK11], é uma coleção de definições matemáticas e teoremas sobre terminação de sistemas de reescrita de termos e que compreende a formalização de várias técnicas para verificação de terminação, junto com funções booleanas para verificação das condições que os parâmetros das definições devem satisfazer, e que são provadas corretas. Esta biblioteca é usada pelo Rainbow [BL12], que por sua vez é uma ferramenta que transforma provas de terminação em um formato específico em uma prova em Coq certificando a terminação através do emprego de resultados da biblioteca CoLoR. Neste sentido, Rainbow é um certificador para provas de terminação que aplica CoLoR. Além disso existem outras ferramentas que utilizam CoLoR, como os já mencionados AProVE e $\mathsf{T_TT_2}$, dentre outros, como por exemplo MatchBox [Wal04].

- em Isabelle/HOL, existe a biblioteca IsaFoR [TS] que compreende a formalização de várias técnicas para verificação de terminação em reescrita de termos, que em conjunto com o certificador CeTA [TS09, STWZ12], gerado a partir da formalização IsaFoR, da origem a ferramenta IsaFoR/CeTA para certificar provas de terminação de sistemas de reescrita de termos. IsaFoR/CeTA tem se destacado a cada ano na competição anual de terminação e atualmente, o certificador CeTA é capaz de certificar grande parte das técnicas aplicadas em ferramentas como AProVE, $\mathsf{T_TT_2}$, MatchBox, detre outras.

- ainda em Isabelle/HOL, o princípio de mudança de tamanho foi formalizado [Kra07, Kra09], porém não há uma implementação do mesmo neste provador.

- em ACL2, o princípio de mudança de tamanho, mencionado na introdução, foi implementado com sucesso em ACL2 [MV06, Vro07], através da estrutura de dados Grafos de Contextos de Chamado (CCG), porém sem uma formalização desta

técnica.

- em PVS, o princípio de mudança de tamanho, também tem sido implementado por Ayala-Rincón e Muñoz, e aplicado com sucesso na verificação automática de terminação de definições recursivas na linguagem de especificação do PVS. E este trabalho apresenta uma formalização em PVS deste princípio através da nova estrutura Grafos com Matrizes de Medida (MWG), que organiza as medidas aplicadas em CCG em uma matriz quadrada, falicitando a combinação entre essas medidas em circuitos de tamanho arbitrário do digrafo, e cuja noção de terminação é provada ser equivalente à noção de terminação em CCG's, o que torna este trabalho semelhante á implementação em ACL2, com o diferencial de trazer a verificação formal do método. Neste sentido este trabalho é semelhante à formalização desenvolvida em Isabelle/HOL por Krauss. Contudo a formalização de Isabelle/HOL é desenvolvida com base também em digrafos, mas tendo em mente que grafos formam um outra estrutura denominada álgebra de Kleene, que é uma estrutura mais complicada de lidar no sentido de dificultar as provas, do que a nossa estrura MWG. Além disso, verifica-se, como será apresentado no Capítulo 6, não através de uma prova formal, mas de uma prova analítica, que a noção de terminação em MWG's captura a noção de terminação do PVS, o que leva a conclusão de que a técnica implementada pode ser aplicada corretamente para verificação de terminação em PVS.

Devido à indecidiblidade do problema, nenhuma das ferramentas mencionadas pode funcionar em todos os casos. Porém avanços no sentido de obter ferramentas cada vez mais poderosas têm sido alcançados. Os resultados de pesquisas sobre terminação são largamente disceminados através da competicação anual de teminação, onde grupos de pesquisadores têm a oportunidade de demonstrar a capacidade de suas ferramentas para análise de terminação completamente automatizada, dentre estas as já mencionadas neste capítulo.

Este trabalho apresenta contribuições neste sentido, através da especificação da estrutura MWG para análise de terminação de linguagens funcionais de primeira ordem, e formalização de propriedades indicando que esta estrutura é adequada para análise de terminação. Veja Figura 1.1, onde mostra-se a conexão entre as *teorias* envolvidas e principalmente, a equivalência entre a noção de terminação para MWG's e a noção de terminação para CCG's. Esta equivalência é conclusiva neste trabalho, uma vez que a proposta inicial era desenvolver uma verificação formal da técnica implementada através da estrutura CCG's, e tal verificação é desenvolvida através de MWG's, uma vez que

MWG's se mostra uma estrutura elegante onde suas propriedades são mais simples de especifcar e formalizar. Além disso, partindo do princípio de que CCG's são uma estrutura adequada para análise de terminação, esta equivalência entre as duas estruturas confere uma certificação para a estrutura MWG.

Capítulo 3

Preliminares

Neste capítulo serão estabelecidos os conceitos e resultados que formam a base para o desenvolvimento deste trabalho. Esta base é composta de propriedades sobre digrafos e digrafos com peso, sobre a tecnologia *Calling Context Graphs*, além de conhecimentos sobre a linguagem de especificação e poder dedutivo oferecido pelo assistente de prova PVS. Desta forma este capítulo é organizado em três seções: na Seção 3.1 os principais resultados sobre digrafos e digrafos com peso são apresentados; na Seção 3.2 a tecnologia *Calling Context Graphs* é descrita; e na Seção 3.3 alguns aspectos acerca da sintaxe e semântica do PVS serão abordados.

3.1 Digrafos

Digrafos com peso formam a base para as definições apresentadas neste trabalho acerca de *Grafos com Matrizes de Medida*. Esta estrutura de dados é aplicada na representação de possíveis fluxos de chamado durante a execução de uma definição recursiva. Assim, em um digrafo relativo a um programa, cada vértice é relacionado a um chamado recursivo, e cada aresta significa que sob certas condições é possível realizar um chamado recursivo após o outro. Neste capítulo, pesos sobre arestas em um digrafo com peso serão tratados de maneira geral. A noção de peso em *Grafos com Matrizes de Medida* é mais específica, pois neste caso matrizes de medida construídas a partir de combinações de funções de medida serão usadas. Porém todos estes conceitos acerca de *Grafos com Matrizes de Medida* serão apresentados no Capítulo 4. Por enquanto apenas uma apresentação formal de digrafos e digrafos com peso será tratada.

A teoria apresentada a seguir sobre digrafos está especificada e formalizada em PVS e disponível nas bibliotecas da NASA LaRC para o PVS, sob o nome de `digraphs` [BRSA13].

Foram realizadas contribuições nesta *teoria*, principalmente na especificação de definições e formalização de propriedades acerca de *circuitos* e *ciclos* de um digrafo e *digrafos com peso*. Assim, muitos dos resultados apresentados nesta seção foram incorporados à *teoria* `digraphs`, para fornecer suporte ao desenvolvimento da formalização posterior, e que será descrita nos capítulos subsequentes.

Vértices em um digrafo são elementos de um determinado tipo. É importante notar que em um digrafo pares ordenados são relevantes, uma vez que uma aresta é um par de vértices, e a ordem deste par determina em que direção esta aresta deve ser percorrida, ou seja, de que vértice sair e para qual ir. Assim, define-se um digrafo como segue:

Definição 3.1.1 (Digrafos)**:** *Um digrafo é um par $G = \langle V_G, E_G \rangle$, onde V_G é um conjunto finito e E_G é uma relação binária sobre V_G. Isto é, se $(u, v) \in E_G$ então $u \in V_G$ e $v \in V_G$.*

Notação: O conjunto V_G é o *conjunto de vértices de G* e seus elementos são os *vértices* de G. O conjunto E_G é o *conjunto de arestas de G* e seus elementos são as *arestas* de G. Vétices serão representados por u, v, u_1, v_1, etc. E arestas por $e = (u, v)$, etc.

Notação: Neste trabalho, para uma sequência qualquer *seq* de elementos de um determinado tipo, o seu comprimento será um número denotado por $\ell(seq)$ e corresponde a quantidade de elementos que compõe a sequência. Assim, para $seq = (a_0, a_1, \ldots, a_n)$, $\ell(seq) = n - 1$.

Definição 3.1.2 (Pré-caminhos)**:** *Um pré-caminho $w = (v_0, \ldots, v_{n-1})$ é uma sequência de vértices cujo comprimento $\ell(w)$ é maior que 0. Um conjunto de pré-caminhos será denotado por $\mathcal{P}_w$.*

Notação: Os elementos da sequência $w \in \mathcal{P}_w$ de comprimento $\ell(w) = n$, também serão denotados por $w[i]$, para $0 \leqslant i \leqslant n - 1$.

Definição 3.1.3 (Sub-caminhos)**:** *Seja $w = (v_0, \ldots, v_{n-1}) \in \mathcal{P}_w$, com $\ell(w) = n$. Para quaisquer $i, j \in \mathbb{N}$, com $0 \leqslant i \leqslant j \leqslant n - 1$, $w^{(i,j)} = (v_i, \ldots, v_j)$ é um sub-caminho de w.*

No que segue será estabelecida uma relação de equivalência sobre um conjunto de pré-caminhos. Tal relação é importante na definição de uma relação de equivalência para circuitos. O que por sua vez, é importante na estrutura de um esquema de indução em circuitos, e também na verificação de propriedades (que sendo satisfeitas por um circuito o serão por todos os outros na mesma classe de equivalência), relevantes para terminalidade de uma definição recursiva.

Considere o seguinte predicado $\overset{w}{\sim}$: $\mathcal{P}_w \times \mathcal{P}_w \to \{\texttt{true}, \texttt{false}\}$, definido por:

$$w_1 \overset{w}{\sim} w_2 \;:=\; w_1 = w_2 \;\vee\; \exists\, i \in \mathbb{N},\; 0 \leqslant i \leqslant \ell(w_1) - 1 : w_2 = w_1^{(i,\ell(w_1)-1)} \circ w_1^{(0,i-1)} \tag{3.1}$$

onde $w_1, w_2 \in \mathcal{P}_w$ e $\circ$ denota a composição usual de sequências. Além disso, observe que $w_1^{(i,\ell(w_1)-1)} \circ w_1^{(0,i-1)} \in \mathcal{P}_w$.

Definição 3.1.4 (Pré-caminhos equivalentes)**:** *Dado um conjunto de pré-caminhos $\mathcal{P}_w$ e dois pré-caminhos $w_1, w_2 \in \mathcal{P}_w$, diz-se que w_1 e w_2 são equivalentes se $w_1 \overset{w}{\sim} w_2$ é válido.*

Lema 3.1.5 (Relação de equivalência sobre pré-caminhos)**:** *Seja um conjunto de pré-caminhos $\mathcal{P}_w$. O predicado $\overset{w}{\sim}$: $\mathcal{P}_w \times \mathcal{P}_w \to \{\texttt{true}, \texttt{false}\}$, definido na Equação 3.1, estabelece uma relação de equivalência sobre $\mathcal{P}_w$.*

Prova: Deve-se verificar que a relação estabelecida pelo predicado $\overset{w}{\sim}$ é:

i) Reflexiva: $\forall\, w \in \mathcal{P}_w : w \overset{w}{\sim} w$. Segue direto da definição de $\overset{w}{\sim}$, pois $w = w$.

ii) Simétrica: $\forall\, w_1, w_2 \in \mathcal{P}_w : w_1 \overset{w}{\sim} w_2 \Rightarrow w_2 \overset{w}{\sim} w_1$. Neste caso, deve-se verificar que

$$w_1 = w_2 \;\vee\; \exists\, i \in \mathbb{N},\; 0 \leqslant i \leqslant \ell(w_1) - 1 : w_2 = w_1^{(i,\ell(w_1)-1)} \circ w_1^{(0,i-1)}$$

$$\Downarrow$$

$$w_2 = w_1 \;\vee\; \exists\, j \in \mathbb{N},\; 0 \leqslant j \leqslant \ell(w_2) - 1 : w_1 = w_2^{(j,\ell(w_2)-1)} \circ w_2^{(0,j-1)}$$

Se $w_1 = w_2$, então segue o resultado. Assim, suponha que $w_1 \neq w_2$ e que

$$w_2 = w_1^{(i,\ell(w_1)-1)} \circ w_1^{(0,i-1)} \tag{3.2}$$

para algum $i \in \mathbb{N}$, $0 \leqslant i \leqslant \ell(w_1) - 1$. Observe que

$$\ell(w_2) = \ell(w_1^{(i,\ell(w_1)-1)} \circ w_1^{(0,i-1)}) = (\ell(w_1) - 1 - i + 1) + (i - 1 + 1) = \ell(w_1)$$

Tome $j = \ell(w_1) - i$. Deve-se verificar que

$$w_1 = w_2^{(\ell(w_1)-i,\ell(w_2)-1)} \circ w_2^{(0,\ell(w_1)-i-1)} \tag{3.3}$$

De fato, observe que $\ell(w_2^{(\ell(w_1)-i,\ell(w_2)-1)} \circ w_2^{(0,\ell(w_1)-i-1)}) = \ell(w_2)$, que por hipótese é igual a $\ell(w_1)$. Além disso, para todo $k \in \mathbb{N}$, $0 \leqslant k \leqslant \ell(w_1) - 1$, decompondo a Equação (3.3), deve-se verificar que

$$w_1[k] = (w_2^{(\ell(w_1)-i,\ell(w_2)-1)} \circ w_2^{(0,\ell(w_1)-i-1)})[k] = \begin{cases} w_2[\ell(w_1) - i + k], & \text{se } k < i \\ w_2[k - i], & \text{se } k \geqslant i \end{cases}$$

No caso em que $k < i$, observe que $\ell(w_1) - i + k \geqslant \ell(w_1) - i$. Portanto de (3.2) segue que

$$w_2[\ell(w_1) - i + k] = w_1[\ell(w_1) - i + k - (\ell(w_1) - i)] = w_1[k]$$

No caso em que $k \geqslant i$, observe que $k - i \leqslant \ell(w_1) - i - 1 < \ell(w_1) - i$. Portanto de (3.2) segue que

$$w_2[k - i] = w_1[k - i + i] = w_1[k]$$

O que conclui a verificação da simetria de $\overset{w}{\sim}$.

iii) Transitiva: $\forall\, w_1, w_2, w_3 \in \mathcal{P}_w : w_1 \overset{w}{\sim} w_2 \wedge w_2 \overset{w}{\sim} w_3 \Rightarrow w_1 \overset{w}{\sim} w_3$. Neste caso, deve-se verificar que

$$w_1 = w_2 \ \vee\ \exists\, i \in \mathbb{N},\ 0 \leqslant i \leqslant \ell(w_1) - 1 : w_2 = w_1^{(i,\ell(w_1)-1)} \circ w_1^{(0,i-1)} \quad \wedge$$

$$w_2 = w_3 \ \vee\ \exists\, j \in \mathbb{N},\ 0 \leqslant j \leqslant \ell(w_2) - 1 : w_3 = w_2^{(j,\ell(w_2)-1)} \circ w_2^{(0,j-1)}$$

$$\Downarrow$$

$$w_1 = w_3 \ \vee\ \exists\, k \in \mathbb{N},\ 0 \leqslant k \leqslant \ell(w_1) - 1 : w_3 = w_1^{(k,\ell(w_1)-1)} \circ w_1^{(0,k-1)}$$

Caso, $w_1 = w_2$, basta tomar $k = j$. Caso, $w_2 = w_3$, basta tomar $k = i$. Assim, resta considerar o caso em que $w_1 \neq w_2$ e $w_2 \neq w_3$. Sejam $i, j \in \mathbb{N}$, $0 \leqslant i \leqslant \ell(w_1) - 1$ e $0 \leqslant j \leqslant \ell(w_2) - 1$, tais que $w_2 = w_1^{(i,\ell(w_1)-1)} \circ w_1^{(0,i-1)}$ e $w_3 = w_2^{(j,\ell(w_2)-1)} \circ w_2^{(0,j-1)}$. Observe que:

- Se $j < \ell(w_1) - i$, então

$$(w_1^{(i,\ell(w_1)-1)} \circ w_1^{(0,i-1)})^{(j,\ell(w_2)-1)} = w_1^{(j+i,\ell(w_1)-1)} \circ w_1^{(0,i-1)}$$

$$(w_1^{(i,\ell(w_1)-1)} \circ w_1^{(0,i-1)})^{(0,j-1)} = w_1^{(i,j+i-1)}$$

$$\Downarrow \text{ Tome } k = j + i$$

$$w_3 = w_1^{(j+i,\ell(w_1)-1)} \circ w_1^{(0,j+i-1)}$$

- Se $j \geqslant \ell(w_1) - i$, então

$$(w_1^{(i,\ell(w_1)-1)} \circ w_1^{(0,i-1)})^{(j,\ell(w_2)-1)} = w_1^{(j+i-\ell(w_1),i-1)}$$

$$(w_1^{(i,\ell(w_1)-1)} \circ w_1^{(0,i-1)})^{(0,j-1)} = w_1^{(i,\ell(w_1)-1)} \circ w_1^{(0,j+i-\ell(w_1)-1)}$$

$$\Downarrow \text{ Tome } k = j + i - \ell(w_1)$$

$$w_3 = w_1^{(j+i-\ell(w_1),\ell(w_1)-1)} \circ w_1^{(0,j+i-\ell(w_1)-1)}$$

Portanto, de (i), (ii), e (iii) segue que $\overset{w}{\sim}$ é uma relação de equivalência. ■

Definição 3.1.6 (Caminhos)**:** *Dado um digrafo G e um pré-caminho $w = (v_0, \ldots, v_{n-1})$, diz-se que w é um caminho de G se:*

$$\forall\, i,\ 0 \leqslant i \leqslant n-1 : v_i \in V_G \quad \wedge \quad \forall\, i,\ 0 \leqslant i \leqslant n-2 : (v_i, v_{i+1}) \in E_G,$$

onde os pares (v_i, v_{i+1}), para todo i, $0 \leqslant i \leqslant n-2$, são ditas as arestas de w. O conjunto de caminhos de um digrafo G será denotado por $\mathcal{W}_G$.

Observe que, dado um digrafo G, para o comprimento de um caminho $w \in \mathcal{W}_G$ vale que $\ell(w) \geqslant 1$. As definições estabelecidas para pré-caminhos são naturalmente estendidas para caminhos, uma vez que caminhos são também pré-caminhos. Contudo $\overset{w}{\sim}$ não é uma relação de equivalência em $\mathcal{W}_G$, pois nem sempre é verdade que $w_1^{(i,\ell(w_1)-1)} \circ w_1^{(0,i-1)} \in \mathcal{W}_G$, para $0 \leqslant i \leqslant \ell(w_1)-1$, a menos que exista uma aresta conectando os vértices $w_1[\ell(w_1)-1]$ e $w_1[0]$.

Definição 3.1.7 (Arestas e vértices de um caminho)**:** *Seja um digrafo G e $w \in \mathcal{W}_G$. O conjunto de vértices de w será denotado por $V_G(w)$; e o conjunto de arestas de w será denotado por $E_G(w)$.*

Definição 3.1.8 (Composição de caminhos)**:** *Dados um digrafo G e dois caminhos $w_1 \in \mathcal{W}_G$ e $w_2 \in \mathcal{W}_G$, define-se a composição de w_1 com w_2, para o caso em que $w_1[\ell(w_1) - 1] = w_2[0]$, pela aplicação:*

$$\circ_w : \{(w_1, w_2) \mid w_1, w_2 \in \mathcal{W}_G \ \wedge\ w_1[\ell(w_1) - 1] = w_2[0]\} \to \mathcal{W}_G$$

definida por

$$\circ_w(w_1, w_2) := w_1 \circ w_2^{(1,\ell(w_2)-1)}$$

Lema 3.1.9 (Arestas da composição de caminhos)**:** *Dados um digrafo G e $w_1, w_2 \in \mathcal{W}_G$, se $w_1[\ell(w_1) - 1] = w_2[0]$ então $E_G(\circ_w(w_1, w_2)) = E_G(w_1) \cup E_G(w_2)$.*

Prova: Se $\ell(w_2) = 1$, então $\circ_w(w_1, w_2) = w_1$ e $E_G(w_2)$ é um conjunto vazio. Logo, $E_G(\circ_w(w_1, w_2)) = E_G(w_1) = E_G(w_1) \cup E_G(w_2)$. Analogamente, no caso em que $\ell(w_1) = 1$ e $\ell(w_2) > 1$, tem-se que $\circ_w(w_1, w_2) = w_2$ e $E_G(w_1)$ é um conjunto vazio. E portanto, $E_G(\circ_w(w_1, w_2)) = E_G(w_2) = E_G(w_1) \cup E_G(w_2)$. Resta considerar o caso em que $\ell(w_1) > 1$ e $\ell(w_2) > 1$. Sejam, $w_1 = (u_0, u_1, \ldots, u_{n-1})$ e $w_2 = (v_0, v_1, \ldots, v_{m-1})$, onde $u_{n-1} = v_0 = \bar{u}$.

Assim, $\circ_w(w_1, w_2) = (u_0, u_1, \ldots, u_{n-2}, \bar{u}, v_1, \ldots, v_{m-1})$. E portanto,

$$\begin{aligned} E_G(\circ_w(w_1, w_2)) &= \{(u_0, u_1), (u_1, u_2), \ldots, (u_{n-2}, \bar{u}), (\bar{u}, v_1), \ldots, (v_{m-2}, v_{m-1})\} \\ &= \{(u_0, u_1), (u_1, u_2), \ldots, (u_{n-2}, u_{n-1}), (v_0, v_1), \ldots, (v_{m-2}, v_{m-1})\} \\ &= \{(u_0, u_1), (u_1, u_2), \ldots, (u_{n-2}, u_{n-1})\} \ \cup \ \{(v_0, v_1), \ldots, (v_{m-2}, v_{m-1})\} \\ &= E_G(w_1) \cup E_G(w_2) \end{aligned}$$

O que conclui a prova. ■

3.1.1 Circuitos e Ciclos

Nesta seção são apresentados definições e resultados sobre ciclos e circuitos, que são os elementos do digrafo mais diretamente relacionados com a terminação ou não de uma definição recursiva, uma vez que circuitos representam possíveis sequências de chamados recursivos, como mencionado no início desta seção.

Definição 3.1.10 (Circuitos)**:** *Dado um digrafo G, um circuito $c = (v_0, \ldots, v_{n-1})$ é um caminho de G tal que $v_0 = v_{n-1}$ e $\ell(c) > 1$.*

Notação: $\mathcal{C}ir(G)$ denota o conjunto de circuitos do digrafo G.

Definição 3.1.11 (Ciclos)**:** *Dado um digrafo G, um ciclo $c = (v_0, \ldots, v_{n-1})$ é um circuito tal que $\forall\, i, j < \ell(c) - 1 : i \neq j \Rightarrow c[i] \neq c[j]$.*

Notação: $\mathcal{C}yc(G)$ denota o conjunto de ciclos do digrafo G.

Da definição anterior note que, em um ciclo, todos os elementos são distintos exceto pelo primeiro e último vértices.

Como sugerido nas definições anteriores, circuitos e ciclos serão preferencialmente representados pela letra c.

Definição 3.1.12 (Circuitos equivalentes)**:** *Dados um digrafo G e $c_1, c_2 \in \mathcal{P}_w$, considere o seguinte predicado $\sim\colon \mathcal{P}_w \times \mathcal{P}_w \to \{\texttt{true}, \texttt{false}\}$ definido por:*

$$c_1 \sim c_2 := c_1 \in \mathcal{C}ir(G) \ \wedge \ c_2 \in \mathcal{C}ir(G) \ \wedge \ c_1^{(1,\ell(c_1)-1)} \overset{w}{\sim} c_2^{(1,\ell(c_2)-1)}$$

Se $c_1 \sim c_2$, diz-se que c_1 e c_2 são circuitos equivalentes.

Observe que $\sim$ é uma relação de equivalência sobre $\mathcal{C}ir(G)$. Este fato segue diretamente do fato, verificado no Lema 3.1.5, de que $\overset{w}{\sim}$ é uma relação de equivalência sobre $\mathcal{P}_w$.

As definições e resultados para pré-caminhos e caminhos estendem-se naturalmente para ciclos e circuitos, uma vez que ciclos e circuitos são também pré-caminhos.

Lema 3.1.13 (Posição em circuitos equivalentes)**:** *Dados um digrafo G e $c_1, c_2 \in \mathcal{P}_w$. Se $c_1 \sim c_2$, então existe $j < \ell(c_1)$ tal que $c_2 = c_1^{(j,\ell(c_1)-2)} \circ c_1^{(0,j)}$.*

Prova: De fato, como $c_1^{(1,\ell(c_1)-1)} \overset{w}{\sim} c_2^{(1,\ell(c_2)-1)}$, segue da Equação 3.1 que

$$c_1^{(1,\ell(c_1)-1)} = c_2^{(1,\ell(c_2)-1)}, \tag{a}$$

ou

$$\exists\, i \in \mathbb{N},\ 0 \leqslant i \leqslant \ell(c_1) - 2 : c_2^{(1,\ell(c_2)-1)} = (c_1^{(1,\ell(c_1)-1)})^{(i,\ell(c_1)-2)} \circ (c_1^{(1,\ell(c_1)-1)})^{(0,i-1)}. \tag{b}$$

No caso em que (a), tem-se que $c_1 = c_2$ e basta tomar $j = 0$. No caso em que (b) vale, note que $(c_1^{(1,\ell(c_1)-1)})^{(i,\ell(c_1)-2)} \circ (c_1^{(1,\ell(c_1)-1)})^{(0,i-1)} = c_1^{(i+1,\ell(c_1)-1)} \circ c_1^{(1,i)} = c_1^{(i+1,\ell(c_1)-2)} \circ c_1^{(0,i)}$, para algum $i \in \mathbb{N},\ 0 \leqslant i \leqslant \ell(c_1) - 2$. Assim, basta tomar $j = i$. ∎

Lema 3.1.14 (Arestas de circuitos equivalentes)**:** *Sejam um digrafo G e $c_1, c_2 \in \mathcal{P}_w$. Se $c_1 \sim c_2$, então $E_G(c_1) = E_G(c_2)$.*

Prova: De fato, pelo Lema 3.1.13, existe $j < \ell(c_1)$ tal que $c_2 = c_1^{(j,\ell(c_1)-2)} \circ c_1^{(0,j)}$. Assim, $E_G(c_2) = E_G(c_1^{(j,\ell(c_1)-2)} \circ c_1^{(0,j)})$. Denotando $c_1 = (u_0, u_1, \ldots, u_j, \ldots, u_{n-1})$, onde $\ell(c_1) = n$, tem-se que $c_1^{(j,\ell(c_1)-2)} \circ c_1^{(0,j)} = (u_j, \ldots, u_{n-2}, u_0, u_1, \ldots, u_j)$. Logo, observando que $u_0 = u_{n-1}$, pois c_1 é um circuito, tem-se que

$$\begin{aligned} E_G(c_1^{(j,\ell(c_1)-2)} \circ c_1^{(0,j)}) &= \{(u_j, u_{j+1}), \ldots, (u_{n-2}, u_0), (u_0, u_1), \ldots, (u_{j-1}, u_j)\} \\ &= \{(u_j, u_{j+1}), \ldots, (u_{n-2}, u_{n-1}), (u_0, u_1), \ldots, (u_{j-1}, u_j)\} \\ &= \{(u_0, u_1), \ldots, (u_{j-1}, u_j), (u_j, u_{j+1}), \ldots, (u_{n-2}, u_{n-1})\} \\ &= E_G(c_1) \end{aligned}$$

O que conclui a prova do lema. ∎

Os lemas seguintes estabelecem um esquema de indução em circuitos baseado em sua estrutura. Este esquema de indução está ilustrado na Figura 3.1. Basicamente verifica-se que em um circuito c qualquer sempre é possível destacar um ciclo (Lema 3.1.15) e, desta forma, obter um circuito equivalente a c onde o sub-caminho inicial é um ciclo, obviamente o sub-caminho restante será um circuito (Lema 3.1.16).

Lema 3.1.15 (Sub-caminho ciclo)**:** *Sejam um digrafo G e um pré-caminho c. Assim,*

$$c \in \mathcal{C}ir(G) \Rightarrow \exists\, i, j \in \mathbb{N}, 0 \leqslant i \leqslant j \leqslant \ell(c) - 1 : c^{(i,j)} \in \mathcal{C}yc(G)$$

Isto é, c possui um sub-caminho que é um ciclo.

Prova: A prova se dá por indução na estrutura de c. Considere dois casos:

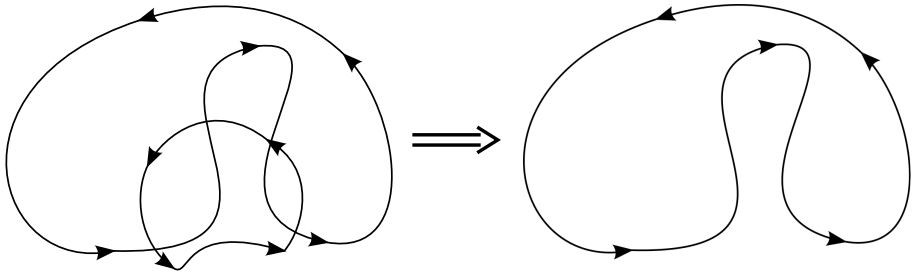


Figura 3.1: Esquema de indução aplicado na estrutura de um circuito, onde formalizou-se que em todo circuito é possível destacar um ciclo.

B.I. $c \in \mathcal{C}yc(G)$: Neste caso, basta tomar $i = 0$ e $j = \ell(c) - 1$, ou seja, c é o próprio sub-caminho.

P.I. $c \notin \mathcal{C}yc(G)$: Neste caso, da Definição 3.1.11, tem-se que

$$\exists\, i', j' < \ell(c) - 1 : i' \neq j' \Rightarrow c[i'] = c[j']$$

Sem perda de generalidade, suponha $i' < j'$. Assim, considere o subcaminho $c^{(i',j')}$ e observe que $c^{(i',j')}$ é um circuito e que $\ell(c^{(i',j')}) < \ell(c)$. Logo, por hipótese de indução,

$$\exists\, i'', j'' \in \mathbb{N}, 0 \leqslant i'' \leqslant j'' \leqslant \ell(c^{(i',j')}) - 1 : (c^{(i',j')})^{(i'',j'')} \in \mathcal{C}yc(G)$$

Mas note que $(c^{(i',j')})^{(i'',j'')} = c^{(i'+i'',j'+j'')}$ é um sub-caminho de c que é um ciclo. Assim, basta tomar $i = i' + i''$ e $j = j' + j''$.

De (i) e (ii) segue a prova do lema. ∎

Lema 3.1.16 (Esquema de indução na estrutura de um circuito)**:** *Seja um digrafo G e um pré-caminho c. Assim,*

$$c \in \mathcal{C}ir(G) \Rightarrow \begin{cases} c \in \mathcal{C}yc(G) \vee \\ \exists\, c_1, c_2 \in \mathcal{P}_w :\ c \sim \circ_w(c_1, c_2)\ \wedge\ c_1 \in \mathcal{C}yc(G)\ \wedge\ c_2 \in \mathcal{C}ir(G) \end{cases}$$

Prova: Como $c \in \mathcal{C}ir(G)$, pelo Lema 3.1.15, segue que existem $i, j \in \mathbb{N}$, com $0 \leqslant i \leqslant j \leqslant \ell(c) - 1$ tais que $c^{(i,j)} \in \mathcal{C}yc(G)$. Assim, considere os casos:

i) $i = 0\ \wedge\ j = \ell(c) - 1$: Neste caso, $c^{(i,j)} = c \in \mathcal{C}yc(G)$.

ii) $i \neq 0 \vee j \neq \ell(c) - 1$: Neste caso, $c^{(i,j)} \in \mathcal{C}yc(G)$ e de fato $i < j$, pois $\ell(c^{(i,j)}) > 1$. Além disso, observe que $c[i] = c[j]$. Assim, denotando $c = (v_0, \ldots, v_{n-1})$, onde $\ell(c) = n$,

tem-se que:

$$\begin{aligned} c &= (v_0, \ldots, v_i, \ldots, v_j, \ldots, v_{n-1}) \\ &\sim (v_i, \ldots, v_j, \ldots, v_{n-1}, v_1, \ldots, v_i) \\ &= \circ_w((v_i, \ldots, v_j), (v_j, \ldots, v_{n-1}, v_1, \ldots, v_i)) \end{aligned}$$

Assim, $c_1 = (v_i, \ldots, v_j) \in \mathcal{C}yc(G)$, $c_2 = (v_j, \ldots, v_{n-1}, v_1, \ldots, v_i) \in \mathcal{C}ir(G)$ e $c \sim \circ_w(c_1, c_2)$.

De (i) e (ii) segue a prova do lema. ■

3.1.2 Digrafos com Peso

Nesta seção a definição de digrafos com peso é introduzida. Um digrafo com peso G é um digrafo com uma função de medidas associada. Esta função designa um rótulo para cada aresta de G, que é chamado o peso da aresta.

Observação 3.1.1: A função de medida associada a um digrafo G será representada por $\boldsymbol{\mu}$, e a imagem de $\boldsymbol{\mu}$ é uma estrutura algébrica, representada por W, com uma operação binária + satisfazendo o seguinte:

- Associatividade: $\forall\, a, b, c \in W : (a + b) + c = a + (b + c)$;
- Identidade: $\exists\, id \in W,\ \forall\, a \in W : id + a = a + id = a$.

Definição 3.1.17 (ω-digrafos)**:** *Um digrafo G com peso (ou ω-digrafo), é uma tripla $\langle V_G, E_G, \boldsymbol{\mu} : E_G \to W\rangle$, onde $\langle V_G, E_G\rangle$ é um digrafo e $\boldsymbol{\mu} : E_G \to W$ é a função de medida do conjunto de arestas E_G para W. As imagens de $\boldsymbol{\mu}$ são os pesos do digrafo.*

As seguintes definições estendem a função $\boldsymbol{\mu}$ para caminhos no digrafo.

Definição 3.1.18: *Dados um ω-digrafo $G = \langle V_G, E_G, \boldsymbol{\mu} : E_G \to W\rangle$, um caminho $w \in \mathcal{W}_G$, e dois naturais $i, j < \ell(w)$, define-se a aplicação*

$$\boldsymbol{\mu}_{aux} : \mathcal{W}_G \times \{0, \ldots, \ell(w) - 1\} \times \{0, \ldots, \ell(w) - 1\} \to W$$

por:

$$\boldsymbol{\mu}_{aux}(w, i, j) := \begin{cases} id, & \text{se } j \leqslant i \\ \boldsymbol{\mu}_{aux}(w, i, j-1) + \boldsymbol{\mu}(w[j-1], w[j]), & \text{se } j > i \end{cases}$$

Definição 3.1.19 (Peso sobre caminhos)**:** *Dados um ω-digrafo $G = \langle V_G, E_G, \boldsymbol{\mu} : E_G \to W\rangle$ e um caminho $w \in \mathcal{W}_G$, atribui-se um peso w através da aplicação $\hat{\boldsymbol{\mu}} : \mathcal{W}_G \to W$ definida por:*

$$\hat{\boldsymbol{\mu}}(w) := \boldsymbol{\mu}_{aux}(w, 0, \ell(w) - 1)$$

Observe que o resultado de $\hat{\boldsymbol{\mu}}(w)$ é a "soma" (ordenada, pois não se tem que + é comutativa) de $\boldsymbol{\mu}(w[j-1], w[j])$, para $0 < j < \ell(w)$. Ou seja, é a "soma" dos pesos de cada aresta de w.

O seguinte lema estabelece que o peso de um caminho w segundo a aplicação $\hat{\boldsymbol{\mu}}$ pode ser decomposto.

Lema 3.1.20 (Decomposição do peso de um caminho)**:** *Sejam um ω-digrafo G, um caminho $w \in \mathcal{W}_G$ e $j < \ell(w)$. Então, $\hat{\boldsymbol{\mu}}(w) = \hat{\boldsymbol{\mu}}(w^{(0,j)}) + \hat{\boldsymbol{\mu}}(w^{(j,\ell(w)-1)})$.*

Prova: De fato, observe que, se $j = 0$, então segue da Definição 3.1.19 que $\hat{\boldsymbol{\mu}}(w^{(0,j)}) = id$ e além disso $w^{(j,\ell(w)-1)} = w$. Assim, suponha que $0 < j < \ell(w)$ e considere $w = (v_0, v_1, \ldots, v_j, \ldots, v_{n-1})$, onde $n = \ell(w)$. Logo,

$$\begin{aligned}\hat{\boldsymbol{\mu}}(w) &= \boldsymbol{\mu}(v_0, v_1) + \ldots + \boldsymbol{\mu}(v_{j-1}, v_j) + \boldsymbol{\mu}(v_j, v_{j+1}) + \ldots + \boldsymbol{\mu}(v_{n-2}, v_{n-1}) \\ &= \hat{\boldsymbol{\mu}}(v_0, v_1, \ldots, v_{j-1}, v_j) + \hat{\boldsymbol{\mu}}(v_j, v_{j+1}, \ldots, v_{n-2}, v_{n-1}) \\ &= \hat{\boldsymbol{\mu}}(w^{(0,j)}) + \hat{\boldsymbol{\mu}}(w^{(j,\ell(w)-1)})\end{aligned}$$

O que conclui a prova. ∎

3.2 A Tecnologia *Calling Context Graphs* (CCG)

A tecnologia *Calling Context Graphs* (CCG) foi introduzida em 2006 por Manolios e Vroon em [MV06] e se trata de uma estrutura de dados aplicada para análise de terminação de funções recursivas. A tecnologia CCG implementa a análise desenvolvida através do princípio de mudança de tamanho [LJBA01], introduzido em 2001 por Lee, Jones e Ben-Amram. Por sua vez, o princípio de mudança de tamanho estabelece que *um programa termina para qualquer entrada se toda sequência infinita de chamados recursivos levar a uma sequência infinita decrescente em algum conjunto de valores.* Basicamente um CCG é um digrafo com uma família de medidas, que abstrai o comportamento de uma definição recursiva da seguinte forma: cada elemento do conjunto de vértices, denominados *contextos de chamado*, está relacionado a um chamado recursivo na definição da função, e carrega informações sobre o respectivo chamado, como parâmetros formais e atuais, além de condições para que o chamado ocorra; cada aresta do CCG representa uma

possível sequência de dois chamados recursivos; assim, os ciclos e circuitos de um CCG aproximam possíveis sequências infinitas de chamados recursivos. Desta forma, a análise de terminação através de CCG's é desenvolvida da seguinte maneira: dada uma família de medidas sobre o domínio de uma relação bem fundada, deve-se mostrar que, para qualquer possível sequência infinita (de vértices), existe uma sequência correspondente de medidas infinitamente decrescente.

A tecnologia CCG foi implementada no assistente de prova ACL2, por Chamarthi *et al*, dando origem ao ACL2 Sedan [CDMV11]. Esta tecnologia também foi implementada em PVS por Muñoz e Ayala-Rincón.

Uma apresentação mais formal de CCG's exigiria começar por introduzir a sintaxe de uma linguagem funcional de primeira ordem. Contudo esta sintaxe será abordada apenas na Seção 5.1 para em seguida apresentar-se a formalização em PVS de CCG's. Aqui apenas uma apresentação minimal será desenvolvida. Em seguida algumas definições formais acerca de CCG's.

Notação: Nas definições a seguir será considerada uma função recursiva f, de aridade n. Os parâmetros formais de f serão denotados por $(x_1, \ldots, x_n)$ os parâmetros atuais em um dado chamado recursivo por $(e_1, \ldots, e_n)$; e $f(e_1, \ldots, e_n)$ representará um chamado recursivo de f em sua definição. É importante notar que os parâmetros atuais são funções dos parâmetros formais.

Definição 3.2.1 (Contexto de Chamado)**:** *Seja f uma função recursiva de aridade n, e $f(e_1, \ldots, e_n)$ um chamado recursivo na definição de f que ocorre sob determinadas condições* `cnd`*. Um contexto de chamado é uma tripla $\langle (x_1, \ldots, x_n), \texttt{cnd}, (e_1, \ldots, e_n) \rangle$.*

Notação: Contextos de chamado serão denotados por cc, cc_1, etc.

Em um contexto de chamado $cc = \langle (x_1, \ldots, x_n), \texttt{cnd}, (e_1, \ldots, e_n) \rangle$, a condição `cnd` é uma função dos parâmetros formais.

Exemplo 3.2.1 (Máximo divisor comum)**:** Considere a função $mdc : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ definida abaixo:

$$\begin{aligned} mdc(m,n) \;\; = \;\; & \texttt{IF} \;\; m = 0 \;\vee\; n = 0 \;\; \texttt{THEN} \;\; m + n \\ & \texttt{ELSIF} \;\; m \geqslant n \;\; \texttt{THEN} \;\; mdc(m-n, n) \;\; \texttt{ELSE} \;\; mdc(n, m) \end{aligned}$$

Observe que a função mdc tem dois chamados recursivos em sua definição: $mdc(m-n, n)$ e $mdc(n, m)$. A condição para que o primeiro ocorra é $\neg(m = 0 \;\vee\; n = 0) \;\wedge\; (m \geqslant n)$; e a condição para que o segundo ocorra é $\neg(m = 0 \;\vee\; n = 0) \;\wedge\; \neg(m \geqslant n)$. Assim, os contextos de chamado relacionados aos chamados recursivos que ocorrem na definição de

mdc são dois, a saber:

$$cc_1 \;=\; \langle\, (m,n),\; (\neg(m=0 \;\vee\; n=0) \;\wedge\; (m \geqslant n)),\; (m-n,n)\,\rangle$$

$$cc_2 \;=\; \langle\, (m,n),\; (\neg(m=0 \;\vee\; n=0) \;\wedge\; \neg(m \geqslant n)),\; (n,m)\,\rangle$$

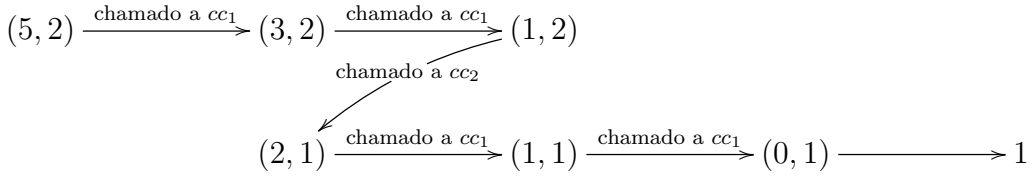
Na seguinte definição uma designação de valores para os parâmetros formais de uma definição recursiva será considerada. Assim, $\mathcal{V}ar$ é um conjunto de variáveis que contém os parâmetros formais da função, e $\mathcal{V}al$ um conjunto de valores onde as variáveis são mapeadas, e a aplicação $\beta : \mathcal{V}ar \to \mathcal{V}al$ representa uma designação de valores para as variáveis.

Definição 3.2.2 (Sequência de contextos bem formada)**:** *Dada uma definição de função recursiva f, de aridade n com parâmetros formais $\boldsymbol{x} = (x_1, \ldots, x_n)$, e uma sequência $s_{cc} = (cc_{i_1}, cc_{i_2}, cc_{i_3}, \ldots)$ de contextos de chamado de f, onde $cc_{i_k} = \langle \boldsymbol{x}, \mathtt{cnd}_{i_k}, \boldsymbol{e}_{i_k} \rangle$, com $\boldsymbol{e}_{i_k} = (e_1^{i_k}, e_2^{i_k}, \ldots, e_n^{i_k})$ e $\mathtt{cnd}_{i_k}$ representando os parâmetros atuais e as condições do chamado cc_{i_k}, respectivamente, para $k \geqslant 1$. Diz-se que s_{cc} é uma sequência bem formada se*

$$\exists\, \beta_{i_1} : \mathcal{V}ar \to \mathcal{V}al \;:\; \mathtt{cnd}_{i_1}(\beta_{i_1}(\boldsymbol{x})) \;\wedge\; \forall\, k \geqslant 1 : \mathtt{cnd}_{i_{k+1}}(\boldsymbol{e}_{i_k}(\beta_{i_k}(\boldsymbol{x}))),$$

onde $\beta_{i_k}(\boldsymbol{x}) = \boldsymbol{e}_{i_{k-1}}(\beta_{i_{k-1}}(\boldsymbol{x}))$.

Exemplo 3.2.2 (Máximo divisor comum)**:** Voltando ao exemplo da função *mdc*, considere a seguinte sequência de contextos $s_{cc} = (cc_1, cc_1, cc_2, cc_1, cc_1)$. Esta sequência é bem formada pois $\beta_1 : \mathcal{V}ar \to \mathcal{V}al$ definida por $\beta_1(m,n) = (5,2)$ é uma designação inicial que a torna possível. De fato, a partir de $(5,2)$ gera-se a seguinte sequência de valores:



Note que terminação de uma definição recursiva pode ser expressa em termos de sequências bem formadas de contextos de chamado da seguinte forma:

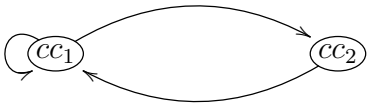
Teorema 3.2.3 (Teorema 1 de [MV06])**:** *Uma função recursiva f termina em todas as entradas se, e somente se, toda sequência bem formada de contextos de f é finita.*

A prova do teorema anterior segue das definições de contexto de chamado e de sequência de contextos bem formada.

Em seguida a definição de *Calling Context Graphs*.

Definição 3.2.4: *Um CCG é um digrafo $G = \langle V_G, E_G \rangle$, onde V_G é um conjunto de contextos de chamado, e para qualquer par de contextos $cc_1, cc_2 \in V_G$, se a sequência (cc_1, cc_2) é bem formada, então $(cc_1, cc_2) \in E_G$.*

Exemplo 3.2.3 (Máximo divisor comum)**:** Para a função *mdc* que, como visto no Exemplo 3.2.1, tem dois contextos de chamado cc_1 e cc_2, gera-se o seguinte CCG:



Observe que a sequência (cc_2, cc_2) não está presente no CCG, pois não é bem formada, uma vez que:

$$\forall\ \beta : \mathcal{V}ar \to \mathcal{V}al :\ \mathtt{cnd}_2(\beta(m,n)) \Rightarrow \neg \mathtt{cnd}_2(e_2(\beta(m,n)))$$

Note que, dada uma definição de uma função recursiva f e o seu respectivo CCG $G = \langle V_G, E_G \rangle$ toda sequência bem formada de contextos de chamado de V_G é um caminho em G, mas nem todo caminho de G é uma sequência bem formada de contextos, isto porque em um CCG somente se verifica boa formação local, enquanto dizer que uma sequência é bem formada significa boa formação em cada aresta da sequência.

Neste ponto, são adicionadas medidas ao CCG. Medidas são aplicações que mapeiam o conjunto de parâmetros formais de uma função em um conjunto com uma ordem bem fundada. O conjunto de medidas será representado por $\mathcal{F}$. A seguinte definição estabalece um mecanismo para comparar medidas de $\mathcal{F}$ aplicadas a dois vértices adjacentes do CCG.

Definição 3.2.5: *Seja G um CCG com uma aresta $e = (cc_1, cc_2)$, onde $cc_i = \langle \boldsymbol{x}, \mathtt{cnd}_i, \boldsymbol{e}_i \rangle$. Seja ainda $(S, \succ)$ uma estrutura bem fundada e considere um conjunto de medidas $\mathcal{F}$. Dadas duas medidas $\mu_1, \mu_2 \in \mathcal{F}$ para cc_1 e cc_2 respectivamente, uma função de medida para a aresta e, cujo papel é comparar as medidas μ_1 e μ_2, é dada pela seguinte aplicação $\phi : \mathcal{F} \times \mathcal{F} \to \{>, =, \times\}$ definida por:*

$$\phi(\mu_1, \mu_2) := \begin{cases} >, & \text{se } \forall\ \beta : \mathcal{V}ar \to \mathcal{V}al : \mu_1(\beta(\boldsymbol{x})) \succ \mu_2(\boldsymbol{e}_1(\beta(\boldsymbol{x}))) \\ =, & \text{se } \forall\ \beta : \mathcal{V}ar \to \mathcal{V}al : \mu_1(\beta(\boldsymbol{x})) \succcurlyeq \mu_2(\boldsymbol{e}_1(\beta(\boldsymbol{x}))) \\ \times, & \text{caso contrário} \end{cases}$$

Assim, a partir da definição anterior define-se um CCG bem fundado como segue:

Definição 3.2.6: *Seja $G = \langle V_G, E_G \rangle$ um CCG. Diz-se que G é bem fundado se existe uma estrutura bem fundada $(S, \succ)$ e um conjunto de medidas $\mathcal{F}$ sobre S tal que o seguinte vale: para todo caminho infinito $s_{cc} = (cc_{i_1}, cc_{i_2}, cc_{i_3}, \ldots)$ de contextos de G existe $k' \geqslant 1$ e uma*

sequência de medidas $(\mu_{i_{k'}}, \mu_{i_{k'+1}}, \ldots)$ *tal que para todo* $k > k'$, $\phi(\mu_{i_k}, \mu_{i_k+1}) \in \{>, =\}$, *e para infinitos* $k > k'$, $\phi(\mu_{i_k}, \mu_{i_k+1}) \in \{>\}$.

E assim o seguinte resultado é estabelecido:

Teorema 3.2.7 (Teorema 2 de [MV06])**:** *Seja G um CCG relacionado a uma função recursiva f. Se toda componente fortemente conexa maximal de G é bem fundada, então f termina em todas as entradas.*

Este teorema é provado, verificando-se que, se toda componente fortemente conexa maximal de G é bem fundada, então toda sequência bem formada de contextos de G é finita. Daí, segue do Teorema 3.2.3 que a função f é terminante.

3.3 PVS: Sintaxe e Semântica

O PVS[1] (*Prototype Verification System*) é um sistema de especificação e verificação que provê um ambiente integrado para desenvolvimento e análise de especificações formais, e suporta uma ampla gama de atividades envolvendo criação, análise, modificação, gerenciamento e documentação de teorias e provas. O PVS tem sido aplicado com sucesso em formalização de unificação de primeira ordem [AdMGA10, AGdMAR12, AGdMAR14], formalização de propriedades elaboradas em teoria de reescrita [GAR10, GAR09], certificação de protocolos criptográficos [ARR13], correção de sistemas de gerenciamento de tráfico áereo [NM12, NMHZH13], verificação de algoritmos e hardware [ORR$^+$96, NM14, Cyr94, GS97], formalizações de teorias matemáticas [DLMn09], extração de código formal [LMG09], dentre outras. O PVS é basicamente composto por uma *Linguagem de Especificação* fortemente integrada com um poderoso *Assistente de Prova Interativo*, um *Typechecker*, além de outras ferramentas como *bibliotecas de especificação*. Neste capítulo é apresentada uma visão geral sobre a semântica do PVS e as principais ferramentas utilizadas na especificação/formalização apresentada nos Capítulos 4, 5 e 6. Para mais detalhes sobre a semântica do PVS consultar [OS97], o qual é a base das seções 3.3.3 e 3.3.4. Também é possível encontrar, na página do sistema, manuais e documentos que descrevem em detalhes a linguagem de especificação do PVS, o provador e o sistema.

[1]Disponível em `http://pvs.csl.sri.com`

3.3.1 A Linguagem de Especificação do PVS

A *linguagem de especificação* do PVS é baseada em lógica de ordem superior simplesmente tipada, e é desenvolvida para permitir especificações suscintas e legíveis, além de construções efetivas de prova. Dentro de uma *teoria* os tipos podem ser definidos a partir de tipos mais básicos, como booleanos, números naturais, etc.

Exemplo 3.3.1: Por exemplo, observe o seguinte trecho de código pvs extraído da *teoria* `measures` a ser apresentada no Capítulo 4, onde alguns tipos são definidos:

```
measures : TYPE = Sign3


measure_matrix : TYPE = [below(N) -> [below(N) -> measures]]
```

Para definir o tipo `measures`, que são números tomados no conjunto $\{-1, 0, 1\}$, ou o tipo `measure_matrix`, que são matrizes quadradas cujas entradas são do tipo `measures`, foram empregados os tipos: `Sign3`, que são inteiros no conjunto $\{-1, 0, 1\}$; e o tipo `below(N)`, que são números naturais menores que ou iguais a $N - 1$.

Uma especificação em PVS consiste de uma coleção de *teorias*. E cada *teoria* consiste de um conjunto de símbolos para os nomes dos tipos e constantes introduzidas na *teoria*, além de axiomas, definições e teoremas associados.

3.3.2 O Assistente de Prova

O objetivo principal do assistente de prova do PVS é conferir suporte à construção de provas legíveis, isto é, o processo de verificação busca ser próximo de uma prova em papel e lápis. Logo, permite interação humana de forma que a prova pode ser facilmente entendida e comunicada a outras pessoas. No sentido de possibilitar que as provas sejam facilmente desenvolvidas, o assistente de prova do PVS provê uma coleção poderosa de comandos de prova. Quando combinados corretamente, tais comandos desenvolvem uma estratégia de prova onde se realiza raciocínios lógicos, proposicionais e aritméticos, com o uso de definições e lemas.

O assistente de prova do PVS foi desenvolvido com base na semântica usual de Gentzen, da Teoria da Prova. Isto significa que os objetivos em PVS são apresentados na forma de sequentes $\Sigma \vdash \Lambda$, onde Σ e Λ são sequências finitas de fórmulas.

O assistente de prova mantém uma árvore de prova durante a formalização de uma propriedade especificada, onde cada nó é um objetivo de prova. A formalização em curso é completa quando se obtém uma árvore de prova que seja completa, isto é, uma árvore

em que todas as folhas são axiomas. Cada nó da árvore é um objetivo de prova, representado por um sequente da forma $\Sigma \vdash \Lambda$, que consiste de uma sequência de fórmulas Σ denominadas antecedentes e uma sequência de fórmulas Λ denominadas consequentes. A interpretação de um sequente é que a conjunção dos antecedentes implica a disjunção dos consequentes, isto é, para $\Sigma = \{A_1, A_2, A_3, \ldots\}$ e $\Lambda = \{B_1, B_2, B_3, \ldots\}$, tem-se que $(A_1 \wedge A_2 \wedge A_3 \ldots) \supset (B_1 \vee B_2 \vee B_3 \ldots)$.

3.3.3 A Checagem de Tipos em PVS

Os tipos formam um mecanismo poderoso para detectar erros sintáticos e semânticos em uma especificação. O PVS explora este mecanismo através de uma operação de checagem de tipos, que é mantida pelo *typechecker*. A lógica expressiva do PVS proporciona uma boa integração entre o *typechecker* e o *assistente de prova*. O *typechecker* explora o poder dedutivo do *assistente de prova* para verificar automaticamente as condições de correção de tipos, ou TCC's (*type correctness conditions*), que são obrigações de prova geradas pela operação de checagem de tipos. Estas obrigações de prova surgem, por exemplo, quando realiza-se a checagem de tipos de um termo em confronto com um subtipo de um predicado. Tais obrigações também aparecem como sub-objetivos de prova durante uma formalização.

Em PVS os tipos básicos consistem de *Booleanos*, `bool`, e *números reais*, `real`. O PVS é uma linguagem de especificação fortemente tipada, onde os tipos são construídos a partir dos tipos básicos, através de funções e produtos de tipos, e expressões são construídas a partir das constantes e variáveis por meio de aplicações, abstrações e sequências.

A operação de checagem de tipos é desenvolvida dentro de um *contexto*. Em PVS, um contexto Γ é uma sequência de declarações, onde cada declaração é: ou uma declaração de tipo `s:TYPE`; ou uma declaração de constante `c:T`, onde `T` é um tipo; ou uma declaração de variável `x:VAR T`, onde `T` é um tipo. Esta operação pode ser vista como uma função parcial que associa a cada símbolo uma *espécie* que pode ser ou `TYPE`, ou `CONSTANT`, ou `VARIABLE` e um *tipo* a cada símbolo de constante e variável. De maneira mais formal, define-se:

Definição 3.3.1: *Seja Γ um contexto e s um símbolo com declaração D e r um símbolo qualquer. Então,*

1. $(\Gamma, s : D)(s) = D$

2. $r \neq s \Rightarrow (\Gamma, s : D)(r) = \Gamma(r)$

3. *Se s não é declarado em Γ, então $\Gamma(s)$ é indefinido.*

4. *Para qualquer símbolo s, a espécie de s em Γ é dada por $kind(\Gamma(s))$.*

5. *Se $kind(\Gamma(s))$ é* `CONSTANT` *ou* `VARIABLE`*, então o tipo de $\Gamma(s)$ é o tipo associado a s em Γ.*

As regras de tipos em uma teoria simplesmente tipada são dadas pela definição recursiva de uma função parcial τ que associa:

(i) a um termo a, bem tipado em relação a um contexto Γ, um tipo $\tau(\Gamma)(a)$.

(ii) a um tipo A, bem formado em relação a um contexto Γ, a palavra-chave `TYPE` como resultado de $\tau(\Gamma)(A)$.

(iii) a um contexto Δ, bem formado em relação a um contexto Γ, a palavra-chave `CONTEXT` como resultado de $\tau(\Gamma)(\Delta)$.

Normalmente, as regras de tipos são apresentadas como regras de inferência, mas em PVS uma apresentação funcional é mais apropriada, pois desta forma obtem-se uma argumentação de correção de provas mais natural e direta. Assim tem-se a seguinte definição para as regras de tipo, que em PVS representam a operação de *typechecking*.

Definição 3.3.2: *Regras de tipos*

$$
\begin{aligned}
\tau()(\{\}) &= \texttt{CONTEXT} \\
\tau()(\Gamma, s : \texttt{TYPE}) &= \texttt{CONTEXT},\ \textit{se}\ \Gamma(s)\ \textit{é indefinido}\ \ \textit{e}\ \tau()(\Gamma) = \texttt{CONTEXT} \\
\tau()(\Gamma, c : T) &= \texttt{CONTEXT},\ \textit{se}\ \Gamma(c)\ \textit{é indefinido},\ \tau(\Gamma)(T) = \texttt{TYPE} \\
&\quad\ \textit{e}\ \tau()(\Gamma) = \texttt{CONTEXT} \\
\tau()(\Gamma, x : \texttt{VAR}\ T) &= \texttt{CONTEXT},\ \textit{se}\ \Gamma(x)\ \textit{é indefinido},\ \tau(\Gamma)(T) = \texttt{TYPE} \\
&\quad\ \textit{e}\ \tau()(\Gamma) = \texttt{CONTEXT} \\
\tau(\Gamma)(s) &= \texttt{TYPE},\ \textit{se}\ kind(\Gamma(s)) = \texttt{TYPE} \\
\tau(\Gamma)([A \to B]) &= \texttt{TYPE},\ \textit{se}\ \tau(\Gamma)(A) = \tau(\Gamma)(B) = \texttt{TYPE} \\
\tau(\Gamma)([A_1, A_2]) &= \texttt{TYPE},\ \textit{se}\ \tau(\Gamma)(A_i) = \texttt{TYPE}\ \textit{para}\ 1 \leq i \leq 2 \\
\tau(\Gamma)(s) &= type(\Gamma(s)),\ \textit{se}\ kind(\Gamma(s)) \in \{\texttt{CONSTANT}, \texttt{VARIABLE}\} \\
\tau(\Gamma)(f\ a) &= B,\ \textit{se}\ \tau(\Gamma)(f) = [A \to B]\ \textit{e}\ \tau(\Gamma)(a) = A \\
\tau(\Gamma)(\lambda(x : T) : a) &= [T \to \tau(\Gamma, x : \texttt{VAR}\ T)(a)],\ \textit{se}\ \Gamma(x)\ \textit{é indefinido} \\
&\quad\ \textit{e}\ \tau(\Gamma)(T) = \texttt{TYPE} \\
\tau(\Gamma)((a_1, a_2)) &= [\tau(\Gamma)(a_1), \tau(\Gamma)(a_2)] \\
\tau(\Gamma)(p_i\ a) &= T_i,\ \textit{onde}\ \tau(\Gamma)(a) = [T_1, T_2]
\end{aligned}
$$

Exemplo 3.3.2: Seja Ω um contexto onde `bool` : `TYPE`, `TRUE` : `bool` e `FALSE` : `bool`. Assim, neste contexto as regras de tipo são dadas por:

$$\begin{aligned}
\tau()(\{\}) &= \texttt{TYPE} \\
\tau()(\Omega) &= \texttt{TYPE} \\
\tau(\Omega)([[\texttt{bool}, \texttt{bool}] \to \texttt{bool}]) &= \texttt{TYPE} \\
\tau(\Omega)((\texttt{TRUE}, \texttt{FALSE})) &= [\texttt{bool}, \texttt{bool}] \\
\tau(\Omega)(p_2(\texttt{TRUE}, \texttt{FALSE})) &= \texttt{bool} \\
\tau(\Omega)(\lambda(x : \texttt{boll}) : \texttt{TRUE}) &= [\texttt{bool} \to \texttt{bool}]
\end{aligned}$$

Um termo bem tipado s com um tipo designado por τ dentro de um contexto Γ é dito um termo de tipo $\tau(\Gamma)(s)$ no contexto Γ.

Note que nas regras de tipos a boa formação do contexto em questão não é explicitamente verificada, contudo tais regras preservam a boa formação do contexto em cada chamado recursivo, então se o contexto inicial é bem formado, todos os contextos intermediários o serão também.

O PVS ainda oferece o recurso de declarar tipos dependentes, isto é, alguns dos tipos dos parâmetros de uma especificação podem ser dependentes de parâmetros anteriores.

3.3.4 As Regras de Prova do PVS

As regras de prova do PVS são apresentadas em termos de cálculo de sequentes. Como mencionado na Seção 3.3.2, um sequente é da forma $\Sigma \vdash_\Gamma \Lambda$, onde Γ é o contexto, Σ é o conjunto das fórmulas que compõem o antecedente e Λ representa o conjunto das fórmulas que compõem o consequente. Sobre um sequente desta forma deve-se fazer a seguinte leitura: a conjunção das fórmulas de Σ implica a disjunção das fórmulas de Λ.

Regras Estruturais

Com as regras estruturais pode-se rearranjar um sequente ou enfraquecê-lo, introduzindo novas fórmulas na conclusão.

A regra (W), apresentada em (3.4), representa uma poderosa regra de enfraquecimento. Todas as regras estruturais podem ser expressas em termos desta. Esta regra permite derivar um sequente mais fraco de um mais forte, por meio de introdução de fórmulas no antecedente ou no consequente. Este fato é expresso pela condição que se impõe de que $\Sigma_1 \subseteq \Sigma_2$ e $\Lambda_1 \subseteq \Lambda_2$.

$$\frac{\Sigma_1 \vdash_\Gamma \Lambda_1}{\Sigma_2 \vdash_\Gamma \Lambda_2}\ (W), \quad \text{se } \Sigma_1 \subseteq \Sigma_2 \text{ e } \Lambda_1 \subseteq \Lambda_2 \tag{3.4}$$

As regras de contração $(C\ \vdash)$ e $(\vdash\ C)$, apresentadas abaixo, permitem substituir várias ocorrências de uma mesma fórmula no antecedente ou no consequente por uma única ocorrência.

$$\frac{a, a, \Sigma \vdash_\Gamma \Lambda}{a, \Sigma \vdash_\Gamma \Lambda}\ (C\ \vdash) \qquad\qquad \frac{\Sigma \vdash_\Gamma a, a, \Lambda}{\Sigma \vdash_\Gamma a, \Lambda}\ (\vdash\ C)$$

As regras de comutação $(X\ \vdash)$ e $(\vdash\ X)$, afirmam que a ordem das fórmulas tanto do antecedente quanto do consequente é insignificante.

$$\frac{\Sigma_1, b, a, \Sigma_2 \vdash_\Gamma \Lambda}{\Sigma_1, a, b, \Sigma_2 \vdash_\Gamma \Lambda}\ (X\ \vdash) \qquad\qquad \frac{\Sigma \vdash_\Gamma \Lambda_1, b, a, \Lambda_2}{\Sigma \vdash_\Gamma \Lambda_1, a, b, \Lambda_2}\ (\vdash\ X)$$

Regra de Corte

A regra de corte (Cut) pode ser usada para introduzir a análise de casos sobre uma fórmula a, dentro de uma prova de um sequente da forma $\Sigma \vdash_\Gamma \Lambda$. Isto leva a dois sub-objetivos da forma $\Sigma, a \vdash_\Gamma \Lambda$ e $\Sigma \vdash_\Gamma a, \Lambda$, que podem ser vistos como assumindo a em um ramo da prova e $\neg a$ no outro ramo da prova.

$$\frac{(\tau(\Gamma)(a) \sim \texttt{bool})_\Gamma \quad \Sigma, a \vdash_\Gamma \Lambda \quad \Sigma \vdash_\Gamma a, \Lambda}{\Sigma \vdash_\Gamma \Lambda}\ (Cut)$$

Regras para Axiomas Proposicionais

Na regra (Ax), simplesmente tem-se afirmação trivial de que a segue de a.

$$\frac{}{\Sigma, a \vdash_\Gamma a, \Lambda}\ (Ax)$$

As regras $(\texttt{FALSE}\ \vdash)$ e $(\vdash\ \texttt{TRUE})$, afirmam que, se em um dado sequente existe uma ocorrência de `FALSE` no antecedente ou uma ocorrência de `TRUE` no consequente, então este sequente é um axioma.

$$\frac{}{\Sigma, \texttt{FALSE} \vdash_\Gamma \Lambda}\ (\texttt{FALSE}\ \vdash) \qquad\qquad \frac{}{\Sigma \vdash_\Gamma \texttt{TRUE}, \Lambda}\ (\vdash\ \texttt{TRUE})$$

Regras de Contexto

Algumas fórmulas valem em um contexto simplesmente porque elas já fazem parte do contexto, ou como uma fórmula mesmo ou como uma declaração de constante. Nisto consiste a assertiva das regras $(ContextFormula)$ e $(ContextDefinition)$ abaixo.

$$\frac{}{\vdash_\Gamma a}\ (ContextFormula), \quad \text{se } a \text{ é uma fórmula em } \Gamma.$$

$$\frac{}{\vdash_\Gamma s = a}\ (ContextDefinition), \quad \text{se } s : T = a \text{ é uma definição de constante em } \Gamma$$

Um contexto Γ pode ser estendido, através das regras $(Context \vdash)$ e $(\vdash Context)$, por fórmulas no antecedente ou negação de fórmulas no consequente.

$$\frac{\Sigma, a \vdash_{\Gamma,a} \Lambda}{\Sigma, a \vdash_\Gamma \Lambda}\ (Context \vdash) \qquad \frac{\Sigma \vdash_{\Gamma,\neg a} a, \Lambda}{\Sigma \vdash_\Gamma a, \Lambda}\ (\vdash Context)$$

A regra $(ContextW)$, é uma regra de enfraquecimento do contexto que é bastante útil, pois mostra que uma derivação é monótona em relação a um contexto.

$$\frac{\Sigma \vdash_\Gamma \Lambda}{\Sigma \vdash_{\Gamma'} \Lambda}\ (ContextW), \quad \text{se } \Gamma \text{ é um prefixo de } \Gamma'$$

Regras Condicionais

As regras $(IF \vdash)$ e $(\vdash IF)$ têm o objetivo de eliminar as ocorrências de `IF-THEN-ELSE` em uma prova. Contudo estas regras não são usuais pois elas aumentam o contexto.

$$\frac{\Sigma, a, b \vdash_{\Gamma,a} \Lambda \quad \Sigma, c \vdash_{\Gamma,\neg a} a, \Lambda}{\Sigma, \mathtt{IF}(a, b, c) \vdash_\Gamma \Lambda}\ (IF \vdash) \qquad \frac{\Sigma, a \vdash_{\Gamma,a} b, \Lambda \quad \Sigma \vdash_{\Gamma,\neg a} a, c, \Lambda}{\Sigma \vdash_\Gamma \mathtt{IF}(a, b, c), \Lambda}\ (\vdash IF)$$

Regras de Igualdade

As regras de transitividade e simetria para a igualdade podem ser derivadas das regras de igualdade $(Refl)$ e $(Repl)$ abaixo. Nestas regras a notação $a[e]$ indica uma ou mais ocorrências de e na fórmula a, tal que não existem ocorrências de variáveis livres em e. Similarmente, a notação $\Lambda[e]$ indica ocorrências de e em Λ.

$$\frac{}{\Sigma \vdash_\Gamma a = a, \Lambda}\ (Refl) \qquad \frac{a = b, \Sigma[b] \vdash_\Gamma \Lambda[b]}{a = b, \Sigma[a] \vdash_\Gamma \Lambda[a]}\ (Repl)$$

Regras de Igualdade Booleana

Na regra (*Repl* `TRUE`) tem-se a asserção de que uma fórmula a no antecedente pode ser tratada como uma fórmula de igualdade no antecedente da forma $a = \texttt{TRUE}$. Similarmente a regra (*Repl* `FALSE`) estabelece que uma fórmula a no consequente pode ser vista como uma igualdade da forma $a = \texttt{FALSE}$, no antecedente. Já a regra `TRUE-FALSE`, afirma que as constantes booleanas `TRUE` e `FALSE` são distintas.

$$\frac{\Sigma[\texttt{TRUE}], a \vdash_\Gamma \Lambda[\texttt{TRUE}]}{\Sigma[a], a \vdash_\Gamma \Lambda[a]}\ (Refl\ \texttt{TRUE}) \qquad \frac{\Sigma[\texttt{FALSE}], a \vdash_\Gamma \Lambda[\texttt{FALSE}]}{\Sigma[a] \vdash_\Gamma a, \Lambda[a]}\ (Repl\ \texttt{FALSE})$$

$$\frac{}{\Sigma, \texttt{TRUE} = \texttt{FALSE} \vdash_\Gamma \Lambda}\ (\texttt{TRUE} - \texttt{FALSE})$$

Regras de Redução

As regras de redução (β) e (π) são axiomas de igualdade que possibilitam realizar simplificações óbvias, para aplicações envolvendo lambda abstrações e projeção de produtos.

$$\frac{}{\vdash_\Gamma (\lambda(x:T):a)(b) = a[b/x]}\ (\beta) \qquad \frac{}{\vdash_\Gamma p_i(a_1, a_2) = a_i}\ (\pi), \quad \text{para } i = 1, 2$$

Regras de Extensionalidade

As regras de extensionalidade $(FunExt)$ e $(TupExt)$, são regras que estabelecem igualdade para expressões funcionais e de produto, respectivamente. A regra de extensionalidade para funções introduz uma constante de Skolem s para estabelecer que duas funções f e g são iguais quando os resultados das aplicações destas funções a um argumento qualquer de s são iguais. A regra de extensionalidade para produto diz que dois produtos são iguais se as suas projeções correspondentes são iguais.

$$\frac{\Sigma \vdash_{\Gamma, s:A} (f\ s) =_{B[s/x]} (g\ s), \Lambda}{\Sigma \vdash_\Gamma f =_{[x:A \to B]} g, \Lambda}\ (FunExt), \quad \text{se } \Gamma(s) \text{ é indefinido}$$

$$\frac{\Sigma \vdash_\Gamma p_1(a) =_{T_1} p_1(b), \Lambda \quad \Sigma \vdash_\Gamma p_2(a) =_{T_2[(p_1\ a)/x]} p_2(b), \Lambda}{\Sigma \vdash_\Gamma a =_{[x:T_1 T_2]} b, \Lambda}\ (TupExt)$$

Regra de Restrição de Tipo

A regra de restrição de tipo ($Typepred$), é usada para suprir a necessidade de uma regra que introduza a restrição de tipo sobre um termo como uma fórmula no antecedente de um dado sequente.

$$\frac{\tau(\Gamma)(a) = A \qquad \pi(A)(a), \Sigma \vdash_\Gamma \Lambda}{\Sigma \vdash_\Gamma \Lambda} \ (Typepred)$$

Exemplos de formalização em PVS com aplicação das regras de prova serão apresentadas na Seção 4.5.

Capítulo 4

Grafos com Matrizes de Medida

Neste capítulo apresenta-se uma aplicação interessante de ω-digrafos, utilizando esta estrutura de dados na análise de terminação de definições recursivas, assim como CCGs. Grafos com Matrizes de Medida (ou *Matrix Weighted Graphs - MWG*) são ω-digrafos, onde a função de medida sobre as arestas é escolhida de modo a modelar o comportamento dos parâmetros da definição recursiva. Para uma dada função recursiva, os digrafos nas definições de MWG e CCG são idênticos, contudo o espaço de medidas de um MWG é o espaço de matrizes quadradas, onde cada matriz associada a uma dada aresta informa todas as possíveis comparações da família de medidas associada ao CCG.

As definições e demonstrações apresentadas neste capítulo correspondem às definições especificadas e às demonstrações formalizadas em PVS como parte da *teoria* `mwg`, cuja hierarquia pode ser visualizada na Figura 4.1 e será apresentada na próxima seção.

Neste capítulo apresenta-se na Seção 4.1 a estrutura hierárquica da especificação de MWG bem como uma justificativa intuitiva para a abstração presente na especificação e formalização da nova álgebra desenvolvida para tratar MWG que será apresentada na Seção 4.2; na Seção 4.3 é apresentada a definição formal de MWG's e a semântica operacional da terminação para esta estrutura; na Seção 4.4 dois critérios de terminação para MWG são descritos; finalmente na Seção 4.5 apresenta-se um pouco da formalização em PVS dos critérios descritos na seção anterior.

4.1 Especificação de MWG

A especificação dessa nova estrutura denominada MWG foi desenvolvida para tratar definições recursivas e principalmente formalizar critérios de terminação. Contudo em nenhum momento na especificação da *teoria* em PVS serão encontrados termos como "pro-

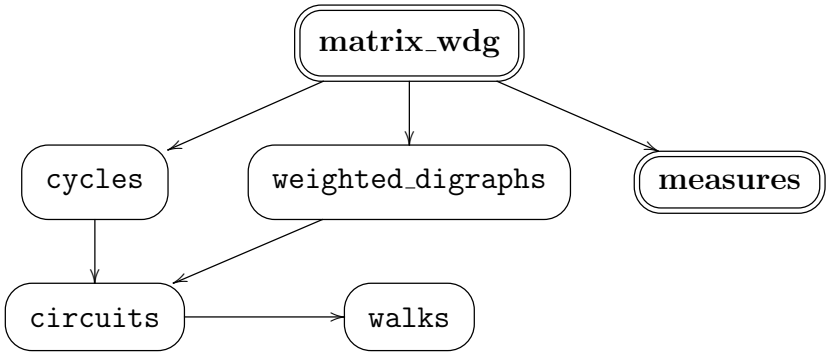


Figura 4.1: Hierarquia da *sub-teoria* `matrix_wdg`

grama" ou "para", da maneira como tais termos estão presentes no Capítulo 2. Mas todas estas ideias estão presentes na *teoria* de uma maneira abstrata, e o objetivo desta seção é apresentar uma intuição da conexão entre o que se encontra na especificação e na formalização e o objeto de estudo que é a formalização de critérios para verificação de terminação de linguagens funcionais de primeira ordem.

Apresenta-se a estrutura hierárquica da especificação de MWG. A Figura 4.1 mostra as *teorias* `matrix_wdg` e `measures` em destaque a serem apresentadas neste capítulo e as *sub-teorias* (da *teoria* `digraphs`) sobre as quais a especificação e formalização de `matrix_wdg` está construída. A descrição completa das *teorias* `cycles`, `circuits`, `walks` e `weighted_digraphs` é encontrada nas Seções 3.1 e 3.1.2.

Como pode ser observado na definição de ω-digrafos (Definição 3.1.17), a função peso $\boldsymbol{\mu}$, definida no conjunto das arestas do ω-digrafo, toma valores em um conjunto (denotado por W) que possui a estrutura de um monoide, ou seja, é um conjunto com uma operação binária associativa e um elemento neutro. A *teoria* `measures` consiste da especificação de uma estrutura de monoide, que neste caso será um conjunto especial de matrizes-quadradas $N \times N$, denotado por $\mathcal{M}_{N\times N}$, e da formalização de algumas propriedades sobre tais matrizes incluindo a prova de que a operação binária definida é associativa e a prova de existência do elemento neutro; a *teoria* `measures` é apresentada na Seção 4.2. Tendo definido formalmente a estrutura de monoide $\mathcal{M}_{N\times N}$ é possível aplicar a tecnologia geral para tratar digrafos com peso, desenvolvida na *teoria* `weighted_digraphs`, na formalização de critérios de terminação em MWG's. Observe que tomar W, como definido em `weighted_digraphs`, igual a $\mathcal{M}_{N\times N}$, como definido em `measures`, dá origem a nova estrutura denominada MWG; a especificação/formalização dos critérios de terminação em MWG's é apresentada na Seção 4.4.

A fim de desenvolver um ambiente formal com ferramentas matemáticas onde a nova estrutura de dados MWG pudesse ser caracterizada através da formalização de proprie-

dades relevantes, como por exemplo critérios de terminação, uma nova (porém padrão) álgebra de matrizes foi especificada/formalizada. As definições e formalizações desta álgebra apresentam propriedades que abstraem o comportamento dos objetos que se deseja modelar, no caso funções recursivas de primeira-ordem, e uma conexão mais formal entre o que se apresenta nesta álgebra e os objetos reais do estudo é assunto dos Capítulos 5 e 6.

A álgebra apresentada neste capítulo é construída sobre o conjunto $\{-1, 0, 1\}$. Para entender o que cada um desses números representa é necessário falar um pouco sobre o que se deseja abstrair. Com este objetivo, destacam-se alguns pontos (que não estão presentes concretamente na especificação/formalização da álgebra, mas sim abstratamente), através de uma descrição dos elementos de um MWG, que são os mesmos elementos de um ω-digrafo: vértices, arestas e uma função de medida sobre arestas.

vértices: Cada vértice v do MWG representa um chamado recursivo na definição da função, da mesma forma que um contexto de chamado na Definição 3.2.1. Assim, cada vértice tem a informação implícita de um *estado* na computação com parâmetros formais, condições sobre os parâmetros e parâmetros atuais, que são funções dos formais em novos parâmetros;

arestas: Da mesma forma que em um ω-digrafo, arestas são pares de vértices conectados. Portanto, assim como apresentado na tecnologia CCG (Seção 3.2), a presença de uma aresta $e = (v_i, v_j)$ significa que é possível realizar o chamado representado por v_j logo após o chamado representado por v_i;

medida: Na tecnologia CCG a análise de terminação envolve medidas sobre um domínio de uma relação bem fundada. Assim, considere uma família $\mathcal{F} = \{\mu_0, \ldots, \mu_{N-1}\}$ de medidas. O peso de cada aresta $e = (v_1, v_2)$ de um MWG será uma $N \times N$-matriz, onde N é a cardinalidade da família $\mathcal{F}$. Desta forma, cada entrada m_{ij}, $i, j = 0, \ldots N-1$, da matriz será um número em $\{-1, 0, 1\}$ obtido da seguinte maneira:

$$m_{ij} := \begin{cases} 1, & \text{se } \mu_i(v_1) > \mu_j(v_1); \\ 0, & \text{se } \mu_i(v_1) \geqslant \mu_j(v_1); \\ -1, & \text{caso contrário.} \end{cases} \tag{4.1}$$

Onde $\mu_k(v_1)$, $k = i, j$, representa o resultado de aplicar a medida μ_k aos parâmetros formais ou atuais do chamado representado pelo vértice v_1.

Observe que a ideia intuitiva apresentada na Equação 4.1 é:

- Se o resultado encontrado é igual a 1, então existe uma medida que decresce;
- Se o resultado encontrado é igual a 0, então existe uma medida menor ou igual mas não é possível responder se existe uma medida que decresce;
- Se o resultado encontrado é igual a -1, então não é possível responder se existe uma medida menor ou igual ou que decresce.

4.2 Uma Álgebra Para Matrizes de Medidas

Tendo em mente os aspectos que se deseja abstrair, apresentam-se as definições formais relacionadas a esta nova álgebra de matrizes.

Definição 4.2.1 (Medidas)**:** *Uma medida é um número em* $\{-1, 0, 1\}$.

Definição 4.2.2 (Adição e Máximo de Medidas)**:** *A adição de medidas é definida como uma operação binária* $+ : \{-1, 0, 1\} \times \{-1, 0, 1\} \to \{-1, 0, 1\}$ *satisfazendo as seguintes equações:*

$$-1 + x := -1,\ x + (-1) := -1,\ 0 + x := x,\ x + 0 := x \ e\ 1 + 1 := 1$$

A relação de máximo sobre medidas é dada pela relação binária comutativa usual de máximo sobre $\{-1, 0, 1\}$*, de acordo com a ordem* $-1 < 0 < 1$*: Isto é,*

$$max(x, 1) := 1,\ max(x, -1) := x \ e\ max(0, 0) := 0$$

Note que a adição e a relação de máximo são operadores binários associativos e comutativos. Portanto, as estruturas dadas por $(\{-1, 0, 1\}, +, 0)$ e $(\{-1, 0, 1\}, max, -1)$ são monoides comutativos.

A relação de máximo pode ser estendida a vetores de medidas da seguinte maneira:

Definição 4.2.3 (Máximo de um Vetor)**:** *Seja* $\mathbf{A} = [a_0 \dots a_{N-1}]$ *um vetor sobre* $\{-1, 0, 1\}$*, então o elemento máximo de* $\mathbf{A}$*, denotado por* $max(\mathbf{A})$*, é definido por:*

$$max(\mathbf{A}) := \{a \mid \forall i < N,\ \mathbf{A}(i) \leqslant a\ \wedge\ \exists j < N,\ \mathbf{A}(j) = a\}$$

Notação: A seguinte notação será adotada para o máximo de um vetor de medidas: dado $\mathbf{A} = [a_0\ \dots\ a_{N-1}]$ de comprimento N, o elemento máximo de $\mathbf{A}$ é denotado por $max(\mathbf{A}) = \underset{0 \leqslant k < N}{max}\{a_k\}$.

Observação 4.2.1: Dadas duas medidas a e b, a seguinte situação será frequentemente considerada: (i) $a \geqslant 0 \wedge b \geqslant 0 \wedge (a = 1 \vee b = 1)$. Mas observe que isto pode ser reescrito de maneira muito mais simples e elegante por: (ii) $a + b = 1$. Portanto a expressão (ii) será preferencialmente usada ao invés de (i). Assim como $a + b \geqslant 0$, ao invés de $a \geqslant 0 \wedge b \geqslant 0$.

Lema 4.2.4: *Para todo vetor de medidas* $\mathbf{A} = [a_0 \dots a_{N-1}]$ *e para toda medida* b*, vale a seguinte igualdade:*

$$max(\mathbf{A}) + b = \underset{0 \leqslant k < N}{max}\{a_k + b\}$$

Prova: Com efeito, como $max(\mathbf{A}) + b = \underset{0 \leqslant k < N}{max}\{a_k\} + b$, o seguinte fato deve ser verificado: $\underset{0 \leqslant k < N}{max}\{a_k\} + b = \underset{0 \leqslant k < N}{max}\{a_k + b\}$. Assim, considere os possíveis casos para b:

$b = -1$: $\underset{0 \leqslant k < N}{max}\{a_k + b\} = \underset{0 \leqslant k < N}{max}\{a_k + (-1)\} = \underset{0 \leqslant k < N}{max}\{-1\} = -1$ e
$\underset{0 \leqslant k < N}{max}\{a_k\} + b = \underset{0 \leqslant k < N}{max}\{a_k\} + (-1) = -1$

$b = 0$: $\underset{0 \leqslant k < N}{max}\{a_k + b\} = \underset{0 \leqslant k < N}{max}\{a_k + 0\} = \underset{0 \leqslant k < N}{max}\{a_k\}$ e
$\underset{0 \leqslant k < N}{max}\{a_k\} + b = \underset{0 \leqslant k < N}{max}\{a_k\} + 0 = \underset{0 \leqslant k < N}{max}\{a_k\}$

$b = 1$: neste caso, suponha que $\underset{0 \leqslant k < N}{max}\{a_k + b\} = \underset{0 \leqslant k < N}{max}\{a_k + 1\} = a_{k_1} + 1$ e $\underset{0 \leqslant k < N}{max}\{a_k\} + b = \underset{0 \leqslant k < N}{max}\{a_k\} + 1 = a_{k_2} + 1$. Além disso, vale que para todo k, $0 \leqslant k < N$, $a_k + 1 \leqslant a_{k_1} + 1$ e $a_k \leqslant a_{k_2}$. Em particular, tem-se que $a_{k_2} + 1 \leqslant a_{k_1} + 1$ e $a_{k_1} \leqslant a_{k_2}$. Portanto, considerando as possibilidades para a_{k_1} e a_{k_2}, concluí-se que $a_{k_1} + 1 = a_{k_2} + 1$.

Logo, tem-se que $\underset{0 \leqslant k < N}{max}\{a_k + b\} = \underset{0 \leqslant k < N}{max}\{a_k\} + b$ em todos os casos. ∎

Definição 4.2.5 (Matrizes de medida)**:** *Uma matriz de medida é uma matriz quadrada cujas entradas são medidas.*

Notação: Matrizes de medida são representadas por letras maiúsculas $\mathbf{M}$, $\mathbf{A}$, $\mathbf{B}$, etc. Suas respectivas entradas são representadas pelas letras minúsculas correspondentes com sub-índices m_{ij}, a_{ij}, b_{ij}, etc; ou ainda por $(\mathbf{M})_{ij}$, $(\mathbf{A})_{ij}$, $(\mathbf{B})_{ij}$, etc. Além disso, convenciona-se que matrizes de medida são $N \times N$-matrizes e o espaço de tais matrizes é denotado por $\mathcal{M}_{N \times N}$.

A fim de ter uma álgebra de matrizes que satisfaça as condições da Observação 3.1.1, e que consequentemente forneça um espaço de matrizes de medida $\mathcal{M}_{N \times N}$ adequado à caracterização de ω-digrafos, é necessário definir uma operação associativa sobre $\mathcal{M}_{N \times N}$ e uma (matriz de medida) identidade em $\mathcal{M}_{N \times N}$.

A operação $* : \mathcal{M}_{N \times N} \times \mathcal{M}_{N \times N} \to \mathcal{M}_{N \times N}$ é definida similarmente à multiplicação

usual de matrizes, porém usando a operação + e a relação *max* sobre medidas, da seguinte forma:

Definição 4.2.6 (Multiplicação de matrizes de medida)**:** *Sejam* $\mathbf{A}, \mathbf{B} \in \mathcal{M}_{N\times N}$ *matrizes de medida, a multiplicação* $\mathbf{A} * \mathbf{B}$ *é definida por:*

$$\forall\, i,j \in \{0,\ldots,N-1\},\ (\mathbf{A}*\mathbf{B})_{ij} := \max_{0\leqslant k<N}\{a_{ik}+b_{kj}\}.$$

Definição 4.2.7 (Matriz de Medida **Id**)**:** *Defina a matriz de medida, denotada por* **Id**, *cujas entradas sejam tais que:*

$$id_{ij} = \begin{cases} 0, & \text{se } i = j \\ -1, & \text{se } i \neq j \end{cases}$$

para i *e* j *variando em* $\{0,\ldots,N-1\}$.

Lema 4.2.8 (Matriz identidade)**:** *A matriz* **Id** *definida acima é identidade à direita e à esquerda em* $\mathcal{M}_{N\times N}$*, isto é, se* $\mathbf{A} \in \mathcal{M}_{N\times N}$ *então* $\mathbf{Id} * \mathbf{A} = \mathbf{A} * \mathbf{Id} = \mathbf{A}$.

Prova: De fato, pela Definição 4.2.6, tem-se que:

$$\forall\, i,j \in \{0,\ldots,N-1\},\ (\mathbf{Id}*\mathbf{A})_{ij} = \max_{0\leqslant k<N}\{id_{ik}+a_{kj}\} \qquad (\star)$$

Isto significa que existe um $k_1 \in \{0,\ldots,N-1\}$ tal que $(\mathbf{Id}*\mathbf{A})_{ij} = id_{ik_1} + a_{k_1 j}$, para cada $i,j \in \{0,\ldots,N-1\}$. Considere as seguintes possibilidades sobre k_1:

$$\begin{array}{rcll} k_1 = i & \Rightarrow & id_{ii} = 0, & \text{Definição 4.2.7} \\ & \Rightarrow & id_{ii} + a_{ij} = a_{ij}, & \text{Definição 4.2.2} \\ k_1 \neq i & \Rightarrow & id_{ik_1} = -1, & \text{Definição 4.2.7} \\ & \Rightarrow & id_{ik_1} + a_{k_1 j} = -1, & \text{Definição 4.2.2} \end{array}$$

Além disso, note que $a_{ij} \geqslant -1$. Portanto, o valor de k para o qual a soma $id_{ik} + a_{kj}$ é máxima, é sempre igual a i para todos $i,j \in \{0,\ldots,N-1\}$. Assim, pela Equação $(\star)$

$$\begin{array}{rl} \forall\, i,j \in \{0,\ldots,N-1\},\quad (\mathbf{Id}*\mathbf{A})_{ij} & = id_{ii} + a_{ij} \\ & = a_{ij} \\ & = (\mathbf{A})_{ij} \end{array}$$

Portanto, $\mathbf{Id} * \mathbf{A} = \mathbf{A}$. De maneira análoga, verifica-se que $\mathbf{A} * \mathbf{Id} = \mathbf{A}$. ∎

Lema 4.2.9 (Associatividade da multiplicação de matrizes de medida)**:** *A operação* $*$ *é associativa, isto é, se* $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C} \in \mathcal{M}_{N\times N}$*, então* $(\mathbf{A}*\mathbf{B})*\mathbf{C} = \mathbf{A}*(\mathbf{B}*\mathbf{C})$.

Prova: De fato, para todos $i, j \in \{0, \ldots, N-1\}$ vale que:

$$\begin{aligned}
((\mathbf{A} * \mathbf{B}) * \mathbf{C})_{ij} &= \max_{0 \leqslant k < N} \{(\mathbf{A} * \mathbf{B})_{ik} + c_{kj}\}, && \text{Definição 4.2.6} \\
&= \max_{0 \leqslant k < N} \{\max_{0 \leqslant l < N} \{a_{il} + b_{lk}\} + c_{kj}\}, && \text{Definição 4.2.6} \\
&= \max_{0 \leqslant k < N} \{\max_{0 \leqslant l < N} \{a_{il} + b_{lk} + c_{kj}\}\}, && \text{Lema 4.2.4}
\end{aligned}$$

e

$$\begin{aligned}
(\mathbf{A} * (\mathbf{B} * \mathbf{C}))_{ij} &= \max_{0 \leqslant l < N} \{a_{il} + (\mathbf{B} * \mathbf{C})_{lj}\}, && \text{Definição 4.2.6} \\
&= \max_{0 \leqslant l < N} \{a_{il} + \max_{0 \leqslant k < N} \{b_{lk} + c_{kj}\}\}, && \text{Definição 4.2.6} \\
&= \max_{0 \leqslant l < N} \{\max_{0 \leqslant k < N} \{a_{il} + b_{lk} + c_{kj}\}\}, && \text{Lema 4.2.4}
\end{aligned}$$

Assim, conclui-se que: $\forall\, i, j \in \{0, \ldots, N\} : ((\mathbf{A} * \mathbf{B}) * \mathbf{C})_{ij} = (\mathbf{A} * (\mathbf{B} * \mathbf{C}))_{ij}$ ■

Os Lemas 4.2.8 e 4.2.9 garantem que a álgebra dada por $(\mathcal{M}_{N \times N}, *, \mathbf{Id})$ é um monoide.

Note que, em uma definição recursiva, se existe uma medida definida sobre o domínio de uma relação bem fundada que decresce nos parâmetros da definição, então é possível garantir terminação de tal definição recursiva. Tendo em mente que, de acordo com a motivação apresentada na Equação 4.1, se a matriz de medida de uma caminho no digrafo tem na posição (i, i), para $0 \leqslant i < N$, entrada igual a 1, então isto representa que a medida μ_i da família de medidas $\mathcal{F}$ decresce. Assim, no que segue define-se o que vem a ser uma matriz de medida positiva, o que mais tarde será crucial para a definição da semântica de terminação para um grafo com matrizes de medida.

Definição 4.2.10 (Matriz de medida positiva)**:** *Seja* $\mathbf{M} \in \mathcal{M}_{N \times N}$ *uma matriz de medida. Diz-se que* $\mathbf{M}$ *é positiva sempre que existir* j*,* $0 \leqslant j < N$*, tal que* $(\mathbf{M})_{jj} = 1$*.*

A seguinte definição introduz a nomenclatura utilizada para tratar posições $(i, j) \in \{0, \ldots, N-1\} \times \{0, \ldots, N-1\}$ de matrizes cuja respectiva entrada em (i, j) seja 0 ou 1.

Definição 4.2.11 (Matriz com posição definida ou positiva)**:** *Sejam* $\mathbf{M} \in \mathcal{M}_{N \times N}$ *e* $i, j \in \{0, \ldots, N-1\}$*. Assim, adota-se a seguinte notação para as situações descritas:*

- *se* $(\mathbf{M})_{ij} \geqslant 0$*, então* (i, j) *é uma posição definida;*
- *se* $(\mathbf{M})_{ij} = 1$*, então* (i, j) *é uma posição positiva.*

Vários resultados envolvendo matrizes de medida positivas, onde basicamente são estabelecidas relações entre as posições de matrizes resultantes da multiplicação de outras matrizes satisfazendo certas propriedades, são formalizados na *teoria* `measures` em PVS. Nesta apresentação, apenas os resultados mais relevantes e necessários à discussão que se segue ao longo do trabalho são apresentados.

O lema seguinte trata de posições positivas em uma multiplicação de matrizes de medida.

Lema 4.2.12 (Multiplicação de matrizes com posição positiva)**:** *Sejam* $\mathbf{A}, \mathbf{B} \in \mathcal{M}_{N\times N}$ *e* $i, j \in \{0, \ldots, N-1\}$*. Assim,*

$$(\mathbf{A} * \mathbf{B})_{ij} = 1 \iff \exists\, k \in \{0, \ldots, N-1\} :\ a_{ik} + b_{kj} = 1$$

Prova: De fato, sejam $i, j \in \{0, \ldots, N-1\}$. Assim,

$$\begin{aligned}
&(\mathbf{A} * \mathbf{B})_{ij} = 1 \\
&\quad\iff \max_{0\leqslant k<N}\{a_{ik} + b_{kj}\} = 1, && \text{Definição 4.2.6} \\
&\quad\iff \exists\, k \in \{0, \ldots, N-1\} : a_{ik} + b_{kj} = 1, && \text{pela existência do máximo}
\end{aligned}$$

O que conclui a prova. ∎

O lema seguinte é sobre posições definidas em uma multiplicação de matrizes de medida.

Lema 4.2.13 (Multiplicação de matrizes com posição definida)**:** *Sejam* $\mathbf{A}, \mathbf{B} \in \mathcal{M}_{N\times N}$ *e* $i, j \in \{0, \ldots, N-1\}$*. Assim,*

$$(\mathbf{A} * \mathbf{B})_{ij} \geqslant 0 \iff \exists\, k \in \{0, \ldots, N-1\} :\ a_{ik} + b_{kj} \geqslant 0$$

Prova: De fato, sejam $i, j \in \{0, \ldots, N-1\}$. Assim,

$$\begin{aligned}
&(\mathbf{A} * \mathbf{B})_{ij} \geqslant 0 \\
&\quad\iff \max_{0\leqslant k<N}\{a_{ik} + b_{kj}\} \geqslant 0, && \text{Definição 4.2.6} \\
&\quad\iff \exists\, k \in \{0, \ldots, N-1\} : a_{ik} + b_{kj} \geqslant 0, && \text{pela existência do máximo}
\end{aligned}$$

O que conclui a prova. ∎

O próximo lema é de extrema relevância na verificação de positividade de matrizes de medida para circuitos equivalentes, pois leva a um resultado onde verifica-se que, se um circuito tem matriz positiva, então todos os circuitos na mesma classe de equivalência do anterior têm matrizes positivas.

Lema 4.2.14 (Multiplicação positiva comuta)**:** *Sejam* $\mathbf{A}, \mathbf{B} \in \mathcal{M}_{N\times N}$*. A matriz* $\mathbf{A} * \mathbf{B}$ *é positiva se, e somente se, a matriz* $\mathbf{B} * \mathbf{A}$ *é positiva.*

Prova: De fato, $\mathbf{A} * \mathbf{B}$ é positiva

$$\begin{aligned}
&\iff \exists\, j, k \in \{0, \ldots, N-1\} : a_{jk} + b_{kj} = 1, && \text{Lema 4.2.12} \\
&\iff \exists\, k, j \in \{0, \ldots, N-1\} : b_{kj} + a_{jk} = 1, && + \text{ é comutativa} \\
&\iff \mathbf{B} * \mathbf{A} \text{ é positiva}, && \text{Lema 4.2.12}
\end{aligned}$$

O que conclui a prova. ∎

4.3 Terminação Para Grafos com Matrizes de Medida

Nesta breve seção apresenta-se a definição formal de grafos com matrizes de medida e a semântica operacional da terminação para estas estruturas.

A partir da construção formal da álgebra $(\mathcal{M}_{N\times N}, *, \mathbf{Id})$, é possível definir Grafos com Matrizes de Medida (ou MWG's), que são um tipo especial de ω-digrafos, onde a função peso sobre arestas serão matrizes de $\mathcal{M}_{N\times N}$.

Definição 4.3.1 (Grafos com Matrizes de Medida)**:** *Um MWG é um ω-digrafo dado por* $G = \langle V, E, \boldsymbol{\mu} : E \to \mathcal{M}_{N\times N}\rangle$.

Neste momento a construção da função $\boldsymbol{\mu}$ não é apresentada, mas a motivação para tal construção está presente na Equação 4.1. A definição formal de $\boldsymbol{\mu}$ será apresentada no Capítulo 5.

Como mencionado anteriormente, a noção de matriz de medida positiva é de suma importância para a definição de terminação para MWG's, pois garante a existência de uma sequência estritamente decrescente no domínio de uma relação bem fundada sobre o qual as funções de medida da família $\mathcal{F}$ estão definidas. Assim, terminação para MWG's é definida como segue, onde $\hat{\boldsymbol{\mu}}$ é definida em 3.1.19.

Definição 4.3.2 (Terminação em MWG)**:** *Dado um MWG G, a semântica de terminação para G é expressa pela seguinte aplicação booleana* $T_{mwg} : MWG \to \texttt{bool}$ *definida por:*

$$T_{mwg}(G) \;:=\; \forall\, c \in \mathcal{C}ir(G),\; \exists\, j \in \{0, \ldots, N-1\} :\; (\hat{\boldsymbol{\mu}}(c))_{jj} = 1$$

Basicamente a definição anterior expressa que um MWG G é dito terminante se, para todo circuito c de G, a matriz $\hat{\boldsymbol{\mu}}(c)$ é positiva. Observe que $\hat{\boldsymbol{\mu}}(c)$ denota a multiplicação das matrizes (conforme Definição 4.2.6) rotulando cada aresta do circuito c.

4.4 Formalização de Critérios de Terminação para MWG

Nesta seção, dois critérios para verificar terminação em MWGs são apresentados. Estes critérios consistem de condições sobre os ciclos do MWG, que são os circuitos mais simples, conforme Definição 3.1.11, e portanto computáveis, uma vez que um digrafo finito possui um número finito de ciclos. Assim, satisfeitas as condições propostas em cada

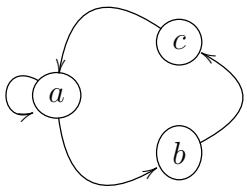
critério, verifica-se que todo circuito do MWG possui uma matriz de medida positiva, o que significa que para todo circuito existe uma sequência estritamente decrescente de medidas, levando a concluir terminação da definição recursiva representada pelo MWG, já que todas as sequências de chamados no fluxo do programa possivelmente infinitas estão representadas por circuitos no MWG.

Umas das ideias iniciais na especificação destes critérios, era tem um critério onde fosse possível verificar a positividade de todo circuito de um MWG através da hipótese de positividade de todo ciclo. Mas como apresentado no exemplo a seguir, positividade de ciclos não é suficiente.

Exemplo 4.4.1: Dado um MWG G, suponha que o seguinte fosse verdade:

$$\begin{array}{c} \forall\, c \in \mathcal{C}yc(G),\ \exists\, j \in \{0,\ldots,N-1\} :\ (\hat{\boldsymbol{\mu}}(c))_{jj} = 1 \\ \Downarrow \\ \forall\, c \in \mathcal{C}ir(G),\ \exists\, j \in \{0,\ldots,N-1\} :\ (\hat{\boldsymbol{\mu}}(c))_{jj} = 1 \end{array} \tag{4.2}$$

Seja G o seguinte MWG:



Dados os ciclos $c_1 = (a, a)$ e $c_2 = (a, b, c, a)$, suponha que suas matrizes de medida sejam dadas por:

$$\hat{\boldsymbol{\mu}}(c_1) = \begin{pmatrix} 1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & 0 & -1 \end{pmatrix} \qquad \hat{\boldsymbol{\mu}}(c_2) = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 1 & 1 \\ -1 & -1 & -1 \end{pmatrix}$$

Assim, a matriz de medida para o circuito $c_3 = (a, a, b, c, a)$ é:

$$\hat{\boldsymbol{\mu}}(c_3) = \hat{\boldsymbol{\mu}}(c_1) * \hat{\boldsymbol{\mu}}(c_2) = \begin{pmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & 1 & 1 \end{pmatrix}$$

Desta forma, tem-se que as matrizes dos circuitos c_1, c_2 e c_3 são positivas, mas a matriz do circuito $c_4 = (a, a, a, b, c, a)$ não é positiva. De fato,

$$\hat{\boldsymbol{\mu}}(c_4) = \hat{\boldsymbol{\mu}}(c_1) * \hat{\boldsymbol{\mu}}(c_3) = \begin{pmatrix} 1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & 0 & -1 \end{pmatrix} * \begin{pmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

O que contradiz (4.2), uma vez que G é um MWG onde todos os ciclos tem matriz de medida positiva, mas existe um circuito com matriz de medida não positiva.

Descartada a ideia de que positividade dos ciclos seria suficiente, cogitou-se a possibilidade de que positividade de ciclos juntamente com *compatibilidade* entre ciclos com vértices em comum fosse suficiente, onde *compatibilidade de ciclos* define-se como segue:

Definição 4.4.1 (Compatibilidade de ciclos)**:** *Dado um MWG G, quaisquer dois ciclos c_1 e c_2 de G são ditos compatíveis se o seguinte é verdade:*

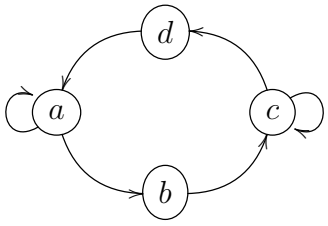
$$c_1[0] = c_2[0] \;\Rightarrow\; \exists j \in \{0, \ldots, N-1\} : (\hat{\boldsymbol{\mu}}(c_1))_{jj} + (\hat{\boldsymbol{\mu}}(c_2))_{jj} = 1.$$

Mas como verifica-se no seguinte exemplo, positividade de ciclos juntamente com compatibilidade de ciclos também não era suficiente para verificar positividade de todo circuito.

Exemplo 4.4.2: Dado um MWG G, suponha que o seguinte fosse verdade:

$$\begin{gathered}
\forall c_1, c_2 \in \mathcal{C}yc(G) : (c_1[0] = c_2[0] \Rightarrow \exists j \in \{0, \ldots, N-1\} : (\hat{\boldsymbol{\mu}}(c_1))_{jj} + (\hat{\boldsymbol{\mu}}(c_2))_{jj} = 1) \\
\wedge \\
\forall\, c \in \mathcal{C}yc(G),\ \exists\, j \in \{0, \ldots, N-1\} :\ (\hat{\boldsymbol{\mu}}(c))_{jj} = 1 \\
\Downarrow \\
\forall\, c \in \mathcal{C}ir(G),\ \exists\, j \in \{0, \ldots, N-1\} :\ (\hat{\boldsymbol{\mu}}(c))_{jj} = 1
\end{gathered} \tag{4.3}$$

Seja G o sequinte MWG:



Considere os caminhos $c_1 = (a, a)$, $c_2 = (a, b, c)$, $c_3 = (c, c)$ e $c_4 = (c, d, a)$. Suponha que suas matrizes de medida sejam dadas por:

$$\hat{\boldsymbol{\mu}}(c_1) = \begin{pmatrix} 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \end{pmatrix} \qquad \hat{\boldsymbol{\mu}}(c_2) = \begin{pmatrix} -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 0 \\ -1 & -1 & -1 & -1 \end{pmatrix}$$

$$\hat{\boldsymbol{\mu}}(c_3) = \begin{pmatrix} -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 \end{pmatrix} \qquad \hat{\boldsymbol{\mu}}(c_4) = \begin{pmatrix} -1 & 1 & -1 & -1 \\ 0 & -1 & -1 & -1 \\ -1 & -1 & -1 & 0 \\ -1 & -1 & 1 & -1 \end{pmatrix}$$

Considere o ciclo $c_5 = (a, b, c, d, a)$ que tem a seguinte matriz de medida:

$$\hat{\boldsymbol{\mu}}(c_5) = \hat{\boldsymbol{\mu}}(c_2) * \hat{\boldsymbol{\mu}}(c_4) = \begin{pmatrix} 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & -1 \end{pmatrix}$$

Note que os ciclos c_1 e c_5 são compatíveis, uma vez que $a \in E_G(c_1) \cap E_G(c_5)$ e $(\hat{\boldsymbol{\mu}}(c_1))_{11} + (\hat{\boldsymbol{\mu}}(c_5))_{11} = 1$.

Agora, considere o ciclo $c_6 = (c, d, a, b, c)$ que tem a seguinte matriz de medida:

$$\hat{\boldsymbol{\mu}}(c_6) = \hat{\boldsymbol{\mu}}(c_4) * \hat{\boldsymbol{\mu}}(c_2) = \begin{pmatrix} -1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 \end{pmatrix}$$

Note que os ciclos c_3 e c_6 são compatíveis, uma vez que $c \in E_G(c_3) \cap E_G(c_6)$ e $(\hat{\boldsymbol{\mu}}(c_3))_{44} + (\hat{\boldsymbol{\mu}}(c_6))_{44} = 1$.

Então, cada dois ciclos do MWG que tenham o vértice inicial em comum são compatíveis e portanto um circuito formado por esses dois ciclos tem matriz de medida positiva como pode ser verificado abaixo.

- $c_7 = (a, b, c, c, d, a)$ - formado por c_5 e c_3, tem a seguinte matriz de medida positiva:

$$\hat{\boldsymbol{\mu}}(c_7) = (\hat{\boldsymbol{\mu}}(c_2) * \hat{\boldsymbol{\mu}}(c_3)) * \hat{\boldsymbol{\mu}}(c_4) = \begin{pmatrix} -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 \\ -1 & -1 & -1 & -1 \end{pmatrix} * \begin{pmatrix} -1 & 1 & -1 & -1 \\ 0 & -1 & -1 & -1 \\ -1 & -1 & -1 & 0 \\ -1 & -1 & 1 & -1 \end{pmatrix}$$

$$= \begin{pmatrix} -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & -1 \end{pmatrix}$$

- $c_8 = (c, d, a, a, b, c)$ - formado por c_6 e c_1, tem a seguinte matriz de medida positiva:

$$\hat{\boldsymbol{\mu}}(c_8) = (\hat{\boldsymbol{\mu}}(c_4) * \hat{\boldsymbol{\mu}}(c_1)) * \hat{\boldsymbol{\mu}}(c_2) = \begin{pmatrix} -1 & -1 & -1 & -1 \\ 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \end{pmatrix} * \begin{pmatrix} -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 0 \\ -1 & -1 & -1 & -1 \end{pmatrix}$$

$$= \begin{pmatrix} -1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \end{pmatrix}$$

Contudo, o circuito $c_9 = (a, b, c, c, d, a, a) = c_2 \circ_w c_3 \circ_w c_4 \circ_w c_1$ não tem uma matriz de medida positiva, como pode ser verificado a seguir:

$$\hat{\boldsymbol{\mu}}(c_8) = (\hat{\boldsymbol{\mu}}(c_2) * \hat{\boldsymbol{\mu}}(c_3)) * (\hat{\boldsymbol{\mu}}(c_4) * \hat{\boldsymbol{\mu}}(c_1)) =$$

$$\begin{pmatrix} -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 \\ -1 & -1 & -1 & -1 \end{pmatrix} * \begin{pmatrix} -1 & -1 & -1 & -1 \\ 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \end{pmatrix} = \begin{pmatrix} -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \end{pmatrix}$$

O que contradiz (4.3), uma vez que G é um MWG onde todos os ciclos tem matriz de medida positiva, cada dois ciclos com vértice inicial em comum são compatíves, mas existe um circuito com matriz de medida não positiva.

Assim, tendo descartado (4.2) e (4.3), formulou-se outros critérios suficientes para verificação de positividade da matriz de medida de todo circuito em um MWG. Tais critérios são apresentados nas subseções 4.4.1 e 4.4.2.

Notação: Deste ponto em diante, nesta seção a matriz de medida $\mathbf{M}$ de um caminho w (resp. um circuito c, uma aresta e) no MWG G será denotada por $\mathbf{M}(w)$ (resp. $\mathbf{M}(c)$, $\mathbf{M}(e)$)

4.4.1 Um Critério Baseado em Ordens Lexicográficas

Para um dado caminho $w = (v_0, \ldots, v_{n-1})$ no MWG G, denote por $e_i = (v_i, v_{i+1})$, $0 \leqslant i < n-1$ as arestas de w. Assim, a entrada m_{ij} na sua matriz de medida $\mathbf{M}(w)$ é obtida pela seguinte equação:

$$(\mathbf{M}(w))_{ij} = (\mathbf{M}(e_0))_{ik_0} * (\mathbf{M}(e_1))_{k_0k_1} * \ldots * (\mathbf{M}(e_{i-2}))_{k_{i-2}j},$$

para alguns $k_0, k_1, \ldots, k_{i-2}$ entre 0 e $N-1$. Note que, se apenas a matriz $\mathbf{M}(w)$ é conhecida, então não é possível, a partir da posição (i,j), determinar completamente todos os k_α, para $0 \leqslant \alpha \leqslant i-2$, envolvidos na equação acima. Esta situação leva a um alto grau de liberdade, pois não necessariamente a sequência $k_0, k_1, \ldots, k_{i-2}$ é única, tornando impossível verificar positividade de todo circuito do MWG apenas a partir de positividade de todo ciclo. Algo mais era necessário, surge então a ideia de medida limitante com condição de positividade sobre ciclos-duplos, definidos a seguir. A inspiração para estabelecer a existência de uma medida limitante surge da definição de ordens lexicográficas (que pode ser encontrada em [BN98]), onde existe uma ordem atuando como fator de controle, que é exatamente a mesma ideia por trás da medida limitante aqui definida. Esta medida, se existe, "controla" o MWG e, juntamente com a hipótese de positividade de todo ciclo, resulta em que todo circuito tenha uma matriz de medida positiva.

Definição 4.4.2 (Ciclo duplo)**:** *Seja c um circuito no MWG G, diz-se que c é um* ciclo duplo *se existem ciclos c_1 e c_2 tais que $c \sim \circ_w(c_1, c_2)$.*

Definição 4.4.3 (Medida limitante)**:** *Dado um MWG G, uma medida limitante para G é um k, $0 \leqslant k \leqslant N-1$, tal que:*

1. *Para toda aresta $e \in E_G$: $(\mathbf{M}(e))_{kk} \geqslant 0$.*

2. *Para todo ciclo-duplo c, existe uma aresta $e \in E_G(c)$ tal que $(\mathbf{M}(e))_{kk} = 1$.*

Assim, o primeiro critério é formulado da seguinte maneira:

Definição 4.4.4 (Critério I)**:** *Dado um MWG G define-se o operador $\mathcal{CI}$ em G por*

$$\mathcal{CI}(G) := \begin{cases} \exists\, k \in \{0, \ldots N-1\} \text{ tal que } k \text{ é medida limitante} \\ \bigwedge\ \forall\, c \in \mathcal{P}_w, \text{ se } c \text{ é um ciclo, então } \mathbf{M}(c) \text{ é positiva} \end{cases}$$

Lema 4.4.5 (Caminhos com matriz de medida definida)**:** *Sejam G um MWG e k uma medida limitante. Para cada $w \in \mathcal{W}_G$, vale que $(\mathbf{M}(w))_{kk} \geqslant 0$.*

Prova: Seja $w \in \mathcal{W}_G$. A prova segue por indução no comprimento do caminho w. É sabido que $\ell(w) > 0$, pela Definição 3.1.6. Assim:

$\ell(w) = 1$**.** Neste caso, da Definição 3.1.19, tem-se que a matrix $\mathbf{M}(w)$ é a matrix identidade $\mathbf{Id}$. Portanto, $(\mathbf{M}(w))_{kk} = (\mathbf{Id})_{kk} = 0$, pela Definição 4.2.7.

$\ell(w) = 2$**.** Neste caso, $w = (v_0, v_1)$ é uma aresta de G. Portanto, segue da Definição 4.4.3 de medida limitante que $(\mathbf{M}(w))_{kk} \geqslant 0$.

$\ell(w) > 2$. Suponha que $\ell(w) = n$ e $w = (v_0, v_1, \ldots, v_{n-1})$. Segue do Lema 3.1.20 que $\mathbf{M}(w) = \mathbf{M}(w^{(0,1)}) * \mathbf{M}(w^{(1,n)})$. Além disso, como $\ell(w^{(0,1)}) = 2$, tem-se que $(\mathbf{M}(w^{(0,1)}))_{kk} \geqslant 0$; e por hipótese de indução, como $\ell(w) = n < n-1 = \ell(w^{(1,n-1)})$, segue que $(\mathbf{M}(w^{(1,n-1)}))_{kk} \geqslant 0$. Portanto, $(\mathbf{M}(w^{(0,1)}))_{kk} + (\mathbf{M}(w^{(1,n-1)}))_{kk} \geqslant 0$ e, pelo Lema 4.2.13, conclui-se que $(\mathbf{M}(w^{(0,1)}) * \mathbf{M}(w^{(1,n-1)}))_{kk} \geqslant 0$.

Logo, $(\mathbf{M}(w))_{kk} \geqslant 0$, para todo $w \in \mathcal{W}_G$. ■

Lema 4.4.6 (Aresta com matriz de medida positiva)**:** *Seja G um MWG. Se existe k uma medida limitante para G, então para todo circuito c de G, que não seja um ciclo, existe uma aresta $e \in E_G(c)$ tal que $(\mathbf{M}(e))_{kk} = 1$.*

Prova: A prova segue por indução na estrutura do circuito c. Como c não é um ciclo, pelo Lema 3.1.16, existe um ciclo c_1 e um circuito c_2 tais que $c \sim \circ_w(c_1, c_2)$. Então existem duas possibilidade para c_2:

a) c_2 é um ciclo: Neste caso c é um ciclo duplo, pois é equivalente à composição de dois ciclos. Assim, segue da Definição 4.4.3, que existe $e \in E_G(c)$ tal que $(\mathbf{M}(e))_{kk} = 1$.

b) c_2 não é um ciclo: Neste caso, por hipótese de indução segue que existe $e \in E_G(c_2)$ tal que $(\mathbf{M}(e))_{kk} = 1$. Além disso, $E_G(c) = E_G(\circ_w(c_1, c_2))$, pelo Lema 3.1.14, e $E_G(\circ_w(c_1, c_2)) = E_G(c_1) \cup E_G(c_2)$, pelo Lema 3.1.9, ou seja, $E_G(c) = E_G(c_1) \cup E_G(c_2)$. Portanto, $e \in E_G(c)$.

Assim, de (a) e (b) segue a prova do lema. ■

Teorema 4.4.7 (Circuitos com matriz positiva via medida limitante)**:** *Seja G um MWG. Se $\mathcal{CI}(G)$ é verdadeiro, então para todo circuito c de G, a matriz $\mathbf{M}(c)$ é positiva.*

Prova: Suponha que $\mathcal{CI}(G)$ é verdadeiro, ou seja, $\exists\ k \in \{0, \ldots N-1\}$ tal que k é medida limitante e $\forall\ c \in \mathcal{P}_w$, se c é um ciclo, então $\mathbf{M}(c)$ é positiva. Seja c um circuito de comprimento m de G, existem duas possibilidades para c:

a) c é um ciclo: por hipótese, $\mathbf{M}(c)$ é positiva.

b) c não é um ciclo: suponha por contradição que $\mathbf{M}(c)$ não é positiva. Assim, para todo l, $0 \leqslant l \leqslant N-1$, $(\mathbf{M}(c))_{ll} \leqslant 0$. Em particular, tomando $l = k$, $(\mathbf{M}(c))_{kk} \leqslant 0$. Além disso, pelo Lema 4.4.6, existe uma aresta $e \in E_G(c)$ tal que $(\mathbf{M}(e))_{kk} = 1$. Suponha que $e = (v_i, v_{i+1})$, então a matriz de medida do circuito c na posição (k, k) é dada por:

$$(\mathbf{M}(c))_{kk} = (\ \mathbf{M}(c^{(0,i)}) * \mathbf{M}(c^{(i,i+1)}) * \mathbf{M}(c^{(i+1,m)})\ )_{kk} \qquad (\star)$$

Como, $c^{(0,i)}$ e $c^{(i+1,n)}$ são caminhos de G, pelo Lema 4.4.5, $(\mathbf{M}(c^{(0,i)}))_{kk} \geqslant 0$ e $(\mathbf{M}(c^{(i+1,m)}))_{kk} \geqslant 0$. Ou seja,

$$(\mathbf{M}(c^{(0,i)}))_{kk} + (\mathbf{M}(c^{(i,i+1)}))_{kk} + (\mathbf{M}(c^{(i+1,m)}))_{kk} = 0 + 1 + 0 = 1 \qquad (\star\star)$$

Portanto, como na multiplicação de matrizes de medida aplica-se maximização (Definição 4.2.6), de $(\star)$ e $(\star\star)$ conclui-se que $(\mathbf{M}(c))_{kk} = 1$. O que é uma contradição.

Logo, de (a) e (b), segue que $\mathbf{M}(c)$ é positiva e conclui-se a prova do teorema. ■

Em seguida um exemplo de aplicação do critério $\mathcal{CI}$.

Exemplo 4.4.3: Considere a função *mdc*, para calcular o máximo divisor comum entre dois números, definida por:

$$\begin{aligned} mdc(m,n) \;=\; & \texttt{IF}\ \ m = 0 \ \vee\ n = 0\ \ \texttt{THEN}\ \ m + n \\ & \texttt{ELSIF}\ \ m \geqslant n\ \ \texttt{THEN}\ \ \boxed{1}\ mdc(m-n,n)\ \ \texttt{ELSE}\ \ \boxed{2}\ mdc(n,m) \end{aligned}$$

E tome como medidas sobre os parâmetros:

$$\mu_1(m,n) = m$$
$$\mu_2(m,n) = n$$

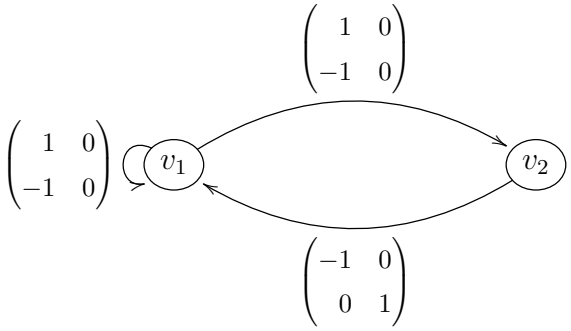
Observe que a função *mdc* tem dois chamados recursivos em sua definição. Portanto o seu MWG possui dois vértices v_1 e v_2, representando os chamados numerados por $\boxed{1}$ e $\boxed{2}$, respectivamente. Observe ainda que o segundo chamado recursivo não pode ser executado novamente depois dele mesmo, uma vez que sua condição não pode ser novamente satisfeita após este chamado ter sido executado. Isto permite excluir a aresta (v_2, v_2) do MWG. Note ainda que: relativamente ao chamado $\boxed{1}$, tem-se que

$$\begin{aligned} \mu_1(m,n) &= m \;>\; m-n &&= \mu_1(m-n,n) \\ \mu_1(m,n) &= m \;\geqslant\; n &&= \mu_2(m-n,n) \\ \mu_2(m,n) &= n \;?\; m-n &&= \mu_1(m-n,n) \\ \mu_2(m,n) &= n \;=\; n &&= \mu_2(m-n,n) \end{aligned}$$

e relativamente ao chamado $\boxed{2}$, tem-se que

$$\begin{aligned} \mu_1(m,n) &= m \;<\; n &&= \mu_1(n,m) \\ \mu_1(m,n) &= m \;=\; m &&= \mu_2(n,m) \\ \mu_2(m,n) &= n \;=\; n &&= \mu_1(n,m) \\ \mu_2(m,n) &= n \;>\; m &&= \mu_2(n,m) \end{aligned}$$

Assim, tem-se o seguinte MWG para a função *mdc*:



Observe que a medida limitante (de acordo com a definição 4.4.3) no MWG acima é $k = 2$, pois para toda aresta a respectiva matriz de medida tem entrada maior ou igual a zero na posição $(2, 2)$. Além disso, o ciclo duplo $c = (v_1, v_1, v_2, v_1)$ do MWG tem a seguinte matriz de medida:

$$\boldsymbol{M}(c) = \begin{pmatrix} 1 & 0 \\ -1 & 0 \end{pmatrix} * \begin{pmatrix} 1 & 0 \\ -1 & 0 \end{pmatrix} * \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

que é positiva. E pelo Lema 4.2.14, ciclos equivalentes ao ciclo c também tem matriz positiva. Note ainda que o ciclo (v_1, v_1) tem matriz positiva e, também o ciclo (v_1, v_2, v_1) tem matriz positiva. Portanto, segue do Teorema 4.4.7, que todo circuito do MWG acima tem uma matriz de medida positiva.

4.4.2 Um Critério Baseado em uma Rotulagem Fixa

O segundo critério apresentado para verificar terminação é baseado em rotulagens dos vértices, onde os rótulos atribuídos representam medidas da família $\mathcal{F}$. Então, positividade das matrizes de medida de todos os circuitos do MWG é verificada sob a condição de existência de uma rotulagem que seja limitante, no sentido de que fornece uma sequência de medidas estritamente decrescente para cada ciclo e decrescente (não estritamente) em cada aresta.

Uma rotulagem fixa dos vértices (ou simplesmente rotulagem) é atribuída segundo uma função que mapeia o conjunto de vértices V_G do MWG G no conjunto $\{0, \ldots, N-1\}$, denotada por: $\mathcal{R}_G : V_G \to \{0, \ldots, N-1\}$. Em seguida a definição formal de rotulagem limitante é apresentada.

Definição 4.4.8 (Rotulagem limitante)**:** *Dado um MWG G e uma rotulagem $\mathcal{R}_G$, diz-se que $\mathcal{R}_G$ é uma* rotulagem limitante *se:*

i) Para toda aresta $e = (u, v) \in E_G$, seja $p \in \{0, \ldots, N-1\} \times \{0, \ldots, N-1\}$ dado por $(\mathcal{R}_G(u), \mathcal{R}_G(v))$, então $(\mathbf{M}(e))_p \geqslant 0$.

ii) Para todo ciclo $c = (v_0, \ldots, v_k, v_0)$ *em* G*, seja* $p \in \{0, \ldots, N-1\} \times \{0, \ldots, N-1\}$ *dado por* $(\mathcal{R}_G(v_0), \mathcal{R}_G(v_0))$*, então* $(\mathbf{M}(c))_p = 1$.

Assim, o segundo critério é formulado da seguinte maneira:

Definição 4.4.9 (Critério II)**:** *Dado um MWG* G *define-se o operador* $\mathcal{CII}$ *em* G *por*

$$\mathcal{CII}(G) := \exists\, \mathcal{R}_G : V_G \to \{0, \ldots, N-1\} \text{ tal que } \mathcal{R}_G \text{ é rotulagem limitante}$$

Lema 4.4.10 (Caminhos com matriz de medida com posição definida via rotulagem limitante)**:** *Sejam* G *um MWG e* $w = (v_0, \ldots, v_{n-1}) \in \mathcal{W}_G$*. Se* $\mathcal{CII}(G)$ *é verdadeiro, então* $(\mathbf{M}(w))_p \geqslant 0$*, onde* $p = (\mathcal{R}_G(v_0), \mathcal{R}_G(v_{n-1}))$.

Prova: A prova é por indução no comprimento do caminho w. Como $\mathcal{CII}(G)$ é verdadeiro, então existe uma rotulagem limitante $\mathcal{R}_G : V_G \to \{0, \ldots, N-1\}$. Além disso, $\ell(w) > 0$ pela Definição 3.1.6. Assim, considere as seguintes possibilidades para $\ell(w)$:

$\ell(w) = 1$**.** Assim, $w = (v_0)$ e denotando $\mathcal{R}_G(v_0) = k$, tem-se $p = (k, k)$. Pela Definição 3.1.19 a matrix $\mathbf{M}(w)$ é a matrix identidade $\mathbf{Id}$. Portanto, $(\mathbf{M}(w))_{kk} = (\mathbf{Id})_{kk} = 0$;

$\ell(w) = 2$**.** Neste caso, $w = (v_0, v_1)$ é uma aresta de G. Portanto, segue da Definição 4.4.8 de rotulagem limitante que, tomando $p = (\mathcal{R}_G(v_0), \mathcal{R}_G(v_1))$, temos $(\mathbf{M}(w))_p \geqslant 0$;

$\ell(w) > 2$**.** Suponha que $\ell(w) = n$ e $w = (v_0, v_1, \ldots, v_{n-1})$. Segue do Lema 3.1.20 que $\mathbf{M}(w) = \mathbf{M}(w^{(0,1)}) * \mathbf{M}(w^{(1,n)})$. Assim, denote $i = \mathcal{R}_G(v_0)$, $k = \mathcal{R}_G(v_1)$ e $j = \mathcal{R}_G(v_{n-1})$. Além disso, como $\ell(w^{(0,1)}) = 2$, tem-se que $(\mathbf{M}(w^{(0,1)}))_{ik} \geqslant 0$; e por hipótese de indução, como $\ell(w) = n < n - 1 = \ell(w^{(1,n-1)})$, segue que $(\mathbf{M}(w^{(1,n-1)}))_{kj} \geqslant 0$. Portanto, $(\mathbf{M}(w^{(0,1)}))_{ik} + (\mathbf{M}(w^{(1,n-1)}))_{kj} \geqslant 0$ e, pelo Lema 4.2.13, conclui-se que $(\mathbf{M}(w^{(0,1)}) * \mathbf{M}(w^{(1,n-1)}))_{ij} \geqslant 0$.

Logo, conclui-se que, para todo caminho $w = (v_0, \ldots, v_{n-1})$ de G, a matriz $\mathbf{M}(w)$ é tal que, tomando $p = (\mathcal{R}_G(v_0), \mathcal{R}_G(v_{n-1}))$, temos $(\mathbf{M}(w))_p \geqslant 0$. ∎

O seguinte teorema é o resultado principal sobre o critério $\mathcal{CII}$, pois estabelece que, se existe uma rotulagem limitante para um dado MWG G, então todo circuito de G tem uma matriz de medida positiva, o que, como discutido anteriormente, leva a concluir terminação da definição recursiva representada por G.

Teorema 4.4.11 (Circuitos com matriz positiva via rotulagem limitante)**:** *Dado um MWG* G*, se* $\mathcal{CII}(G)$ *é verdadeiro, então para todo circuito* c *de* G*, a matriz* $\mathbf{M}(c)$ *é positiva.*

Prova: Da hipótese de que o critério $\mathcal{CII}(G)$ é verdadeiro, tem-se que existe uma rotulagem $\mathcal{R}_G : V_G \to \{0, \ldots, N-1\}$ limitante. Assim, por indução na estrutura do circuito $c = (v_0, v_1, \ldots, v_{n-1} = v_0)$, verifica-se que a matriz $\mathbf{M}(c)$ é positiva na posição $p = (\mathcal{R}_G(v_0), \mathcal{R}_G(v_0))$. Assim, observando que $\ell(c) = n$, considere as seguintes possibilidades:

a) c é um ciclo: por hipótese, $(\mathbf{M}(c))_p = 1$.

b) c não é um ciclo: pelo Lema 3.1.16, existem c_1 e c_2 tais que

$$c \sim \circ_w(c_1, c_2), \tag{A}$$

onde c_1 é um ciclo e c_2 um circuito. Observe que, denotando $c_1 = (u_0, u_1, \ldots, u_{m_1-1})$ e $c_2 = (u'_0, u'_1, \ldots, u'_{m_2-1})$, tem-se que $\ell(c_1) = m_1$ e $\ell(c_2) = m_2$. Além disso, como c_1 é um ciclo e c_2 um circuito,

$$u_0 = u_{m_1-1} \ \text{ e } \ u'_0 = u'_{m_2-1}. \tag{B}$$

E do fato de $\circ_w(c_1, c_2) = c_1 \circ c_2^{(1,m_2-1)} = (u_0, u_1, \ldots, u_{m_1-1}, u'_1, \ldots, u'_{m_2-1})$ ser um circuito, tem-se

$$u_0 = u'_{m_2-1}. \tag{C}$$

Assim, denotando $u_0 = u$, segue de (B) e (C) que $c_1 = (u, u_1, \ldots, u_{m_1-2}, u)$ e $c_2 = (u, u'_1, \ldots, u'_{m_2-2}, u)$; e da hipótese de que $\mathcal{R}_G$ é uma rotulagem limitante e aplicando hipótese de indução a c_2, denotanto $q = (\mathcal{R}_G(u), \mathcal{R}_G(u))$, tem-se que as matrizes de c_1 e c_2 são positivas na posição q, isto é,

$$(\mathbf{M}(c_1))_q = 1 \ \text{ e } \ (\mathbf{M}(c_2))_q = 1. \tag{D}$$

Assim, voltando à equação (A) e observando que $\ell(\circ_w(c_1, c_2)) = m_1 + m_2 - 1$, pelo Lema 3.1.13, existe um natural $i < \ell(\circ_w(c_1, c_2))$ tal que

$$\begin{array}{rcl} c & = & (\circ_w(c_1, c_2))^{(i,\ell(\circ_w(c_1,c_2))-2)} \ \circ \ (\circ_w(c_1, c_2))^{(0,i)} \\ & = & (c_1 \circ c_2^{(1,m_2-1)})^{(i,m_1+m_2-3)} \ \circ \ (c_1 \circ c_2^{(1,m_2-1)})^{(0,i)} \end{array} \tag{E}$$

Considerando as possibilidades para i, tem-se que:

i) $i = 0$

Neste caso, de (E) tem-se que $c = c_1 \circ c_2^{(1,m_2-1)}$ e portanto $p = q$. Além disso, pelo Lema 3.1.20 e de (B) e (C) segue que $\mathbf{M}(c) = \mathbf{M}(c_1) * \mathbf{M}(c_2)$. Assim, por maximização na operação $*$ e de (D), tem-se $(\mathbf{M}(c))_p = (\mathbf{M}(c_1))_p + (\mathbf{M}(c_2))_p = 1$;

ii) $0 < i < m_1 - 1$

Neste caso, de (E) tem-se que $c = c_1^{(i,m_1-2)} \circ c_2 \circ c_1^{(1,i)}$. Pelo Lema 3.1.20 e das Equações (B) e (C) segue que $\mathbf{M}(c) = \mathbf{M}(c_1^{(i,m_1-1)}) * \mathbf{M}(c_2) * \mathbf{M}(c_1^{(0,i)})$. Pelo Lema 4.4.10, como $\mathcal{R}_G$ é rotulagem limitante, denotando $(\mathcal{R}_G(u_i), \mathcal{R}_G(u))$ por q_1 e $(\mathcal{R}_G(u), \mathcal{R}_G(u_i))$ por q_2 segue que $(\mathbf{M}(c_1^{(i,m_1-1)}))_{q_1} \geqslant 0$ e $(\mathbf{M}(c_1^{(0,i)}))_{q_2} \geqslant 0$. Além disso, note que neste caso $v_0 = u_i$. Portanto, por maximização na operação $*$ e de (D), tem-se $(\mathbf{M}(c))_p = (\mathbf{M}(c_1^{(i,m_1-1)}))_{q_1} + (\mathbf{M}(c_2))_q + (\mathbf{M}(c_1^{(0,i)}))_{q_2} = 1$;

iii) $i = m_1 - 1$

Neste caso, de (E) tem-se que $c = c_2 \circ c_1^{(1,m1-1)}$ e, como $v_0 = u$, conclui-se $p = q$. Logo, pelo Lema 3.1.20 e de (B) e (C) segue que $\mathbf{M}(c) = \mathbf{M}(c_1) * \mathbf{M}(c_2)$. Por maximização na operação $*$ e de (D), tem-se $(\mathbf{M}(c))_p = (\mathbf{M}(c_2))_p + (\mathbf{M}(c_1))_p = 1$;

iv) $m_1 - 1 < i < m_1 + m_2 - 2$

Neste caso, de (E) tem-se que $c = c_2^{(i-m_1+1,m_2-2)} \circ c_1 \circ c_2^{(1,i-m_1+1)}$. Portanto, pelo Lema 3.1.20 e pelas Equações (B) e (C) segue que a matriz de c é dada por $\mathbf{M}(c) = \mathbf{M}(c_2^{(i-m_1+1,m_2-1)}) * \mathbf{M}(c_1) * \mathbf{M}(c_2^{(0,i-m_1+1)})$. Pelo Lema 4.4.10, como $\mathcal{R}_G$ é rotulagem limitante, denotando por q_1' a posição $(\mathcal{R}_G(u'_{i-m_1+1}), \mathcal{R}_G(u))$ e por q_2' a posição $(\mathcal{R}_G(u), \mathcal{R}_G(u'_{i-m_1+1}))$, segue que $(\mathbf{M}(c_2^{(i-m1+1,m_2-1)}))_{q_1'} \geqslant 0$ e $(\mathbf{M}(c_2^{(0,i-m_1+1)}))_{q_2'} \geqslant 0$. Além disso, note que neste caso, $v_0 = u'_{i-m_1+1}$. Portanto, por maximização na operação $*$ e de (D), conclui-se que $(\mathbf{M}(c))_p = (\mathbf{M}(c_2^{(i-m_1+1,m_2-1)}))_{q_1'} + (\mathbf{M}(c_1))_q + (\mathbf{M}(c_2^{(0,i-m_1+1)}))_{q_2'} = 1$;

v) $i = m_1 + m_2 - 2$

Neste caso, de (E) tem-se que $c = c_1 \circ c_2^{(1,m_2-1)}$ e, portanto, a prova deste item segue idêntica ao item (i).

Portanto, de (a) e (b) segue que $(\mathbf{M}(c))_p = 1$. Ou seja, temos que a matriz $\mathbf{M}(c)$ é positiva. ■

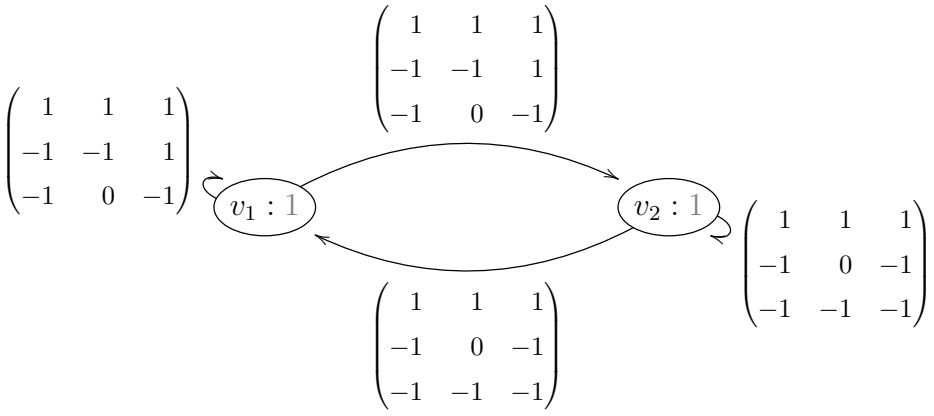
Exemplo 4.4.4: Considere a função $p : \mathbb{N}^3 \to \mathbb{N}$, com dois chamados recursivos, definida como segue:

$$\begin{aligned} p(m,n,r) \;=\; & \texttt{IF } r > 0 \texttt{ THEN } \boxed{1}\, p(m, r-1, n) \\ & \texttt{ELSIF } n > 0 \texttt{ THEN } \boxed{2}\, p(r, n-1, m) \\ & \texttt{ELSE } m \end{aligned}$$

Aplicando as seguintes medidas sobre os parâmetros:

$$\mu_1(m,n,r) = m + n + r$$
$$\mu_2(m,n,r) = m + r$$
$$\mu_3(m,n,r) = m + n$$

Tem-se o seguinte MWG:



Observe que, atribuindo o rótulo 1 a cada vértice do MWG acima, tem-se que a matriz de cada aresta na posição $(1,1)$ possui entrada maior ou igual a zero, e cada ciclo tem matriz positiva na posição $(1,1)$, o que caracteriza uma rotulagem limitante, satisfazendo assim o critério $\mathcal{CII}$. Assim, aplicando o Teorema 4.4.11, segue que todo circuito do MWG de p tem uma matriz de medida positiva.

4.5 A Prova Formal em PVS dos Critérios

Nesta seção apresenta-se a prova formalizada em PVS dos Teoremas 4.4.7 e 4.4.11, assim como a especificação em PVS dos respectivos critérios de terminação apresentados nas Definições 4.4.4 e 4.4.9.

A especificação em PVS de terminação para MWG's, que corresponde à Definição 4.3.2, é apresentada como segue:

```
mwg_termination?(G): bool =
   FORALL (c | circuit?(dg(G), c)): positive?(wgt_walk(G, c))
```

4.5.1 Formalização em PVS da Prova do Teorema 4.4.7

Nesta subseção apresenta-se a formalização do Teorema `lm_pstv_cycles_pstv_circuits`, cuja especificação está presente na *teoria* `matrix_wdg` da Figura 4.1. Este teorema corresponde ao Teorema 4.4.7, onde se verifica que, se o critério $\mathcal{CI}$ é verdadeiro, então todos os

```
double_cycle?(G)(c): bool =
  circuit?(dg(G), c) AND
  EXISTS(c1, c2: {w: prewalk | cycle?(dg(G), w)}):
    eq_circuit?(dg(G), c, c1 o rest(c2))

positive_double_cycle?(G,k)(c: (double_cycle?(G))): bool =
  EXISTS(e: (edges_of(c))): wgt(G)(e)(k)(k) = 1

limiting_measure?(G)(k): bool =
  (FORALL(e: edge(dg(G))): wgt(G)(e)(k)(k) >= 0) AND
  (FORALL(c: (double_cycle?(G))): positive_double_cycle?(G,k)(c))

criterion_one?(G): bool =
  (EXISTS k: limiting_measure?(G)(k)) AND
  (FORALL cy: cycle?(dg(G),cy) IMPLIES positive?(wgt_walk(G, cy)))
```

Figura 4.2: Definição na linguagem de especificação do PVS do critério $\mathcal{CI}$. As definições nesta figura correspondem às definições 4.4.2, 4.4.3 e 4.4.4.

circuitos de um MWG são tais que sua matriz de medida é positiva, a saber, na posição definida pela medida limitante determinada pelo critério $\mathcal{CI}$. A Figura 4.2 apresenta a parte da especificação onde o critério $\mathcal{CI}$ é definido. Quatro funções são apresentadas nesta figura e descritas a seguir:

`double_cycle?`: Booleano que determina condições para que um pré-caminho `c` de um MWG `G` seja classificado como um ciclo-duplo. Tais condições consistem em verificar se `c` é um circuito de `G` e se é equivalente à composição de dois ciclos.

`positive_double_cycle?`: Booleano para determinar se um ciclo duplo de um MWG `G` possui um aresta `e` cuja matriz é positiva em uma dada posição `k` menor que `N`.

`limiting_measure?`: Booleano para determinar se um dado `k` menor que `N` é uma medida limitante para o MWG `G`, o que significa dizer que toda aresta `e` de `G` é tal que a matriz de `e` é positiva na posição `k` e todo ciclo duplo é positivo relativamente à definição anterior. Esta função corresponde à Definição 4.4.3.

`criterion_one?`: booleano que representa o critério $\mathcal{CI}$. Se verdadeiro para um dado MWG `G`, significa que existe uma medida limitante para `G` e que todos os ciclos de `G` tem matriz de medida positiva, o que é suficiente para verificar que todo circuito de `G` possui matriz de medida positiva, em particular, na posição `k`. Tal verificação é apresentada no Teorema `lm_pstv_cycles_pstv_circuits`.

```
lm_defined_walks: LEMMA
  limiting_measure?(G)(k) AND walk?(dg(G), w)
  IMPLIES wgt_walk(G, w)(k)(k) >= 0

lm_positive_edge: LEMMA
  limiting_measure?(G)(k) AND NOT cycle?(dg(G),c) AND circuit?(dg(G), c)
  IMPLIES EXISTS(e: (edges_of(c))): wgt(G)(e)(k)(k) = 1

lm_pstv_cycles_pstv_circuits: THEOREM
  criterion_one?(G) IMPLIES mwg_termination?(G)
```

Figura 4.3: Definição na linguagem de especificação do PVS dos lemas auxiliares `lm_defined_walks`, que corresponde ao Lema 4.4.5, e `lm_positive_edge`, que corresponde ao Lema 4.4.6. Apresenta-se também a especificação do teorema principal, com relação ao Critério 1, o Teorema `lm_pstv_cycles_pstv_circuits`. Este teorema corresponde ao Teorema 4.4.7.

Na Figura 4.3 é apresentado o Teorema `lm_pstv_cycles_pstv_circuits` na linguagem de especificação do PVS. Em seguida apresenta-se a formalização do teorema, onde alguns sequentes de prova são descritos. Na Figura 4.4 a árvore de prova pode ser visualizada. No que segue a formalização do teorema é apresentada com apenas alguns sequentes descritos. A correspondência entre os sequentes de prova e os nós da árvore é descrita ao longo da formalização e destacada na árvore através dos rótulos entre colchetes.

Formalização do teorema `lm_pstv_cycles_pstv_circuits`:

O primeiro objetivo de prova na formalização deste teorema é correspondente ao nó [**raiz**] na Figura 4.4. Este sequente é apresentado por:

- Sequente [**raiz**], Figura 4.4

```
  |-------
{1}   FORALL (G: wdg): criterion_one?(G) IMPLIES
        (FORALL (c: prewalk | circuit?(dg(G), c)): positive?(wgt_walk(G, c)))
```

Após skolemização a fim de dar nomes às variáveis ligadas `G` e `c`, assim como apresentado na prova do Teorema 4.4.7, é considerado o caso em que `c` é um ciclo de `G`. Isto é executado através do comando de prova `(case "cycle?(dg(G),c)")` visualizado na árvore de prova da Figura 4.4. Após este comando, dois ramos são gerados:

- Na sub-árvore a esquerda com três nós (rotulada por **A1**), é considerado o caso em que `c` é um ciclo. Neste caso é facilmente verificado que a matriz de `c` é positiva. Isto segue pela definição do Critério 1 (verdadeiro por hipótese), que pode ser visualizada na Figura 4.2. Este ramo corresponde ao item (a) na prova do Teorema 4.4.7.

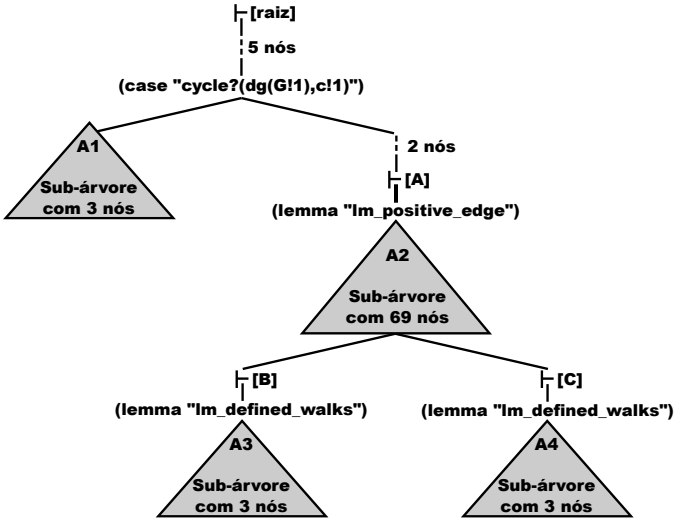


Figura 4.4: Árvore de prova do Lema `lm_pstv_cycles_pstv_circuits` cuja especificação pode ser visualizada na Figura 4.3. Esta árvore possui um total de 116 nós, que são os sequentes na formalização, onde foram empregados 116 comandos de prova.

- Na sub-árvore à direita, o caso em que `c` não é um ciclo, mas um circuito qualquer de `G` é considerado. Este ramo corresponde ao item (*b*) na prova do Teorema 4.4.7. Esta parte da prova é descrita a seguir com destaque para alguns sequentes e lemas aplicados.

Assim, na sub-árvore à direita o sequente [**A**] em destaque é apresentado abaixo. Note que, neste sequente, no consequente `[1]` aparece a fórmula que caracteriza o caso em que `c` não é um ciclo. Note que as hipóteses presentes são: o comprimento de `c` é maior que zero - antecedente {`-1`}; o precaminho `c` é um circuito de `G` - antecedente {`-2`}; e `k` é uma medida limitante - antecedente {`-3`}.

- Sequente [**A**], Figura 4.4

```
{-1}  length(c) > 0
{-2}  circuit?(dg(G), c)
[-3]  limiting_measure?(G)(k)
  |-------
[1]   cycle?(dg(G), c)
[2]   positive?(wgt_walk(G, c))
```

Assim como apresentado na prova do Teorema 4.4.7, neste ponto da prova é aplicado o Lema `lm_positive_edge`, que corresponde ao Lema 4.4.6. Este passo da formalização é efetuado através do comando `(lema "lm_positive_edge")`. Após este passo da prova, uma sub-árvore (rotulada por **A2**) com sessenta e nove nós é gerada. É nesta sub-árvore que aparece o objetivo mais relevante na formalização: verificar que a conclusão apresentada na Equação ($\star\star$) na prova do Teorema 4.4.7 é válida. Com este objetivo foi aplicado

o Lema 4.4.5, que na Figura 4.4 aparece com o nome `lm_defined_walks`. Note que este lema deve ser aplicado duas vezes: uma vez na verificação de que a matriz do caminho `c ^ (i, i + 1)` é definida na posição `(k)(k)`, outra vez na verificação de que a matriz do caminho `c ^ (i + 1, length(c) - 1)` é definida na posição `(k)(k)`. Em PVS isto representa dois ramos na árvore de prova. Assim, tais situações estão ilustrados nos sequentes [**B**] e [**C**] apresentados a seguir:

- Sequente [**B**], Figura 4.4

```
[-1]  (wgt_walk(G, c^(i, i + 1)) *
         wgt_walk(G, c^(i + 1, length(c) - 1)))(k)(k) = -1
[-2]  e = (c`seq(i), c`seq(1 + i))
[-3]  wgt(G)(e)(k)(k) = 1
[-4]  length(c) > 0
[-5]  walk?(dg(G), c)
[-6]  c!1`seq(0) = c`seq(length(c) - 1)
[-7]  limiting_measure?(G)(k)
  |-------
{1}   wgt_walk(G, c^(i, i + 1))(k)(k) /= -1
```

Note que, tanto no sequente [**B**] quanto o sequente [**C**], é possível visualizar as hipóteses de que: a multiplicação das matrizes de `c ^ (i, i + 1)` e `c ^ (i + 1, length(c) - 1)` é igual a `-1` na posição `(k)(k)`, consiste da hipótese que leva ao consequente `[1]` em ambos os casos, gerando assim uma contradição - antecedente `[-1]`; o caminho `c` possui uma aresta `e` cuja matriz é positiva na posição `(k)(k)`, tal conclusão é resultado do Lema `("lm_positive_edge")` - antecedentes `[-2]` e `[-3]`; O caminho `c` é um circuito do MWG `G` - antecedentes `[-5]` e `[-6]`; e a medida `k` é uma medida limitante para `G` - antecedente `[-7]`.

- Sequente [**C**], Figura 4.4

```
{-1}  (wgt_walk(G, c^(i, i + 1)) *
         wgt_walk(G, c^(i + 1, length(c) - 1)))(k)(k) = -1
[-2]  e = (c`seq(i), c`seq(1 + i))
[-3]  wgt(G)(e)(k)(k) = 1
[-4]  length(c) > 0
[-5]  walk?(dg(G), c)
[-6]  c!1`seq(0) = c`seq(length(c) - 1)
[-7]  limiting_measure?(G)(k)
  |-------
{1}   wgt_walk(G, c ^ (i + 1, length(c) - 1))(k)(k) /= -1
```

Como mencionado anteriormente, tanto no sequente [**B**] quanto no sequente [**C**], a

```
vertex_labeling(G): TYPE = [vert(dg(G)) -> below[N]]

limiting_labeling?(G: wdg, F: vertex_labeling(G)): bool =
  (FORALL(e: edge(dg(G))): wgt(G)((e))(F(e‘1))(F(e‘2)) >= 0) AND
  (FORALL c: cycle?(dg(G),c) => wgt_walk(G, c)(F(first(c)))(F(last(c))) = 1)

criterion_two?(G): bool =
  EXISTS (F: vertex_labeling(G)): limiting_labeling?(G, F)
```

Figura 4.5: Definição na linguagem de especificação do PVS do critério $\mathcal{CII}$. As definições nesta figura correspondem às definições 4.4.8 e 4.4.9.

fórmula {1} no consequente leva a uma contradição, pois indicam que as entradas das matrizes dos caminhos `c ^ (i, i + 1)` e `c ^ (i + 1, length(c) - 1)`, na posição `(k)(k)`, são iguais a `-1`, o que pelo Lema `lm_defined_walks` não pode ser verdade. Assim, o passo de prova que fecha estes ramos consiste da aplicação deste lema através do comando `(lemma "lm_defined_walks")`, que pode ser também visualizado na árvore de prova, presente na Figura 4.4.

4.5.2 Formalização em PVS da Prova do Teorema 4.4.11

Nesta subseção apresenta-se a formalização do lema `ll_pstv_pos_circuits` e do teorema `ll_pstv_circuits` que estão formalizados em PVS na *teoria* `matrix_wdg` e correspondem ao Teorema 4.4.11. O lema corresponde à parte da prova do Teorema 4.4.11 onde se verifica que, se o critério $\mathcal{CII}$ é verdadeiro, então todo circuito tem uma matriz de medida com uma posição definida, a saber a posição correspondente ao rótulo dos vértices extremos do circuito dado pela rotulagem limitante. O teorema corresponde à parte da prova onde se verifica que, se o critério $\mathcal{CII}$ é verdadeiro, então todo circuito tem uma matriz de medida positiva, o que se torna uma verificação simples, dado o lema anterior. A Figura 4.5 apresenta a parte da especificação onde o critério $\mathcal{CII}$ é definido. Note que três funções são apresentadas nesta figura e descritas a seguir:

`vertex_labeling:` aplicação dos vértices do MWG G no conjunto $\{0, \ldots, N-1\}$ que determina a rotulagem dos vértices;

`limiting_labeling?:` booleano que determina condições para que, dado um MWG G e uma rotulagem dos vértices de G, tal rotulagem seja limitante;

`criterion_two?:` booleano que representa o critério $\mathcal{CII}$. Se verdadeiro, significa que existe uma rotulagem limitante para um dado MWG G.

```
ll_pstv_pos_circuits: LEMMA
  FORALL(G: wdg, F: vertex_labeling(G)): limiting_labeling?(G, F) IMPLIES
  FORALL(c | circuit?(dg(G), c)): wgt_walk(G, c)(F(first(c)))(F(last(c))) = 1

ll_pstv_circuits: THEOREM
  criterion_two?(G) IMPLIES mwg_termination?(G)
```

Figura 4.6: Definição na linguagem de especificação do PVS do Lema `ll_pstv_pos_circuits` e Teorema `ll_pstv_circuits` respectivamente. Ambos correspondem ao Teorema 4.4.11.

Na Figura 4.6 são apresentados o Lema `ll_pstv_pos_circuits` e o Teorema `ll_pstv_circuits`, conforme é escrito na linguagem de especificação do PVS. Em seguida, apresenta-se a formalização do lema `ll_pstv_pos_circuits`, onde alguns sequentes de prova são descritos. Na Figura 4.7 a árvore de prova do lema é apresentada. A correspondência entre os sequentes de prova e os nós da árvore é mostrada ao longo da prova e destacada na árvore através do rótulos em vermelho.

Formalização do lema `ll_pstv_pos_circuits`:

Como primeiro objetivo de prova na formalização deste lema, tem-se Sequente [**raiz**], na Figura 4.7. Neste sequente existe apenas uma fórmula no consequente, onde as variáveis `G` e `F`, representando respectivamente um MWG e uma rotulagem dos vértices do MWG, ocorrem universalmente quantificadas. Assim, aplica-se uma regra de skolemização, a fim de dar nomes às variáveis `G` e `F`, eliminando o quantificador. Desta forma, existe um MWG `G` e uma rotulagem limitante `F` para `G`. O objetivo é verificar que, para todo pré-caminho `c`, se este for um circuito de `G`, a matriz de medida de `c` é positiva na posição determinada pela rotulagem limitante correspondente ao rótulo do primeiro e último vértices de `c`, que são iguais, já que `c` é um circuito.

- Sequente [**raiz**], Figura 4.7

```
  |-------
{1}   FORALL (G: wdg, F: vertex_labeling(G)): limiting_labeling?(G, F) IMPLIES
         (FORALL (c: prewalk[T] | circuit?(dg(G), c)):
             wgt_walk(G, c)(F(first(c)))(F(last(c))) = 1)
```

Neste ponto, a estratégia é aplicar indução no comprimento do circuito. Esta estratégia é desenvolvida primeiro através do comando `(measure-induct+ "length(c)"("c"))`, que pode ser visualizado na árvore de prova, gerando um novo nome para a variável de indução, que aparecerá como `x` nos sequentes seguintes. Mas observe que, com esta regra, `measure-induct+`, a indução é sobre o comprimento do circuito `c`. Porém, com o auxílio

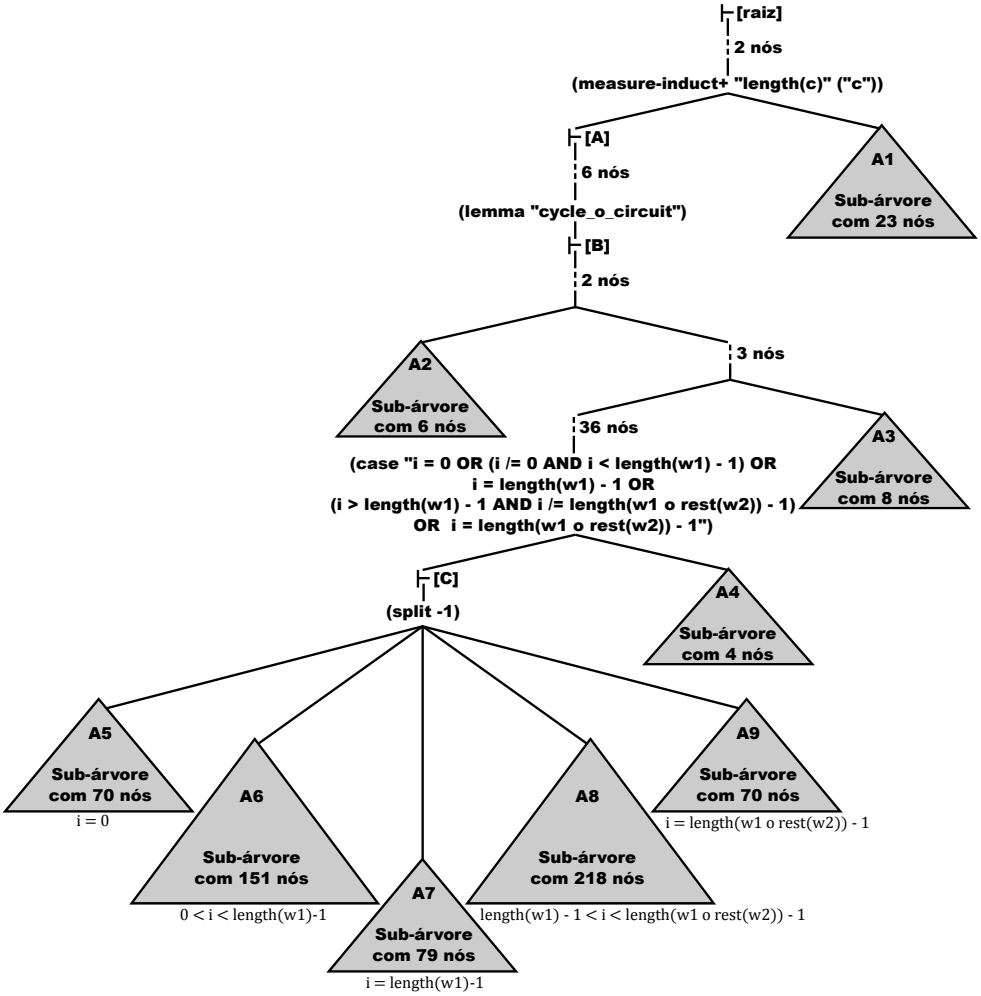


Figura 4.7: Árvore de prova do Lema `ll_pstv_pos_circuits`. A especificação consta na Figura 4.6. A árvore possui 678 nós, que são os sequentes na formalização. Foram empregados 678 comandos de prova.

do Lema `cycle_o_circuit`, que corresponde ao Lema 3.1.16, fica claro que a indução é desenvolvida de fato sobre a estrutura do caminho `c`.

Após o comando de indução, uma ramificação é gerada na árvore de prova, na figura representada por uma sub-árvore à direita, rotulada por **A1**, e o restante da prova à esquerda com algumas sub-árvores. A sub-árvore **A1** possui vinte e três nós. Os objetivos desta sub-árvore consistem em condições de checagem de tipos geradas pelo *typechecker* do PVS durante a prova. As sub-árvores à esquerda correspondem à base de indução e ao passo indutivo da prova, e serão descritas a seguir.

Assim, é apresentado na sub-árvore à esquerda após o comando de indução o sequente [**A**]. Observe que neste sequente é possível visualizar, expresso na fórmula antecedente `{-1}`, a hipótese de indução, onde sempre que um circuito `y` de `G` for tal que o comprimento de `y` é menor que o comprimento do circuito `x`, então a matriz de `y` é positiva na posição correspondente aos rótulos dos extremos de `y`, determinados pela rotulagem `F`. As

fórmulas `[-2]` e `1` do sequente [**A**] correspondem, respectivamente, à hipótese de que `F` é uma rotulagem limitante e ao principal objetivo de prova, que é verificar que a matriz do circuito `x` é positiva na posição determinada pela rotulagem `F` aplicada aos vértices extremos do circuito `c`.

• Sequente [**A**], Figura 4.7

```
{-1}  FORALL (y: {c | circuit?(dg(G), c)}): length(y) < length(x) IMPLIES
         wgt_walk(G, y)(F(first(y)))(F(last(y))) = 1
[-2]  limiting_labeling?(G, F)
  |-------
{1}   wgt_walk(G, x)(F(first(x)))(F(last(x))) = 1
```

Assim, alguns passos após o sequente [**A**] é aplicado o comando de prova `(lemma "cycle_o_circuit")`, para trazer às fórmulas do antecedente a hipótese correspondente ao lema, que pode ser visualizada no sequente [**B**], apresentado a seguir, na fórmula antecedente {`-1`}. Como mencionado anteriormente, o Lema `cycle_o_circuit` possibilita uma prova cujo esquema de indução é sobre a estrutura do circuito `x`, ao invés de ser sobre o comprimento de `x` como sugerido inicialmente. Além disso, é possível visualizar no sequente [**B**]: a hipótese de que `x` é um circuito de `G` - antecedentes `[-3]`, `[-4]` e `[-6]`; uma nova variável `k1`, que dá nome ao rótulo do vértice extremo de `x` - antecedente `[-2]`; hipótese de indução e hipótese inicial de que `F` é limitante - antecedentes `[-7]` e `[-8]`, respectivamente; e tese principal do teorema - consequente `[1]`.

• Sequente [**B**], Figura 4.7

```
{-1}  FORALL (G: digraph[T], w: prewalk[T]): circuit?(G, w) IMPLIES
       (cycle?(G, w) OR (EXISTS (w1, w2: prewalk[T]):
       eq_circuit?(G, w, w1 o rest(w2)) AND cycle?(G, w1) AND circuit?(G, w2)))
[-2]  F(x`seq(0)) = k1
[-3]  length(x) > 0
[-4]  walk?(dg(G), x)
[-5]  x`seq(0) = x!1`seq(length(x) - 1)
[-6]  length(x) > 1
[-7]  FORALL (y: {c | circuit?(dg(G), c)}): length(y) < length(x) IMPLIES
         wgt_walk(G, y)(F(y`seq(0)))(F(y`seq(y`length - 1))) = 1
[-8]  limiting_labeling?(G, F)
  |-------
[1]   wgt_walk(G, x)(k1)(k1) = 1
```

Assim, instanciando o antecende {`-1`} do sequente [**B**] com o MWG `G` e o circuito `x` tem-se duas possibilidades para `x`, ou este é um ciclo ou é equivalente à composição de um ciclo e um circuito. Desta forma uma nova ramificação é gerada na prova, denotada

na Figura 4.7 por uma ramificação abaixo do sequente [**B**] onde uma sub-árvore denotada por **A2** é destacada à esquerda e o restante da prova em várias sub-árvores à direita.

A sub-árvore **A2** corresponde à prova do caso em que `x` é um ciclo, ou seja, corresponde à base de indução na prova do Teorema 4.4.11. Pela hipótese de `F` ser uma rotulagem limitante, este caso é facilmente completado.

A sub-árvore à direita do nó [**B**] corresponde ao caso em que o circuito `x` é equivalente à composição de um ciclo e um circuito. Nomes são dados a essas novas variáveis: o ciclo é denominado `w1` e o circuito `w2`. Assim, a hipótese de indução é instanciada com o circuito `w2`, e a sub-árvore **A3** corresponde à prova de que `w2` é uma instância válida para a hipótese de indução, ou seja, tem comprimento menor que o comprimento de `x`.

Na sub-árvore à esquerda de **A3**, o primeiros passos de prova são dedicados a obter as conclusões correspondentes às Equações (B), (C), (D) e (E) na prova do Teorema 4.4.11. Este objetivo é alcançado em 36 passos de prova. Assim, a igualdade dada por

```
x = (w1 o rest(w2))^(i, length(w1 o rest(w2)) - 2) o (w1 o rest(w2))^(0, i)
```

é obtida. Esta igualdade corresponde à Equação (E). Assim, casos são considerados para a variável `i` através do comando de prova

```
(case "i = 0 OR (i /= 0 AND i < length(w1) - 1) OR i = length(w1) - 1
 OR (i > length(w1) - 1 AND i /= length(w1 o rest(w2)) - 1)
 OR i = length(w1 o rest(w2)) - 1")
```

que também pode ser visualizado na Árvore 4.7. Após considerar os casos para `i`, uma ramificação é gerada e a sub-árvore **A4** corresponde à prova de que a disjunção destes casos, como apresentada acima, é válida. Na parte da árvore de prova à esquerda de **A4** tem-se a parte principal do passo indutivo, onde cada caso para `i` é analisado separadamente e o sequente [**C**] é inicialmente apresentado.

No sequente [**C**] é possível visulizar: na fórmula antecedente {`-1`}, a disjunção dos casos para a variável `i`; o antecedente `[-2]`, já mencionado anteriormente, corresponde à Equação (E) na prova do Teorema 4.4.11; o antecedente `[-5]` corresponde à Equação (C); os antecedentes `[-7]` e `[-8]` correspondem à Equação (D), e compõem uma parte importante na prova do teorema, que é a conclusão de que as matrizes do ciclo `w1` e do circuito `w2` são positivas na posição dada pela rotulagem limitante `F`; os antecedentes `[-10]` a `[-15]` falam sobre as sequências `w1` e `w2`, basicamente que são caminhos de comprimento maior que 1 começando e terminado no mesmo vértice; os antecedentes `[-18]` a `[-20]` trazem as mesmas conclusões sobre o circuito `x`; finalmente, no antecedente `[-21]`, a hipótese principal de que `F` é uma rotulagem limitante e no consequente `[1]` a conclusão

principal a que se quer chegar na prova, de que a matriz de `x` é positiva na posição determinada por `F`.

• Sequente [**C**], Figura 4.7

```
{-1}      i = 0 OR (i /= 0 AND i < length(w1) - 1) OR i = length(w1) - 1
       OR (i > length(w1) - 1 AND i /= length(w1 o rest(w2)) - 1)
       OR i = length(w1 o rest(w2)) - 1
[-2]  i < length(w1 o rest(w2))
[-3]  x = (w1 o rest(w2))^(i, length(w1 o rest(w2)) - 2) o (w1 o rest(w2))^(0, i)
[-4]  F(w1`seq(0)) = k2
[-5]  w1`seq(0) = w2`seq(w2`length - 1)
[-6]  length(w1 o rest(w2)) > 1
[-7]  wgt_walk(G, w1)(k2)(k2) = 1
[-8]  wgt_walk(G, w2)(k2)(k2) = 1
[-9]  eq_circuit?(dg(G), x, w1 o rest(w2))
[-10] walk?(dg(G), w1)
[-11] w1`seq(0) = w1`seq(length(w1) - 1)
[-12] length(w1) > 1
[-13] walk?(dg(G), w2)
[-14] w2`seq(0) = w2`seq(length(w2) - 1)
[-15] length(w2) > 1
[-16] F(x`seq(0)) = k1
[-17] length(x) > 0
[-18] walk?(dg(G), x)
[-19] x`seq(0) = x`seq(length(x) - 1)
[-20] length(x) > 1
[-21] limiting_labeling?(G, F)
  |-------
[1]   wgt_walk(G, x)(k1)(k1) = 1
```

Assim, logo após o sequente [**C**], a regra de prova `split -1` é aplicada. Esta é uma regra de simplificação proposicional que elimina a disjunção da fórmula antecedente {`-1`}, gerando uma ramificação na árvore de prova, onde, para cada fórmula na disjunção, um novo ramo aparece. Portanto, como em {`-1`} existem 5 possibilidades para `i`, uma ramificação com 5 sub-árvores é gerada, na Figura 4.7 destacadas pelos rótulos **A5** a **A9**, onde é possível observar que cada umas dessas sub-árvores corresponde a um dos casos para `i` e em destaque encontra-se quantos comandos de prova foram necessários para cada caso. Estes casos correspondem, na prova do Teorema 4.4.11, aos subitens (i) a (v) constando no item (b).

Capítulo 5

Equivalências entre CCG e MWG

Neste capítulo será apresentada a especificação em PVS de CCG bem como a conexão entre CCG's e MWG's. Na Figura 5.1 é apresentada a hierarquia das *sub-teorias* envolvidas nesta construção, com destaque para as *sub-teorias* a serem apresentadas neste capítulo.

As definições e demonstrações apresentadas neste capítulo correspondem às definições especificadas e às demonstrações formalizadas em PVS como parte da *teoria* `ccg`, cuja hierarquia pode ser visualizada na Figura 5.1.

As *sub-teorias* `cc_def`, `ccg_def` e `ccg` estão relacionadas com a especificação de CCG. Nesta especificação um CCG G' é tratado como um digrafo, assim como apresentado na Seção 3.2, onde cada vértice é um contexto de chamado (Definição 3.2.1). Assim, um vértice representa um chamado recursivo na definição recursiva que está sendo abstraída pelo CCG, e a presença de uma aresta significa que o chamado representado pelo vértice de entrada, após análise local sobre os parâmetros atuais do vértice de saída e sobre as condições do vértice de entrada, pode ser executado logo após o chamado representado pelo vértice de saída. Além disso, uma família $\mathcal{F}_{G'}$ de medidas sobre o domínio de uma relação bem fundada é associada ao CCG G'. Mais especificamente, na *sub-teoria* `ccg` é especificada uma definição para terminação em CCG, condicionada à existência de uma *combinação de medidas* da família associada ao CCG estritamente decrescente para um caminho do CCG.

A *sub-teoria* `ccg_to_mwg` consiste da especificação de um operador, assim como da formalização de propriedades deste operador, para a construção de um MWG G a partir de um CCG G' com sua respectiva família $\mathcal{F}_{G'}$ de medidas. Além da formalização de que, se G' é terminante, então G também é terminante.

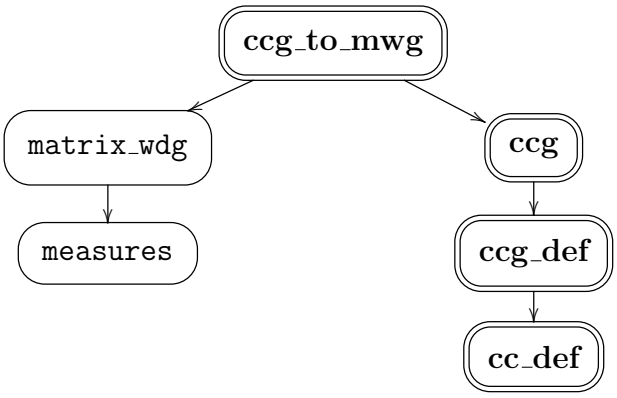


Figura 5.1: Hierarquia da *sub-teoria* `ccg_to_mwg`

5.1 Sintaxe de uma Linguagem Funcional de Primeira Ordem

Como mencionado anteriormente, um grafo de contexto de chamado é uma estrutura de dados através da qual modela-se o comportamento de uma definição recursiva em uma linguagem funcional de primeira ordem. Assim, começamos por introduzir a sintaxe de uma linguagem funcional de primeira ordem. Uma definição recursiva em uma linguagem funcional de primeira ordem é uma expressão construída sobre um conjunto de símbolos desta linguagem. Este conjunto é formado por: um conjunto enumerável de variáveis, denotado por $\mathcal{V}ar$; símbolos de constantes e operadores n-ários efetivamente computáveis, denotado por $\mathcal{S}$; um símbolo para expressão condicional *ite*; e o conjunto de símbolos de função n-ária, denotado por $\mathcal{D}ef$, onde símbolos para funções possivelmente não computáveis podem estar contidos. Assim o conjunto de expressões construído sobre tais símbolos é denotado por $\mathcal{E}xp$. E o corpo de um símbolo de função sobre $\mathcal{D}ef$, também denominado *definição de função*, é uma expressão $e \in \mathcal{E}xp$ definida como segue:

Definição 5.1.1 (Sintaxe da definição de função)**:** *A definição de uma função n-ária $f \in \mathcal{D}ef$ é uma expressão e_f com a seguinte sintaxe: $f(x_1, \ldots, x_n) = e_f$, onde $x_i \in \mathcal{V}ar$, para todo $1 \leqslant i \leqslant n$, são denominadas parâmetros formais e o corpo (ou definição) de f é uma expressão $e_f \in \mathcal{E}xp$ da forma:*

$$e_f \ := \ c \mid x \in \{x_1 \ldots, x_n\} \mid g(e_1, \ldots, e_k) \mid ite(e_1, e_2, e_3) \ \mid f(e_1, \ldots, e_n),$$

onde

- $c \in \mathcal{S}$ *é uma constante;*
- $x \in \{x_1 \ldots, x_n\} \subset \mathcal{V}ar$ *é uma variável;*

- $g \in \mathcal{S}$ *é um operador* $k-$*ário conhecido terminante;*
- $ite(e_1, e_2, e_3)$ *é uma expressão condicional, onde* e_1 *é uma expressão booleana e representa a condição para* e_2 *e* $\neg e_1$ *representa a condição para* e_3*;*
- $f(e_1, \ldots, e_n)$ *é um chamado recursivo, onde os parâmetros* e_i*, para todo* $i \in \{1, \ldots, n\}$*, são denominados os atuais do chamado.*

Exemplo 5.1.1: A função de Ackermann abaixo

$$Ack(m,n) = \begin{cases} n+1, & \text{se } m = 0 \\ Ack(m-1,1), & \text{se } m > 0 \wedge n = 0 \\ Ack(m-1, Ack(m, n-1)), & \text{se } m > 0 \wedge n > 0 \end{cases}$$

definida para m e n naturais, de acordo com a sintaxe apresentada em 5.1.1, é especificada como segue:

$$Ack(m,n) = ite(m = 0, n+1, ite(n = 0, Ack(m-1,1), Ack(m-1, Ack(m, n-1)))),$$

onde os operadores binários $=$, $+$, e $-$ estão presentes nas sub-expressões "$m = 0$", "$n = 0$", "$n+1$", "$m-1$" e "$n-1$". Note que estes operadores são aplicados a expressões que são constantes ou variáveis do conjunto de parâmetros formais da definição.

Observe que, como chamados recursivos são expressões, estes podem ser aninhados, sem violar a descrição gramatical. Em geral a expressão condicional e_1, em expressões do tipo $ite(e_1, e_2, e_3)$, admite ocorrências de expressões que contenham chamados recursivos. A condição sob a qual cada chamado recursivo é realizado é construída por conjugação da condição nas expressões do tipo $ite(e_1, e_2, e_3)$ ou sua negação, conforme o chamado recursivo ocorra na expressão e_2 ou e_3, respectivamente, conforme definição abaixo.

Definição 5.1.2 (Condição do chamado recursivo)**:** *Seja* $f(x_1, \ldots, x_n) = e_f$ *uma definição recursiva. Se na definição de* f *ocorre um chamado recursivo* $f(e_1, \ldots, e_n) = e'$*, então a condição deste chamado, denotada por* $\mathcal{C}(e')$*, é definida recursivamente da seguinte maneira:*

- *se* $e_f = e'$ *então* $\mathcal{C}(e') = true$*;*
- *se* $e_f = g(e_1, \ldots, e_j, \ldots, e_n)$ *e* e' *ocorre em* e_j*, então* $\mathcal{C}(e')$ *é a condição de* e' *em* e_j*;*
- *se* $e_f = f(e_1, \ldots, e_j, \ldots, e_n)$ *e* e' *ocorre em* e_j*, da mesma forma* $\mathcal{C}(e')$ *é a condição de* e' *em* e_j*;*

- *se* $e_f = ite(e_1, e_2, e_3)$*, então:*
  - *se* e' *ocorre em* e_1*, então* $\mathcal{C}(e')$ *é a condição de* e' *em* e_1*;*
  - *se* e' *ocorre em* e_2*, então* $\mathcal{C}(e')$ *é a conjunção de* e_1 *com a condição de* e' *em* e_2*;*
  - *se* e' *ocorre em* e_3*, então* $\mathcal{C}(e')$ *é a conjunção de* $\neg e_1$ *com a condição de* e' *em* e_3*.*

Exemplo 5.1.2: Voltando à função de Ackermann, note que o chamado $Ack(m-1, 1)$ tem condição $\neg m = 0 \wedge n = 0$; e os chamados $Ack(m-1, Ack(m, n-1))$ e $Ack(m, n-1)$ tem condição $\neg m = 0 \wedge \ \neg n = 0$.

5.2 Grafos de Contextos de Chamado

A próxima definição caracteriza um contexto de chamado que, como mencionado anteriormente, são os vértices de um CCG e representam um chamado recursivo na definição de função.

Definição 5.2.1 (Contexto de Chamado)**:** *Seja* e' *um chamado recursivo de* f *em* $f(x_1, \ldots, x_n) = e_f$*. Assim, o contexto de chamado relativo a* e' *no CCG é uma tripla* $\langle \mathcal{F}(e'),\ \mathcal{C}(e'),\ \mathcal{A}(e') \rangle$*, onde* $\mathcal{F}(e')$ *são os parâmetros formais do chamado* e'*,* $\mathcal{C}(e')$ *é a condição de* e' *em* e_f *e* $\mathcal{A}(e')$ *os parâmetros atuais de* e'*.*

A construção de um CCG, como definido em PVS, é desenvolvida sobre um conjunto $\mathcal{MT}$ com uma relação bem fundada $\succ$, isto quer dizer que qualquer subconjunto $\mathcal{B}$ não vazio de $\mathcal{MT}$ possui um elemento mínimo, ou seja, um $b \in \mathcal{B}$ tal que não vale $b \succ b'$ qualquer que seja $b' \in \mathcal{B}$. A relação $\succ$ é estendida por $\succcurlyeq$ da seguinte maneira:

$$\forall\, m_1, m_2 \in \mathcal{MT};\ \ m_1 \succcurlyeq m_2 \ := \ m_1 = m_2 \ \vee \ m_1 \succ m_2$$

Uma *medida* é uma função definida no conjunto de variáveis $\mathcal{V}ar$ sobre o conjunto $\mathcal{MT}$. Um conjunto enumerável de medidas $\{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$ será, como mencionado anteriormente, denotado por $\mathcal{F}$ e denominado uma *família de medidas*. Assim, um elemento da família $\mathcal{F}$ é uma aplicação $\mu_k : \mathcal{V}ar^n \to \mathcal{MT}$, para $0 \leqslant k \leqslant N-1$, onde N é a cardinalidade da família $\mathcal{F}$ e $\mathcal{V}ar^n = \underbrace{\mathcal{V}ar \times \ldots \times \mathcal{V}ar}_{n-vezes}$, ou seja, μ_k tem aridade n. O conjunto $\{k \in \mathbb{N} \mid 0 \leqslant k \leqslant N-1\}$ dos índices dos elementos de $\mathcal{F}$ será denotado por $\mathcal{N}$, isto é, $\mathcal{N} = \{0, 1, \ldots, N-1\}$.

Definição 5.2.2 (Grafo de contextos de chamado - CCG)**:** *Um CCG é um digrafo com uma família de medidas associada, onde o conjunto de vértices é um conjunto de contextos de chamado e o conjunto de arestas é formado por pares de vértices, como definido em Definição 3.1.1. CCG's são denotados por* $G' = \langle V_{G'}, E_{G'}, \mathcal{F}_{G'} \rangle$.

5.3 Especificação de terminação em CCG

Note que, no contexto deste capítulo, pré-caminhos e caminhos no digrafo de um CCG são sequências de contextos de chamado. Assim, preferencialmente, vértices serão representados pelos símbolos $cc, cc_0, cc_1, \ldots$ e caminhos de comprimento n por $w = (cc_0, cc_1, \ldots, cc_{n-1})$. E por simplicidade os parâmetros formais, as condições e os parâmetros atuais de um vértice cc serão denotados respectivamente por $\mathcal{F}_{cc}$, $\mathcal{C}_{cc}$ e $\mathcal{A}_{cc}$.

A definição da noção de terminação para um CCG G' baseia-se na existência de uma sequência de medidas da família $\mathcal{F}_{G'}$, para cada caminho $w \in \mathcal{W}_{G'}$, satisfazendo condições para garantir que esta sequência de medidas gere uma sequência estritamente decrescente sobre $\mathcal{MT}$. No que segue, a notação para tratar tais sequências de medidas é introduzida e em seguida as definições referentes à *combinação de medidas* para um caminho no CCG, que são sequências sobre $\mathcal{N}$.

Notação: O conjunto

$$\boldsymbol{\mathcal{N}} = \bigcup_{k \in \mathbb{N}} \mathcal{N}^k = \bigcup_{k \in \mathbb{N}} \{0, \ldots, N-1\}^k$$

denota o conjunto de todas as possíves sequências de índices sobre $\mathcal{N}$ com comprimento finito arbitrário.

Definição 5.3.1 (Combinação de Medidas)**:** *Uma* combinação de medidas *é uma aplicação* $\mathcal{MC} : \mathcal{P}_w \to \boldsymbol{\mathcal{N}}$ *definida por* $\mathcal{MC}(w) = \{mc \in \boldsymbol{\mathcal{N}} \mid \ell(mc) = \ell(w)\}$*, ou seja,* $\mathcal{MC}(w)$ *é o conjunto de sequências finitas de índices tomadas sobre o conjunto* $\mathcal{N} = \{0, \ldots, N-1\}$*, tal que o comprimento destas sequências é igual ao comprimento do pré-caminho* w*.*

Note que, para um caminho w em um CCG G' com uma família de medidas $\mathcal{F}_{G'}$ de cardinalidade N, uma combinação de medidas para w determina um conjunto de sequências de medidas sobre $\mathcal{F}_{G'}$. Assim, nas definições que seguem uma família de medidas de cardinalidade N é considerada.

Considere uma família de medidas $\mathcal{F}$, um conjunto de contextos de chamado V e o conjunto $\texttt{bool} = \{\texttt{true}, \texttt{false}\}$. Define-se uma aplicação booleana

$$\mathcal{G} \ : \ \mathcal{F} \times V \times \mathcal{N}^2 \times \texttt{bool} \ \to \ \texttt{bool}$$

por

$$\mathcal{G}(\mathcal{F}, cc, i, j, \beta) \ := \ \mathcal{C}_{cc}(\mathcal{F}_{cc}) \ \Rightarrow \ \mu_i(\mathcal{F}_{cc}) \succcurlyeq \mu_j(\mathcal{A}_{cc}) \ \wedge \ (\ \mu_i(\mathcal{F}_{cc}) = \mu_j(\mathcal{A}_{cc}) \ \Rightarrow \ \beta \ )$$

que é crucial na definição de combinação de medidas que leva a uma sequência decrescente em $\mathcal{MT}$, denominada *combinação de medidas decrescente*, e na definição de combinação de medidas que leva a uma sequência estritamente descrescente em $\mathcal{MT}$, denominada *combinação de medidas estritamente decrescente*.

Definição 5.3.2 (Combinação de medidas decrescente)**:** *Dada uma família de medidas $\mathcal{F}$, define-se a aplicação booleana*

$$\mathcal{G}te : \mathcal{F} \times \mathcal{P}_w \times \mathcal{MC}(w) \to \texttt{bool},$$

por

$$\mathcal{G}te(\mathcal{F}, w, mc) \ := \ \forall\, i \in \{0, \ldots, n-2\} : \ \mathcal{G}(\mathcal{F}, cc_i, k_i, k_{i+1}, \texttt{true})$$

onde $w = (cc_0, \ldots, cc_{n-1}) \in \mathcal{P}_w$, com $\ell(w) = n$, $mc = \{k_0, \ldots, k_{n-1}\} \in \mathcal{MC}(w)$ e $k_j \in \{0, \ldots, N-1\}$ para todo $j \in \{0, \ldots, n-1\}$.

Note que, para uma família de medidas $\mathcal{F}$, um pré-caminho $w = (cc_0, cc_1, \ldots, cc_{n-1})$, com $\ell(w) = n$, e uma combinação de medidas $mc = \{k_0, \ldots, k_{n-1}\} \in \mathcal{MC}(w)$, se $\mathcal{G}te(\mathcal{F}, w, mc) = \texttt{true}$, então a combinação de medidas leva a uma sequência decrescente de valores em $\mathcal{MT}$, pois resulta que, para todo $i \in \{0, \ldots, n-2\}$, tem-se que $\mu_{k_i}(\mathcal{F}_{cc_i}) \succcurlyeq \mu_{k_{i+1}}(\mathcal{A}_{cc_i})$.

Definição 5.3.3 (Combinação de medidas estritamente decrescente)**:** *Dada uma família de medidas $\mathcal{F}$, define-se a aplicação booleana*

$$\mathcal{G}t : \mathcal{F} \times \mathcal{P}_w \times \mathcal{MC}(w) \to \texttt{bool},$$

por

$$\mathcal{G}t(\mathcal{F}, w, mc) \ := \ \mathcal{G}te(\mathcal{F}, w, mc) \ \wedge \ \exists\, i \in \{0, \ldots, n-2\} : \ \mathcal{G}(\mathcal{F}, cc_i, k_i, k_{i+1}, \texttt{false})$$

onde $w = (cc_0, \ldots, cc_{n-1}) \in \mathcal{P}_w$, com $\ell(w) = n$, $mc = \{k_0, \ldots, k_{n-1}\} \in \mathcal{MC}(w)$ e $k_j \in \{0, \ldots, N-1\}$ para todo $j \in \{0, \ldots, n-1\}$.

Observe que, para uma família de medidas $\mathcal{F}$, um pré-caminho $w = (cc_0, cc_1, \ldots, cc_{n-1})$, com $\ell(w) = n$, e uma combinação de medidas $mc = \{k_0, \ldots, k_{n-1}\} \in \mathcal{MC}(w)$, se $\mathcal{G}t(\mathcal{F}, w, mc) = \texttt{true}$, então a combinação de medidas mc leva a uma sequência estritamente decrescente de valores em $\mathcal{MT}$, pois resulta que $\mathcal{G}te(\mathcal{F}, w, mc) = \texttt{true}$ e existe um $i \in \{0, \ldots, n-2\}$, tal que $\mu_{k_i}(\mathcal{F}_{cc_i}) \succ \mu_{k_{i+1}}(\mathcal{A}_{cc_i})$.

Assim, com base nas definições sobre combinação de medidas para um caminho, define-se a noção de terminação para um CCG como segue:

Definição 5.3.4 (Terminação em CCG)**:** *Dado um CCG G', a semântica de terminação para G' é expressa pela seguinte aplicação booleana $T_{ccg} : CCG \to \texttt{bool}$ definida por:*

$$\begin{aligned} T_{ccg}(G) \ := \forall\ c = (cc_0, \ldots, cc_{n-1}) \in \mathcal{C}ir(G'), \\ \exists\ mc = (k_0, \ldots, k_{n-1}) \in \mathcal{MC}(c)\ :\ k_0 = k_{n-1}\ \wedge\ \mathcal{G}t(\mathcal{F}_{G'}, c, mc) \end{aligned}$$

A definição anterior estabelece que um CCG G' é dito terminante se, para todo circuito c de G', com $\ell(c) = n$, existe uma combinação de medidas $mc = (k_0, \ldots, k_{n-1})$ para c, ou seja, de comprimento n, cujos elementos extremos sejam iguais, isto é, $k_0 = k_{n-1}$, tal que $\mathcal{G}t(\mathcal{F}_{G'}, w, mc) = \texttt{true}$. O fato de os extremos de mc serem iguais garante que a sequência de funções de medidas de $\mathcal{F}_{G'}$ tomada começa e termina com a mesma função de medida, gerando uma sequência decrescente sobre $\mathcal{MT}$.

Lema 5.3.5 (Combinação de medidas decrescente para sub-caminhos)**:** *Seja G' um CCG. Sejam ainda $w \in \mathcal{W}_{G'}$, com $\ell(w) = n$, dois números naturais i e j tomados no conjunto $\{0, \ldots, n-1\}$, e uma combinação de medidas $mc \in \mathcal{MC}(w)$. Assim,*

$$i \leqslant j\ \wedge\ \mathcal{G}te(\mathcal{F}_{G'}, w, mc) \quad \Rightarrow \quad \mathcal{G}te(\mathcal{F}_{G'}, w^{(i,j)}, mc^{(i,j)}) \tag{5.1}$$

Prova: Note que, se $w = (cc_0, \ldots, cc_{n-1})$ e $mc = (k_0, \ldots, k_{n-1})$ então, para i e j tais que $0 \leqslant i \leqslant j \leqslant n-1$, tem-se $w^{(i,j)} = (cc_i, \ldots, cc_j)$ e $mc^{(i,j)} = (k_i, \ldots, k_j)$, e portanto, o α-ésimo elemento de $w^{(i,j)}$ é $cc_{i+\alpha}$ e de $mc^{(i,j)}$ é $k_{i+\alpha}$. Além disso, $\ell(w^{(i,j)}) = \ell(mc^{(i,j)}) = j - i + 1$. Logo, expandindo a definição de $\mathcal{G}te$ na Equação 5.1, tem-se que:

$$i \leqslant j\ \wedge\ \forall\ \alpha_1 \in \{0, \ldots, n-2\} :\ \mathcal{G}(\mathcal{F}, cc_{\alpha_1}, k_{\alpha_1}, k_{\alpha_1+1}, \texttt{true})$$

$$\Rightarrow \quad \forall\ \alpha_2 \in \{0, \ldots, j-i-1\} :\ \mathcal{G}(\mathcal{F}, cc_{i+\alpha_2}, k_{i+\alpha 2}, k_{i+\alpha_2+1}, \texttt{true})$$

Assim, suponha que

$$\exists\ \alpha_2 \in \{0, \ldots, j-i-1\} :\ \neg\mathcal{G}(\mathcal{F}, cc_{i+\alpha_2}, k_{i+\alpha 2}, k_{i+\alpha_2+1}, \texttt{true}) \tag{5.2}$$

Note que, como $0 \leqslant \alpha_2 \leqslant j-i-1$, então $0 \leqslant i + \alpha_2 \leqslant n-2$. Portanto, por hipótese, para $\alpha_1 = i + \alpha_2$ tem-se que $\mathcal{G}(\mathcal{F}, cc_{i+\alpha_2}, k_{i+\alpha_2}, k_{i+\alpha_2+1}, \texttt{true})$, o que é uma contradição. Logo (5.2) é uma suposição falsa e segue a prova do lema. ∎

Lema 5.3.6 (Composição decrescente de combinação de medidas)**:** *Seja G' um CCG. Sejam ainda dois caminhos quaisquer $w_1 = (cc_0^1, \ldots, cc_{n_1-1}^1)$ e $w_2 = (cc_0^2, \ldots, cc_{n_2-1}^2)$*

em $\mathcal{W}_{G'}$ *e duas combinações de medidas* $mc_1 = (k_0^1, \ldots, k_{n_1-1}^1) \in \mathcal{MC}(w_1)$ *e* $mc_2 = (k_0^2, \ldots, k_{n_2-1}^2) \in \mathcal{MC}(w_2)$. *Assim,*

$$cc_{n_1-1}^1 = cc_0^2 \quad \wedge \quad k_{n_1-1}^1 = k_0^2 \quad \wedge \quad \mathcal{G}te(\mathcal{F}_{G'}, w_1, mc_1) \quad \wedge \quad \mathcal{G}te(\mathcal{F}_{G'}, w_2, mc_2)$$

$$\Downarrow$$

$$\mathcal{G}te(\mathcal{F}_{G'}, \circ_w(w_1, w_2), \circ_w(mc_1, mc_2))$$

Prova: Segue da definição de $\mathcal{G}te$. ∎

Lema 5.3.7 (Composição estritamente decrescente de combinação de medidas)**:** *Seja* G' *um CCG. Sejam ainda dois caminhos quaisquer de* $\mathcal{W}_{G'}$, $w_1 = (cc_0^1, \ldots, cc_{n_1-1}^1)$ *e* $w_2 = (cc_0^2, \ldots, cc_{n_2-1}^2)$, *e duas combinações de medidas* $mc_1 = (k_0^1, \ldots, k_{n_1-1}^1) \in \mathcal{MC}(w_1)$ *e* $mc_2 = (k_0^2, \ldots, k_{n_2-1}^2) \in \mathcal{MC}(w_2)$. *Assim,*

$$cc_{n_1-1}^1 = cc_0^2 \quad \wedge \quad k_{n_1-1}^1 = k_0^2 \quad \wedge \quad \mathcal{G}te(\mathcal{F}_{G'}, w_1, mc_1) \quad \wedge \quad \mathcal{G}te(\mathcal{F}_{G'}, w_2, mc_2)$$

$$\wedge \quad (\ \mathcal{G}t(\mathcal{F}_{G'}, w_1, mc_1) \ \vee \ \mathcal{G}t(\mathcal{F}_{G'}, w_2, mc_2)\ )$$

$$\Downarrow$$

$$\mathcal{G}t(\mathcal{F}_{G'}, \circ_w(w_1, w_2), \circ_w(mc_1, mc_2))$$

Prova: Segue da definição de $\mathcal{G}t$. ∎

5.4 Formalização da Equivalência entre Terminação em CCG e MWG

Nesta seção apresenta-se a construção da função $\boldsymbol{\mu} : E \to \mathcal{M}_{N\times N}$, descrita na Seção 4.2 e, portanto, a construção de um MWG G a partir de um CCG G'. Apresenta-se ainda os resultados sobre equivalência entre terminação em G e terminação em G'.

Definição 5.4.1 (Função indicadora)**:** *Dado um CCG* $G' = \langle V_{G'}, E_{G'}, \mathcal{F}_{G'}\rangle$, *define-se a aplicação* $\mathcal{I} : \{0, \ldots, N-1\} \times \{0, \ldots, N-1\} \times V_{G'} \to \{-1, 0, 1\}$, *por:*

$$\mathcal{I}(i, j, cc) := \begin{cases} 1, & \textit{se } \mathcal{G}(cc, \mu_i, \mu_j, false) \\ 0, & \textit{se } \mathcal{G}(cc, \mu_i, \mu_j, true) \\ -1, & \textit{caso contrário} \end{cases}$$

A seguinte definição atribui a cada aresta de um CCG um peso em forma de uma matriz quadrada cuja ordem é igual à cardinalidade da família de funções do CCG e cujas entradas são resultados de comparações entre as medidas da família de medidas do CCG.

Definição 5.4.2 (Função peso)**:** *Dado um CCG G' com uma família de medidas $\mathcal{F}_{G'}$ de cardinalidade N, a função que associa a cada aresta $e = (cc_1, cc_2) \in E_{G'}$ uma matriz quadrada de ordem N é uma aplicação $\boldsymbol{\mu} : E_{G'} \to \mathcal{M}_{N\times N}$ definida por:*

$$\boldsymbol{\mu}(e) \in \mathcal{M}_{N\times N},\ \textit{onde}\ \forall\, i,j \in \{0,\ldots,N\} :\ (\boldsymbol{\mu}(e))_{ij} = \mathcal{I}(i,j,cc_1)$$

Assim, dado um CCG G', a seguinte definição estabelece uma construção de um MWG G a partir de G'.

Definição 5.4.3 (Transformação de grafos de contexto de chamado para grafos com matrizes de medida)**:**

$$\mathcal{T}^{ccg}_{mwg} : CCG \to MWG$$

$$\mathcal{T}^{ccg}_{mwg}(G') = G = \langle V_{G'},\ E_{G'},\ \boldsymbol{\mu} : E_{G'} \to \mathcal{M}_{N\times N}\rangle$$

Observe que, dado um CCG G', a parte de G' que determina o digrafo, ou seja, $\langle V_{G'},\ E_{G'}\rangle$ permanece inalterada em $\mathcal{T}^{ccg}_{mwg}(G')$. Portanto, todo vértice, aresta ou caminho de G' é também um vértice, aresta ou caminho de $G = \mathcal{T}^{ccg}_{mwg}(G')$. Logo, o mesmo vale para circuitos e ciclos.

5.4.1 Matrizes de Medida com Posição Definida e Combinação de Medidas Decrescente

Lema 5.4.4 (Combinação de medidas decrescente implica matriz com posição definida)**:** *Seja G' um CCG e considere o MWG G obtido de G' por $\mathcal{T}^{ccg}_{mwg}(G')$. Assim, para todo caminho $w \in \mathcal{W}_{G'}$ e toda combinação de medidas $mc = (k_0, k_1, \ldots, k_{n-1}) \in \mathcal{MC}(w)$, vale que:*

$$\mathcal{G}te(\mathcal{F}_{G'}, w, mc) \Rightarrow (\hat{\boldsymbol{\mu}}(w))_{k_0 k_{n-1}} \geqslant 0$$

Prova: Dado G' um CCG, considere o MWG $G = \mathcal{T}^{ccg}_{mwg}(G')$ obtido de G'. Considere ainda, um caminho $w = (cc_0, \ldots, cc_{n-1}) \in \mathcal{W}_{G'}$ e uma combinação de medidas $mc = (k_0, k_1, \ldots, k_{n-1}) \in \mathcal{MC}(w)$ tais que $\mathcal{G}te(\mathcal{F}_{G'}, w, mc)$. A prova segue por indução em $\ell(w) = n$. Assim,

B.I. $\ell(w) = 1$: Neste caso, observe que $\ell(mc) = 1$ e portanto $k_0 = k_{n-1}$. Além disso, pela Definição 3.1.19, $\hat{\boldsymbol{\mu}}(w) = \mathbf{Id}$. Ou seja, $(\hat{\boldsymbol{\mu}}(w))_{k_0k_0} = (\mathbf{Id})_{k_0k_0} = 0$ (Definição 4.2.7).

P.I. $\ell(w) > 1$: Neste caso, w é a composição $w = \circ_w(w^{(0,1)}, w^{(1,n-1)})$ e segue do Lema 3.1.20 que $\hat{\boldsymbol{\mu}}(w) = \boldsymbol{\mu}(w^{(0,1)}) * \hat{\boldsymbol{\mu}}(w^{(1,n-1)})$.

Assim, de $\mathcal{G}te(\mathcal{F}_{G'}, w, mc)$ e expandindo a definição de $\mathcal{G}te$, tem-se que: $\forall\ i \in \{0, \ldots, n-2\}$: $\mathcal{G}(\mathcal{F}, cc_i, k_i, k_{i+1}, \texttt{true})$, em particular para $i = 0$, tem-se que $\mathcal{G}(\mathcal{F}, cc_0, k_0, k_1, \texttt{true})$. Assim, segue das Definições 5.4.1 e 5.4.2 que:

$$(\boldsymbol{\mu}(w^{(0,1)}))_{k_0,k_1} \geqslant 0 \tag{5.3}$$

Note ainda que $\ell(w^{(1,n-1)}) < \ell(w)$ e, além disso, pelo Lema 5.3.5 tem-se que $\mathcal{G}te(\mathcal{F}_{G'}, w^{(1,n-1)}, mc^{(1,n-1)})$. Portanto, por hipótese de indução, segue que:

$$(\hat{\boldsymbol{\mu}}(w^{(1,n-1)}))_{k_1 k_{n-1}} \geqslant 0 \tag{5.4}$$

Assim, das Equações 5.3 e 5.4 e do Lema 4.2.13, segue que:

$$(\boldsymbol{\mu}(w^{(0,1)}) * \hat{\boldsymbol{\mu}}(w^{(1,n-1)}))_{k_0 k_{n-1}} \geqslant 0$$

Ou seja, $(\hat{\boldsymbol{\mu}}(w))_{k_0 k_{n-1}} \geqslant 0$.

O que conclui a prova do lema. ■

Lema 5.4.5 (Matriz com posição definida implica combinação de medidas decrescente)**:** *Seja G' um CCG e considere o MWG $G = \mathcal{T}^{ccg}_{mwg}(G')$. Assim, para todo caminho $w \in \mathcal{W}_{G'}$ de comprimento n e para quaisquer $i, j \in \{0, \ldots, N-1\}$, vale que:*

$$\begin{aligned}(\hat{\boldsymbol{\mu}}(w))_{ij} \geqslant 0 \quad \Rightarrow \quad & \exists\, mc = (k_0, k_1, \ldots, k_{n-1}) \in \mathcal{MC}(w) : \\ & k_0 = i \ \ \wedge \ \ k_{n-1} = j \ \ \wedge\ \mathcal{G}te(\mathcal{F}_{G'}, w, mc)\end{aligned}$$

Prova: Sejam G' um CCG e $G = \mathcal{T}^{ccg}_{mwg}(G')$ o MWG obtido de G'. Seja ainda o caminho $w = (cc_0, \ldots, cc_{n-1}) \in \mathcal{W}_{G'}$ e $i, j \in \{0, \ldots, N-1\}$ tais que $(\hat{\boldsymbol{\mu}}(w))_{ij} \geqslant 0$. A prova segue por indução em $\ell(w) = n$. Assim,

B.I. $\ell(w) = 1$: Neste caso, $w = (cc_0)$ e segue da Definição 3.1.19 que $\hat{\boldsymbol{\mu}}(w) = \mathbf{Id}$. E portanto $(\hat{\boldsymbol{\mu}}(w))_{ij} = (\mathbf{Id})_{ij} \geqslant 0$. Logo, da Definição 4.2.7, $i = j$. Assim, tome $mc \in \mathcal{MC}(w)$, ou seja $\ell(mc) = \ell(w)$, dada por $mc = (i)$. Observe que $k_0 = i$ e $k_{n-1} = j = i$. Resta verificar que $\mathcal{G}te(\mathcal{F}_{G'}, w, mc)$, ou seja, que para todo i, com $0 \leqslant i \leqslant n-2$, vale $\mathcal{G}(\mathcal{F}_{G'}, cc_i, k_i, k_{i+1}, \texttt{true})$. O que é válido por vacuidade, pois $n = 1$ e não pode existir i tal que $0 \leqslant i \leqslant -1$.

P.I. $\ell(w) > 1$: Neste caso, $w = (cc_0, cc_1, \ldots, cc_{n-1})$ com $n > 2$. Assim, w pode ser escrito como a composição $w = (cc_0, cc_1) \circ_w (cc_1, \ldots, cc_{n-1})$ e segue do Lema 3.1.20 que $\hat{\boldsymbol{\mu}}(w) = \boldsymbol{\mu}(cc_0, cc_1) * \hat{\boldsymbol{\mu}}(cc_1, \ldots, cc_{n-1})$. Portanto,

$$(\boldsymbol{\mu}(cc_0, cc_1) * \hat{\boldsymbol{\mu}}(cc_1, \ldots, cc_{n-1}))_{ij} \geqslant 0$$

e pelo Lema 4.2.13, existe $l \in \{0, \ldots, N-1\}$ tal que:

$$(\boldsymbol{\mu}(cc_0, cc_1))_{il} + (\hat{\boldsymbol{\mu}}(cc_1, \ldots, cc_{n-1}))_{lj} \geqslant 0$$

Logo, denotando $(cc_0, cc_1) = w'$ e $(cc_1, \ldots, cc_{n-1}) = w''$, tem-se que $(\boldsymbol{\mu}(w'))_{il} \geqslant 0$ e $(\hat{\boldsymbol{\mu}}(w''))_{lj} \geqslant 0$. Assim, note que $\ell(w'') = n-1$ e por hipótese de indução, existe $mc'' = (k''_0, \ldots, k''_{n-2}) \in \mathcal{MC}(w'')$ tal que $k''_0 = l$ e $k''_{n-2} = j$ e $\mathcal{G}te(\mathcal{F}_{G'}, w'', mc'')$. Além disso, observe que $(\boldsymbol{\mu}(w'))_{il} = \mathcal{I}(i, l, cc_0) \geqslant 0$. Daí segue da Definição 5.4.1, que $\mathcal{G}(cc, \mu_i, \mu_l, \texttt{false})$ ou $\mathcal{G}(cc, \mu_i, \mu_l, \texttt{true})$. Em ambos os casos, conclui-se que $\mathcal{G}te(\mathcal{F}_{G'}, w', mc')$ (Definições 5.3.2 e 5.3.3), onde $mc' = (k'_0, k'_1) = (i, l)$. Portanto,

$$k'_1 = k''_0 \;\; \wedge \;\; \mathcal{G}te(\mathcal{F}_{G'}, w', mc') \;\; \wedge \;\; \mathcal{G}te(\mathcal{F}_{G'}, w'', mc'')$$

$$\Downarrow \text{ (Lema 5.3.6)}$$

$$\mathcal{G}te(\mathcal{F}_{G'}, \circ_w(w', w''), \circ_w(mc', mc''))$$

Logo, conclui-se que existe $mc = (k_0, \ldots, k_{n-1}) = \circ_w(mc', mc'') \in \mathcal{MC}(w)$ tal que $k_0 = i$, $k_{n-1} = j$ e $\mathcal{G}te(\mathcal{F}_{G'}, w, mc)$. ■

5.4.2 Matrizes de Medida com Posição Positiva e Combinação de Medidas Estritamente Decrescente

Lema 5.4.6 (Combinação de medidas estritamente decrescente implica matriz com posição positiva)**:** *Seja G' um CCG. Seja ainda o MWG $G = \mathcal{T}^{ccg}_{mwg}(G')$. Assim, para todo caminho $w \in \mathcal{W}_{G'}$ e para toda combinação de medidas $mc = (k_0, \ldots, k_{n-1}) \in \mathcal{MC}(w)$, vale:*

$$\mathcal{G}t(\mathcal{F}_{G'}, w, mc) \Rightarrow (\hat{\boldsymbol{\mu}}(w))_{k_0 k_{n-1}} = 1$$

Prova: Denote $w = (cc_0, \ldots, cc_{n-1})$. Expandindo a definição da aplicação booleana $\mathcal{G}t$, tem-se que:

$$\mathcal{G}te(\mathcal{F}_{G'}, w, mc) \tag{5.5}$$

e

$$\exists\, i \in \{0, \ldots, n-2\} : \; \mathcal{G}(\mathcal{F}, cc_i, k_i, k_{i+1}, \texttt{false}) \tag{5.6}$$

Seja $i \in \{0, \ldots, n-2\}$ dado pela Equação 5.6. Assim, $w = w' \circ_w (cc_i, cc_{i+1}) \circ_w w''$, onde $w' = (cc_0, \ldots, cc_i)$ e $w'' = (cc_{i+1}, \ldots, cc_{n-1})$ e pelo Lema 3.1.20 segue que:

$$\hat{\boldsymbol{\mu}}(w) = \hat{\boldsymbol{\mu}}(w') * \boldsymbol{\mu}(cc_i, cc_{i+1}) * \hat{\boldsymbol{\mu}}(w'') \tag{5.7}$$

De $G = \mathcal{T}^{ccg}_{mwg}(G')$ tem-se que $(\boldsymbol{\mu}(cc_i, cc_{i+1}))_{k_i k_{i+1}} = \mathcal{I}(k_i, k_{i+1}, cc_0)$, pela Definição 5.4.2. Assim, da Definição 5.4.1 e de (5.6) segue que:

$$(\boldsymbol{\mu}(cc_i, cc_{i+1}))_{k_i k_{i+1}} = 1 \tag{5.8}$$

Note ainda que, de (5.5) e do Lema 5.3.5, segue que $\mathcal{G}te(\mathcal{F}_{G'}, w', mc')$ e $\mathcal{G}te(\mathcal{F}_{G'}, w'', mc'')$, onde $mc' = (k_0, \ldots, k_i)$ e $mc'' = (k_{i+1}, \ldots, k_{n-1})$. Assim, segue do Lema 5.4.4 que:

$$(\hat{\boldsymbol{\mu}}(w'))_{k_0 k_i} \geqslant 0 \quad \wedge \quad (\hat{\boldsymbol{\mu}}(w''))_{k_{i+1} k_{n-1}} \geqslant 0 \tag{5.9}$$

Além disso, de (5.8) e de (5.9), segue que:

$$(\hat{\boldsymbol{\mu}}(w'))_{k_0 k_i} + (\boldsymbol{\mu}(cc_i, cc_{i+1}))_{k_i k_{i+1}} + (\hat{\boldsymbol{\mu}}(w''))_{k_{i+1} k_{n-1}} = 1 \tag{5.10}$$

Assim, de (5.10) e do Lema 4.2.12, segue que

$$(\hat{\boldsymbol{\mu}}(w') * \boldsymbol{\mu}(cc_i, cc_{i+1}) * \hat{\boldsymbol{\mu}}(w''))_{k_0 k_{n-1}} = 1 \tag{5.11}$$

Portanto, de (5.7) e de (5.11), segue que $(\hat{\boldsymbol{\mu}}(w))_{k_0 k_{n-1}} = 1$. ∎

Lema 5.4.7 (Matriz com posição positiva implica combinação de medidas estritamente decrescente)**:** *Sejam G' um CCG e considere o MWG $G = \mathcal{T}^{ccg}_{mwg}(G')$. Assim, para todo $w \in \mathcal{W}_{G'}$ de comprimento n e quaisquer $i, j \in \{0, \ldots, N-1\}$, vale que:*

$$\begin{aligned}(\hat{\boldsymbol{\mu}}(w))_{ij} = 1 \quad \Rightarrow \quad & \exists\, mc = (k_0, \ldots, k_{n-1}) \in \mathcal{MC}(w) : \\ & k_0 = i \ \wedge \ k_{n-1} = j \ \wedge \ \mathcal{G}t(\mathcal{F}_{G'}, w, mc)\end{aligned}$$

Prova: A prova segue por indução em $\ell(w) = n$.

Note que, se $\ell(w) = 1$, o lema não se aplica, pois neste caso $\hat{\boldsymbol{\mu}}(w) = \mathbf{Id}$ (Definição 3.1.19) e quaisquer que sejam $i, j \in \{0, \ldots, N-1\}$, $(\mathbf{Id})_{ij} \in \{-1, 0\}$ (Definição 4.2.7). Logo, neste caso, não podem existir $i, j \in \{0, \ldots, N-1\}$ tais que $(\hat{\boldsymbol{\mu}}(w))_{ij} = 1$.

Assim, seguindo com a indução para $\ell(w) \geqslant 2$, tem-se que:

B.I. $\ell(w) = 2$: Neste caso, $w = (cc_0, cc_1) \in E_{G'} = E_G$. Portanto, $\hat{\boldsymbol{\mu}}(w) = \boldsymbol{\mu}(cc_0, cc_1)$. Sejam $i, j \in \{0, \ldots, N-1\}$ tais que $(\boldsymbol{\mu}(cc_0, cc_1))_{ij} = 1$. Observe que também $(\boldsymbol{\mu}(cc_0, cc_1))_{ij} \geqslant 0$. Logo, pelo Lema 5.4.5, existe uma combinação de medidas $mc = (k_0, k_1)$ em $\mathcal{MC}(w)$, tal que $k_0 = i$, $k_1 = j$ e $\mathcal{G}te(\mathcal{F}_{G'}, w, mc)$. Assim, tome esta combinação de medidas, resta verificar que $\mathcal{G}t(\mathcal{F}_{G'}, w, mc)$. De fato, pela definição de $\mathcal{G}t$, tem-se que: $\mathcal{G}t(\mathcal{F}_{G'}, w, mc) = \mathcal{G}te(\mathcal{F}_{G'}, w, mc) \ \wedge \ \mathcal{G}(\mathcal{F}_{G'}, cc_0, k_0, k_1, \texttt{false})$. Logo, resta verificar que $\mathcal{G}(\mathcal{F}_{G'}, cc_0, k_0, k_1, \texttt{false})$. De $G = \mathcal{T}^{ccg}_{mwg}(G')$ tem-se que $(\boldsymbol{\mu}(cc_0, cc_1))_{ij} = \mathcal{I}(i, j, cc_0)$, pela Definição 5.4.2. Ou seja, $\mathcal{I}(i, j, cc_0) = 1$. Assim, segue da definição 5.4.1, que $\mathcal{G}(cc, \mu_i, \mu_j, \texttt{false})$.

P.I. $\ell(w) > 2$: Neste caso, $w = (cc_0, cc_1, \ldots, cc_{n-1})$, com $n > 2$. Assim, pelo Lema 3.1.20 e de $w = (cc_0, cc_1) \circ_w (cc_1, \ldots, cc_{n-1})$, a matriz de w pode ser decomposta da seguinte forma: $\hat{\boldsymbol{\mu}}(w) = \boldsymbol{\mu}(cc_0, cc_1) * \hat{\boldsymbol{\mu}}(cc_1, \ldots, cc_{n-1})$. Sejam $i, j \in \{0, \ldots, N-1\}$ tais que $(\hat{\boldsymbol{\mu}}(w))_{ij} = 1$. Assim, $(\boldsymbol{\mu}(cc_0, cc_1) * \hat{\boldsymbol{\mu}}(cc_1, \ldots, cc_{n-1}))_{ij} = 1$ e pelo Lema 4.2.12, existe um $l \in \{0, \ldots, N-1\}$ tal que:

$$(\boldsymbol{\mu}(cc_0, cc_1))_{il} + (\hat{\boldsymbol{\mu}}(cc_1, \ldots, cc_{n-1}))_{lj} = 1$$

Assim, denotando $(cc_0, cc_1) = w'$ e $(cc_1, \ldots, cc_{n-1}) = w''$, considere dois casos:

i) $(\boldsymbol{\mu}(w'))_{il} = 1$ e $(\hat{\boldsymbol{\mu}}(w''))_{lj} \geqslant 0$

Por hipótese de indução, existe $mc' = (k'_0, k'_1) \in \mathcal{MC}(w')$ tal que $k'_0 = i$, $k'_1 = l$ e $\mathcal{G}t(\mathcal{F}_{G'}, w', mc')$. Além disso, pelo Lema 5.4.5, existe uma combinação de medidas $mc'' = (k''_0, \ldots, k''_{n-2}) \in \mathcal{MC}(w'')$ tal que $k''_0 = l$ e $k''_{n-2} = j$ e $\mathcal{G}te(\mathcal{F}_{G'}, w'', mc'')$. Logo,

$$k'_1 = k''_0 \ \wedge \ \mathcal{G}t(\mathcal{F}_{G'}, w', mc') \ \wedge \ \mathcal{G}te(\mathcal{F}_{G'}, w'', mc'')$$

$$\Downarrow \text{(Lema 5.3.7)}$$

$$\mathcal{G}t(\mathcal{F}_{G'}, \circ_w(w', w''), \circ_w(mc', mc''))$$

Ou seja, existe $mc = (k_0, \ldots, k_{n-1}) = \circ_w(mc', mc'') \in \mathcal{MC}(w)$ tal que $k_0 = i$, $k_{n-1} = j$ e $\mathcal{G}t(\mathcal{F}_{G'}, w, mc)$.

ii) $(\boldsymbol{\mu}(w'))_{il} \geqslant 0$ e $(\hat{\boldsymbol{\mu}}(w''))_{lj} = 1$

Por hipótese de indução, existe $mc'' = (k''_0, \ldots, k''_{n-2}) \in \mathcal{MC}(w'')$ tal que $k''_0 = l$, $k''_{n-2} = j$ e $\mathcal{G}t(\mathcal{F}_{G'}, w'', mc'')$. Além disso, pelo Lema 5.4.5, existe uma combinação de medidas $mc' = (k'_0, k'_1) \in \mathcal{MC}(w')$ tal que $k'_0 = i$, $k'_1 = l$ e $\mathcal{G}te(\mathcal{F}_{G'}, w', mc')$. Logo,

$$k'_1 = k''_0 \ \wedge \ \mathcal{G}te(\mathcal{F}_{G'}, w', mc') \ \wedge \ \mathcal{G}t(\mathcal{F}_{G'}, w'', mc'')$$

$$\Downarrow \text{(Lema 5.3.7)}$$

$$\mathcal{G}t(\mathcal{F}_{G'}, \circ_w(w', w''), \circ_w(mc', mc''))$$

Ou seja, existe $mc = (k_0, \ldots, k_{n-1}) = \circ_w(mc', mc'') \in \mathcal{MC}(w)$ tal que $k_0 = i$, $k_{n-1} = j$ e $\mathcal{G}t(\mathcal{F}_{G'}, w, mc)$.

Portanto, segue que, para todos $w \in \mathcal{W}_{G'}$ e $i, j \in \{0, \ldots, N-1\}$, se $(\hat{\boldsymbol{\mu}}(w))_{ij} = 1$ então existe uma combinação de medidas $mc = (k_0, \ldots, k_{n-1}) \in \mathcal{MC}(w)$ tal que $k_0 = i$, $k_{n-1} = j$ e $\mathcal{G}t(\mathcal{F}_{G'}, w, mc)$. ■

5.4.3 Prova da Equivalência entre Terminação em CCG's e MWG's

Teorema 5.4.8 (Equivalência entre matriz de medida positiva e combinação de medidas estritamente decrescente)**:** *Dados um CCG G' e um caminho $w \in \mathcal{W}_{G'}$ com $\ell(w) = n$, considere o MWG $G = \mathcal{T}^{ccg}_{mwg}(G')$. Assim, a matriz do caminho w, denotada por $\hat{\boldsymbol{\mu}}(w)$, é positiva se, e somente se, existe uma combinação de medidas $mc = (k_0, k_1, \ldots, k_{n-1}) \in \mathcal{MC}(w)$, tal que $k_0 = k_{n-1}$ e $\mathcal{G}t(\mathcal{F}_{G'}, w, mc)$.*

Prova: Sejam G' um CCG e G o MWG $\mathcal{T}^{ccg}_{mwg}(G')$. Seja ainda $w \in \mathcal{W}_{G'}$, com $\ell(w) = n$. Note que também $w \in \mathcal{W}_G$. Assim,

(⇒) Suponha que $\hat{\boldsymbol{\mu}}(w)$ é positiva, ou seja, existe $j \in \{0, \ldots, N-1\}$ tal que $(\hat{\boldsymbol{\mu}}(w))_{jj} = 1$. Logo, deve existir uma combinação de medidas $mc = (k_0, k_1, \ldots, k_{n-1}) \in \mathcal{MC}(w)$, tal que $k_0 = k_{n-1}$ e $\mathcal{G}t(\mathcal{F}_{G'}, w, mc)$. Portanto, tomando $i = j$ no Lema 5.4.7, segue este sentido da prova.

(⇐) Suponha que existe uma combinação de medidas $mc = (k_0, k_1, \ldots, k_{n-1}) \in \mathcal{MC}(w)$, tal que $k_0 = k_{n-1}$ e $\mathcal{G}t(\mathcal{F}_{G'}, w, mc)$. Assim, deve existir $j \in \{0, \ldots, N-1\}$ tal que $(\hat{\boldsymbol{\mu}}(w))_{jj} = 1$. De fato, pelo Lema 5.4.6, $(\hat{\boldsymbol{\mu}}(w))_{k_0 k_{n-1}} = 1$. Assim, como $k_0 = k_{n-1}$, basta tomar $j = k_0$, e segue este sentido da prova.

O que conclui a prova do teorema. ■

Teorema 5.4.9 (Equivalência entre terminação para CCG e terminação para MWG)**:** *Seja um CCG G' e considere o MWG G obtido de G' por $\mathcal{T}^{ccg}_{mwg}(G')$. Assim,*

$$T_{ccg}(G') \iff T_{mwg}(G)$$

Prova: De fato, deve-se verificar que:

$$\forall\, c \in \mathcal{C}ir(G'),\ \text{com } \ell(c) = n,\ \exists\, mc = (k_0, k_1, \ldots, k_{n-1}),\ \text{com } k_0 = k_{n-1},$$

$$\text{tal que } \mathcal{G}t(\mathcal{F}_{G'}, c, mc)$$

$$\Updownarrow$$

$$\forall\, c \in \mathcal{C}ir(G),\ \exists\, j \in \{0, \ldots, N-1\} : (\hat{\boldsymbol{\mu}}(c))_{jj} = 1$$

Assim, aplicando o Teorema 5.4.8, e observando que $\mathcal{C}ir(G) = \mathcal{C}ir(G') \subset \mathcal{W}_{G'} = \mathcal{W}_G$, conclui-se a prova. ■

Capítulo 6

Relacionando a Semântica Operacional da Terminação para Linguagens Funcionais de Primeira Ordem e a Terminação em MWGs (e CCGs)

Neste capítulo apresenta-se a especificação da semântica da terminação para a linguagem funcional de primeira ordem, cuja sintaxe foi apresentada na Seção 5.1. As definições e resultados aqui apresentados encontram-se especificados e formalizados em PVS, na *teoria* `PVS0`, exceto pelos resultados apresentados a partir do Lema 6.1.11 e que culminam no Teorema 6.2.10, onde é apresentada uma prova analítica de terminação para MWG's implica em terminação da linguagem funcional de primeira ordem. A formalização em PVS destes resultados entra para a lista de trabalhos futuros, contudo apresenta-se aqui uma prova analítica do mesmos, que constitui um roteiro para a formalização.

6.1 Semântica Operacional da Terminação para Linguagens Funcionais de Primeira Ordem

A semântica operacional da terminação para uma linguagem de primeira-ordem, cuja sintaxe é apresentada na Seção 5.1, é definida com base em uma pré-interpretação para os símbolos de operadores e símbolos de constante de $\mathcal{S}$, denotada por

$\mathfrak{I} : \mathcal{S} \to \mathcal{V}al^k \to \mathcal{V}al$, onde k é um natural; além de uma designação para as variáveis sobre o conjunto de valores, denotada por $\beta : \mathcal{V}ar \to \mathcal{V}al$. Assim, em uma definição de função n-ária $f(x_1, \ldots, x_n) = e_f$, especificada de acordo com a sintaxe apresentada na Definição 5.1.1, cada constante $c \in \mathcal{S}$ é interpretada segundo $\mathfrak{I}$, e a sua interpretação $c^{\mathfrak{I}}$ é um elemento de $\mathcal{V}al$; da mesma forma, cada operador k-ário $g \in \mathcal{S}$ é interpretado segundo $\mathfrak{I}$, e a sua interpretação $g^{\mathfrak{I}}$ é um operador k-ário sobre $\mathcal{V}al$ completamente computável. A definição seguinte estende a pré-interpretação $\mathfrak{I}$ para os símbolos de função $f \in \mathcal{D}ef$.

Definição 6.1.1 (Relação semântica de uma expressão)**:** *Dada uma definição de função n-ária , $f(x_1, \ldots, x_n) = e_f \in \mathcal{E}xp$. A expressão e é interpretada em $\mathcal{V}al$ segundo a relação indutiva $\varepsilon : \mathcal{E}xp \times \mathcal{E}xp \times \beta \times \mathcal{V}al \to$ `bool` definida por:*

$$
\begin{array}{rcl}
\varepsilon(e, e_f, \beta, \nu) & := & \texttt{CASES } e \texttt{ OF} \\
& & \qquad\qquad c: \;\; \nu = c^{\mathfrak{I}}; \\
& & \qquad\qquad x: \;\; \nu = \beta(x); \\
& & g(e_1, \ldots, e_k): \;\; \exists \nu_1, \ldots, \nu_k \in \mathcal{V}al : \varepsilon(e_1, e_f, \beta, \nu_1) \wedge \ldots \wedge \varepsilon(e_k, e_f, \beta, \nu_k) \wedge \\
& & \qquad\qquad \nu = g^{\mathfrak{I}}(\nu_1, \ldots, \nu_k) \\
& & ite(e_1, e_2, e_3): \;\; \exists \nu_1 : \varepsilon(e_1, e_f, \beta, \nu_1) \wedge \\
& & \qquad\qquad \texttt{IF } \nu_1 \texttt{ THEN } \varepsilon(e_2, e_f, \beta, \nu) \texttt{ ELSE } \varepsilon(e_3, e_f, \beta, \nu) \\
& & f(e_1, \ldots, e_n): \;\; \exists \nu_1, \ldots, \nu_n \in \mathcal{V}al : \varepsilon(e_1, e_f, \beta, \nu_1) \wedge \ldots \wedge \varepsilon(e_n, e_f, \beta, \nu_n) \wedge \\
& & \qquad\qquad \varepsilon(e_f, e_f, \beta', \nu), \;\; \textit{onde } \beta'(x_i) = \nu_i, \;\; \textit{para } i = 1, \ldots, n
\end{array}
$$

Portanto, dada uma designação β e uma pré-interpretação $\mathfrak{I}$, $\varepsilon(e_f, e_f, \beta, \nu)$, quando verdadeira para algum valor $\nu \in \mathcal{V}al$, fornece uma interpretação sobre $\mathcal{V}al$ para o símbolo de função n-ário $f \in \mathcal{D}ef$ cuja definição é dada pela expressão e_f.

Desta forma, a semântica da terminação é definida com base na relação semântica ε como segue:

Definição 6.1.2 (Semântica da terminação segundo ε)**:** *Estabelece-se que uma definição de função n-ária , $f(x_1, \ldots, x_n) = e_f \in \mathcal{E}xp$, é terminante segundo a aplicação booleana $T_\varepsilon : \mathcal{E}xp \to$ `bool`, definida por:*

$$T_\varepsilon(e_f) := \forall\, \beta,\ \exists\, \nu \in \mathcal{V}al :\ \varepsilon(e_f, e_f, \beta, \nu)$$

No que segue, define-se uma aplicação, onde uma definição de função n-ária é avaliada dado um número natural. A existência deste natural implica que, dada uma n-upla inicial de valores, não pode ocorrer uma sequência infinita de chamados recursivos durante a execução de uma função n-ária, já que o natural decresce a cada chamado recursivo.

Definição 6.1.3 (Avaliação semântica de uma expressão)**:** *Dada uma definição de função n-ária* $f(x_1, \ldots, x_n) = e_f$ *a aplicação* $\chi : \mathbb{N} \times \mathcal{E}xp \times \mathcal{E}xp \times \beta \to \mathcal{V}al \cup \diamondsuit$*, onde* $\diamondsuit \notin \mathcal{V}al$*, definida por:*

$$
\begin{array}{rcl}
\chi(m, e, e_f, \beta) & := & \texttt{IF } m = 0 \texttt{ THEN } \diamondsuit \\
& & \texttt{ELSE CASES } e \texttt{ OF} \\
& & \qquad\qquad c: \; c^{\mathfrak{I}}; \\
& & \qquad\qquad x: \; \beta(x); \\
& & \qquad g(e_1, \ldots, e_k): \; \texttt{IF } \forall i \in \{1, \ldots, k\} : \chi(m, e_i, e_f, \beta) \neq \diamondsuit \\
& & \qquad\qquad\qquad \texttt{THEN } g^{\mathfrak{I}}(\chi(m, e_1, e_f, \beta), \ldots, \chi(m, e_k, e_f, \beta)) \\
& & \qquad\qquad\qquad \texttt{ELSE } \diamondsuit; \\
& & \qquad ite(e_1, e_2, e_3): \; \texttt{IF } \chi(m, e_1, e_f, \beta) \neq \diamondsuit \texttt{ THEN} \\
& & \qquad\qquad\qquad \texttt{IF } \chi(m, e_1, e_f, \beta) \texttt{ THEN } \chi(m, e_2, e_f, \beta) \\
& & \qquad\qquad\qquad \texttt{ELSE } \chi(m, e_3, e_f, \beta); \\
& & \qquad\qquad\qquad \texttt{ELSE } \diamondsuit; \\
& & \qquad f(e_1, \ldots, e_n): \; \texttt{IF } \forall i \in \{1, \ldots, n\} : \chi(m, e_i, e_f, \beta) \neq \diamondsuit \texttt{ THEN} \\
& & \qquad\qquad\qquad \chi(m-1, e_f, e_f, \beta'), \\
& & \qquad\qquad\qquad \textit{onde } \beta'(x_i) = \chi(m, e_i, e_f, \beta), \textit{ para } i = 1, \ldots, n \\
& & \qquad\qquad\qquad \texttt{ELSE } \diamondsuit
\end{array}
$$

estabelece uma avaliação sobre o conjunto $\mathcal{V}al$ *da expressão e dado um número natural* m *que funciona como um limite para os níveis de recursão na computação da função* f *em* e*.*

O seguinte lema estabelece que quando a avaliação semântica de uma expressão retorna um valor sobre $\mathcal{V}al$, este valor é sempre o mesmo independente do parâmetro m, a partir de um limite mínimo de níveis de recursão.

Lema 6.1.4: *Sejam uma expressão* e*, uma definição de função* $f(x_1, \ldots, x_n) = e_f$ *e uma designação* $\beta : \mathcal{V}ar \to \mathcal{V}al$*. Sejam ainda,* $n \in \mathbb{N}$ *e* $m \geqslant n$*. Assim,*

$$\chi(n, e, e_f, \beta) \in \mathcal{V}al \;\wedge\; \chi(n, e, e_f, \beta) = \nu \;\;\Rightarrow\;\; \chi(m, e, e_f, \beta) = \nu$$

Prova: A prova segue diretamente da definição 6.1.3. ∎

A seguinte definição estabelece a semântica da terminação de uma expressão a partir da existência de um natural m, qualquer que seja a designação β de valores para as variáveis. Basicamente estipula-se que uma definição de função se relaciona com um valor ν sobre $\mathcal{V}al$, segundo a relação semântica ε, caso exista um natural m suficientemente grande tal que a avaliação semântica χ da definição de função é o valor ν. Note que a

relação semântica de uma expressão é aplicada na seguinte definição, contudo dado que se a semântica da avaliação fornece um valor para uma dada expressão, então este valor deve satisfazer a relação semântica para a expressão em questão.

Definição 6.1.5 (Semântica da terminação segundo χ)**:** *Dada uma definição de função n-ária* $f(x_1, \ldots, x_n) = e_f \in \mathcal{E}xp$*, a aplicação booleana* $T_\chi : \mathcal{E}xp \to \texttt{bool}$ *definida por*

$$T_\chi(e_f) := \forall\, \beta,\ \exists\, m \in \mathbb{N} :\ \chi(m, e_f, e_f, \beta) = \nu \in \mathcal{V}al\ \wedge\ \varepsilon(e_f, e_f, \beta, \nu),$$

determina se f *termina, em um número finito de níveis de recursão, para quaisquer valores/argumentos de entrada.*

Nos lemas seguintes, estabelece-se uma relação entre a interpretação de uma expressão e sobre $\mathcal{V}al$ dada pela relação semântica ε, e a interpretação da mesma expressão dada pela avaliação semântica χ.

Lema 6.1.6 (Semântica da avaliação por ε implica em número finito de níveis de recursão)**:** *Sejam uma definição de função* $f(x_1, \ldots, x_n) = e_f$*, uma expressão* e*, uma designação de variáveis* $\beta : \mathcal{V}ar \to \mathcal{V}al$ *e um valor* $\nu \in \mathcal{V}al$ *quaisquer. Assim, vale que:*

$$\varepsilon(e, e_f, \beta, \nu)\ \Rightarrow\ \exists\, m \in \mathbb{N} :\ \chi(m, e, e_f, \beta) \in \mathcal{V}al\ \wedge\ \chi(m, e, e_f, \beta) = \nu$$

Prova: A prova segue por indução na estrutura de e. Assim, $\varepsilon(e, e_f, \beta, \nu)$ e considere os casos para e:

$e = c$: Neste caso, segue da Definição 6.1.1 que $\nu = c^{\mathfrak{I}} \in \mathcal{V}al$ e para qualquer natural $m \geqslant 1$, segue da Definição 6.1.3 que $\chi(m, e, e_f, \beta) = c^{\mathfrak{I}}$.

$e = x$: Analogamente ao anterior, segue da Definição 6.1.1 que $\nu = \beta(x) \in \mathcal{V}al$ e para qualquer natural $m \geqslant 1$, segue da Definição 6.1.3 que $\chi(m, e, e_f, \beta) = \beta(x)$.

$e = g(e_1, \ldots, e_k)$: Neste caso, segue da Definição 6.1.1 que existem $\nu_1, \ldots, \nu_k \in \mathcal{V}al$ tais que $\varepsilon(e_1, e_f, \beta, \nu_1), \ldots, \varepsilon(e_k, e_f, \beta, \nu_k)$ e $\nu = g^{\mathfrak{I}}(\nu_1, \ldots, \nu_k)$. Logo, por hipótese de indução, segue que, para todo $i \in \{1, \ldots, k\}$,

$$\exists\, m_i \in \mathbb{N} :\ \chi(m_i, e_i, e_f, \beta) \in \mathcal{V}al\ \wedge\ \chi(m_i, e_i, e_f, \beta) = \nu_i$$

Seja $m = \max\{m_i \mid 1 \leqslant i \leqslant k\}$. Assim, segue do Lema 6.1.4, que para todo $i \in \{1, \ldots, k\}$ vale $\chi(m, e_i, e_f, \beta) = \nu_i$. Logo,

$$\chi(m, e, e_f, \beta) = g^{\mathfrak{I}}(\chi(m, e_1, e_f, \beta), \ldots, \chi(m, e_k, e_f, \beta)) = g^{\mathfrak{I}}(\nu_1, \ldots, \nu_k) = \nu$$

$e = ite(e_1, e_2, e_3)$: Neste caso, segue da Definição 6.1.1 que existe $\nu_1 \in \mathcal{V}al$ tal que $\varepsilon(e_1, e_f, \beta, \nu_1)$ e por hipótese de indução, segue que existe $m_1 \in \mathbb{N}$ tal que $\chi(m_1, e_1, e_f, \beta) = \nu_1$. Assim, duas possibilidades devem ser consideradas:

$\nu_1 = \texttt{true}$: Neste caso, tem-se $\varepsilon(e_2, e_f, \beta, \nu)$ e por hipótese de indução, segue que:

$$\exists\, m_2 \in \mathbb{N} :\ \chi(m_2, e_2, e_f, \beta) \in \mathcal{V}al \ \wedge\ \chi(m_2, e_2, e_f, \beta) = \nu$$

Portanto, tome o máximo entre m_1 e m_2. Seja m este máximo. Assim, do Lema 6.1.4, $\chi(m, e_1, e_f, \beta) = \texttt{true}$ e $\chi(m, e, e_f, \beta) = \chi(m, e_2, e_f, \beta) = \nu_2$.

$\nu_1 = \texttt{false}$: De maneira análoga, tem-se $\varepsilon(e_3, e_f, \beta, \nu)$ e por hipótese de indução, segue que:

$$\exists\, m_3 \in \mathbb{N} :\ \chi(m_3, e_3, e_f, \beta) \in \mathcal{V}al \ \wedge\ \chi(m_3, e_3, e_f, \beta) = \nu$$

Portanto, seja m o máximo entre m_1 e m_3. Assim, do Lema 6.1.4, segue que $\chi(m, e_1, e_f, \beta) = \texttt{false}$ e $\chi(m, e, e_f, \beta) = \chi(m, e_3, e_f, \beta) = \nu$.

$e_f = f(e_1, \ldots, e_n)$: Neste caso, segue da Definição 6.1.1, que existem valores ν_i tais que $\varepsilon(e_i, e_f, \beta, \nu_i)$, para $1 \leqslant i \leqslant n$, e $\varepsilon(e_f, e_f, \beta', \nu)$, onde $\beta' : \mathcal{V}ar \to \mathcal{V}al$ é definida por $\beta'(x_i) = \nu_i$. Por hipótese de indução, segue que:

$$\forall\, i \in \{1, \ldots, n\}, \exists\, m_i \in \mathbb{N} :\ \chi(m_i, e_i, e_f, \beta) \in \mathcal{V}al \ \wedge\ \chi(m_i, e_i, e_f, \beta) = \nu_i$$

e

$$\exists\, k \in \mathbb{N} :\ \chi(k, e_f, e_f, \beta') \in \mathcal{V}al \ \wedge\ \chi(k, e_f, e_f, \beta') = \nu$$

Assim, seja $\bar{m}$ o máximo dentre os m_i's e k. Desta forma, do Lema 6.1.4, segue que:

$$\forall\, i \in \{1, \ldots, n\} :\ \chi(\bar{m}+1, e_i, e_f, \beta) \in \mathcal{V}al \ \wedge\ \chi(\bar{m}+1, e_i, e_f, \beta) = \nu_i \qquad \text{(A)}$$

e

$$\chi(\bar{m}, e_f, e_f, \beta') \in \mathcal{V}al \ \wedge\ \chi(\bar{m}, e_f, e_f, \beta') = \nu$$

Além disso, da Definição 6.1.3 e de (A), segue que $\chi(\bar{m}+1, e, e_f, \beta) = \chi(\bar{m}, e_f, e_f, \beta')$. Portanto, tomando $m = \bar{m} + 1$, tem-se que:

$$\exists\, m \in \mathbb{N} :\ \chi(m, e, e_f, \beta) \in \mathcal{V}al \ \wedge\ \chi(m, e, e_f, \beta) = \nu$$

O que conclui a prova. ■

O seguinte lema estabelece que a relação semântica de uma expressão é determinística.

Lema 6.1.7: *Sejam uma expressão e, uma definição de função $f(x_1, \ldots, x_n) = e_f$ e uma designação $\beta : \mathcal{V}ar \to \mathcal{V}al$. Sejam ainda $\nu_1, \nu_2 \in \mathcal{V}al$. Assim,*

$$\varepsilon(e, e_f, \beta, \nu_1) \ \wedge \ \varepsilon(e, e_f, \beta, \nu_2) \ \Rightarrow \ \nu_1 = \nu_2$$

Prova: Sejam $\nu_1, \nu_2 \in \mathcal{V}al$ tais que $\varepsilon(e, e_f, \beta, \nu_1)$ e $\varepsilon(e, e_f, \beta, \nu_2)$. Segue do Lema 6.1.6 que:

$$\exists\, m_1 \in \mathbb{N} : \ \chi(m_1, e, e_f, \beta) \in \mathcal{V}al \ \wedge \ \chi(m_1, e, e_f, \beta) = \nu_1$$

e

$$\exists\, m_2 \in \mathbb{N} : \ \chi(m_2, e, e_f, \beta) \in \mathcal{V}al \ \wedge \ \chi(m_2, e, e_f, \beta) = \nu_2.$$

Seja m o máximo entre m_1 e m_2. Assim, segue do Lema 6.1.4 que $\chi(m, e, e_f, \beta) = \nu_1$ e $\chi(m, e, e_f, \beta) = \nu_2$. Logo, $\nu_1 = \nu_2$. ■

Lema 6.1.8 (Equivalência entre T_ε e T_χ)**:** *Seja um símbolo de função n-ário $f \in \mathcal{D}ef$ com definição dada pela expressão e_f. Assim,*

$$T_\varepsilon(e_f) \iff T_\chi(e_f)$$

Prova: De fato, deve-se verificar que:

$$\forall\, \beta, \ \exists\, \nu \in \mathcal{V}al : \ \varepsilon(e_f, e_f, \beta, \nu)$$

$$\Updownarrow$$

$$\forall\, \beta, \ \exists\, m \in \mathbb{N} : \ \chi(m, e_f, e_f, \beta) = \nu \in \mathcal{V}al \ \wedge \ \varepsilon(e_f, e_f, \beta, \nu)$$

(⇒) Suponha que, para toda designação β, existe um valor ν tal que $\varepsilon(e_f, e_f, \beta, \nu)$. Então, pelo Lema 6.1.6 e considerando a mesma designação β, existe $m \in \mathbb{N}$ tal que $\chi(m, e_f, e_f, \beta) \in \mathcal{V}al$ e $\chi(m, e_f, e_f, \beta) = \nu$.

(⇐) Suponha que, para toda designação β, existe um número natural m tal que $\chi(m, e_f, e_f, \beta) = \nu \in \mathcal{V}al$ e $\varepsilon(e_f, e_f, \beta, \nu)$. Assim, para toda designação β, existe $\nu \in \mathcal{V}al$ tal que $\varepsilon(e_f, e_f, \beta, \nu)$.

O que conclui a prova. ■

A seguinte definição estabelece um predicado que relaciona uma expressão e com uma medida definida sobre o domínio de uma relação bem fundada, no sentido de que a cada chamado recursivo a medida aplicada aos parâmetros atuais decresce comparativamente à mesma medida aplicada aos parâmetros formais. Desta forma, captura-se a noção de terminação utilizada no assistente de prova PVS para verificação de terminação de funções

recursivas. Observando que em PVS o domínio de uma medida aplicada para verificação de terminação de uma definição recursiva é o mesmo da definição recursiva, mas a sua imagem é o domínio de uma relação bem fundada, de modo que a seguinte notação é adotada: $\mu : \mathcal{V}al^n \to (\mathcal{MT}, \succcurlyeq)$, isto é, uma medida μ definida sobre o domínio $\mathcal{MT}$ de uma relação bem fundada $\succcurlyeq$.

Definição 6.1.9 (Relação entre uma expressão e uma medida)**:** *Dada uma definição de função n-ária, $f(x_1, \ldots, x_n) = e_f \in \mathcal{E}xp$ e uma conjunção de condições* `cnd`*, uma expressão e é relacionada a uma medida μ definida sobre $(\mathcal{MT}, \succcurlyeq)$, onde $\succcurlyeq$ é uma relação bem fundada, segundo a seguinte relação indutiva $\zeta : \mathcal{E}xp \times \mathcal{E}xp \times \mu \times \mathcal{E}xp \to$* `bool`

$$
\begin{array}{rcl}
\zeta(e, e_f, \mu, \texttt{cnd}) & := & \texttt{CASES } e \texttt{ OF} \\
& & c: \ \texttt{true}; \\
& & x: \ \texttt{true}; \\
& & g(e_1, \ldots, e_k): \ \zeta(e_1, e_f, \mu, \texttt{cnd}) \wedge \ldots \wedge \zeta(e_k, e_f, \mu, \texttt{cnd}); \\
& & ite(e_1, e_2, e_3): \ \zeta(e_1, e_f, \mu, \texttt{cnd}) \wedge \zeta(e_2, e_f, \mu, \texttt{cnd} \wedge e_1) \wedge \\
& & \qquad \zeta(e_3, e_f, \mu, \texttt{cnd} \wedge \neg e_1); \\
& & f(e_1, \ldots, e_n): \ \zeta(e_1, e_f, \mu, \texttt{cnd}) \wedge \ldots \wedge \zeta(e_n, e_f, \mu, \texttt{cnd}) \wedge \\
& & \qquad \forall\, \beta : \texttt{cnd}(\beta(\boldsymbol{x})) \Rightarrow \mu(\beta(\boldsymbol{x})) \succcurlyeq \mu(e_1(\beta(\boldsymbol{x})), \ldots, e_n(\beta(\boldsymbol{x}))),
\end{array}
$$

onde, $\beta(\boldsymbol{x}) = (\beta(x_1), \ldots, \beta(x_n))$.

A seguinte definição estabelece a semântica da terminação para uma expressão a partir da existência de uma medida que decresce a cada chamado recursivo segundo a relação ζ.

Definição 6.1.10 (Semântica da terminação segundo ζ)**:** *Dada uma definição de função n-ária, $f(x_1, \ldots, x_n) = e_f \in \mathcal{E}xp$, define-se a seguinte aplicação booleana $T_\zeta : \mathcal{E}xp \to$* `bool` *por:*

$$T_\zeta(e_f) := \exists\, \mu : \mathcal{V}al^n \to (\mathcal{MT}, \succcurlyeq) : \zeta(e_f, e_f, \mu, \texttt{true})$$

Lema 6.1.11 (Terminação segundo ζ e sequências bem-formadas finitas)**:** *Seja uma definição de função n-ária $f(x_1, \ldots, x_n) = e_f$. Assim, tem-se que $T_\zeta(e_f)$ se, e somente se, toda sequência bem formada (Definição 3.2.2) $s_{cc} = (cc_{i_1}, cc_{i_2}, cc_{i_3}, \ldots)$ de contextos de chamado de f é finita.*

Prova: De fato, lembrando que cada contexto de chamado da sequência s_{cc} é da forma $cc_{i_k} = \langle \boldsymbol{x}, cnd_{i_k}, \boldsymbol{e}_{i_k} \rangle$, onde $\boldsymbol{x} = (x_1, \ldots, x_n)$ e $\boldsymbol{e}_{i_k} = (e_1^{i_k}, e_2^{i_k}, \ldots, e_n^{i_k})$, deve-se verificar que:

$$\exists\, \mu : \mathcal{V}al^n \to (\mathcal{MT}, \succcurlyeq) : \zeta(e_f, e_f, \mu, \texttt{true})$$

$$\Updownarrow$$

$$\exists\, \beta_{i_1} : \mathcal{V}ar \to \mathcal{V}al : \texttt{cnd}_{i_1}(\beta_{i_1}(\boldsymbol{x})) \wedge \forall\, k \geqslant 1 : \texttt{cnd}_{i_{k+1}}(\boldsymbol{e}_{i_k}(\beta_{i_k}(\boldsymbol{x}))),$$

onde $\beta_{i_k}(\boldsymbol{x}) = \boldsymbol{e}_{i_{k-1}}(\beta_{i_{k-1}}(\boldsymbol{x}))$. Assim,

(⇒) Suponha que $T_\zeta(e_f)$ e seja $s_{cc} = (cc_{i_1}, cc_{i_2}, \ldots)$ uma sequência bem-formada infinita de contextos de chamado. Como vale que existe uma medida μ tal que $\zeta(e_f, e_f, \mu, \texttt{true})$, então para cada $k \geqslant 1$ vale que $\mu(\beta_{i_k}(\boldsymbol{x})) \succcurlyeq \mu(\boldsymbol{e}_{ik}(\beta_{i_k}(\boldsymbol{x})))$. Mas, como $\succcurlyeq$ é uma relação bem-fundada, esta sequência não pode ser infinita, o que é uma contradição. Logo, s_{cc} é finita.

(⇐) Suponha que toda sequência bem-formada é finita. Como a definição recursiva e_f é finita, então o número de chamados recursivos que podem ocorrer após um chamado qualquer é finito. Considere a árvore com raiz $f(x_1, \ldots, x_n) = e_f$ cujos filhos são apenas aqueles chamados que geram uma sequência bem formada de contextos de chamado, e assim por diante. Pelo lema de König a árvore originada, como é finitamente ramificável, é finita, uma vez que só poderia ser infinita se, e somente se, possuísse um caminho infinito, ou seja, uma sequência bem-formada infinita. Seja h a altura desta árvore e seja $f(e_1, \ldots, e_n)$ um chamado recursivo em um nível $k \leqslant h$ da árvore. Assim, considere a medida $\mu : \mathcal{V}al^n \to (\mathcal{MT}, \succcurlyeq)$ definida por: $\mu(e_1(\beta(\boldsymbol{x}), \ldots, \beta(\boldsymbol{x}))) = h - k$, para qualquer designação β. Note que, $\mu(\beta(x_1), \ldots, \beta(x_n)) = h$. Assim, em cada nível da árvore, o resultado da medida aplicada aos formais do chamado recursivo naquele nível é menor que h. Logo, para esta medida, vale que $\zeta(e_f, e_f, \mu, \texttt{true})$. Ou seja, a cada chamado recursivo sob condições `cnd` em e_f, vale que

$$\forall\, \beta : \texttt{cnd}(\beta(\boldsymbol{x})) \Rightarrow \mu(\beta(\boldsymbol{x})) \succcurlyeq \mu(e_1(\beta(\boldsymbol{x})), \ldots, e_n(\beta(\boldsymbol{x}))).$$

O que conclui a prova. ∎

Teorema 6.1.12 (Terminação segundo ζ e terminação segundo χ)**:** *Seja uma definição de função n-ária $f(x_1, \ldots, x_n) = e_f$. Assim, tem-se que*

$$T_\zeta(e_f) \iff T_\chi(e_f)$$

Prova: Deve-se verificar que:

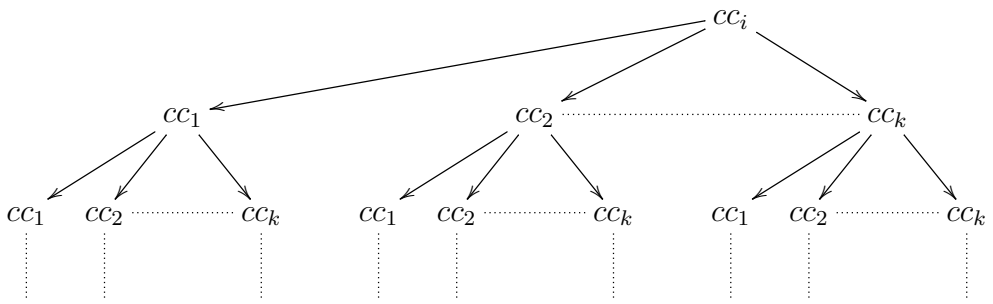


Figura 6.1: Dada uma definição de função n-ária $f(x_1, \ldots, x_n) = e_f$, com os seguintes contextos de chamado $\{cc_1, \ldots, cc_k\}$ correspondendo aos chamados recursivos de f em e_f, a partir de um chamado específico cc_i, somente um número finito de chamados subsequentes pode ser realizado.

$$\exists\, \mu : \mathcal{V}al^n \to (\mathcal{MT}, \succcurlyeq) : \zeta(e_f, e_f, \mu, \texttt{true})$$

$$\Updownarrow$$

$$\forall\, \beta,\ \exists\, m \in \mathbb{N} :\ \chi(m, e_f, e_f, \beta) = \nu \in \mathcal{V}al\ \wedge\ \varepsilon(e_f, e_f, \beta, \nu)$$

(⇒) Suponha que exista uma medida $\mu : \mathcal{V}al^n \to (\mathcal{MT}, \succcurlyeq)$ tal que $\zeta(e_f, e_f, \mu, \texttt{true})$. Assim,

i) Do Teorema 6.1.11 segue que toda sequência de contextos bem formada é finita. Assim, para toda sequência bem formada, existe um número natural que limita o seu comprimento.

ii) Um chamado específico gera um número finito de sequências bem formadas, uma vez que em cada nível de recursão somente pode ser realizado um número finito de chamados recursivos subsequentes, conforme Figura 6.1.

Assim, de (i) e (ii), aplicando o Lema de König, conclui-se que as sequências bem formadas originadas a partir de um chamado específico são limitadas por um natural h, que é o máximo comprimento dessas sequências. Portanto, tome m igual ao máximo h e observe que, para todo β, $\chi(h, e_f, e_f, \beta)$ decrementa h em uma unidade a cada chamado recursivo. Desta forma, $T_\chi(e_f)$ vale para h.

(⇐) Suponha que para qualquer designação β de valores para as variáveis, existe um natural m tal que $\chi(m, e_f, e_f, \beta) = \nu \in \mathcal{V}al$ e $\varepsilon(e_f, e_f, \beta, \nu)$. Assim, para uma designação β defina k_β como sendo o menor número natural tal que $\chi(k_\beta, e_f, e_f, \beta) \in \mathcal{V}al$. Desta forma, para $f(x_1, \ldots, x_n) = e_f$, defina a medida $\mu : \mathcal{V}al^n \to (\mathcal{MT}, \succcurlyeq)$ por:

$$\mu : (\beta(x_1), \ldots, \beta(x_n)) \mapsto k_\beta.$$

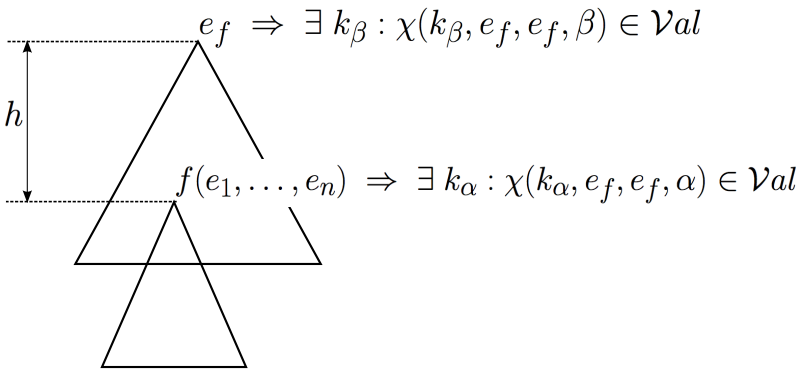


Figura 6.2: Dada uma definição de função n-ária $f(x_1, \ldots, x_n) = e_f$, e uma designação inicial β, a cada chamado recursivo a designação muda e existe um mínimo k_β tal que $\chi(k_\beta, e_f, e_f, \beta)$.

Note que, a cada chamado recursivo de f ocorrendo em e_f, a designação de valores para as variáveis muda de acordo com a Definição 6.1.3. Assim, se para um dado chamado recursivo $f(e_1, \ldots, e_n)$ tem-se uma nova designação α, então existe um mínimo k_α tal que $\chi(k_\alpha, e_f, e_f, \alpha) \in \mathcal{V}al$, de acordo com a Figura 6.2, e

$$\mu : (\alpha(x_1), \ldots, \alpha(x_n)) \mapsto k_\alpha.$$

Note que se a distância entre a raiz na definição de e_f e o chamado recursivo $f(e_1, \ldots, e_n)$ é h, então $k_\beta - h \geqslant k_\alpha$, pois caso contrário, $\chi(k_\beta - h, e_f, e_f, \alpha)$ não seria um valor, uma vez que k_α é o mínimo necessário para se obter um valor. Logo, para qualquer designação β de valores para as variáveis a função $\mu : (\beta(x_1), \ldots, \beta(x_n)) \mapsto k_\beta$ assim definida é uma função de medida e $T_\zeta(e_f)$ vale para μ.

O que conclui a prova. ■

6.2 Terminação em MWG's e Terminação para Linguagens Funcionais de Primeira Ordem

Uma possível verificação da conexão entre a semântica da terminação para linguagens de primeira ordem (LPO) e terminação em MWG's deve ser desenvolvida via CCG's, primeiro verificando que a semântica da terminação em LPO's implica terminação em CCG's, e em seguida aplicando o Teorema 5.4.9, onde verifica-se equivalência entre terminação em CCG's e MWG's. O primeiro passo é apresentar a construção de um CCG a partir de uma definição de função especificada segundo a sintaxe apresentada na Definição 5.1.1. Tal construção é desenvolvida por partes, primeiro formando o conjunto de vértices, em

seguida a partir de análises locais sobre as condições de cada vértice, algumas arestas são removidas e assim é formado o conjunto de arestas, então o digrafo está pronto e basta incluir uma família de medidas (que são funções sobre o domínio de uma relação bem fundada) para então obter o CCG. Em PVS, esta construção é desenvolvida a partir da especificação das seguintes funções:

Definição 6.2.1 (Função para construir o pré-conjunto de vértices)**:** *Dada uma definição de função $f(x_1, \ldots, x_n) = e_f$, o pré-conjunto de vértices do CCG associado a e_f é obtido através da aplicação $PV^{lpo}_{ccg} : \mathcal{E}xp \times \mathcal{E}xp \to V$, definida por:*

$$
\begin{array}{rcl}
PV^{lpo}_{ccg}(e_f, \texttt{cnd}) & := & \texttt{CASES } e_f \texttt{ OF} \\
& & c: \ \emptyset; \\
& & x: \ \emptyset; \\
& & g(e_1, \ldots, e_k): \ \bigcup_{i=1}^{k} PV^{lpo}_{ccg}(e_i, \texttt{cnd}); \\
& & ite(e_1, e_2, e_3): \ PV^{lpo}_{ccg}(e_1, \texttt{cnd}) \ \cup \ PV^{lpo}_{ccg}(e_2, e_1 \wedge \texttt{cnd}) \ \cup \ PV^{lpo}_{ccg}(e_3, \neg e_1 \wedge \texttt{cnd}); \\
& & f(e_1, \ldots, e_n): \ \langle (x_1, \ldots, x_n), \texttt{cnd}, (e_1, \ldots, e_n) \rangle \ \cup \ (\bigcup_{i=1}^{n} PV^{lpo}_{ccg}(e_i, \texttt{cnd})),
\end{array}
$$

onde V é um conjunto de vértices cujos elementos são contextos de chamado, `cnd` *representa uma conjunção de condições e $\emptyset$ denota o conjunto vazio. No caso recursivo, $\langle (x_1, \ldots, x_n), \texttt{cnd}, (e_1, \ldots, e_n) \rangle$ denota o contexto de chamado com formais $(x_1, \ldots, x_n)$, condição* `cnd` *e atuais $(e_1, \ldots, e_n)$.*

Observe que, na definição anterior, existe uma condição inicial $\texttt{cnd} \in \mathcal{E}xp$ utilizada para construir a condição de cada contexto de chamado. A condição de um contexto de chamado é construida por conjunção das condições presentes nos casos *if-then-else* da definição da função, conforme Definição 5.1.2. Assim, inicialmente na construção do conjunto de vértices não existem condições, então o símbolo ϕ será empregado para representar a condição *vazia* e na seguinte definição a função especificada para efetivamente construir o conjunto de vértices do CCG é apresentada.

Definição 6.2.2 (Função para construir o cojunto de vértices)**:** *Seja uma definição de função $f(x_1, \ldots, x_n) = e_f$. O conjunto de vértices do CCG associado à expressão e_f é dado pela aplicação $V^{lpo}_{ccg} : \mathcal{E}xp \to V$, definida por:*

$$V^{lpo}_{ccg}(e_f) := PV^{lpo}_{ccg}(e_f, \phi)$$

A fim de construir o conjunto de arestas, é necessário desenvolver uma análise para remover algumas arestas. Isto significa que inicialmente todas as possíveis arestas (pares de vértices) estão presentes no CCG, mas algumas serão descartadas. A seguinte definição

estabelece como é desenvolvida esta análise local, que basicamente consiste em verificar se os parâmetros atuais de um determinado vértice satisfazem as condições de um possível vértice subsequente.

Definição 6.2.3 (Função para verificação das condições)**:** *A análise para verificar se os parâmetros de um determinado chamado recursivo na definição de função e_f satisfazem a condição* `cnd`*, é desenvolvida através da aplicação $vc : \mathcal{E}xp \times \mathcal{E}xp \times \beta \to$ `bool` definida por:*

$$\begin{aligned} vc(e_f, \texttt{cnd}, \beta) \;:=\; & \texttt{CASES cnd OF} \\ & \phi \;:\; \texttt{true} \\ & a \wedge \texttt{cnd}' \;:\; \texttt{CASES } a \texttt{ OF} \\ & \qquad e \;:\; \exists \nu \in \mathcal{V}al : \varepsilon(e, e_f, \beta, \nu) \;\wedge\; \nu \\ & \qquad \neg e \;:\; \exists \nu \in \mathcal{V}al : \varepsilon(e, e_f, \beta, \nu) \;\wedge\; \neg\nu \end{aligned}$$

Desta forma, a partir da definição anterior é possível remover arestas do CCG aplicando a seguinte função:

Definição 6.2.4 (Remoção de arestas)**:** *Dados $f(x_1, \ldots, x_n) = e_f$ uma definição de função e $cc_1 = \langle (x_1, \ldots, x_n), \texttt{cnd}_1, (e_1^1, \ldots, e_n^1)\rangle$ e $cc_2 = \langle (x_1, \ldots, x_n), \texttt{cnd}_2, (e_1^2, \ldots, e_n^2)\rangle$ dois contextos de chamado, a aplicação $re : \mathcal{E}xp \times V \times V \to$ `bool` definida por:*

$$\begin{aligned} re(e_f, cc_1, cc_2) := (\; & \forall\, \beta : \mathcal{V}ar \to \mathcal{V}al,\; \forall\, \nu_i \in \mathcal{V}al \;: \\ & vc(e_f, \texttt{cnd}_1, \beta) \;\wedge\; \varepsilon(e_i^1, e_f, \beta, \nu_i)\; i = 1, \ldots, n\;) \\ & \Rightarrow \neg vc(e_f, \texttt{cnd}_2, \beta'),\;\; \textit{onde } \beta'(x_i) = \nu_i,\; i = 1, \ldots, n \end{aligned}$$

determina se é possível existir a aresta (cc_1, cc_2) no CCG associado a e_f ou não.

A partir da definição anterior, é possível construir o conjunto de arestas do CCG, como segue na Definição 6.2.5. Mas note que, através da análise proposta na Definição 6.2.4, nem sempre é possível responder se uma aresta pode ser removida ou não, uma vez que tal análise é indecidível

Definição 6.2.5 (Função para construir o conjunto de arestas)**:** *Seja uma definição de função $f(x_1, \ldots, x_n) = e_f$. O conjunto de arestas do CCG associado à expressão e_f é dado pela aplicação $E_{ccg}^{lpo} : \mathcal{E}xp \to E$, definida por:*

$$E_{ccg}^{lpo}(e_f) := \{(cc_1, cc_2) \mid cc_1, cc_2 \in V_{ccg}^{lpo}(e_f) \;\wedge\; \neg re(e_f, cc_1, cc_2)\}$$

Assim, dada uma família fixa $\mathcal{F}$ de medidas, que são funções definidas sobre o domínio de uma relação bem fundada, é possível definir a construção de um CCG a partir de uma expressão escrita segundo a sintaxe definida em 5.1.

Definição 6.2.6 (CCG de uma expressão em $\mathcal{E}xp$)**:** *Seja uma expressão $e \in \mathcal{E}xp$ e uma família de medidas $\mathcal{F}$. O CCG G' associado a e, com medidas $\mathcal{F}$, é obtido pela aplicação $\mathcal{T}_{ccg}^{lpo} : \mathcal{E}xp \times \mathcal{F} \to CCG$ definida por:*

$$\mathcal{T}_{ccg}^{lpo}(e, \mathcal{F}) := \langle V_{ccg}^{lpo}(e),\ E_{ccg}^{lpo}(e),\ \mathcal{F} \rangle$$

Mas, note que devido a indecidibilidade da análise desenvolvida na remoção de arestas, o CCG de uma expressão e, dada uma família de medidas $\mathcal{F}$ obtido através de $\mathcal{T}_{ccg}^{lpo}$ não é um CCG *exato* no sentido de que seu conjunto de caminhos pode conter mais sequências de chamados do que realmente são possíveis no fluxo do programa representado por e. Contudo é um CCG *correto* no sentido de que o conjunto de seus caminhos contém todas as sequências de chamados que são possíveis de acordo com o fluxo do programa representado por e. Desta forma, um CCG (teórico) de uma expressão e, que contenha apenas arestas que são possíveis de acordo com o fluxo do programa representado por e, será denominado um *CCG exato*, e o CCG da mesma expressão e, obtido através da definição 6.2.6, será denominado um *CCG correto*.

Teorema 6.2.7: *Dada uma definição de função $f(x_1, \ldots, x_n) = e_f$ e uma família de medidas $\mathcal{F}$, considere o CCG exato obtido a partir de e_f. Assim,*

$$T_{ccg}(G) \Leftrightarrow T_{\zeta}(e_f)$$

Prova: Deve-se verificar que: para todo circuito do CCG existe uma combinação de medidas que leva a uma sequência decrescente no domínio de uma relação bem fundada se, e somente se, existe uma medida que decresce a cada chamado recursivo.

(⇒) Tem-se que $T_{ccg}(G)$ implica que todas as sequências bem formadas de contextos de chamado são finitas (Teorema 2 de [MV06] - Teorema 3.2.7). Assim, segue do Lema 6.1.11 que $T_{\zeta}(e_f)$.

(⇐) Por outro lado, suponha que $T_{\zeta}(e_f)$, ou seja, existe uma medida $\mu : \mathcal{V}al^n \to (\mathcal{MT}, \succcurlyeq)$ que decresce a cada chamado recursivo em e_f. Assim, use μ no CCG exato, pois μ decresce em cada aresta do CCG.

■

Teorema 6.2.8: *Considere uma definição de função $f(x_1, \ldots, x_n) = e_f$ e uma família de medidas $\mathcal{F}$. Sejam G_{ex} o CCG exato e G_{cr} o CCG correto obtidos a partir de e_f. Assim,*

$$T_{ccg}(G_{cr}) \Rightarrow T_{ccg}(G_{ex})$$

Prova: Em primeiro lugar, é importante remarcar que o CCG correto G_{cr} pode ter mais arestas que o CCG exato G_{ex}. Assim, se $T_{ccg}(G_{cr})$ vale, então para cada circuito de G_{cr} existe uma combinação de medidas que leva a uma sequência estritamente decrescente de valores tomados no domínio de uma relação bem fundada, mas todo circuito de G_{ex} é também um circuito de G_{cr}. Assim, para todo circuito de G_{ex} existe uma combinação de medidas que leva a uma sequência estritamente decrescente de valores tomados no domínio de uma relação bem fundada, ou seja, vale $T_{ccg}(G_{ex})$. ■

Teorema 6.2.9: *Dada uma definição de função $f(x_1, \ldots, x_n) = e_f$ e uma família de medidas $\mathcal{F}$, considere o CCG obtido a partir de e_f por $\mathcal{T}^{lpo}_{ccg}(e_f, \mathcal{F})$. Assim,*

$$T_{ccg}(\mathcal{T}^{lpo}_{ccg}(e_f, \mathcal{F})) \Rightarrow T_{\zeta}(e_f)$$

Prova: Considere o CCG exato G_{ex} obtido a partir de e_f. Assim, temos que:

$$T_{ccg}(\mathcal{T}^{lpo}_{ccg}(e_f, \mathcal{F})) \overset{\text{Teorema } 6.2.8}{\Longrightarrow} T_{ccg}(G_{ex}) \overset{\text{Teorema } 6.2.7}{\Longleftrightarrow} T_{\zeta}(e_f)$$

O que conclui a prova. ■

Teorema 6.2.10: *Dada uma definição de função $f(x_1, \ldots, x_n) = e_f$ e uma família de medidas $\mathcal{F}$, considere o CCG obtido a partir de e_f por $\mathcal{T}^{lpo}_{ccg}(e_f, \mathcal{F})$. Considere ainda o MWG $\mathcal{T}^{ccg}_{mwg}(\mathcal{T}^{lpo}_{ccg}(e_f, \mathcal{F}))$. Assim,*

$$T_{mwg}(\mathcal{T}^{ccg}_{mwg}(\mathcal{T}^{lpo}_{ccg}(e_f, \mathcal{F}))) \Rightarrow T_{\zeta}(e_f)$$

Prova: Fato que se verifica da seguinte maneira:

$$T_{mwg}(\mathcal{T}^{ccg}_{mwg}(\mathcal{T}^{lpo}_{ccg}(e_f, \mathcal{F}))) \overset{\text{Teorema } 5.4.9}{\Longleftrightarrow} T_{ccg}(\mathcal{T}^{lpo}_{ccg}(e_f, \mathcal{F})) \overset{\text{Teorema } 6.2.9}{\Longrightarrow} T_{\zeta}(e_f)$$

O que conclui a prova. ■

Remarcando que:

- Se, para uma definição de função $f(x_1, \ldots, x_n) = e_f$, tem-se que $T_{\chi}(e_f)$, então qualquer entrada válida para f se executa em um tempo finito. O que implica que, para qualquer designação de valores para as variáveis, não existem sequências bem formadas infinitas de contextos de f. Este fato é equivalente ao Teorema 1 de [MV06] (também em Teorema 3.2.3).

- Se, para uma definição de função $f(x_1, \ldots, x_n) = e_f$ e uma família de medidas $\mathcal{F}$, o CCG obtido por $\mathcal{T}^{lpo}_{ccg}(e_f)$ é tal que, para cada caminho infinito de G existe uma combinação de medidas que leva a uma sequência decrescente no domínio

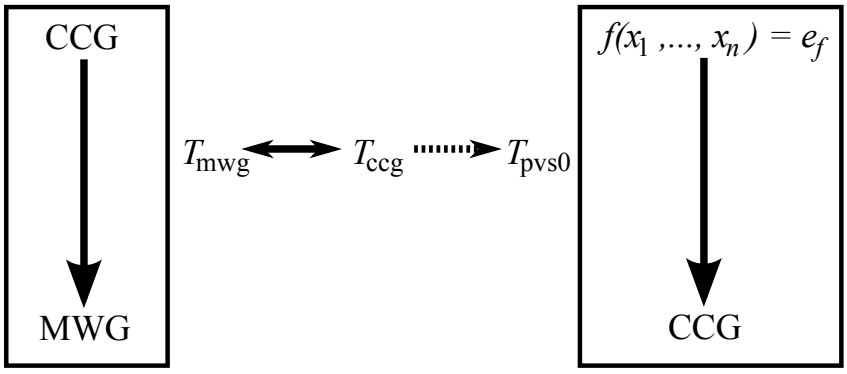


Figura 6.3: Conexão entre as *teorias* especificadas (MWG, CCG e PVS0) e entre as propriedades formalizadas.

de uma relação bem fundada, então para nenhuma designação de valores para as variáveis existe uma sequência bem formada de contextos infinita. Este fato remete ao Teorema 2 de [MV06] (Teorema 3.2.7).

Então a formalização em PVS do Teorema 6.2.10, deve se dar neste sentido: dada uma definição de função $f(x_1, \ldots, x_2) = e_f$ e o respectivo CCG $G' = \mathcal{T}_{ccg}^{lpo}(e_f)$ verificar que terminação em G' implica inexistência de sequências infinitas de contexto, o que por sua vez implica que existe um $n \in \mathbb{N}$ tal que $T_\chi(e_f)$, o que por sua vez implica em $T_\zeta(e_f)$.

Assim, uma vez concluída a formalização, a seguinte conexão, expressa na Figura 6.3, entre as teorias estará estabelecida:

- Tem-se a construção de MWG's a partir de CCG's, e equivalência entre terminação em MWG's e CCG's;
- Tem-se a construção de CCG's a partir de um termo da linguagem de primeira ordem especificada;
- Resta a verificação formal em PVS de que terminação em CCG's implica em terminação da respectiva definição de função em linguagem de primeira ordem.

Capítulo 7

Conclusão e Trabalhos Futuros

Neste trabalho foi introduzida uma nova estrutura de dados, baseada em digrafos com peso, para verificação de terminação de programas funcionais. Denominou-se esta estrutura *Grafos com Matrizes de Medida* (MWG's). Esta estrutura está especificada no assistente de prova PVS e encontra-se na *sub-teoria* `matrix_wdg`.

A Figura 7.1 mostra a hierarquia completa da especificação e formalização de todas as *teorias* em PVS apresentadas neste trabalho, onde é possível visualizar as dependências entre as *sub-teorias* e a qual *teoria* cada uma pertence. No que segue apresenta-se uma descrição resumida de cada uma delas.

1. `digraphs`:

   Foi necessário ampliar a formalização da *teoria* `digraphs` já existente em PVS, visando desenvolver a base para a especificação/formalização de MWG's. As contribuições desenvolvidas na *teoria* `digraphs` encontran-se disponíveis em [BRSA13] e consistem principalmente das *sub-teorias* `circuits`, `cycles` e `weighted_digraphs`, embora existam contribuições em outras *sub-teorias* de `digraphs`, como mencionado na Seção 3.1.

   Uma nova *sub-teoria*, denominada `weighted_digraphs`, contendo especificação de digrafos com peso, bem como a formalização de vários resultados relativos a esta estrutura, foi acrescentada à *teoria* [BRSA13]. O principal aspecto da estrutura especificada em `weighted_digraphs` reside no fato de que os rótulos (pesos) são atribuídos às arestas, por uma aplicação definida no conjunto de arestas do digrafo sobre um conjunto com estrutura de um monóide, isto é, um conjunto com uma operação binária associativa e que possui elemento neutro. Desta maneira, a função de medida definida nas arestas em um digrafo com pesos é tratada de maneira geral

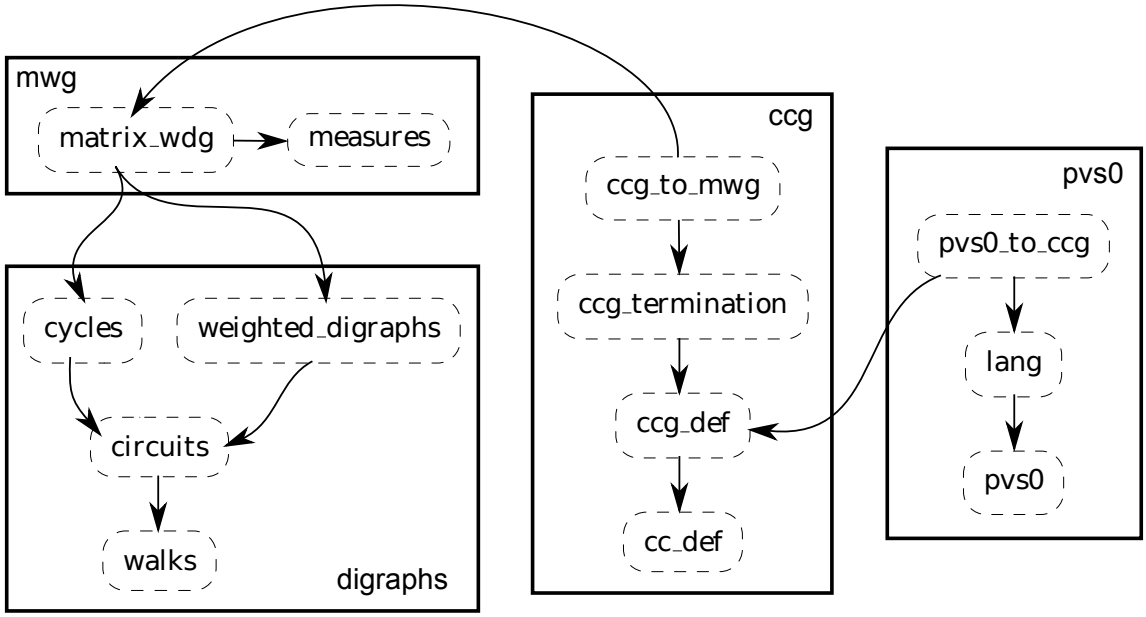


Figura 7.1: Hierarquia da especificação/formalização: mostra as principais *teorias* em PVS envolvidas no desenvolvimento.

e pode ser parametrizada com qualquer função definida nas arestas do digrafo sobre um monóide. Assim, foi possível especificar MWG's com sua álgebra matricial de medidas como um parâmetro na *teoria* `matrix_wdg`.

Os desenvolvimentos constantes da *teoria* `digraphs` correspondem à parte da segunda contribuição desta tese, descritas na introdução.

2. `MWG`:

   Uma nova álgebra de matrizes foi desenvolvida, na *sub-teoria* `measures`, a fim de obter uma estrutura adequada à instanciação dos pesos da *sub-teoria* `weighted_digraphs`, onde era necessário um conjunto com estrutura de um monóide. Desta forma, instanciando a *sub-teoria* `weighted_digraphs` com a álgebra desenvolvida em `measures`, tem-se a estrutura MWG's em `matrix_wdg`.

   A definição de terminação para MWG's equivale à positividade de matrizes de medida, que sendo uma propriedade unicamente sobre matrizes de medida encontra-se especificada na *sub-teoria* `measures`, mas é aplicada na *sub-teoria* `matrix_wdg` na especificação de MWG's. A anterior constitui a especificação e formalização da primeira contribuição desta tese, segundo mencionado na introdução. Dois critérios de terminação para MWG's foram formalizados em PVS na *sub-teoria* `matrix_wdg`, sendo estes a terceira contribuição desta tese, conforme apresentado na introdução. Este critérios baseiam-se em observar positividade dos ciclos e arestas do MWG, que são elementos computáveis. Tais são os critérios:

- Critério 1: pode ser visualizado na Figura 4.2 e corresponde a Definição 4.4.4. Este critério lida com a existência de uma medida de controle, denominada medida limitante, juntamente com a hipótese de positividade de todo ciclo do MWG. A verificação de que este critério implica terminação em MWG's é formalizada no teorema `lm_pstv_cycles_pstv_circuits`, cuja especificação consta da figura 4.3 e corresponde ao Teorema 4.4.7. Assim, verificou-se que, se todo ciclo do MWG tem uma matriz de medida positiva e, se existe uma medida k limitante, isto é, toda aresta tem uma matriz com posição k definida ($\geqslant 0$) e todo ciclo duplo (circuito formado por dois ciclos) possui uma aresta com posição k positiva ($= 1$), então todo circuito do MWG possui matriz de medida positiva, em particular positiva na posição k.
- Critério 2: pode ser visualizado na Figura 4.5 e corresponde a Definição 4.4.9. Este critério lida com a existência de uma rotulagem fixa dos vértices do MWG, denominada rotulagem limitante. A verificação de que este critério implica terminação em MWG's é formalizada no teorema `ll_pstv_circuits`, cuja especificação consta da figura 4.6 e corresponde ao Teorema 4.4.11. Assim, verificou-se que se existe uma rotulagem limitante, isto é, toda aresta tem uma matriz definida ($\geqslant 0$) na posição correspondente aos rótulos de cada vértice da aresta, e todo ciclo possui uma matrix positiva ($= 1$) na posição correspondente ao rótulo do primeiro (ou último) vértice do ciclo, então todo circuito do MWG possui matriz de medida positiva, em particular positiva na posição correspondente ao rótulo do primeiro (ou último) vértice do circuito.

3. `CCG`:

   Foi desenvolvida uma especificação de *Calling Context Graphs* (CCG's), disponível nas *sub-teorias* `cc_def` e `ccg_def`, sendo que na primeira tem-se a especificação de contextos de chamado. A especificação de CCG's é baseada em digrafos, portanto os resultados de `digraphs` tamém são aplicados aqui. Uma família de medidas sobre o domínio de uma relação bem fundada é associada à especificação de um CCG e, a partir da definição de uma combinação de medidas para cada um dos circuitos do CCG que leve a uma sequência decrescente em relação à relação bem fundada escolhida, define-se terminação para CCG's. A definição de terminação para CCG's faz parte da *sub-teoria* `ccg_termination` e corresponde à noção introduzida por Manolios e Vroon em [MV06] e também em [Vro07].

   Funções para construir um MWG a partir de um CCG são especificadas na *sub-teoria*

`ccg_to_mwg`. A construção é desenvolvida por funções que constroem as comparações entre as medidas da família de medidas associada ao CCG, para cada aresta, a parte do CCG correspondente ao digrafo é idêntica ao digrafo do MWG. Ainda em `ccg_to_mwg` é verificada de forma construtiva a equivalência entre terminação para MWG's e terminação para CCG's. Esta prova de equivalência estabelece uma certificação para a estrutura MWG's no sentido de que, assumindo que CCG's são uma ferramente adequada para verificação de terminação, MWG's também o são.

Os desenvolvimentos constantes da *teoria* `ccg` correspondem à parte da segunda contribuição desta tese, descritas na introdução.

4. `PVS0`:

   Uma especificação de uma linguagem de primeira ordem, com variáveis, constantes, operadores assumidos bem definidos, totais e efetivamente computáveis, expressões condicionais e chamados recursivos foi desenvolvida na *teoria* `PVS0`. A especificação da semântica de terminação para esta linguagem e algumas propriedades relacionando equivalência entre a semântica da terminação e terminação via contagem de chamados recursivos foram formalizadas na *sub-teoria* `lang`. A especificação de funções que constroem um CCG para termos da linguagem `PVS0` está presente na *sub-teoria* `pvs_to_ccg`. A verificação de que terminação em CCG implica em terminação para termos desta linguagem ainda não foi formalizada em PVS. Este é um trabalho futuro imediato, onde pretende-se verificar que terminação em MWG's implica terminação de funções definidas em `PVS0`. Esta formalização deve ser desenvolvida verificando-se que terminação em CCG's implica em terminação de funções definidas em `PVS0`, para em seguida aplicar a equivalência entre terminação em CCG's e MWG's.

Adicionalmente, é importante ampliar o atual desenvolvimento através de:

- Formalizar outros critérios de terminação, mais poderosos, baseados na positividade dos ciclos de um MWG;
- Estender a linguagem funcional especificada em `PVS0`, de maneira que: *i*) outros comandos além de condicionais `if-then-else` sejam possíveis, como por exemplo `case of`, `let in`; *ii*) o uso de variáveis globais seja possível; *iii*) seja possível a abordagem de casos de recorrência mútua, etc;
- Estender o método para verificação de terminação em outros formalismos, como por exemplo sistemas de reescrita de termos e linguagens de ordem superior.

Referências Bibliográficas

[AdMGA10] Andréia B. Avelar, Flávio L. C. de Moura, André Luiz Galdino, and Mauricio Ayala-Rincón. Verification of the Completeness of Unification Algorithms à la robinson. In *Logic, Language, Information and Computation, 17th International Workshop, WoLLIC 2010, Brasilia, Brazil, July 6-9, 2010. Proceedings*, volume 6188 of *Lecture Notes in Computer Science*, pages 110–124. Springer, 2010.

[AG00] Thomas Arts and Jürgen Giesl. Termination of Term Rewriting Using Dependency Pairs. *Theoretical Computer Science*, 236(1):133–178, 2000.

[AGdMAR12] Andréia Borges Avelar, André Luiz Galdino, Flávio L.C. de Moura, and Mauricio Ayala-Rincón. A Formalization of the Theorem of Existence of First-Order Most General Unifiers. *Electronic Proceedings in Theoretical Computer Science*, 81:63–78, 2012.

[AGdMAR14] Andréia Borges Avelar, André Luiz Galdino, Flávio L.C. de Moura, and Mauricio Ayala-Rincón. First-order Unification in the PVS Proof Assistant. *Logic Journal of IGPL*, No prelo 2014.

[ARR13] Mauricio Ayala-Rincón and Yuri Santos Rego. Formalization in PVS of Balancing Properties Necessary for Proving Security of the Dolev-Yao Cascade Protocol Model. *Journal of Formalized Reasoning*, 6(1):31–61, 2013.

[BK11] Frédéric Blanqui and Adam Koprowski. CoLoR: a Coq Library on Well-founded Rewrite Relations and its Application to the Automated Verification of Termination Certificates. *Mathematical Structures in Computer Science*, 21(04):827–859, 2011.

[BL12] Frédéric Blanqui and Kim Quyen Ly. Automated Verification of Termination Certificates. *CoRR*, abs/1212.2350, 2012.

[BN98] Franz Baader and Tobias Nipkow. *Term Rewriting and* All That. Cambridge University Press, 1998.

[BRSA13] Ricky Butler, Kristin Rozier, Jon Sjogren, and Andréia Borges Avelar. A PVS *Theory* for Digraphs, 2013. Disponível: `http://shemesh.larc.nasa.gov/fm/ftp/larc/PVS-library/pvslib.html` (Visitada em 29/07/2014).

[CDMV11] Harsh Raju Chamarthi, Peter Dillinger, Panagiotis Manolios, and Daron Vroon. The ACL2 Sedan Theorem Proving System. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6605:291–295, 2011.

[CGBA+11] Michael Codish, Igor Gonopolskiy, Amir M Ben-Amram, Carsten Fuhs, and Jürgen Giesl. Sat-based termination analysis using monotonicity constraints over the integers. *Theory and Practice of Logic Programming*, 11(4-5):503–520, 2011.

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

[CPR06a] Byron Cook, Andreas Podelski, and Andrey Rybalchenko. Termination Proofs for Systems Code. In Michael I. Schwartzbach and Thomas Ball, editors, *Proceedings of the ACM SIGPLAN 2006 Conference on Programming Language Design and Implementation, Ottawa, Ontario, Canada, June 11-14, 2006*, pages 415–426. ACM, 2006.

[CPR06b] Byron Cook, Andreas Podelski, and Andrey Rybalchenko. Terminator: Beyond Safety. *In CAV06, Lecture Notes in Computer Science*, 4144:415–418, 2006.

[CPR07] Byron Cook, Andreas Podelski, and Andrey Rybalchenko. Proving thread termination. In Jeanne Ferrante and Kathryn S. McKinley, editors, *Proceedings of the ACM SIGPLAN 2007 Conference on Programming Language Design and Implementation, San Diego, California, USA, June 10-13, 2007*, pages 320–330. ACM, 2007.

[CPR11] Byron Cook, Andreas Podelski, and Andrey Rybalchenko. Proving Program Termination. *Communications of the ACM*, 54(5):88–98, 2011.

[CSZ13] Byron Cook, Abigail See, and Florian Zuleger. Ramsey vs. Lexicographic Termination Proving. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 47–61. Springer, 2013.

[Cyr94] David Cyrluk. Microprocessor Verification in PVS - A Methodology and Simple Example. Technical report, SRI International, 1994.

[Der85] Nachum Dershowitz. Termination. In *Rewriting Techniques and Applications*, pages 180–224. Springer, 1985.

[Der87] Nachum Dershowitz. Termination of Rewriting. *Journal of symbolic computation*, 3(1):69–115, 1987.

[DLMn09] Marc Daumas, David Lester, and César Muñoz. Verified real number calculations: A library for interval arithmetic. *Computers, IEEE Transactions on*, 58(2):226–237, 2009.

[DSD94] Danny De Schreye and Stefaan Decorte. Termination of Logic Programs: The Never-Ending Story. *The Journal of Logic Programming*, 19:199–260, 1994.

[EFT94] Heinz-Dieter Ebbinghaus, Jörg Flum, and Wolfgang Thomas. *Mathematical logic (2. ed.)*. Undergraduate texts in Mathematics. Springer, 1994.

[FKS11] Stephan Falke, Deepak Kapur, and Carsten Sinz. Termination Analysis of C Programs Using Compiler Intermediate Languages. In *22nd International Conference on Rewriting Techniques and Applications (RTA)*, volume 10, pages 41–50, 2011.

[FKS12] Stephan Falke, Deepak Kapur, and Carsten Sinz. Termination analysis of imperative programs using bitvector arithmetic. In Rajeev Joshi, Peter Müller, and Andreas Podelski, editors, *Verified Software: Theories, Tools, Experiments - 4th International Conference, VSTTE 2012, Philadelphia, PA, USA, January 28-29, 2012. Proceedings*, volume 7152 of *Lecture Notes in Computer Science*, pages 261–277. Springer, 2012.

[GAR09] André Luiz Galdino and Mauricio Ayala-Rincón. A PVS *Theory* for Term Rewriting Systems. *Electronic Notes in Theoretical Computer Science*, 247:67–83, 2009.

[GAR10] André Luiz Galdino and Mauricio Ayala-Rincón. A Formalization of the Knuth–Bendix (–Huet) Critical Pair Theorem. *Journal of Automated Reasoning*, 45(3):301–325, 2010.

[GS97] Susanne Graf and Hassen Saïdi. Construction of Abstract State Graphs with PVS. In Orna Grumberg, editor, *Computer Aided Verification*, volume 1254 of *Lecture Notes in Computer Science*, pages 72–83. Springer, 1997.

[GSKT06] Jürgen Giesl, Peter Schneider-Kamp, and René Thiemann. Automatic Termination Proofs in the Dependency Pair Framework. In Ulrich Furbach and Natarajan Shankar, editors, *IJCAR*, volume 4130 of *Lecture Notes in Computer Science*, pages 281–286. Springer, 2006.

[GTSKF06] Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, and Stephan Falke. Mechanizing and Improving Dependency Pairs. *Journal of Automated Reasoning*, 37(3):155–203, 2006.

[Kra07] Alexander Krauss. Certified Size-Change Termination. In Frank Pfenning, editor, *CADE*, volume 4603 of *Lecture Notes in Computer Science*, pages 460–475. Springer, 2007.

[Kra09] Alexander Krauss. *Automating Recursive Definitions and Termination Proofs in Higher-Order Logic*. PhD thesis, Institut für Informatik der Technischen Universität München, 2009.

[KSZM09] Martin Korp, Christian Sternagel, Harald Zankl, and Aart Middeldorp. Tyrolean Termination Tool 2. In Ralf Treinen, editor, *RTA*, volume 5595 of *Lecture Notes in Computer Science*, pages 295–304. Springer, 2009.

[LJBA01] Chin Soon Lee, Neil D. Jones, and Amir M. Ben-Amram. The Size-Change Principle for Program Termination. *POPL*, pages 81–92, 2001.

[LMG09] Leonard Lensink, César Muñoz, and Alwyn Goodloe. From Verified Models to Verifiable Code. Technical Memorandum NASA/TM-2009-215943, NASA, Langley Research Center, Hampton VA 23681-2199, USA, June 2009.

[MV06] Panagiotis Manolios and Daron Vroon. Termination Analysis with Calling Context Graphs. In Thomas Ball and Robert B. Jones, editors, *Computer*

Aided Verification, volume 4144 of *Lecture Notes in Computer Science*, pages 401–414. Springer, 2006.

[NM12] Anthony Narkawicz and César Muñoz. Formal Verification of Conflict Detection Algorithms for Arbitrary Trajectories. *Reliable Computing*, 17:209–237, December 2012.

[NM14] Anthony Narkawicz and César Muñoz. A Formally Verified Generic Branching Algorithm for Global Optimization. In Ernie Cohen and Andrey Rybalchenko, editors, *Proceedings of the 5th International Conference on Verified Software: Theories, Tools, and Experiments (VSTTE 2013)*, volume 8164 of *Lecture Notes in Computer Science*, pages 326–343. Springer, 2014.

[NMHZH13] Anthony Narkawicz, César Muñoz, Heber Herencia-Zapana, and George Hagen. Formal Verification of Lateral and Temporal Safety Buffers for State-Based Conflict Detection. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 227(9):1412–1424, September 2013.

[ORR+96] Sam Owre, S. Rajan, John M. Rushby, Natarajan Shankar, and Mandayam K. Srivas. PVS: Combining Specification, Proof Checking, and Model Checking. In Rajeev Alur and Thomas A. Henzinger, editors, *CAV*, volume 1102 of *Lecture Notes in Computer Science*, pages 411–414. Springer, 1996.

[ORS92] Sam Owre, John M. Rushby, and Natarajan Shankar. PVS: A Prototype Verification System. In Deepak Kapur, editor, *CADE*, volume 607 of *Lecture Notes in Computer Science*, pages 748–752. Springer, 1992.

[OS97] Sam Owre and Natarajan Shankar. The Formal Semantics of PVS. Technical report, SRI-CSL-97-2, Computer Science Laboratory, SRI International, Menlo Park, CA, August 1997. Disponível: `http://pvs.csl.sri.com/` (Visitada em 04/08/2014).

[STWZ12] Christian Sternagel, René Thiemann, Sarah Winkler, and Harald Zankl. Ceta - A tool for certified termination analysis. *CoRR*, abs/1208.1591, 2012.

[TS] René Thiemann and Christian Sternagel. IsaFoR - Isabelle Formalization of Rewriting. Disponível: `http://cl-informatik.uibk.ac.at/software/ceta/` (Visitada em 03/12/2014).

[TS09] René Thiemann and Christian Sternagel. Certification of termination proofs using ceta. In Stefan Berghofer, Tobias Nipkow, Christian Urban, and Makarius Wenzel, editors, *TPHOLs*, volume 5674 of *Lecture Notes in Computer Science*, pages 452–468. Springer, 2009.

[Tur36] Alan M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936.

[Vro07] Daron Vroon. *Automatically Proving the Termination of Functional Programs*. PhD thesis, Georgia Institute of Technology, 2007.

[Wal04] Johannes Waldmann. Matchbox: A Tool for Match-Bounded String Rewriting. In Vincent van Oostrom, editor, *Rewriting Techniques and Applications, 15th International Conference, RTA 2004, Aachen, Germany, June 3-5, 2004, Proceedings*, volume 3091 of *Lecture Notes in Computer Science*, pages 85–94. Springer, 2004.