UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE CIÊNCIAS
DEPARTAMENTO DE COMPUTAÇÃO
DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

Márcia Roberta Falcão de Farias

Complexidade Descritiva da

Lógica de Ponto Fixo Relacional Inflacionário

Fortaleza
10 de Junho de 2016

Márcia Roberta Falcão de Farias

Complexidade Descritiva da Lógica de Ponto Fixo Relacional Inflacionário

Tese apresentada ao Curso de Doutorado em Ciência da Computação do Departamento de Computação da Universidade Federal do Ceará, como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Área de concentração: Lógica

Orientadora: Prof.ª Dr.ª Ana Teresa de Castro Martins

Co-orientador: Prof. Dr. Francicleber Martins Ferreira.

Fortaleza
10 de Junho de 2016

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

F238c Farias, Marcia Roberta Falcão de.
Complexidade Descritiva da Lógica de Ponto Fixo Relacional Inflacionário / Marcia Roberta Falcão de Farias. – 2016.
115 f. : il.

Tese (doutorado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação , Fortaleza, 2016.
Orientação: Prof. Dr. Ana Teresa de Castro Martins.
Coorientação: Prof. Dr. Francicleber Martins Ferreira.

1. Complexidade Descritiva. 2. Lógica de Ponto Fixo. 3. Lógica de Ponto Fixo Relacional. I. Título.
CDD 005

MÁRCIA ROBERTA FALCÃO DE FARIAS

Complexidade Descritiva da Lógica de Ponto Fixo Relacional Inflacionário

Tese submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação, da Universidade Federal do Ceará, como requisito para a obtenção do grau de Doutor em Ciência da Computação.

Aprovada em 10 de junho de 2016.

BANCA EXAMINADORA

Ana Teresa de Castro Martins

Prof.ª Dr.ª Ana Teresa de Castro Martins
(Presidente)
Universidade Federal do Ceará - UFC

Dr. Francicleber Martins Ferreira
(Coorientador)
Universidade Federal do Ceará - UFC

Prof. Dr. Edward Hermann Haeusler
Pontifícia Universidade Católica do Rio de Janeiro - PUC/RIO

Prof. Dr. Ruy José Guerra Barretto de Queiroz
Universidade Federal de Pernambuco - UFPE

Prof. Dr. Pedro Porfírio Muniz Farias
Universidade de Fortaleza - UNIFOR

João Fernando Lima Alcântara

Prof. Dr. João Fernando Lima Alcântara
Universidade Federal do Ceará – UFC

Aos meus filhos, Linus e Leon, com amor.

Agradecimentos

Agradeço ao Programa de Mestrado e Doutorado em Ciência da Computação da Universidade Federal do Ceará na pessoa da coordenadora Profa. Rossana todo o apoio recebido para o desenvolvimento deste trabalho.

Agradeço a CAPES pelo financiamento deste Doutorado.

Agradeço às secretarias e funcionários do MDCC/DC: Gláucia, Amanda, Juliana, Orley, Marquinhos, Martinha, Rosely, Reginaldo, Dona Margarida, Vlademiro, Daniel, que sempre estiverem disponíveis a ajudar no apoio logístico da organização das apresentações que precisei fazer.

Agradeço aos professores do MDCC/DC por todo tempo de convivência aqui na UFC.

Agradeço aos professores avaliadores desse trabalho Hermann Haeusler, Ruy Queiroz, Pedro Porfírio, João Fernando, Carlos Brito.

Agradeço aos meus colegas e amigos do laboratório LIA pelas horas de descontração vividas: Henrique Viana, Thiago, Daniel, Hugo, Luiz Henrique, Gabriela, José Wellington, Arnaldo, Carlos Filho, Rainara, Florêncio, Prof. João Paulo, Prof. João Fernando, Diego Parente, Jonatas, Saulo e tantos outros que posso ter esquecido aqui de nomear.

Agradeço a todos os amigos/amigas que estiveram presentes em conversas e em trocas de experiências Sandra Greice, Andrea Mendonça, Andrea Giordana, Marcos Silva e Henrique Reffert.

Agradeço às minhas amigas que também estão e/ou estiveram em momentos concomitantes no mesmo processo que eu: Karol, Camila e Cibele.

Agradeço à minha orientadora Profa. Ana Teresa Martins pela confiança atribuída a mim para a execução deste projeto.

Agradeço ao meu coorientador Prof. Francicleber Ferreira pelos conhecimentos passados e pela amizade surgida no desenvolvimento desta Tese.

Agradeço ao meu marido pela paciência, cuidado e vigilância para que eu não caísse em procrastinação muito frequentemente.

Agradeço à minha mãe pelo apoio emocional, mesmo a distância, e a compreensão de que era por um bem maior que eu passasse por esse processo de crescimento tanto intelectual quanto pessoal.

Agradeço aos meus filhos e maiores incentivadores de que era um sacrifício que valeria a pena a custa de nossas saudades e lágrimas a cada despedida.

Agradeço a todos que me substituíram a altura em minha ausência física nos aniversários, festividades familiares e sabendo/acreditando que estava buscando cumprir da melhor maneira a tarefa que escolhi realizar.

E por fim, agradeço a essa força misteriosa, que em minha fé atribuo o nome Deus, visto que não nego que em momentos de profunda angústia recorri a Ele para não desistir em meio às dificuldades.

“Disciplina é liberdade”
Renato Russo

Resumo

Complexidade Descritiva é um ramo da Teoria dos Modelos Finitos que está interessada em caracterizar classes de complexidade computacionais em termos dos recursos lógicos necessários para expressar todos os problemas que estão contidos na classe. O resultado mais celebrado da área é o Teorema de Fagin [Fag74] que prova que a classe NP é capturada pelo fragmento existencial de segunda ordem ($\exists$SO). Por outro lado, a lógica de primeira ordem (FO) não é expressiva o suficiente para definir problemas simples como o problema da conectividade de grafos. A introdução de operadores de ponto fixo é uma técnica padrão para acrescentar poder expressivo à lógica de maneira controlada. De fato, em [Imm82] e [Var82] é provado que FO com o operador de menor ponto fixo, denotada por LFP, é capaz de capturar a classe P sobre estruturas finitas e ordenadas. Neste trabalho, seguiremos uma abordagem diferente e definiremos a noção de ponto fixo sobre uma relação arbitrária. Nós generalizamos as lógicas clássicas de ponto fixo usando relações no lugar de operadores. A ideia básica é que usamos laços em uma relação no lugar de pontos fixos de uma função, isto é, X é ponto fixo de uma relação $\mathcal{R}$ no caso do par (X, X) pertencer à relação $\mathcal{R}$. Nós introduzimos a noção de ponto fixo inicial de uma relação inflacionária $\mathcal{R}$ e o operador **rifp** associado. Chamamos de RIFP a lógica FO com o operador de ponto fixo relacional inflacionário e mostramos que essa lógica captura a hierarquia polinomial usando uma tradução para a lógica de segunda ordem. Também consideramos o fragmento RIFP_1 com a restrição do operador **rifp** poder ser aplicado no máximo uma vez. Mostramos que RIFP_1 captura a classe NP e comparamos nossa lógica com a lógica de ponto fixo não-determinístico proposta por Abiteboul, Vianu e Vardi em [AVV97], que introduz a noção do ponto fixo não-determinístico e prova que a lógica FO com o operador de ponto fixo não-determinístico captura a classe NP sobre estruturas ordenadas. Os

resultados deste trabalho foram aceitos para publicação em [FMF16].

Abstract

Descriptive Complexity is a field of Finite Model Theory, which is interested in characterizing computational complexity classes in terms of the logical resources that are required to express all the problems belonging to the class. The seminal result in the area is the celebrated Fagin's Theorem, which proves that the class NP is captured by the existential fragment of second order logic [Fag74]. On the other side of the spectrum, we have the well known fact that first-order logic is not sufficiently expressive to define even such simple problems as graph connectivity. The introduction of fixed-point operators is a standard technique to increase the expressive power of a logic in a controlled way. Indeed, [Imm82] and [Var82] prove that first-order logic with the least fixed-point operator, denoted LFP, is able to capture the class P over the set of ordered structures. We generalize the classical fixed-point logics using relations instead of operators. The basic idea is that we use loops in a relation instead of fixed-points of a function, that is, X is a fixed-point of the relation $\mathcal{R}$ in case the pair (X, X) belongs to $\mathcal{R}$. We introduce the notion of initial fixed-point of an inflationary relation $\mathcal{R}$ and the associated operator **rifp**. We denote by RIFP the first-order logic with the inflationary relational fixed-point operator **rifp** and show that it captures the polynomial hierarchy using a translation to second-order logic. We also consider the fragment RIFP_1 with the restriction that the **rifp** operator can be applied at most once. We show that RIFP_1 captures the class NP and compare our logic with the non-deterministic fixed-point logic proposed by Abiteboul, Vianu and Vardi in [AVV97], that introduces the notion of non-deterministic fixed-points and proves that the first-order logic with such operators captures the class NP. The results of this work will be published in [FMF16].

Sumário

Capítulo 1

Introdução

A caracterização do poder expressivo das lógicas usando classes de complexidade é o tema central da área de Complexidade Descritiva [Imm99]. Tais caracterizações estabelecem que sentenças em alguma lógica definem problemas de decisão sobre estruturas finitas que pertencem a alguma classe de complexidade específica e vice-versa, ou seja, problemas nesta classe de complexidade podem ser definidos por sentenças nesta linguagem. Este tipo de resultado constrói uma ponte entre a Teoria dos Modelos Finitos [EF99] e a Teoria da Complexidade [Pap94] e abre a possibilidade de utilizar métodos e resultados de uma área para resolver os problemas na outra área e encontrar relações entre os recursos lógicos (por exemplo, quantificação, número de variáveis, quantificação de ordem superior, operadores de ponto fixo, etc.) e recursos computacionais (por exemplo, número de processadores, não-determinismo, alternância, tempo, espaço, etc.).

O modelo relacional de banco de dados introduzidos por Codd [Cod70] explicita a relação estreita entre a Teoria de Banco de Dados e a Teoria de Modelos Finitos. Bancos de dados podem ser vistos como estruturas finitas e linguagens de consulta como linguagens lógicas [AHV95]. O interesse em linguagens lógicas tem crescido, uma vez que tais linguagens com as semânticas associadas, fornecem forte base teórica para a Teoria de Banco de Dados, especialmente no processamento de consultas, e uma grande quantidade de técnicas de Teoria de Modelos Finitos pode ser aplicada. Por sua vez, este interesse na abordagem

lógica para bancos de dados promoveu um desenvolvimento intenso da Teoria dos Modelos Finitos.

Um candidato óbvio para uma linguagem de consulta é a lógica de primeira ordem (FO). A lógica de primeira ordem foi desenvolvida no século passado, como resultado dos esforços para proporcionar um fundamento lógico para a matemática. Nesse contexto, as estruturas infinitas desempenham um papel importante e, de fato, os métodos da teoria dos modelos de lógica de primeira ordem baseiam-se no fato das estruturas poderem ser infinitas [CK90, Hod93]. Embora FO esteja muito bem adaptada ao raciocínio matemático, seu poder expressivo em relação aos modelos finitos é bastante limitado. Muitos problemas, tais como decidir se um grafo é conexo ou não, não podem ser expressos em FO, mesmo quando restrito a estruturas finitas. Tais limitações derivam do fato de que FO não é capaz de expressar definições indutivas.

Esta falta de expressividade de FO foi solucionada com a introdução de operadores de ponto fixo para aumentar seu poder expressivo. Aho e Ullman [AU79] introduziram a ideia de estender a álgebra relacional, que é basicamente a lógica de primeira ordem, em forma algébrica, com operadores de ponto fixo. A ideia foi seguida por Chandra e Harel [CH80], dando origem a uma extensão da lógica de primeira ordem com o operador de menor ponto fixo, conhecida como lógica de menor ponto fixo (LFP).

Um operador é uma função $F : \mathcal{P}(A) \to \mathcal{P}(A)$, que dá origem a uma sequência $F_0, F_1, \ldots$ onde $F_0 = \emptyset$ e $F_{i+1} = F(F_i)$. Quando F é um operador monótono (ou seja, $X \subseteq Y \Rightarrow F(X) \subseteq F(Y)$), tem um menor ponto fixo $\mathbf{lfp}(F)$ que é atingido pela sequência. A lógica de menor ponto fixo estende a linguagem da lógica de primeira ordem com expressões que definem o menor ponto fixo de operadores monótonos adequadamente escolhidos (ver a definição precisa no Capítulo 3). A lógica de menor ponto fixo é estritamente mais expressiva do que FO. Por exemplo, propriedades de grafos, como conectividade e acessibilidade podem ser expressas por sentenças em LFP. Outras lógicas de ponto fixo podem ser definidas desta forma, mudando o tipo de operador e o tipo de ponto fixo escolhido.

Dada uma sentença em alguma lógica, o conjunto de seus modelos finitos definem um problema de decisão. Uma vez que podemos usar sentenças em alguma linguagem para expressar problemas de decisão, uma questão que rapidamente se coloca é: qual é a complexidade de tais problemas? Esta pergunta foi respondida para várias lógicas, veja [Imm99]. A lógica de primeira ordem, por exemplo, corresponde à hierarquia de tempo logarítmico. Um dos primeiros

resultados relativos ao poder expressivo de uma lógica e a complexidade computacional é o celebrado Teorema de Fagin [Fag74]. De acordo com o Teorema de Fagin, o fragmento existencial da lógica de segunda ordem define exatamente os problemas em NP (tempo polinomial não-determinístico, consulte [Pap94]), e foi usado por Stockmeyer [Sto76] para mostrar que a lógica de segunda ordem corresponde à hierarquia polinomial. Um resultado anterior, devido a Büchi [Büc60], mostra que a lógica monádica de segunda ordem define exatamente a classe das linguagens regulares. A lógica LFP tem um poder expressivo intermediário entre a lógica de primeira ordem e lógica existencial de segunda ordem. Na classe das estruturas ordenadas, LFP corresponde aos problemas de decisão na classe P, isto é, a classe dos problemas que podem ser resolvidos em tempo polinomial [Imm99].

A maioria das lógicas de ponto fixo são definidas do modo esboçado acima para o caso de LFP. Em particular, essas lógicas usam os operadores (funções) e seus pontos fixos para definirem conjuntos indutivos. Estamos interessados em estender essa noção considerando não só os operadores, mas as relações. Em vez de uma função $F : \mathcal{P}(A) \to \mathcal{P}(A)$ e seus pontos fixos, consideraremos as relações $\mathcal{R} \subseteq \mathcal{P}(A) \times \mathcal{P}(A)$ e os seus laços, isto é, X tal que $(X, X) \in \mathcal{R}$. Esta generalização abre novas possibilidades para a definição de operadores de ponto fixo e suas lógicas correspondentes. Como no caso dos operadores utilizados nas lógicas de ponto fixo tradicionais, as restrições adequadas sobre os tipos de relações e pontos fixos considerados podem ser utilizados para definirem várias lógicas de maneiras diferentes.

Iremos nos concentrar em um tipo específico de relação que chamaremos de inflacionária (ver Capítulo 4) e pode ser vista como a contrapartida relacional da lógica de ponto fixo inflacionário (IFP) [EF99, GKL$^+$07]. No Capítulo 2, apresentaremos os conceitos básicos utilizados neste trabalho tais como as definições e as notações das lógicas, alguns conceitos da Teoria da Complexidade e da Teoria da Complexidade Descritiva. No Capítulo 3, veremos os fundamentos da teoria de ponto fixo e das lógicas de ponto fixo. No Capítulo 4, definiremos alguns conceitos importantes sobre a teoria de ponto fixo relacional, apresentamos a lógica de ponto fixo relacional inflacionária (RIFP) com a sua sintaxe e sua semântica, e também, mostramos exemplos de problemas expressos nessa lógica. No Capítulo 5, demonstraremos que a lógica RIFP captura a classe de complexidade hierarquia polinomial através de uma tradução de RIFP para a lógica de segunda ordem e vice-versa; definiremos um fragmento da lógica RIFP que chamamos RIFP$_1$. Isso significa que a RIFP$_1$-fórmula contém apenas um ope-

rador **rifp** e demonstraremos que essa lógica captura a classe NP. No Capítulo 6, compararemos a lógica RIFP e a lógica NIFP que foi definida por Abiteboul, Vianu e Vardi [AVV97]. Nossos primeiros resultados, com o fragamento RIFP_1, foram apresentados em [FMF15] e, mais recentemente, os resultados mostrados nos Capítulos 4, 5 e 6 constam no artigo [FMF16] aceito para publicação. Finalmente, concluímos este trabalho e mostramos temas futuros para esta pesquisa no Capítulo 7.

Capítulo 2

Noções Básicas de Lógica e Complexidade

Neste capítulo serão introduzidos os conceitos básicos das lógicas de primeira ordem e de segunda ordem, alguns conceitos da Teoria da Complexidade tais como o modelo computacional da Máquina de Turing e das principais Classes de Complexidade, e, finalmente, os conceitos fundamentais da Teoria da Complexidade Descritiva. Seguiremos a notação de [End01, Ebb94, EF99, Pap94, GKL$^+$07, Lib12, AB09].

2.1 Lógica de Primeira Ordem

Nesta seção introduziremos a lógica de primeira ordem, a qual denotaremos por FO. Para maiores detalhes ver [Ebb94, End01].

2.1.1 Sintaxe das Linguagens de Primeira Ordem

Alfabeto

Antes de definirmos a linguagem precisamos dizer qual será o alfabeto que usaremos para escrever as palavras de nossa linguagem. Após isso, apresentaremos a sintaxe da linguagem de primeira ordem (denotada por FO) [Ebb94]. Cada fórmula da lógica de primeira ordem será uma *string* de símbolos do alfabeto. Veja a seguir a definição do alfabeto.

Definição 2.1.1 (Alfabeto) *O alfabeto da linguagem de primeira ordem contém os seguintes símbolos [Ebb94]:*

(i.) Variáveis (uma para cada inteiro positivo n): $v_1, v_2 \ldots$;

(ii.) Símbolos de conectivos sentenciais: $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$;

(iii.) Símbolos de quantificadores: $\forall$, $\exists$;

(iv.) Símbolo de igualdade: $\equiv$;

(v.) Parênteses: (,);

(vi.) Símbolos relacionais: para todo $k \geq 1$ *um conjunto de símbolos relacionais de aridade k (que pode ser vazio);*

(vii.) Símbolos de constantes: um conjunto de constantes (que pode ser vazio);

(viii.) Símbolos funcionais: para todo $k \geq 1$ *um conjunto de símbolos funcionais de aridade k (que pode ser vazio).*

O conjunto de símbolos que estão listados nos itens (i.) ao item (v.) será denotado por $\mathbb{A}$*, e por* S*, o conjunto que compreende os itens de (vi.) ao (viii.)*

que são os conjuntos dos símbolos relacionais, símbolos funcionais e constantes. Vamos nos referir ao conjunto S como **vocabulário** *ou* **conjunto de símbolos**.

Um conjunto de símbolos é um conjunto $S = \{R_1, \ldots, R_l, f_1, \ldots, f_k, c_1, \ldots, c_m\}$ de símbolos relacionais, símbolos funcionais e símbolos de constantes. Cada R_i é um símbolo relacional de aridade a_i e f_j é um símbolo funcional de aridade r_j. □

Fórmulas

Os **termos** são os nomes e os pronomes da linguagem que podem ser interpretados como objetos.

Definição 2.1.2 (S-termos) *O conjunto dos S-termos é o conjunto das expressões que podem ser construídas usando os símbolos em S, a partir de constantes e variáveis, aplicando as operações de funções, zero ou mais vezes [Ebb94]. As regras são as seguintes:*

(T1) Toda variável em S é um S-termo.

(T2) Toda constante em S é um S-termo.

(T3) Se $t_1, \ldots, t_k$ são S-termos e f é um símbolo funcional de aridade k em S, então $ft_1 \ldots t_k$ também é um S-termo.

Escreveremos $\overline{x} = x_1, \ldots, x_k$, para algum inteiro k, para denotar uma sequência de variáveis e escreveremos $\overline{x}_k$ quando quisermos explicitar o tamanho da sequência. □

Definição 2.1.3 (FO-Fórmulas) *As fórmulas da lógica de primeira ordem do vocabulário S são aquelas* strings *que são obtidas pela aplicação das seguintes regras um número finito de vezes [Ebb94]:*

(F1) Se t_0 e t_1 são S-termos então $t_0 \equiv t_1$ é uma FO[S]-fórmula.

(F2) Se R está em S e possui aridade k e $t_1, \ldots, t_k$ são S-termos então $Rt_1 \ldots t_k$ é uma FO[S]-fórmula.

(F3) Se ϕ é uma FO[S]-fórmula então $\neg\phi$ é uma FO[S]-fórmula.

(F4) Se ϕ e ψ são FO[S]-fórmulas então $(\phi \vee \psi)$, $(\phi \wedge \psi)$, $(\phi \rightarrow \psi)$, $(\phi \leftrightarrow \psi)$ são FO[S]-fórmulas.

(F5) Se ϕ é uma FO[S]-fórmula e x é uma variável então $\exists x\phi$ e $\forall x\phi$ são FO[S]-fórmulas.

*As fórmulas obtidas das regras (F1) e (F2) são chamadas de **fórmulas atômicas**. As fórmulas atômicas são aquelas fórmulas que não possuem símbolos de conectivos ou de quantificadores [End01]. Denotaremos por FO[S] o conjunto de S-fórmulas da lógica de primeira ordem do vocabulário S.* □

Variáveis Livres

Definição 2.1.4 (Variáveis livres) *Uma variável livre na fórmula ϕ será dada de maneira indutiva no tamanho da fórmula [Ebb94]:*

(i.) Se ϕ é uma fórmula atômica então o conjunto free(ϕ) das variáveis livres de ϕ é o conjunto das variáveis que ocorrem em ϕ.

(ii.) $free(\neg\phi) = free(\phi)$

(iii.) $free((\phi * \psi)) = free(\phi) \cup free(\psi)$, *onde* $* = \vee, \wedge, \rightarrow, \leftrightarrow$.

(iv.) $free(\exists x\phi) = free(\phi)\backslash\{x\}$.

(v.) $free(\forall x\phi) = free(\phi)\backslash\{x\}$. □

Definição 2.1.5 (Sentença) *Uma sentença é uma fórmula sem ocorrência de variáveis livres (Definição 2.1.4) [Ebb94].* □

2.1.2 Semântica das Linguagens de Primeira Ordem

Nesta seção definiremos a semântica da lógica de primeira ordem. Vimos os conceitos sintáticos de termos, fórmulas, variáveis livres. Agora vamos definir como dar significado para estes conceitos através de uma interpretação. Antes, precisamos entender que, para interpretar uma fórmula, precisamos atribuir significados a todos os símbolos que aparecem na fórmula.

Uma estrutura atribui significado aos símbolos relacionais, funcionais e de constantes, e mais, uma estrutura não atribui significado a símbolos de variáveis. Variáveis de primeira ordem são interpretadas por elementos do domínio. Para isso, precisamos do conceito de **interpretação**.

Estrutura e Interpretação

Definição 2.1.6 (S-estrutura) *Uma S-estrutura é um par $\mathfrak{A} = (A, \sigma)$ onde A é um conjunto não-vazio e σ é um mapeamento que associa cada símbolo de relação $R \in S$ de aridade k a uma relação $\sigma(R) = R^{\mathfrak{A}} \subseteq A^k$, cada símbolo de função f a uma função de $\sigma(f) = f^{\mathfrak{A}}$ e cada constante $c \in S$ a um elemento $\sigma(c) = c^{\mathfrak{A}}$.*

Denotamos por $STRUCT[S]$ o conjunto das S-estruturas finitas [Ebb94]. □

Definição 2.1.7 (Domínio de estruturas - $\mathcal{D}(S)$) *Um domínio de estruturas é um conjunto ou classe de estruturas $\mathcal{D}$ de conjuntos de símbolos diferentes e não necessariamente contém todas as estruturas (Definição 2.1.6) de um conjunto de símbolos específico. Escreveremos $\mathcal{D}(S)$ para as estruturas do domínio $\mathcal{D}$ com vocabulário fixo S [GKL$^+$07].* □

Definição 2.1.8 (Isormorfismo) *Sejam $\mathfrak{A}$ e $\mathfrak{B}$ S-estruturas.*

(i.) Um mapeamento $\pi : A \to B$ é chamado **isomorfismo** *entre $\mathfrak{A}$ e $\mathfrak{B}$ (escrevemos: $\pi : \mathfrak{A} \cong \mathfrak{B}$ se, e somente se,*

(1.) π é uma bijeção de A em B;

(2.) para $R \in S$ com aridade k e $a_1, \ldots, a_k \in A^k$,

$$R^{\mathfrak{A}} a_1 \ldots a_k \quad sse \quad R^{\mathfrak{B}} \pi(a_1) \ldots \pi(a_k); \tag{2.1}$$

(3.) para $f \in S$ com aridade k e $a_1, \ldots, a_k \in A^k$,

$$\pi(f^{\mathfrak{A}}(a_1, \ldots, a_k)) \quad = \quad f^{\mathfrak{B}}(\pi(a_1), \ldots, \pi(a_k)); \tag{2.2}$$

(4.) para todo $c \in S$, $\pi(c^{\mathfrak{A}}) = c^{\mathfrak{B}}$.

(ii.) $\mathfrak{A}$ e $\mathfrak{B}$ são ditos isomórficos ($\mathfrak{A} \cong \mathfrak{B}$) se, e somente se, existe um isomorfismo $\pi : \mathfrak{A} \cong \mathfrak{B}$ [Ebb94]. □

Definição 2.1.9 (Classe de Estruturas) *Classe de Estruturas é um conjunto de estruturas $\mathcal{K}$ para um vocabulário fixo S que é fechada sob isomorfismo, ou*

seja, se $\mathfrak{A} \in \mathcal{K}$ *e* $\mathfrak{A} \cong \mathfrak{B}$*, então* $\mathfrak{B} \in \mathcal{K}$ *[GKL*$^+$*07].* □

A seguir, definiremos atribuição, outro importante conceito para definirmos interpretação das variáveis.

Definição 2.1.10 (Atribuição) *Uma atribuição em uma* S*-estrutura* $\mathfrak{A}$ *é um mapeamento* $\beta : \{v_n \mid n \in \mathbb{N}\} \to A$ *do conjunto de variáveis no domínio* A*. Se* x *é uma variável de primeira ordem e* $a \in A$*, então* $\beta\frac{a}{x}$ *é a atribuição definida como* $\beta\frac{a}{x}(y) = a$ *se* $y = x$ *e* $\beta\frac{a}{x}(y) = \beta(y)$*, caso contrário [Ebb94].* □

Definição 2.1.11 (*S*-interpretação) *Uma* S*-interpretação* $\mathfrak{I}$ *é um par* $(\mathfrak{A}, \beta)$ *que consiste de uma* S*-estrutura e uma função* β*, que associa a cada variável de primeira ordem* x *um elemento* $\beta(x) \in A$ *[Ebb94].* □

Relação de Satisfatibilidade

A relação de satisfatibilidade estabelece quando uma interpretação satisfaz uma fórmula.

Definição 2.1.12 (Interpretação de *S*-termos) *Fixemos o vocabulário* S*, seja* $\mathfrak{A}$ *uma* S*-estrutura e* $\mathfrak{I}$ *uma* S*-interpretação. Associaremos a cada* S*-interpretação* $\mathfrak{I} = (\mathfrak{A}, \beta)$ *e cada* S*-termo* t *um elemento* $\mathfrak{I}(t)$ *do domínio* A*. Definiremos* $\mathfrak{I}(t)$ *por indução sobre os termos, como segue:*

(i.) para uma variável x *seja* $\mathfrak{I}(x) = \beta(x)$*;*

(ii.) para uma constante $c \in S$ *seja* $\mathfrak{I}(c) = c^{\mathfrak{A}}$*;*

(iii.) para um símbolo funcional $f \in S$ *de aridade* k *e termos* $t_1, \ldots, t_k$ *seja*

$$\mathfrak{I}(ft_1 \ldots t_k) = f^{\mathfrak{A}}(\mathfrak{I}(t_1), \ldots, \mathfrak{I}(t_k))$$ *[Ebb94].* □

Agora, por indução no conjunto das fórmulas ϕ, definiremos a relação de satisfatibilidade entre S-interpretação e S-fórmulas, em símbolos, $\mathfrak{I} \models \phi$ (leia-se $\mathfrak{I}$ *é um modelo de* ϕ), como:

Definição 2.1.13 (Relação de Satisfatibilidade) *Para todas as interpretações* $\mathfrak{I} = (\mathfrak{A}, \beta)$ *seja*

(i.) $\mathfrak{I} \models t_1 \equiv t_2$ *se, e somente se,* $\mathfrak{I}(t_1) \equiv \mathfrak{I}(t_2)$*;*

(ii.) $\mathfrak{I} \models Rt_1 \dots t_k$ *se, e somente se,* $\mathfrak{I}(t_1) \dots \mathfrak{I}(t_k) \in R^{\mathfrak{A}}$*;*

(iii.) $\mathfrak{I} \models \neg\phi$ *se, e somente se, não* $\mathfrak{I} \models \phi$*;*

(iv.) $\mathfrak{I} \models (\phi \wedge \psi)$ *se, e somente se,* $\mathfrak{I} \models \phi$ *e* $\mathfrak{I} \models \psi$*;*

(v.) $\mathfrak{I} \models (\phi \vee \psi)$ *se, e somente se,* $\mathfrak{I} \models \phi$ *ou* $\mathfrak{I} \models \psi$*;*

(vi.) $\mathfrak{I} \models (\phi \rightarrow \psi)$ *se, e somente se, se* $\mathfrak{I} \models \phi$*, então* $\mathfrak{I} \models \psi$*;*

(vii.) $\mathfrak{I} \models (\phi \leftrightarrow \psi)$ *se, e somente se,* $\mathfrak{I} \models \phi$ *sse* $\mathfrak{I} \models \psi$*;*

(viii.) $\mathfrak{I} \models \forall x\phi$ *se, e somente se, para todo* $a \in A$, $\mathfrak{I}^a_x \models \phi$*;*

(ix.) $\mathfrak{I} \models \exists x\phi$ *se, e somente se, existe um* $a \in A$ *tal que* $\mathfrak{I}^a_x \models \phi$ *[Ebb94].* □

Definição 2.1.14 (Classe de Modelos Finitos de ϕ) *Seja* ϕ *uma* S*-sentença. Definimos a classe de modelos finitos de* ϕ *como a classe de* S*-estruturas que satisfazem* ϕ*:*

$$Mod(\phi) = \{\mathfrak{A} \in STRUCT[S] \mid \mathfrak{A} \models \phi\} \tag{2.3}$$

□

Definição 2.1.15 (Lógica) *Definimos uma lógica como um par* $\mathcal{L} = (L, \models)$*, onde* L *é uma linguagem e* $\models$ *é uma relação de satisfatibilidade.* □

Outro importante conceito que usaremos será o conceito de relação definida. Veja a seguir.

Definição 2.1.16 (Relação definida) *Dada uma* S*-interpretação* $\mathfrak{I}$*, a fórmula* ϕ *define uma relação no domínio de* $\mathfrak{I}$*, a saber*

$$\phi(\overline{x})^{\mathfrak{I}} \quad = \quad \{\, \overline{a} \mid \mathfrak{I}\frac{\overline{a}}{\overline{x}} \models \phi(\overline{x}) \,\}, \tag{2.4}$$

onde $\overline{a}$ *é a interpretação da tupla de variáveis* $\overline{x}$ *e possuem o mesmo tamanho [GKL*$^+$*07]; as variáveis que ocorrem livre em* ϕ *estão em* $\overline{x}$*.* □

2.2 Lógica de Segunda Ordem

Nesta seção apresentaremos a lógica de segunda ordem cuja diferença para a lógica de primeira ordem é o poder de quantificar relações no domínio da estrutura, o que na lógica de primeira ordem não é possível [Ebb94].

2.2.1 Sintaxe das Linguagens de Segunda Ordem

Nesta seção definiremos a sintaxe da lógica de segunda ordem.

Definição 2.2.1 (Lógica de Segunda Ordem) *Seja* S *um conjunto de símbolos. O alfabeto da linguagem de segunda ordem,* $SO[S]$*, contém, em adição aos símbolos de* $FO[S]$*, para cada* $k \geq 1$*, as variáveis relacionais* $V_0^k, V_1^k, V_2^k, \ldots$ *de aridade* k*. Para denotar as variáveis relacionais usaremos as letras* $X, Y, \ldots$ *onde indicaremos a aridade por um sobrescrito, se for necessário. Definiremos*

os conjuntos das fórmulas de segunda ordem SO[S] como o conjunto gerado pelas regras do cálculo das fórmulas de primeira ordem (Definição 2.1.2 e Definição 2.1.3) acrescido das duas regras seguintes:

(i.) Se X é uma variável relacional de aridade k e $t_1, \ldots, t_k$ são S-termos, então $Xt_1 \ldots t_k$ é uma SO[S]-fórmula.

(ii.) Se ϕ é uma SO[S]-fórmula e X é uma variável relacional, então $\exists X\phi$ e $\forall X\phi$ são SO[S]-fórmulas [Ebb94]. □

2.2.2 Semântica das Linguagens de Segunda Ordem

Relação de Satisfatibilidade

Um assinalamento de segunda ordem β em $\mathfrak{A}$ é um mapeamento que assinala a cada variável v_i um elemento do domínio A e a cada variável relacional V_i^k de aridade k uma relação de aridade k sobre o domínio A. Analogamente, na Definição 2.1.10, em símbolos temos que $\mathfrak{I}\frac{C}{X} = \left(\mathfrak{A}, \beta\frac{C}{X}\right)$ e $\beta\frac{C}{X}$ é o assinalamento estendido, onde X é uma variável de segunda ordem de aridade k e $C \subseteq A^k$. Logo, $\beta\frac{C}{X}(Y) = C$ se $Y = X$ e $\beta\frac{C}{X}(Y) = \beta(Y)$ caso contrário.

Estendemos a noção de relação de satisfatibilidade da lógica de primeira ordem para a lógica de segunda ordem como apresentamos a seguir.

Definição 2.2.2 (Relação de Satisfatibilidade) *$\mathfrak{A}$ é uma S-estrutura, β um assinalamento de segunda ordem em $\mathfrak{A}$ e $\mathfrak{I} = (\mathfrak{A}, \beta)$, então definimos:*

(i.) $\mathfrak{I} \vDash Xt_1 \ldots t_k$ sse $\mathfrak{I}(t_1) \ldots \mathfrak{I}(t_k) \in \beta(X)$.

(ii.) Para X de aridade k:

$$\mathfrak{I} \models \exists X\varphi \quad sse \quad \text{existe um } C \subseteq A^k \text{ tal que } \mathfrak{I}\,\tfrac{C}{X} \models \varphi. \tag{2.5}$$

□

Denotaremos por SO a lógica de segunda ordem, isto é, o sistema lógico dado pela linguagem SO junto com a relação de satisfatibilidade para essa linguagem.

2.3 Máquina de Turing

Outro conceito importante que usaremos neste trabalho é o conceito de Máquina de Turing. Adotaremos este modelo de computação devido ao fato da maioria das classes de complexidade serem definidas em termos de Máquina de Turing. Usaremos as seguintes referências para as definições nesta seção [Pap94, AB09].

2.3.1 Máquina de Turing Determinística

Uma **Máquina de Turing** M é um dispositivo finito que executa operações em uma fita limitada ao lado esquerdo e ilimitada ao lado direito e esta fita está dividida em células [Pap94].

A seguir, iremos apresentar a definição formal para a Máquina de Turing Determinística [Pap94].

Definição 2.3.1 (Máquina de Turing Determinística (MTD)) *Uma Máquina de Turing é uma sextupla* $M = (Q, \Sigma, q_0, F^+, F^-, \delta)$*, onde* Q *é o conjunto de estados,* Σ *é o alfabeto de* M*,* q_0 *é o estado inicial,* F^+ *e* F^- *são os conjuntos de estados de aceitação e rejeição, respectivamente, e a função parcial* $\delta : (Q \times \Sigma) \to (Q \times \Sigma \times \{-1, 0, 1\})$ *é a função de transição. O conjunto* $\{-1, 0, 1\}$ *significa a codificação referente ao movimento da cabeça de leitura da máquina, i.e., -1 significando que a cabeça de leitura se movimenta para a esquerda, +1 significando que a cabeça de leitura se movimenta para a direita e 0 significando que a cabeça de leitura fica na mesma posição [Pap94, GKL*$^+$*07].* □

Uma Máquina de Turing trabalha somente com cadeias (*strings*), assim precisamos codificar nossos dados em cadeias (ver Seção 2.5.2).

Vamos explicar como a máquina funciona. Fixe um alfabeto finito Σ. A todo momento, cada célula da fita contém um único símbolo do alfabeto Σ ou é o símbolo branco $\sqcup$.

Uma máquina de Turing M contém uma cabeça de leitura que a cada instante encontra-se sobre uma única célula da fita e pode mover-se para esquerda, ou mover-se para a direita, ou permanecer no mesmo lugar, ou ainda, apagar o símbolo atual reescrevendo um novo símbolo em cima do anterior.

A cada instante, a máquina de Turing M encontra-se em um estado pertencente ao conjunto finito Q de estados .

Definição 2.3.2 (Instrução da MTD) *A ação da máquina de Turing Determinística M depende de seu estado atual e do símbolo atual que a cabeça de leitura está lendo. Esta ação é dada pela função δ. Cada instrução é da forma*

$$\delta(s, a) \quad = \quad (s', b, D) \tag{2.6}$$

onde

- $s, s' \in Q$;

- $a, b \in \Sigma$;

- $D \in \{-1, 0, +1\}$

[Pap94]. □

A instrução descrita na expressão (2.6) significa que, se a máquina está no estado s e sua cabeça de leitura lê o símbolo a na célula, ela substitui o símbolo a pelo símbolo b, move sua cabeça de leitura para a célula à esquerda, no caso de $D = -1$, ou move sua cabeça de leitura para a direita, no caso de $D = +1$, ou não move sua cabeça de leitura no caso de $D = 0$. Finalmente, a máquina muda o estado de s para s'.

Consideraremos Σ^* o conjunto de *strings* sobre Σ e Σ^+ o conjunto que não contém a *string* vazia. Dito isto, seja $x \in \Sigma^*$ uma *string* em Σ^*, onde $x =$

$w_1, \ldots, w_r$ com $w_i \in \Sigma$. Caso a *string* seja a *string* vazia, será denotada pelo símbolo ϵ. A máquina M é inicializada com uma *string* x e seu estado inicial é q_0.

A computação ocorre em passos e cada passo da execução corresponderá a uma instrução de δ. A máquina para quando ela está no estado s, lendo o símbolo $a \in \Sigma$ tal que não existe qualquer instrução da forma (2.6) em $\delta(M)$. Além disso, se o estado alcançado é um estado de F^+ dizemos que a máquina aceita a entrada; no caso do estado alcançado ser de F^- dizemos que a entrada foi rejeitada.

A máquina inicialmente encontra-se no estado q_0. A fita é inicializada por uma *string* finita $x \in (\Sigma - \{\sqcup\})^*$. Diremos que x é a entrada da máquina de Turing. O cursor está apontando para a primeira célula da fita que contém o símbolo que denotaremos por $\triangleright$.

Usaremos o conceito de **configuração** que contém a descrição completa da máquina no i-ésimo passo da computação que corresponde às seguintes informações: o estado atual da máquina, o endereço da célula que está sendo lida pela cabeça de leitura e a inscrição na fita após i passos. Denotaremos a i-ésima configuração por C_i. Daremos a seguir a definição formal para tal configuração.

Definição 2.3.3 (Configuração da MTD) *Uma configuração de M é uma tripla (q, w, u), onde $q \in Q$ é um estado, w e u são* strings *em Σ^*. A* string *w é a* string *ao lado esquerdo da cabeça de leitura incluindo o símbolo atual que está sendo lido e u é a* string *do lado direito da cabeça de leitura que pode inclusive ser a* string *vazia ϵ [Pap94].* □

Definição 2.3.4 (Transição da MTD) *Fixe uma máquina de Turing M. Dizemos que a configuração (q, w, u) atinge a configuração (q', w', u') em um passo, denotado por $(q, w, u) \overset{M}{\rightarrow} (q', w', u')$, se em um passo da máquina ela sai da configuração (q, w, u) para a configuração (q', w', u'). Primeiro, seja σ o último símbolo de w, e suponha que $\delta(q, \sigma) = (p, \rho, D)$. Então temos que $q' = p$. Temos três casos. Se $D = +1$, então w' é w com seu último símbolo (que era σ) subs-*

tituído por ρ, e o primeiro símbolo de u anexado a w' ($\sqcup$ se u era a string *vazia); u' é u com o primeiro símbolo removido (ou se u era a* string *vazia, u' continua sendo vazio). Se $D = -1$, então w' é w com σ omitido de seu final, e u' é u com ρ anexado no seu início. Finalmente, se $D = 0$, então w' é w com σ (no final) substituído por ρ, e $u' = u$. Uma vez definida a transição da máquina de Turing em um passo, podemos agora definir o* fecho transitivo da transição*, isto é, como que a configuração (q, w, u) vai para a configuração (q', w', u') em k passos que será denotado por $(q, w, u) \overset{M^k}{\rightarrow} (q', w', u')$, onde $k \geq 0$ é um inteiro, se existe uma configuração (q_i, w_i, u_i), $i = 1, \ldots, k+1$, tal que $(q_i, w_i, u_i) \overset{M}{\rightarrow} (q_{i+1}, w_{i+1}, u_{i+1})$ para $i = 1, \ldots, k$, $(q_1, w_1, u_1) = (q, w, u)$ e $(q_{k+1}, w_{k+1}, u_{k+1}) = (q', w', u')$. Finalmente, dizemos que a configuração (q, w, u) atinge a configuração (q', w', u'), denotado por $(q, w, u) \overset{M^*}{\rightarrow} (q', w', u')$, se existe $k \geq 0$ tal que $(q, w, u) \overset{M^k}{\rightarrow} (q', w', u')$ [Pap94].* □

Definição 2.3.5 (M decide $\mathcal{L}$) *Seja $\mathcal{L} \subset (\Sigma - \{\sqcup\})^*$ uma linguagem. Seja M uma máquina de Turing tal que, para toda* string *$x \in (\Sigma - \{\sqcup\})^*$, se $x \in \mathcal{L}$, então M para em um estado de F^+ começando com a entrada x, e, se $x \notin \mathcal{L}$, então M para no estado de rejeição de F^-. Então dizemos que M decide $\mathcal{L}$. Se $\mathcal{L}$ é decidida por alguma máquina M, então $\mathcal{L}$ é chamada de* **linguagem recursiva** *[Pap94].* □

Definição 2.3.6 (M aceita $\mathcal{L}$) *Dizemos que M aceita $\mathcal{L}$ se, para qualquer* string *$x \in (\Sigma - \{\sqcup\})^*$, se $x \in \mathcal{L}$, então M para em um estado de F^+; entretanto, se $x \notin \mathcal{L}$, então M não para. Se $\mathcal{L}$ é aceita por alguma máquina de Turing M, então $\mathcal{L}$ é chamada* **recursivamente enumerável** *[Pap94].* □

2.3.2 Máquina de Turing Não-determinística

Nesta seção definiremos a máquina de Turing Não-determinística [Pap94].

Definição 2.3.7 (Máquina de Turing Não-determinística (MTND)) *Uma máquina de Turing Não-determinística é uma sextupla* $N = (Q, \Sigma, q_0, F^+, F^-, \Delta)$ *tal como a máquina de Turing determinística.* Q, Σ *e* q_0 *são definidos como anteriormente. Refletindo o fato da máquina de Turing ser não-determinística, não existe uma única próxima ação definida, mas uma escolha entre várias próximas ações expressas pela relação* $\Delta : (Q \times \Sigma) \rightarrow \mathcal{P}(Q \times \Sigma \times \{-1, 0, +1\})$. *Isto é, para cada combinação de estado e símbolo, pode existir mais de um próximo passo associado [Pap94].* □

As configurações de uma máquina de Turing não-determinística são exatamente como as configurações de uma máquina de Turing determinística. Veja a Definição 2.3.8.

Definição 2.3.8 (Transição MTND) *Dizemos que a configuração* (q, w, u) *de uma máquina não-determinística* N *resulta na configuração* (q', w', u') *em um passo, denotado por* $(q, w, u) \overset{N}{\rightarrow} (q', w', u')$, *se existe uma regra em* Δ *que faz esta transição legal. Formalmente, existe um movimento* $(q', \rho, D) \in \Delta(q, \sigma)$ *tal que, se (a)* $D = +1$, w' *é* w *com seu último símbolo substituído por* ρ *(antes era* σ*), e o primeiro símbolo de* u *anexado a* w' *(*$\sqcup$ *se* u *era a* string *vazia), e* u' *é* u *com o primeiro símbolo removido (ou, se* u *era a* string *vazia,* u' *continua vazio); se (b)* $D = -1$, w' *é* w *com* σ *omitido do seu final, e* u' *é* u *com* ρ *anexado no início; finalmente, se (c)* $D = 0$, w' *é* w *com o final* σ *substituído por* ρ, *e* $u' = u$. *A definição de* $\overset{N^k}{\rightarrow}$ *e* $\overset{N^*}{\rightarrow}$ *é como na Definição 2.3.3 da MTD, mas* $\overset{N^k}{\rightarrow}$ *é uma relação [Pap94].* □

Usando a Definição 2.3.2 das instruções da máquina de Turing determinística, segue que, agora, uma configuração pode possuir mais de uma próxima configuração.

Uma entrada é aceita se existe alguma sequência de escolhas não-determinísticas que resulta em algum estado de F^+. Outras escolhas podem resultar em algum estado de F^-. A *string* é rejeitada somente se nenhuma das escolhas resulta no estado de aceitação [Pap94].

Definição 2.3.9 (N aceita uma entrada x) *Seja $\mathcal{L}$ uma linguagem e N uma MTND. Dizemos que N aceita uma entrada $x \in \Sigma^*$ se $(q_0, \triangleright, x) \overset{N^*}{\rightarrow} (q, w, u)$, onde $q \in F^+$ e $x, w, u \in \Sigma^*$. Note que a máquina de Turing não-determinística aceita uma entrada se existe uma computação que para sobre essa entrada em um estado de F^+ [LP97].* □

Definição 2.3.10 (N decide $\mathcal{L}$) *Seja $\mathcal{L}$ uma linguagem e N uma MTND. Dizemos que N decide $\mathcal{L}$ se para toda entrada $x \in \Sigma^*$, o seguinte é verdadeiro: $x \in \mathcal{L}$ se, e somente se, $(q_0, \triangleright, x) \overset{N^*}{\rightarrow} (q, w, u)$ para alguma* string *w e u e $q \in F^+$ [Pap94].* □

Definição 2.3.11 (N decide $\mathcal{L}$ em tempo $f(n)$) *Dizemos que uma MTND N decide uma linguagem $\mathcal{L}$ em tempo $f(n)$, onde f é uma função de inteiros não-negativos para inteiros não-negativos, se N decide $\mathcal{L}$ (Definição 2.3.10), e além disso, para qualquer entrada $x \in \Sigma^*$, se $(q_0, \triangleright, x) \overset{N^k}{\rightarrow} (q, u, w)$, então $k \leq f(|x|)$ [Pap94].* □

2.4 Complexidade Computacional

Nesta seção definiremos como serão medidos os recursos necessários para resolver qualquer entrada dos problemas propostos em uma única computação. A

performance dos algoritmos é medida pela quantidade de tempo e espaço requeridos sobre as entradas de tamanho n, quando estas quantidades são expressas como uma função de n. Dada uma função $T : \mathbb{N} \to \mathbb{N}$, diremos que uma máquina de Turing M terá limite de tempo T, se para toda entrada $x \in \Sigma^*$ aceita por M existe uma execução de M, iniciada com x, que tem número de passos no máximo $T(|x|)$, onde $|x|$ denota o tamanho de x. E para uma função $S : \mathbb{N} \to \mathbb{N}$ dizemos que M terá limite de espaço S para toda entrada $x \in \Sigma^*$ aceita por M existe uma execução de aceitação que executa usando no máximo $S(|x|)$ células para escrita antes de parar [EF99].

2.4.1 Classes

Nesta seção apresentaremos as definições das principais classes de complexidade.

Classe DTIME$(T(n))$

Definição 2.4.1 (Classe DTIME$(T(n))$**)** *Seja $T : \mathbb{N} \to \mathbb{N}$ uma função. Uma linguagem $\mathcal{L} \subseteq \Sigma^*$ está em* **DTIME**$(T(n))$ *se, e somente se, existe uma máquina de Turing determinística M (MTD) que executa em tempo $c \cdot T(n)$ para algum $c > 0$ e decide $\mathcal{L}$ [AB09].* □

Note que **DTIME**$(T(n))$ é uma classe de linguagens. A classe **DTIME**$(T(n))$ contém exatamente aquelas linguagens que podem ser decididas por uma máquina de Turing executando com limite de tempo $T(n)$. Chamaremos **DTIME**$(T(n))$ de **classe de complexidade**.

Uma classe importante é a classe **PTIME** ou classe **P** em que uma linguagem $\mathcal{L} \subseteq \Sigma^*$ está na classe P (tempo polinomial), se ela é decidida por uma máquina de Turing determinística que tem limite de tempo p para algum polinômio p. Veja a seguir a classe de complexidade PTIME.

$$\text{PTIME} \quad = \quad \bigcup_{c=1}^{\infty} \text{DTIME } (n^c). \tag{2.7}$$

Muitas vezes usamos no lugar de uma função T, nesse caso, não uma função em particular, mas uma família de funções, parametrizadas por um inteiro $c >$

0. A classe de complexidade denotada será a união de todas as classes de complexidade individuais, uma para cada valor de c. Mostraremos a seguir outras importantes classes similares a essa.

Um outro exemplo de classe de complexidade é a classe **EXPTIME** ou classe **EXP** em que uma linguagem $\mathcal{L} \subseteq \Sigma^*$ está na classe EXP (tempo exponencial), se ela é decidida por uma máquina de Turing determinística que tem limite de tempo 2^p para algum polinômio p. Veja a seguir a classe EXPTIME.

$$\text{EXPTIME} \quad = \quad \bigcup_{c=1}^{\infty} \text{DTIME } (2^{n^c}). \tag{2.8}$$

Classe NTIME$(T(n))$

O conjunto de linguagens decididas por uma máquina de Turing não-determinística com tempo T é denotado por **NTIME**$(T(n))$. Veja a seguir a definição.

Definição 2.4.2 (Classe NTIME$(T(n))$**)** *Seja $T : \mathbb{N} \to \mathbb{N}$ uma função. Uma linguagem $\mathcal{L} \in \Sigma^*$ está em* **NTIME**$(T(n))$ *se, e somente se, existe uma máquina de Turing não-determinística N (MTND) que executa em tempo $c \cdot T(n)$ para algum $c > 0$ e que decide $\mathcal{L}$ [AB09].* □

Uma importante classe de complexidade não-determinística é a classe NP que significa a união de todas as classes NTIME(n^c). Uma linguagem $\mathcal{L} \subseteq \Sigma^*$ está na classe **NPTIME** ou **NP** (tempo polinomial), se ela é decidida por uma máquina de Turing não-determinística que tem limite de tempo p para algum polinômio p. A seguir, a classe NPTIME.

$$\text{NPTIME} \quad = \quad \bigcup_{c=1}^{\infty} \text{NTIME } (n^c). \tag{2.9}$$

Um outro exemplo de classe é a classe **NEXPTIME** ou classe **NEXP** em que uma linguagem $\mathcal{L} \subseteq \Sigma^*$ está na classe NEXP (tempo exponencial), se ela é decidida por uma máquina de Turing não-determinística que tem limite de tempo 2^p para algum polinômio p. A seguir, a classe NEXPTIME.

$$\text{NEXPTIME} \quad = \quad \bigcup_{c=1}^{\infty} \text{NTIME } (2^{n^c}). \tag{2.10}$$

Classe DSPACE$(S(n))$

As classes de espaço geralmente são definidas usando máquinas de Turing com uma fita de entrada que é apenas para leitura e uma fita de saída. O espaço é medido em termos da quantidade de células utilizadas na fita de saída. O modelo de máquina Turing que definimos acima pode ser facilmente generalizado para o caso de máquinas multi-fitas. Para maiores detalhes veja [Pap94, AB09].

Definição 2.4.3 (Classe DSPACE$(S(n))$**)** *Seja* $S : \mathbb{N} \to \mathbb{N}$ *uma função. Seja* $\mathcal{L} \subseteq \Sigma^*$ *uma linguagem. Dizemos que* $\mathcal{L}$ *está na classe de complexidade de espaço* **DSPACE**$(S(n))$ *se existe uma máquina de Turing determinística* M *com fitas de entrada e saída que decide* $\mathcal{L}$ *(Definição 2.3.5) e opera com limite de espaço* $S(n)$ *[AB09].* □

Um exemplo de classe de espaço é a classe **PSPACE** em que a linguagem $\mathcal{L}$ é aceita por uma máquina de Turing determinística com limite de espaço p (espaço polinomial), onde p é um polinômio. Veja a seguir.

$$\text{PSPACE} \quad = \quad \bigcup_{k=1}^{\infty} \text{DSPACE}\ (n^k). \tag{2.11}$$

Outra importante classe de complexidade é a classe **L** em que a linguagem $\mathcal{L}$ é aceita por uma máquina de Turing determinística com limite de espaço $\log n$ (espaço logarítmico), expressa a seguir.

$$\text{L} \quad = \quad \text{DSPACE}\ (\log n). \tag{2.12}$$

Classe NSPACE$(S(n))$

Definição 2.4.4 (Classe NSPACE$(S(n))$**)** *Dizemos que* $\mathcal{L} \in$ **NSPACE**$(S(n))$ *se existe uma máquina de Turing não-determinística* N *(MTND) que decide* $\mathcal{L}$ *usando não mais que* $c \cdot S(n)$ *espaços não-brancos da fita para uma entrada de tamanho* n*, em suas escolhas não-determinísticas [AB09].* □

Um exemplo de classe é a classe de complexidade **NPSPACE**(n^k) em que a linguagem $\mathcal{L}$ é aceita por uma máquina de Turing não-determinística com limite de espaço p (espaço polinomial), onde p é um polinômio.

Em símbolos, temos,

$$\text{NPSPACE} \quad = \quad \bigcup_{k=1}^{\infty} \text{NSPACE } (n^k). \tag{2.13}$$

Similar a classe **L** definida para a função limite $\log n$ de execução da máquina de Turing determinística, temos a classe **NL**, isto é, a linguagem $\mathcal{L}$ é aceita por uma máquina de Turing não-determinística com limite de espaço $\log n$. A seguir veja a definição para **NL**.

$$\text{NL} \quad = \quad \text{NSPACE } (\log n). \tag{2.14}$$

A seguir, apresentaremos a definição da máquina de Turing alternante (MTA) a qual é uma generalização da máquina de Turing não-determinística.

Definição 2.4.5 (Máquina de Turing Alternante (MTA)) *Uma máquina de Turing Alternante é uma máquina de Turing não-determinística* $N = (Q, \Sigma, q_0, F^+, F^-, \Delta)$ *onde o conjunto de estados* Q *está particionado em dois conjuntos* $Q = Q_\forall \vee Q_\exists$. *Seja* x *uma entrada, e considere a árvore de computações de* N *sobre a entrada* x. *Cada nó nesta árvore é uma configuração da máquina e inclui a quantidade de passos da máquina. Defina, agora, recursivamente, iniciando nas folhas da árvore e indo até um subconjunto das configurações, chamado de configuração de aceitação, como segue: Primeiro, todas as configurações folhas com o estado* F^+ *são configurações de aceitação. Uma configuração* C *com o estado* $Q_\forall$ *é uma configuração de aceitação se, e somente se, todas as suas configurações sucessoras são configurações de aceitação. Uma configuração* C *no estado* $Q_\exists$ *é configuração de aceitação se, e somente*

se, no mínimo uma de suas configurações sucessoras é uma configuração de aceitação. Finalmente, dizemos que N aceita x se a configuração inicial é de aceitação. Dizemos que uma máquina alternante N decide uma linguagem $\mathcal{L}$ se N aceita todas as strings $x \in \mathcal{L}$ *e rejeita todas as* strings $x \notin \mathcal{L}$ *[Pap94].* □

Definição 2.4.6 (Classes ATIME$(T(n))$ e ASPACE$(S(n))$) *Seja* **ATIME**$(T(n))$ *a classe de todas as linguagens decididas por uma máquina de Turing alternante, tais que todas as computações para a entrada x param em no máximo $T(|x|)$ passos;* **ASPACE**$(S(n))$ *é a classe de todas as linguagens decididas por uma máquina de Turing alternante que usa não mais que $T(|x|)$ espaços sobre a entrada x. Finalmente, defina as seguintes classes de complexidade* **AP** $=$ ATIME(n^k) *e* **AL** $=$ ASPACE$(\log n)$ *[Pap94].* □

Definição 2.4.7 (Hierarquia Polinomial (PH)) *A hierarquia polinomial (***PH***) é a seguinte sequência de classes: Primeiro,* $\Sigma_0\mathrm{P} = \Pi_0\mathrm{P} = \mathrm{P}$*; e para todo $i > 0$, a classe* $\Sigma_i\mathrm{P}$ *corresponde aos problemas que podem ser decididos por uma máquina de Turing alternante em tempo polinomial com restrição de que cada computação compreende apenas uma quantidade constante de alternâncias entre estados existenciais e estados universais, começando com estado existencial. Analogamente, a classe* $\Pi_i\mathrm{P}$ *corresponde aos problemas que podem ser decididos por uma máquina de Turing alternante em tempo polinomial com restrição de que cada computação compreende apenas uma quantidade constante de alternâncias entre estados existenciais e estados universais, começando com estado universal.*

Definimos a hierarquia polinomial como sendo a classe

$$\mathrm{PH} \quad = \quad \bigcup_{i \geq 0} \Sigma_i \mathrm{P}. \tag{2.15}$$

□

A hierarquia polinomial PH será relacionada com a lógica RIFP como veremos no Capítulo 5.

2.5 Complexidade Descritiva

Nesta seção, apresentaremos os conceitos e as definições da Teoria da Complexidade Descritiva.

2.5.1 Complexidade da Lógica

Nesta seção, apresentaremos os conceitos básicos que estão envolvidos na relação entre expressividade da lógica e complexidade computacional [GKL$^+$07]. Os dois problemas centrais dessa área são:

- teste de satisfatibilidade;
- *model-checking*.

O **problema da satisfatibilidade** para uma lógica $\mathcal{L}$ (ver Definição 2.1.15) sobre um domínio $\mathcal{D}$ de estruturas é tomar fórmulas $\psi \in \mathcal{L}$ como entrada e responder se existe um modelo em $\mathcal{D}$ que é modelo de ψ [GKL$^+$07].

O **problema de *model-checking*** é, para uma lógica $\mathcal{L}$ e um domínio $\mathcal{D}$ de estruturas (finitas), perguntar se é o caso que $\mathfrak{A} \models \psi$, para uma dada estrutura $\mathfrak{A} \in \mathcal{D}$ e uma fórmula $\psi \in \mathcal{L}$. Um problema semelhante é o de **avaliar uma fórmula** (ou avaliar uma *query*). Definiremos *query* a seguir.

Definição 2.5.1 (k-*Query* e *Query* booleana) *Seja S um vocabulário e k um inteiro positivo.*

- *Uma* k-query *(ou* query *de aridade* k*) sobre uma classe* $\mathcal{K}$ *é um mapeamento* Q *com domínio* $\mathcal{K}$ *e tal que*

  - $Q(\mathfrak{A})$ *é uma relação de aridade* k *sobre* $\mathfrak{A}$*, para* $\mathfrak{A} \in \mathcal{K}$*;*

  - Q *é preservada sob isomorfismo, o que significa que se* $h : \mathfrak{A} \to \mathfrak{B}$ *é um isomorfismo, então* $Q(\mathfrak{B}) = h\big(Q(\mathfrak{A})\big)$.

- *Uma* query *booleana sobre uma classe* $\mathcal{K}$ *é um mapeamento* $Q : \mathcal{K} \to \{0, 1\}$ *que é preservado sob isomorfismo, i.e., se* $\mathfrak{A}$ *é isomórfico a* $\mathfrak{B}$*, então* $Q(\mathfrak{A}) = Q(\mathfrak{B})$*. Consequentemente,* Q *pode ser identificada com a subclasse* $\mathcal{K}' = \big\{\mathfrak{A} \in \mathcal{K} \mid Q(\mathfrak{A}) = 1\big\}$ *de* $\mathcal{K}$*.*

□

Query booleana é um problema de decisão definido por uma classe de estruturas usando o jargão de Banco de Dados. *Queries* são objetos matemáticos que formalizam o conceito de uma propriedade da estrutura e de elementos da estrutura. Esta formalização permite definir e estudar tais propriedade exprimíveis em alguma lógica.

Voltemos a análise do seguinte problema: dada uma estrutura $\mathfrak{A}$ e uma fórmula $\psi(\overline{x})$ (com variáveis livres em $\overline{x}$), computar a relação definida por ψ sobre $\mathfrak{A}$, isto é, o conjunto $\psi^{\mathfrak{A}} = \{\overline{a} \mid \mathfrak{A} \models \psi(\overline{a})\}$ (Definição 2.1.16). Claramente, o problema de avaliar para uma fórmula com k variáveis livre sobre uma estrutura com n elementos reduz a n^k problemas de *model-checking* [GKL$^+$07].

Observe que o problema de *model-checking* possui duas entradas: uma estrutura e uma fórmula. Podemos medir a complexidade em função das duas entradas, e isto é conhecido por **complexidade combinada** (*combined complexity*) do problema de *model-checking*. Se fixarmos a estrutura $\mathfrak{A}$, então o problema de *model-checking* para $\mathcal{L}$ é o de decidir $\mathrm{Th}_{\mathcal{L}}(\mathfrak{A}) = \{\psi \in \mathcal{L} \mid \mathfrak{A} \models \psi\}$ e a complexidade de decidir esse conjunto é chamada de **complexidade de expressão** (*expression complexity*). Um outro problema importante de *model-checking* surge quando fixamos uma fórmula ψ, avaliando a **classe de modelos**

de ψ em $\mathcal{D}$, $\mathrm{Mod}_{\mathcal{D}}(\psi) = \{\mathfrak{A} \in \mathcal{D} \mid \mathfrak{A} \models \psi\}$ (Definição 2.1.14). Essa complexidade é chamada de **complexidade de estrutura** ou **complexidade de dados** (*data complexity*) do problema de *model-checking* [GKL$^+$07].

Por trás da análise algorítmica dos problemas de lógica encontra-se a questão central da complexidade descritiva, e que trataremos nas seções seguintes. Estamos interessados na caracterização do poder expressivo das lógicas usando classes de complexidade.

2.5.2 Codificação de Estruturas em *Strings*

A maioria das classes de complexidade com as quais trabalharemos aqui são baseadas no modelo da **Máquina de Turing (MT)**, cujas entradas são *strings* ou palavras sobre um alfabeto finito, como foi dito na Seção 2.3. Como estamos interessados em medir a complexidade dos problemas sobre estruturas finitas em termos das estruturas finitas e das fórmulas, necessitaremos representar as estruturas como *strings* para serem usadas como entrada para a máquina de Turing. O mínimo necessário para codificar uma estrutura finita como uma *string* requer selecionar uma **representação ordenada da estrutura** (ver [Imm99]).

Definição 2.5.2 (Classe de S-estruturas Finitas e Ordenadas) *Para uma vocabulário S qualquer, escreveremos Fin(S) para a classe das S-estruturas finitas e Ord(S) para a classe de todas estruturas $(\mathfrak{A}, <)$, onde $\mathfrak{A} \in$ Fin(S) e $<$ é uma ordem linear sobre A $(\mathfrak{A} = (A, <))$ [GKL$^+$07].* □

Para qualquer estrutura $(\mathfrak{A}, <) \in \mathrm{Ord}(S)$ de cardinalidade n e para qualquer k, podemos identificar A^k com o conjunto $\{0, \ldots, n^k - 1\}$, associando a cada k-tupla com sua posição na ordem lexicográfica induzida por $<$ sobre A^k. Essas estruturas ordenadas podem ser codificadas como uma *string* binária de muitas maneiras. A escolha particular de uma codificação não é importante e basta que as seguintes condições sejam satisfeitas [GKL$^+$07].

Definição 2.5.3 (Boa codificação) *Uma codificação code : Ord(S) $\to \Sigma^*$ (sobre qualquer alfabeto finito Σ) é* **boa** *se ela identifica estruturas isomórficas*

(Definição 2.1.8), se seus valores são limitados polinomialmente, se ela é definível em primeira ordem, e se as fórmulas atômicas podem ser computadas em tempo polinomial. Formalmente isso significa satisfazer as seguintes condições:

1. $code(\mathfrak{A}, <) = code(\mathfrak{B}, <)$ *se, e somente se,* $(\mathfrak{A}, <) \cong (\mathfrak{B}, <)$.

2. $|code(\mathfrak{A}, <)| \leq p(|A|)$ *para algum polinômio* p.

3. *Para todo* $k \in \mathbb{N}$ *e todo símbolo* $\sigma \in \Sigma$, *existe uma fórmula de primeira ordem* $\beta_\sigma(x_1, \ldots, x_k)$ *do vocabulário* $S \cup \{<\}$ *tal que, para toda estrutura* $(\mathfrak{A}, <) \in Ord(S)$ *e para todo* $\overline{a} \in A^k$, *a seguinte equivalência vale:*

$$(\mathfrak{A}, <) \models \beta_\sigma(\overline{a}) \text{ sse o } \overline{a}\text{-ésimo símbolo de } code(\mathfrak{A}, <) \text{ é } \sigma \tag{2.16}$$

4. *Dado* $code(\mathfrak{A}, <)$, *um símbolo relacional* $R \in S$, *e uma tupla* $\overline{a}$, *pode decidir se* $\mathfrak{A} \models R\overline{a}$ *[GKL*$^+$*07].*

□

Um exemplo de codificação é dado a seguir. Seja $<$ uma ordem linear sobre A e seja $\mathfrak{A} = (A, R_1, \ldots, R_t)$ uma S-estrutura de cardinalidade n. Seja l a aridade máxima de $R_1, \ldots, R_t$. Com cada relação R de aridade j, associaremos $\chi(R) = w_0 \ldots w_{n^j-1} 0^{n^l - n^j} \in \{0,1\}^{n^l}$, onde $w_i = 1$ se a i-ésima tupla de A^j está em R, e $w_i = 0$, caso contrário. Agora, fazemos $code(\mathfrak{A}, <) = 1^n 0^{n^l - n} \chi(R_1) \ldots \chi(R_t)$ [GKL$^+$07].

Podemos fixar uma função de boa codificação (Definição 2.5.3) e entender estruturas ordenadas pelas suas codificações. Com uma estrutura $\mathfrak{A}$ sem ordem, associaremos os conjuntos de todas as codificações $code(\mathfrak{A}, <)$, onde $<$ é uma ordem linear sobre A. Logo, quando dizemos que um algoritmo M decide uma classe $\mathcal{K}$ de S-estruturas, significa que M decide o conjunto de codificações das

estruturas em $\mathcal{K}$, i.e., a linguagem

$$code(\mathcal{K}) = \{code(\mathfrak{A}, <) \mid \mathfrak{A} \in \mathcal{K} \text{ e } < \text{ é uma ordem linear sobre } A\} \quad (2.17)$$

[GKL$^+$07].

Logo, podemos perguntar se uma classe de estruturas é computável ou não, e qual a complexidade envolvida para realizar esta computação. Portanto, podemos considerar que as classes de complexidades, se for o caso, são definidas por *queries* pertencentes a alguma lógica. Apresentaremos na seção seguinte essa questão formalmente. Antes disso, apresentaremos a definição de *query* $\mathcal{L}$-definível.

Definição 2.5.4 (*Query* $\mathcal{L}$-definível) *Seja $\mathcal{L}$ uma lógica e $\mathcal{K}$ uma classe de S-estruturas.*

- *Uma k-*query *Q sobre $\mathcal{K}$ é $\mathcal{L}$-definível se existe uma $\mathcal{L}$-fórmula $\phi(x_1, \ldots, x_k)$ com $x_1, \ldots, x_k$ variáveis livre e tais que para toda $\mathfrak{A} \in \mathcal{K}$,*

$$\mathcal{Q}(\mathfrak{A}) = \big\{(a_1, \ldots, a_k) \in A^k \mid \mathfrak{A} \models \phi(a_1, \ldots, a_k)\big\}. \quad (2.18)$$

- *Uma* query *booleana Q sobre $\mathcal{K}$ é $\mathcal{L}$-definível se existe uma $\mathcal{L}$-sentença ψ tal que para toda $\mathfrak{A} \in \mathcal{K}$,*

$$Q(\mathfrak{A}) = 1 \Leftrightarrow \mathfrak{A} \models \psi. \quad (2.19)$$

- *$\mathcal{L}(\mathcal{K})$ denota a coleção de todas as* queries *$\mathcal{L}$-definíveis sobre $\mathcal{K}$.*

□

2.5.3 Capturando Classe de Complexidade

Nesta seção, buscamos apresentar a conexão entre complexidade e lógica que é o tema de pesquisa da área de Complexidade Descritiva. Dado um domínio $\mathcal{D}$ de estruturas (Definição 2.1.7), uma lógica $\mathcal{L}$ e o conceito de captura de uma classe de complexidade *Comp*, seguem-se as questões: (1) para toda sentença $\psi \in \mathcal{L}$, a complexidade de dados de avaliar ψ sobre as estruturas de $\mathcal{D}$ é um problema na classe de complexidade *Comp*, e (2) toda propriedade das estruturas em $\mathcal{D}$ que podem ser decididas com complexidade *Comp* são definíveis na lógica $\mathcal{L}$.

Apresentamos uma metodologia para fazer esta conexão que pode ser encontrada em [EF99, GKL$^+$07, Imm99]. A seguir, enunciaremos como é feita essa conexão entre classe de complexidade e lógica.

Quando nos referimos aqui a uma **classe de modelos** significa uma classe de estruturas $\mathcal{K}$ para um vocabulário S fixo que é fechada sob isomorfismo (Definições 2.1.8 e 2.1.14), ou seja, se $\mathfrak{A} \in \mathcal{K}$ e $\mathfrak{A} \cong \mathfrak{B}$, então $\mathfrak{B} \in \mathcal{K}$. Chamaremos de domínio da estrutura, se o vocabulário não estiver fixado. Para um domínio $\mathcal{D}$ e vocabulário S, escreveremos $\mathcal{D}(S)$ para a classe de S-estruturas em $\mathcal{D}$ (Definição 2.1.7) [GKL$^+$07].

Intuitivamente, uma lógica $\mathcal{L}$ *captura* uma classe de complexidade *Comp* sobre $\mathcal{D}$ se as propriedades $\mathcal{L}$-definíveis em $\mathcal{D}$ (Definição 2.5.4) são precisamente as que são decidíveis em *Comp*. Formalmente, isso se expressa na definição a seguir.

Definição 2.5.5 (Captura) *Seja $\mathcal{L}$ uma lógica, Comp uma classe de complexidade e $\mathcal{D}$ um domínio de estruturas finitas. Dizemos que $\mathcal{L}$* **captura** *Comp* **sobre** *$\mathcal{D}$ se*

- *$\mathcal{L} \subseteq$ Comp: Para um vocabulário qualquer S e uma sentença qualquer $\psi \in \mathcal{L}(S)$, o problema de complexidade de dados para ψ sobre $\mathcal{D}(S)$ está na classe de complexidade Comp.*

- *Comp $\subseteq \mathcal{L}$: Para uma classe de modelos qualquer $\mathcal{K} \in \mathcal{D}(S)$ cujo problema*

de decisão está em Comp, existe uma sentença $\psi \in \mathcal{L}(S)$ tal que

$$\mathcal{K} \quad = \quad \{\mathfrak{A} \in \mathcal{D}(S) \mid \mathfrak{A} \models \psi\} \tag{2.20}$$

[GKL$^+$07]. □

Algumas vezes escreveremos $\mathcal{L} \subseteq Comp$ para dizer que a condição (1) da Definição 2.5.5 é satisfeita sobre o domínio de todas as estruturas finitas. E $Comp \subseteq \mathcal{L}$ para a condição (2).

Temos dois importantes exemplos desse tipo de resultado. Um deles é o Teorema de Fagin que afirma que a lógica existencial de segunda ordem (Definição 2.5.7) captura a classe NP sobre a classe de todas as estruturas finitas, e o Teorema de Immerman-Vardi que afirma que a lógica de menor ponto fixo (Seção 3.2) captura a classe de complexidade P sobre a classe de todas as estruturas finitas e ordenadas.

A seguir, apresentaremos as provas para o Teorema de Fagin e para o Teorema Immerman-Vardi.

Prova de Fagin para ∃SO = NP

A prova original apresentada por Fagin está em [Fag74]. Apresentaremos a versão descrita em [GKL$^+$07].

Definição 2.5.6 (Espectro) *O espectro (*spectrum*) de uma sentença de primeira ordem ψ é o conjunto de cardinalidades de seus modelos finitos, isto é,*

$$\textit{spectrum}(\psi) \quad = \quad \{\, k \in \mathbb{N} : \psi \textit{ tem um modelo com } k \textit{ elementos}\}. \tag{2.21}$$

□

Definição 2.5.7 (Lógica Existencial de Segunda Ordem (∃SO)) *A lógica existencial de segunda ordem, denotada por ∃SO, é o conjunto de fórmulas*

da forma $\exists R_1 \ldots \exists R_m\, \phi$*, onde* $m \in \mathbb{N}$*,* $R_1, \ldots, R_m$ *são símbolos relacionais de qualquer aridade finita, e* ϕ *é uma fórmula de primeira ordem.* □

Note que o espectro de uma sentença de primeira ordem ψ do vocabulário relacional $S_0 = \{R_1, \ldots, R_m\}$ pode ser visto como o conjunto dos modelos finitos da sentença existencial de segunda ordem $\exists R_1 \ldots \exists R_m\, \psi$. Como todos os símbolos relacionais estão quantificados, esta é uma sentença sobre um vocabulário vazio, i.e., seus modelos são apenas conjuntos.

Definição 2.5.8 (Espectro generalizado) *Espectro generalizado é a classe de modelos finitos de uma sentença em lógica existencial de segunda ordem.* □

Teorema 2.5.1 (Teorema de Fagin) *Seja* $\mathcal{K}$ *uma classe de estruturas finitas de algum vocabulário fixo não-vazio e finito e que seja fechada sob isomorfismo. Então* $\mathcal{K}$ *está em NP se, e somente se,* $\mathcal{K}$ *é definível por uma sentença existencial de segunda ordem, isto é, se, e somente se,* $\mathcal{K}$ *é um espectro generalizado.*

Prova. Mostraremos como decidir um espectro generalizado. Seja

$$\psi \quad = \quad \exists R_1 \ldots \exists R_m\, \phi \tag{2.22}$$

uma sentença no fragmento existencial de segunda ordem. Descreveremos um algoritmo de tempo polinomial não-determinístico N que, dada uma codificação $code(\mathfrak{A}, <)$ de uma estrutura $\mathfrak{A}$, decide se $\mathfrak{A} \models \psi$. Primeiro, N não-deterministicamente adivinha as relações $R_1, \ldots, R_m$ sobre A. Uma relação R_i é determinada por uma *string* binária de tamanho n^{r_i}, onde r_i é a aridade da relação R_i e $n = |A|$. Assim, N decide se $(\mathfrak{A}, R_1, \ldots, R_m) \models \phi$. Como ϕ está em primeira ordem, ela pode ser computada em espaço logarítmico e portanto em tempo polinomial, conforme teorema FO $\subseteq$ L [Imm99].

Como a computação de N consiste em adivinhar um número polinomial de bits, segue uma computação tempo polinomial não-determinística. Claramente N decide a classe de modelos finitos de ψ.

No outro sentido da prova: seja $\mathcal{K}$ uma classe fechada sob isomorfismo de S-estruturas e seja N uma máquina de Turing não-determinística de uma fita,

que dada uma entrada $code(\mathfrak{A}, <)$, decide em tempo polinomial se $\mathfrak{A}$ está em $\mathcal{K}$. Construiremos uma sentença ϕ cujos modelos finitos são exatamente as estruturas em $\mathcal{K}$.

Seja $N = (Q, \Sigma, q_0, F^+, F^-, \Delta)$, onde Q é o conjunto de estados, Σ é o alfabeto de N, q_0 é o estado inicial, F^+ e F^- são os conjuntos de estados de aceitação e rejeição, respectivamente, $\Delta : (Q \times \Sigma) \to \mathcal{P}(Q \times \Sigma\{-1, 0, 1\})$ é a função de transição. Sem perda de generalidade, podemos assumir que toda computação de N atinge um estado de aceitação ou rejeição em no máximo $n^k - 1$ passos, onde n é a cardinalidade do domínio da estrutura $\mathfrak{A}$.

Representaremos uma computação de N para uma entrada $code(\mathfrak{A}, <)$ por uma tupla $\overline{X}$ de relações sobre A construindo uma sentença de primeira ordem ψ_N do vocabulário $S \cup \{<\} \cup \{\overline{X}\}$ tal que

$$(\mathfrak{A}, <, \overline{X}) \models \psi_N \quad sse \quad \begin{array}{l}\text{as relações } \overline{X} \text{ representam uma computação de} \\ \text{aceitação de } N \text{ sobre } code(\mathfrak{A}, <).\end{array} \tag{2.23}$$

Para representar os n^k parâmetros de tempo e espaço da computação identificaremos números até $n^k - 1$ com tuplas em A^k. Dada uma ordem linear, uma relação sucessor associada, o maior e o menor elementos, estão definidos. Note, que se uma relação sucessor Suc e as constantes "0" e "e" para o primeiro e último elemento estão disponíveis, segue que a relação de sucessor induzida será $\overline{y} = \overline{x} + 1$ sobre as k-tuplas é definida pela seguinte fórmula, que é livre de quantificadores

$$\bigvee_{i \leq k} \Big(\bigwedge_{j<i} (x_j = e \wedge y_j = 0) \wedge Suc\, x_i y_i \wedge \bigwedge_{j>i} x_j = y_j \Big) \tag{2.24}$$

Logo, para qualquer inteiro fixo m, a relação $\overline{y} = \overline{x} + m$ pode ser expressa em FO.

A descrição $\overline{X}$ de uma computação de N sobre a codificação $code(\mathfrak{A}, <)$ consiste das seguintes relações.

(1) Para cada estado $q \in Q$, o predicado

$$X_q \;=\; \big\{ \overline{t} \in A^k \mid \text{ no tempo } \overline{t}, N \text{ está no estado } q \big\}. \tag{2.25}$$

(2) Para cada símbolo $\sigma \in \Sigma$, o predicado

$$Y_\sigma \;=\; \big\{(\overline{t},\overline{a}) \in A^{2k} \mid \text{ no tempo } \overline{t},\ \text{a célula } \overline{a} \text{ contém o símbolo } \sigma\big\}. \tag{2.26}$$

(3) O predicado cabeça

$$Z \;=\; \big\{(\overline{t},\overline{a}) \in A^{2k} \mid \text{ no tempo } \overline{t},\ \text{a cabeça de } N \text{ está na posição } \overline{a}\big\}. \tag{2.27}$$

A sentença ψ_N é o fecho universal da conjunção

$$\mathit{START} \;\wedge\; \mathit{COMPUTE} \;\wedge\; \mathit{END}. \tag{2.28}$$

A subfórmula START garante a condição da configuração inicial de N no tempo $t = 0$ sobre a entrada $\mathit{code}(\mathfrak{A}, <)$. Veja a subfórmula a seguir:

$$\mathit{START} \;=\; X_{q_0}(\overline{0}) \wedge Z(\overline{0},\overline{0}) \wedge \bigwedge_{\sigma\in\Sigma} (\beta_\sigma(\overline{x}) \to Y_\sigma(\overline{0},\overline{x})) \tag{2.29}$$

onde β_σ é a fórmula da Definição 2.5.3.

A subfórmula $\mathit{COMPUTE}$ descreve as transições de uma configuração para a configuração seguinte enquanto ainda não atingiu o estado final da computação. Trata-se da conjunção das seguintes fórmulas $\mathit{NOCHANGE} \wedge \mathit{CHANGE}$.

$$\begin{aligned} \mathit{NOCHANGE} \;=\; \bigwedge_{\sigma\in\Sigma} \big(&Y_\sigma(\overline{t},\overline{x}) \wedge (\overline{y} \neq \overline{x}) \wedge (\overline{t}' = \overline{t}+1) \wedge Z(\overline{t},\overline{y}) \\ &\to Y_\sigma(\overline{t}',\overline{x})\big) \end{aligned} \tag{2.30}$$

e

$$\mathit{CHANGE} \;=\; \bigwedge_{q\in K,\ \sigma\in\Sigma} \big(\mathit{PRE}[q,\sigma] \to \bigvee \mathit{POST}[q',\sigma',m]\big) \tag{2.31}$$

onde

$$\mathit{PRE}[q,\sigma] \;=\; X_q(\overline{t}) \;\wedge\; Z(\overline{t},\overline{x}) \;\wedge\; Y_\sigma(\overline{t},\overline{x}) \;\wedge\; (\overline{t}' = \overline{t}+1) \tag{2.32}$$

$$POST[q', \sigma', m] \quad = \quad X_{q'}(\overline{t}') \wedge Y_{\sigma'}(\overline{t}', \overline{x}) \wedge \exists \overline{y} \left((\overline{x} + m = \overline{y}) \wedge Z(\overline{t}', \overline{y}) \right) \tag{2.33}$$

onde a expressão $\overline{x} + m = \overline{y}$, o m indica o tipo do movimento da cabeça de leitura. *NOCHANGE* expressa o fato que o conteúdo das células, que não são aquelas que estão sendo lidas, não devem mudar na configuração seguinte, enquanto *CHANGE* garante a mudança nas relações X_q, Y_σ e Z definidas pela função de transição.

Finalmente, a fórmula

$$END \quad = \quad \bigvee_{q \in F^+} X_q(\overline{t}) \tag{2.34}$$

que garante a aceitação. Note que todas as computações param.

∎

Lema 2.5.2 *Se N aceita code$(\mathfrak{A}, <)$, então $(\mathfrak{A}, <) \models (\exists \overline{X})\ \psi_N$.*

Prova. Isso segue imediatamente da construção de ψ_N, pois para qualquer computação aceita de N sobre *code*$(\mathfrak{A}, <)$ significa que $\overline{X}$ satisfaz ψ_N. ∎

Lema 2.5.3 *Se $(\mathfrak{A}, <, \overline{X}) \models \psi_N$, então N aceita code$(\mathfrak{A}, <)$.*

Prova. Suponha que $(\mathfrak{A}, <, \overline{X}) \models \psi_N$. Para qualquer N-configuração C no estado q, posição da cabeça de leitura na posição p, conteúdo da fita é $w_0 \ldots, w_{n^k-1}$ $\in \Sigma^*$, e para qualquer tempo $j < n^k$, seja $CONF[C, j]$ a conjunção das fórmulas atômicas que valem para C no tempo j, isto é,

$$CONF\ [C, j] \quad = \quad X_q(\overline{j}) \wedge X(\overline{j}, p) \wedge \bigwedge_{i=0}^{n^k-1} Y_{w_i}(\overline{j}, \overline{i}) \tag{2.35}$$

onde $\overline{j}$, $\overline{p}$ e $\overline{i}$ são tuplas de A^k representando os números j, p e i.

(i.) Seja C_0 a configuração inicial de N para a entrada *code*$(\mathfrak{A}, <)$. Como $(\mathfrak{A}, <, \overline{X}) \models START$, segue que,

$$(\mathfrak{A}, <, \overline{X}) \quad \models \quad CONF\ [C_0, 0]. \tag{2.36}$$

(ii.) Devido à subfórmula *COMPUTE* de ψ_N, toda configuração C não-terminal e todo $j < n^k - 1$, temos que,

$$\psi_N \ \wedge \ CONF\,[C,j] \ \models \bigvee_{C' \in \ Next\ (C)} CONF\,[C', j+1], \tag{2.37}$$

onde $Next(C) = \{C' \mid C \vdash_N C'\}$ é o conjunto das configurações sucessoras de C. Segue disso que existe uma computação

$$C_0(\mathfrak{A}, <) \ = \ C_0 \vdash_N C_1 \vdash_N \ldots \vdash_N C_{n^k-1} \ = \ C_{end} \tag{2.38}$$

de N sobre $code(\mathfrak{A}, <)$ tal que, para todo $j < n^k$,

$$(\mathfrak{A}, <, \overline{X}) \ \models \ CONF\,[C_j, j]. \tag{2.39}$$

(iii.) Desde que $(\mathfrak{A}, <) \ \models \ C_{end}$, a configuração C_{end} é de aceitação. Logo, N aceita $code(\mathfrak{A}, <)$.

Isso prova o Lema 2.5.3. Note que o fato de que uma relação binária é uma ordem linear pode ser expresso em lógica de primeira ordem. Portanto,

$$\mathfrak{A} \in \mathcal{K} \quad \text{sse} \quad \mathfrak{A} \models (\exists <)\,(\exists \overline{X})\,(\text{“} < \text{ é uma ordem linear”} \wedge \psi_N) \tag{2.40}$$

Isso prova que $\mathcal{K}$ é definível por uma sentença $\exists$SO. ∎

Outro resultado é o da classe P que pode ser capturada usando a lógica LFP (Seção 3.2) considerando-se as estruturas ordenadas e finitas (Teorema em [Imm99]).

Prova do Immerman-Vardi para FO ⊆ L

A seguir a prova para FO ⊆ L [Imm99].

Teorema 2.5.4 (FO ⊆ L) *O conjunto das* queries *booleanas de primeira ordem está contido no conjunto das* queries *computáveis em* logspace *denotado*

por FO ⊆ L [Imm99] (Definições 2.5.1 e 2.5.4).

Prova. Seja $\tau = \{R_1, \ldots, R_k, c_1, \ldots, c_l\}$. Seja $\phi \in \mathrm{FO}[\tau]$ uma fórmula

$$\phi \;=\; \exists x_1 \, \forall x_2 \; \ldots \; \mathcal{Q} x_m \, \alpha(\overline{x}) \tag{2.41}$$

onde α é uma fórmula livre de quantificadores e Q a *query* booleana definível correspondente, i.e., $Q(\mathfrak{A}) = 1$ se, e somente se, $\mathfrak{A} \models \phi$. Precisamos construir uma máquina de Turing logspace M tal que para toda estrutura $\mathfrak{A} \in \mathcal{D}[\tau]$, $\mathfrak{A}$ satisfaz ϕ *se, e somente se*, M aceita a codificação binária de $\mathfrak{A}$. Em símbolos,

$$\mathfrak{A} \models \phi \quad \Leftrightarrow \quad M \text{ aceita } \mathfrak{A} \tag{2.42}$$

Contruiremos a máquina de Turing M em logspace indutivamente sobre m que é o número de quantificadores que ocorre em ϕ. Se $k = 0$, então $\phi = \alpha$ é uma sentença livre de quantificadores. Logo, α é uma combinação de fórmulas atômicas finita. As fórmulas atômicas são as relações de entrada $Rp_1 \ldots p_r$, onde $p_i \in \{c_1, \ldots, c_l\}$. Uma vez que saibamos que a máquina M pode determinar se $\mathfrak{A}$ satisfaz cada uma das fórmulas atômicas, M pode determinar se $\mathfrak{A} \models \alpha$.

Uma máquina M em logspace recebe a entrada codificada em forma binária de $\mathfrak{A}$, para alguma $\mathfrak{A} \in \mathcal{D}[\tau]$. Para verificar um dos predicados de entrada, a máquina apenas verifica o bit apropriado na codificação da estrutura de entrada. Isso pode ser feito em espaço $O(\log n)$.

Indutivamente, assuma que as consultas de primeira ordem com $k-1$ quantificadores são feitas em logspace. Seja

$$\psi(x_1) \;=\; \forall x_2 \; \ldots \; \mathcal{Q}_k(x_k) \, \alpha(\overline{x}) \tag{2.43}$$

Seja M_0 uma máquina de Turing logspace que computa a consulta $\psi(c)$. Note que c é um novo símbolo de constante para substituir a variável livre x_1. Para computar a consulta $\phi = \exists x_1 \, \psi(x_1)$ construímos uma máquina logspace M que passa por todos os possíveis valores de x_1, substituindo cada um desses por c e executa M_0. Se algum desses valores levar M_0 ao estado de aceitação, então M encerra também no estado de aceitação, caso contrário, M encerra no estado de

rejeição. Note que o espaço extra necessário é apenas $\log n$ para armazenar os possíveis valores de x_1. A simulação do quantificador universal é semelhante.

∎

Capítulo 3

Lógicas de Ponto Fixo

Neste capítulo, apresentaremos as definições e os conceitos das lógicas de ponto fixo [GKL$^+$07, Imm99, Lib12, EF99, Kre02].

A lógica de primeira ordem não é capaz de expressar recursão. Isso levou alguns pesquisadores a buscarem mecanismos que, adicionados a FO, permitissem expressar definições recursivas. Um desses mecanismos são os operadores de pontos fixos, com o objetivo de aumentar a expressividade da lógica de primeira ordem. Outro exemplo de mecanismo que permite aumentar o poder expressivo da lógica são os quantificadores de segunda ordem. Fato este que foi apresentado no Teorema de Fagin (Teorema 2.5.3) que mostrou o exato ganho do poder expressivo que é alcançado quando um único quantificador existencial de segunda ordem é usado na forma de prefixo.

As lógicas de ponto fixo foram criadas para estender a linguagem de FO com construções sintáticas que denotam pontos fixos de operadores definidos na linguagem. A possibilidade de definir novas relações através do recurso de indução [Imm99] foi um dos resultados alcançados. Essas relações definidas de maneira indutiva não são exprimíveis em lógica de primeira ordem. Um exemplo de uma relação que não pode ser expressa em lógica de primeira ordem é o **fecho transitivo**. O fecho transitivo de um grafo é o conjunto dos pares de vértices entre os quais existe um caminho.

3.1 Teoria do Ponto Fixo

Nesta seção, definiremos alguns dos conceitos da Teoria de Ponto Fixo. Considere A um conjunto finito e não-vazio e $\mathcal{P}(A)$ o conjunto potência de A.

Definição 3.1.1 (Função monótona, ponto fixo e menor ponto fixo) *Seja A um conjunto e $F : \mathcal{P}(A) \to \mathcal{P}(A)$ uma função. F é chamada* **monótona** *se para todo $X \subseteq Y \subseteq A$, $F(X) \subseteq F(Y)$. Um* **ponto fixo de** *F é qualquer conjunto $P \subseteq A$ tal que $F(P) = P$. Um* **menor ponto fixo de** *F é um ponto fixo que está contido em qualquer outro ponto fixo, isto é, um ponto fixo X de F tal que $X \subseteq Y$ para todo ponto fixo Y de F. Denotaremos por $\mathbf{lfp}(F)$ o menor ponto fixo de F [Kre02, Lib12, GKL$^+$07].* □

Definição 3.1.2 (Fórmula Positiva) *Dizemos que $\phi(X)$ é* **positiva em** *X se toda ocorrência de X em $\phi(X)$ possui um número par de negações, onde X é símbolo relacional e ϕ é uma fórmula do vocabulário $S \cup \{X\}$ [Lib12, GKL$^+$07, EF99].* □

Seja $\phi(X, \overline{x})$ uma fórmula de alguma linguagem onde X é um símbolo relacional de aridade k e $\overline{x} = x_1, \ldots, x_k$. A fórmula ϕ define um operador sobre o domínio de uma interpretação $\mathfrak{I}$ como

$$\phi^{\mathfrak{I}}(\mathbf{X}) \;=\; \big\{\overline{a} \in A^k \;\mid\; \mathfrak{I}\frac{\mathbf{X}, \overline{a}}{X, \overline{x}} \;\models\; \phi(X, \overline{x})\big\}. \tag{3.1}$$

Como na Definição 2.1.16, o operador, nesse caso, também está definindo uma nova relação.

Lema 3.1.1 (Operador Monótono) *Se $\phi(X, \overline{x})$ é positiva em X, então $\phi^{\mathfrak{I}}$ é* **monotóno** *[Lib12].*

A prova do Lema 3.1.1 é por indução na estrutura da fórmula.

Definição 3.1.3 (Operador Indutivo) *Chamaremos um operador* $F : \mathcal{P}(A) \to \mathcal{P}(A)$ *indutivo se a sequência de estágios* X_α*, onde* α *é um inteiro, definida por*

$$\begin{aligned} X_0 &= \emptyset, \\ X_{\alpha+1} &= F(X_\alpha), \ e \end{aligned} \tag{3.2}$$

é crescente, isto é, se $X_\beta \subseteq X_\alpha$ *para todo* $\beta < \alpha$*. A sequência de estágios de um operador indutivo alcança um ponto fixo (Teorema 3.1.2), que geralmente é denotado por* X_∞ *ou* F_∞*.* □

Assim, se X ocorre positivamente em ϕ, então o operador $\phi^{\mathfrak{I}}$ é monótono (ver Lema 3.1.1), isso implicará que existirá uma relação *mínima* ou *menor* $X \subseteq A^k$ tal que $\phi^{\mathfrak{I}}(X) = X$.

A seguir o Teorema de Tarski-Knaster.

Teorema 3.1.2 (Tarski-Knaster) *Todo operador monótono* $F : \mathcal{P}(A) \to \mathcal{P}(A)$ *tem um menor ponto fixo* $\mathbf{lfp}(F)$ *que pode ser definido como*

$$\mathbf{lfp}(F) = \bigcap \{Y \mid F(Y) = Y\} \tag{3.3}$$

Além disso, $\mathbf{lfp}(F) = X_\infty = \bigcup_i X_i$ *para a sequência* X_i *(Definição 3.2) [Lib12].*

Os tipos de pontos fixos mais estudados são: menor ponto fixo (**lfp**) (Definição 3.1.1), ponto fixo inflacionário (**ifp**) e ponto fixo parcial (**pfp**).

Dizemos que o operador F é **inflacionário** se $X \subseteq F(X)$, para todo $X \in \mathcal{P}(A)$ e que o operador F é **monótono** se $X \subseteq Y$ implica $F(X) \subseteq F(Y)$, para todo $X, Y \in \mathcal{P}(A)$. Denotamos por F_∞ o estágio F_i que é um ponto fixo de F, se existir, e $\emptyset$ caso contrário. A sequência de estágios indutiva atinge um ponto fixo em algum estágio da sequência. Se F é inflacionário, chamamos de **ponto fixo inflacionário** de F e definimos como $\mathbf{inf}(F) = F_\infty$. No caso de F ser um operador arbitrário, chamamos F_∞ de **ponto fixo parcial** de F e denotamos por $\mathbf{pfp}(F)$.

3.2 Lógica de Menor Ponto Fixo (LFP)

Nesta seção vamos apresentar como aumentar a sintaxe da lógica de primeira ordem com o operador de menor ponto fixo.

A lógica de menor ponto fixo (LFP) é obtida estendendo a linguagem da FO com a seguinte regra:

- se X é uma variável relacional de aridade k, $\overline{x} = x_1, \ldots, x_k$ e $\phi(X, \overline{x})$ é uma LFP-fórmula positiva em X, então $\left[lfp_{X,\overline{x}}\phi(X, \overline{x})\right](\overline{t})$ é uma LFP-fórmula, onde $\overline{t} = t_1, \ldots, t_k$ é uma tupla de termos de comprimento k.

Dada uma interpretação $\mathfrak{I}$, a relação de satisfatibilidade para as novas fórmulas é definida

$$\mathfrak{I} \models \left[lfp_{X,\overline{x}}\, \phi(X, \overline{x})\right](\overline{t}) \quad \text{sse} \quad \left(t_1^{\mathfrak{I}}, \ldots, t_k^{\mathfrak{I}}\right) \in \mathbf{lfp}(\phi^{\mathfrak{I}}). \tag{3.4}$$

Lema 3.2.1 *Seja $F : \mathcal{P}(A^k) \to \mathcal{P}(A^k)$ um operador monótono sobre um conjunto finito A. Se F é computável em tempo polinomial (com respeito à $|A|$), então $\mathbf{lfp}(F)$ também o é [GKL$^+$07].*

Do fato que operações em primeira ordem serem computáveis em tempo polinomial e pelo Lema 3.2.1 podemos concluir que toda propriedade definível em LFP de estruturas finitas é computável em tempo polinomial [GKL$^+$07].

Restrita a estruturas ordenadas, LFP é equivalentemente IFP e definem exatamente as *queries* em PTIME ([Imm82, Var82]).

Teorema 3.2.2 (LFP = PTIME, sobre estruturas ordenadas) *Uma classe de estruturas ordenadas pode ser definida em* LFP *se, e somente se, está em* PTIME.

O papel da ordem é importante aqui, uma vez que LFP não pode definir consultas em PTIME sobre estruturas não-ordenadas. Um problema polinomial, como o de decidir se uma estrutura tem um domínio de cardinalidade par, não pode ser definido em LFP [KV96]. Sobre estruturas não-ordenadas, temos somente a inclusão de que LFP $\subseteq$ PTIME, mas não a recíproca.

3.3 Lógica de Ponto Fixo Inflacionário (IFP)

A lógica IFP é outra extensão para a lógica de primeira ordem com operador de ponto fixo inflacionário. Essa extensão permite a construção de um operador de ponto fixo que não é necessariamente monótono.

A lógica IFP é definida semelhante à lógica LFP, mas sem a restrição para que as variáveis de ponto fixo ocorram apenas positivamente na fórmula (Definição 3.1.2), e a semântica é definida por um operador inflacionário associado.

- se X é uma variável relacional de aridade k, $\overline{x} = x_1, \ldots, x_k$ e $\phi(X, \overline{x})$ é uma IFP-fórmula, então $\left[ifp_{X,\overline{x}}\ \phi(X, \overline{x}) \vee X\overline{x}\right](\overline{t})$ é uma IFP-fórmula, onde $\overline{t} = t_1, \ldots, t_k$ é uma tupla de termos de comprimento k.

A relação de satisfatibilidade é definida como:

$$\mathfrak{I} \models \left[ifp_{X,\overline{x}}\ \phi(X, \overline{x})\right](\overline{t}) \quad \text{sse} \quad \left(t_1^{\mathfrak{I}}, \ldots, t_k^{\mathfrak{I}}\right) \in \mathbf{ifp}(\phi^{\mathfrak{I}}) \tag{3.5}$$

As lógicas LFP e IFP têm o mesmo poder expressivo. Este resultado foi provado para estrutura finitas por Gurevich e Shelah em [GS86] e depois a restrição sobre a cardinalidade das estruturas foi removida por Kreutzer [Kre02].

Teorema 3.3.1 (Kreutzer) *Para toda* IFP*-fórmula, existe uma* LFP*-fórmula equivalente [GKL⁺07].*

Proposição 3.3.2 (Capturando PTIME com IFP) IFP *captura* PTIME *sobre estruturas finitas e ordenadas [GKL⁺07].*

3.4 Lógica de Ponto Fixo Parcial (PFP)

Outra importante lógica é a lógica de ponto fixo parcial (PFP):

- se X é uma variável relacional de aridade k, $\overline{x} = x_1, \ldots, x_k$ e $\phi(X, \overline{x})$ é uma PFP-fórmula, então $\left[pfp_{X,\overline{x}}\ \phi(X, \overline{x})\right](\overline{t})$ é uma PFP-fórmula, onde $\overline{t} = t_1, \ldots, t_k$ é uma tupla de termos de comprimento k.

E a relação de satisfatibilidade correspondente é:

$$\mathfrak{I} \models \left[pfp_{X,\overline{x}}\ \phi(X,\overline{x})\right](\overline{t}) \text{ sse } \left(t_1^{\mathfrak{I}},\ldots,t_k^{\mathfrak{I}}\right) \in \mathbf{pfp}(\phi^{\mathfrak{I}}).$$

Em [AV91, Var82], foi mostrado que PFP captura PSPACE, a classe dos problemas de decisão que podem ser resolvidos usando espaço polinomial, sobre estruturas ordenadas:

Teorema 3.4.1 (PFP = PSPACE, sobre estruturas ordenadas) *Uma classe de estruturas ordenadas pode ser definida em* PFP *se, e somente se, está em* PSPACE*.*

Muitos outros resultados de captura têm sido provados para várias lógicas e classes de complexidade, ver referência [Imm99] para tais resultados. Temos as seguintes relações entre as lógicas mencionadas acima:

$$\text{LFP} = \text{IFP} \subseteq \exists\text{SO} \subseteq \text{PFP},$$

onde a última inclusão somente vale para estruturas ordenadas. Queremos generalizar os operadores de ponto fixo considerados na teoria tradicional de ponto fixo a fim de explorar novas maneiras de definir lógicas visando a caracterização de seu poder expressivo através das classes de complexidade dentro da perspectiva da complexidade descritiva. Vamos introduzir a abordagem relacional no próximo capítulo.

Capítulo 4

Lógica de Ponto Fixo Relacional Inflacionário (RIFP)

Como foi visto no Capítulo 3, a teoria do ponto fixo tradicional é baseada no conceito de operador, que essencialmente é uma função (ver mais detalhes em [Lib12, GKL$^+$07]). Nossa proposta é considerar relações no lugar de funções. Assim, neste capítulo iremos apresentar as definições necessárias para definir nossa lógica que será a lógica de primeira ordem acrescida do operador relacional de ponto fixo inflacionário (**rifp**). Em seguida, apresentaremos alguns resultados teóricos de complexidade descritiva para essa nova lógica. Além disso, mostraremos como adicionar à lógica de primeira ordem a capacidade de definir relações que correspondem à união dos pontos fixos inflacionários iniciais das relações definidas. Essa extensão acrescentará poder expressivo maior que o da lógica de ponto fixo inflacionário.

4.1 Teoria do Ponto Fixo Relacional

Nesta seção apresentamos as definições e notações que usaremos posteriormente neste trabalho. Dado um domínio finito A, seja $\mathcal{R} \subseteq \mathcal{P}(A^k) \times \mathcal{P}(A^k)$ uma relação (de segunda ordem) com tuplas de tamanho $k \in \mathbb{N}$.

Podemos definir condições sobre as relações que se assemelham àquelas usadas para definir os tipos de operadores de ponto fixo como visto antes.

Definição 4.1.1 (Relação Total) *Seja A um conjunto finito, não-vazio e $\mathcal{P}(A^k)$ o conjunto das partes do produto cartesiano A^k. Dizemos que uma relação binária $\mathcal{R}$ é* **total** *se para todo $X \in \mathcal{P}(A^k)$ existe um $Y \in \mathcal{P}(A^k)$ tal que $(X, Y) \in \mathcal{R}$. Para $X \in \mathcal{P}(A^k)$ escreveremos $\mathcal{R}(X) = \{Y \in \mathcal{P}(A^k) \mid \mathcal{R}(X, Y)\}$ para denotar os sucessores de X.* □

Definição 4.1.2 (Relação Inflacionária) *Dizemos que uma relação $\mathcal{R} \subseteq \mathcal{P}(A^k) \times \mathcal{P}(A^k)$ é uma relação* ***inflacionária*** *se, para todo $X \in \mathcal{P}(A^k)$ tal que $\mathcal{R}(X) \neq \emptyset$, para todo $Y \in \mathcal{R}(X)$, temos $X \subseteq Y$.* □

Definição 4.1.3 (Cadeia) *Uma* cadeia *em $\mathcal{R}$ é uma sequência $X_0, X_1, \ldots, X_m$ tal que $X_0 = \emptyset$ e $X_i \subseteq X_{i+1}$, $0 \leq i \leq m-1$.*

Definição 4.1.4 (Relação Inflacionária $\mathcal{R}_{\mathbf{Inf}}$) *Dada uma relação $\mathcal{R} \subseteq \mathcal{P}(A^k) \times \mathcal{P}(A^k)$, definimos uma relação total, inflacionária, $\mathcal{R}_{\text{INF}}$ a partir de $\mathcal{R}$ como segue:*

$$\begin{aligned} \mathcal{R}_{\text{INF}} \quad = \quad & \{(X, X \cup Y) \in \mathcal{P}(A^k)^2 \mid (X, Y) \in \mathcal{R}\} \cup \\ & \{(X, X) \in \mathcal{P}(A^k)^2 \mid \nexists Y : (X, Y) \in \mathcal{R}\}. \end{aligned} \tag{4.1}$$

Definição 4.1.5 (Relação Indutiva) *Dizemos que uma relação* $\mathcal{R} \subseteq \mathcal{P}(A^k) \times \mathcal{P}(A^k)$ *é* **indutiva** *se qualquer sequência* $X_0, X_1, \ldots, X_m$, *tal que* $X_0 = \emptyset$ *e* $(X_i, X_{i+1}) \in \mathcal{R}$ *é uma* cadeia, $0 \leq i \leq m-1$. □

Lema 4.1.1 *Se* $\mathcal{R}$ *é inflacionária então é indutiva.*

Definição 4.1.6 (Ponto Fixo) *Um conjunto* $X \in \mathcal{P}(A^k)$ *é um* **ponto fixo da relação** $\mathcal{R}$ *se* $\mathcal{R}(X, X)$. □

Definição 4.1.7 (Ponto Fixo Indutivo - IndFP($\mathcal{R}$)) *Dizemos que* X *é um* **ponto fixo indutivo de** $\mathcal{R}$ *se* X *é um ponto fixo de* $\mathcal{R}$ *e existe uma* cadeia $X_0, X_1, \ldots, X_m$ *tal que* $X_m = X$. *Denotaremos por* $\textsc{IndFP}(\mathcal{R})$ *o conjunto dos pontos fixos indutivos de* $\mathcal{R}$. □

Definição 4.1.8 (Ponto Fixo Inicial - IniFP($\mathcal{R}$)) *Dizemos que um ponto fixo* X *é um* **ponto fixo inicial** *de* $\mathcal{R}$ *se existe uma* cadeia $X_0, X_1, \ldots, X_m$ *tal que* $X_m = X$ *e nenhum* X_j, $j < m$, *é um ponto fixo de* $\mathcal{R}$. *Denotamos por* $\textsc{IniFP}(\mathcal{R})$ *o conjunto de todos os pontos fixos iniciais de* $\mathcal{R}$. □

Na Figura 4.1 note que os pontos fixos iniciais são X_3 e Z_1.

Definição 4.1.9 (Ponto Fixo Inflacionário - InfFP($\mathcal{R}$)) *Um conjunto* X *é um* **ponto fixo inflacionário de** $\mathcal{R}$ *se* X *é um ponto fixo indutivo de* $\mathcal{R}_{\textsc{Inf}}$. □

Definição 4.1.10 (Ponto Fixo Inflacionário Inicial - rifp($\mathcal{R}$)) *Dizemos que* X *é um* **ponto fixo inflacionário inicial de** $\mathcal{R}$ *se* X *é um ponto fixo inicial de* $\mathcal{R}_{\textsc{Inf}}$. *Denotamos por* **rifp**($\mathcal{R}$) *o conjunto de todos os pontos fixos inflacionários iniciais de* $\mathcal{R}$, **rifp**($\mathcal{R}$) = $\textsc{IniFP}(\mathcal{R}_{\textsc{Inf}})$. □

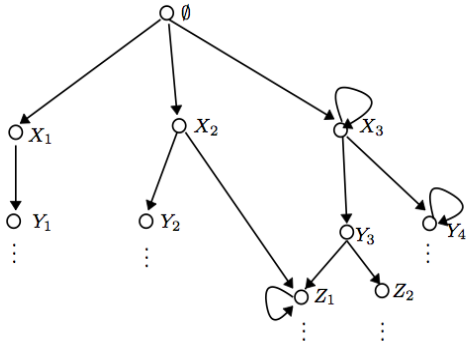

Figura 4.1: Ponto Fixo Inicial

Lema 4.1.2 *Se uma relação* $\mathcal{R} \subseteq \mathcal{P}(A^k) \times \mathcal{P}(A^k)$ *é total e indutiva, então qualquer* cadeia $X_0 = \emptyset, X_1, \ldots, X_m$ *para um* m *suficientemente grande contém um ponto fixo.*

É fácil ver que os operadores são casos particulares de relações. De fato, dado um operador $F : \mathcal{P}(A^k) \to \mathcal{R}(A^k)$, seu grafo é a relação

$$\mathcal{R}_F \quad = \quad \big\{(X, Y) \in \mathcal{P}(A^k) \times \mathcal{P}(A^k) \mid Y = F(X)\big\} \tag{4.2}$$

Assim, a abordagem relacional pode ser usada para representar a funcional.

Desenvolveremos uma lógica que é capaz de expressar o ponto fixo inflacionário inicial de uma relação.

4.2 A Lógica do Ponto Fixo Relacional RIFP

As lógicas de ponto fixo apresentadas no Capítulo 3 usaram sua própria linguagem para definir os operadores e prover expressões que se referem ao ponto fixo de tais operadores. A lógica de ponto fixo relacional fará o mesmo com respeito as relações do tipo $\mathcal{R}^{\phi,\mathfrak{I}}$ que apresentaremos na Definição 4.2.1 para alguma fórmula ϕ. Aqui, focaremos sobre uma lógica específica, a Lógica de Ponto Fixo Relacional Inflacionária (RIFP).

Vamos agora definir a relação $\mathcal{R}^{\phi,\mathfrak{I}}$.

Definição 4.2.1 ($\mathcal{R}^{\phi,\mathfrak{I}}$) *Seja $\phi(X,Y)$ uma S-fórmula com as variáveis relacionais livres X e Y de aridade k, e $\mathfrak{I}$ uma S-interpretação. A fórmula $\phi(X,Y)$ define a relação*

$$\mathcal{R}^{\phi,\mathfrak{I}} \;=\; \big\{(V_1,V_2)\in\mathcal{P}(A^k)\times\mathcal{P}(A^k) \;\mid\; \mathfrak{I}\frac{V_1,V_2}{X,Y} \;\models\; \phi(X,Y)\big\}. \tag{4.3}$$

Se $\phi(X,Y)$ não possui outras variáveis livres além de X e de Y, então, dada uma estrutura $\mathfrak{A}$, a relação $\mathcal{R}^{\phi,\mathfrak{I}}$ é a mesma para todas as interpretações $\mathfrak{I}$ sobre $\mathfrak{A}$. Portanto, escrevemos $\mathcal{R}^{\phi,\mathfrak{A}}$ no lugar. □

Agora, apresentamos a sintaxe e a semântica da lógica RIFP.

Definição 4.2.2 (Sintaxe de RIFP) *A linguagem de* RIFP *estende a linguagem da lógica de primeira ordem com a seguinte regra:*

- *se X e Y são variáveis relacionais de aridade k, $\bar{t} = t_1,\ldots,t_k$ são termos e $\phi(X,Y)$ é uma S-fórmula, então*

$$\big[\mathit{rifp}_{X,Y}\phi(X,Y)\big]\,(\bar{t}) \tag{4.4}$$

é uma S-fórmula de RIFP. □

Definição 4.2.3 (Semântica de RIFP) *A relação de satisfatibilidade é definida usando a relação inflacionária $\mathcal{R}^{\phi,\mathfrak{I}}_{\text{INF}}$ gerada por $\mathcal{R}^{\phi,\mathfrak{I}}$. Dada uma interpretação $\mathfrak{I}$ definimos*

$$\mathfrak{I} \models \big[\mathit{rifp}_{X,Y}\phi(X,Y)\big]\,(\bar{t}) \quad \mathit{sse} \quad (t_1^{\mathfrak{I}},\ldots,t_k^{\mathfrak{I}}) \;\in\; \bigcup \mathbf{rifp}(\mathcal{R}^{\phi,\mathfrak{I}}) \;=\; \bigcup \textsc{IniFP}(\mathcal{R}^{\phi,\mathfrak{I}}_{\text{INF}}) \tag{4.5}$$

Como mencionamos no final da Seção 4.1, podemos usar relações para representar funções. Como consequência, segue o seguinte teorema relacionando as lógicas IFP e RIFP.

Teorema 4.2.1 (IFP $\subset$ RIFP) *Toda* IFP*-fórmula tem uma fórmula equivalente em* RIFP*.*

Prova. Seja ψ a seguinte IFP-fórmula:

$$\psi \;=\; \left[ifp_{X,\overline{x}}\,\phi(X,\overline{x})\right](\overline{t}). \tag{4.6}$$

Seja $\mathfrak{I}$ uma interpretação e $\phi^{\mathfrak{I}}$ o operador definido por ϕ sobre $\mathfrak{I}$. Seja

$$\phi^{\mathfrak{I}}_0 \;=\; \emptyset,\; \phi^{\mathfrak{I}}_1,\; \phi^{\mathfrak{I}}_2,\; \ldots \tag{4.7}$$

a sequência de estágios de $\phi^{\mathfrak{I}}$. Definiremos uma fórmula $\theta(X,Y)$ tal que somente uma *cadeia* em $\mathcal{R}^{\theta,\mathfrak{I}}$ levará ao ponto fixo inicial que coincide com os estágios da sequência de $\phi^{\mathfrak{I}}$. Seja $\theta(X,Y)$ a fórmula

$$\theta(X,Y) \quad = \quad \forall\overline{x}(Y\overline{x} \leftrightarrow \phi(X,\overline{x})) \tag{4.8}$$

A relação $\mathcal{R}^{\theta,\mathfrak{I}}$ é exatamente o grafo do operador $\phi^{\mathfrak{I}}$. Portanto, qualquer segmento inicial da sequência de estágios de $\phi^{\mathfrak{I}}$ é *cadeia* de $\mathcal{R}^{\theta,\mathfrak{I}}$ e vice-versa. Além disso, como $\phi^{\mathfrak{I}}$ é inflacionário, então $\mathcal{R}^{\theta,\mathfrak{I}} = \mathcal{R}^{\theta,\mathfrak{I}}_{\text{INF}}$. Segue que $\mathcal{R}^{\theta,\mathfrak{I}}_{\text{INF}}$ tem apenas um ponto fixo inicial e é igual ao ponto fixo inflacionário de $\phi^{\mathfrak{I}}$. Portanto, a fórmula

$$\psi' \;=\; \left[rifp_{X,Y}\theta(X,Y)\right](\overline{t}) \tag{4.9}$$

é equivalente a ψ. ∎

4.3 Exemplos em RIFP

A seguir, mostraremos como usar RIFP para expressar algumas *queries*.

Apresentamos a seguir *query* EVEN (ver Definição 2.5.4) definível em RIFP mas que não é definível em FO sobre a classe de todas as estruturas de ordens lineares e finitas.

Exemplo 4.3.1 (Cardinalidade Par) *A* query Even Cardinality *ou* ***Cardinalidade Par****, a qual denotaremos por* EVEN*, corresponde à classe das estruturas cujo domínio tem cardinalidade par. Note que um conjunto A finito tem cardinalidade par se existe uma partição $\{A', A''\}$ de A e uma bijeção $f : A' \to A''$. Usaremos o operador* **rifp** *para construir tal bijeção (o grafo da bijeção, que é uma relação binária). Seja Y e X variáveis relacionais binárias e considere as seguintes fórmulas:*

1. *A fórmula FUNC expressa que uma dada relação Y é uma função*

$$FUNC(Y) \;=\; \forall x \forall y \forall z \big[Yxy \wedge Yxz \to y = z \big] \tag{4.10}$$

2. *A fórmula INJ expressa que uma dada relação Y é uma função injetiva*

$$INJ(Y) \;=\; \forall x \forall y \forall z \big[Yxy \wedge Yzy \to x = z \big] \tag{4.11}$$

3. *A fórmula TOT expressa que cada elemento do domínio está associado a algum outro elemento do domínio.*

$$TOT(Y) = \forall x \exists y \big[Yxy \vee Yyx \big] \tag{4.12}$$

4. *A fórmula DIS expressa que nenhum elemento aparece na imagem e no*

domínio da bijeção ao mesmo tempo.

$$DIS(Y) = \forall x \neg \big[\exists y Y x y \wedge \exists z Y z x\big] \tag{4.13}$$

Seja $\phi(X, Y)$ *a fórmula*

$$\phi(X,Y) \;=\; (X = \emptyset) \;\wedge\; FUNC(Y) \;\wedge INJ(Y) \;\wedge\; TOT(Y) \;\wedge\; DIS(Y). \tag{4.14}$$

Seja $\mathfrak{A}$ *uma estrutura e* $\mathcal{R}^{\phi,\mathfrak{A}}$ *a relação definida por* $\phi(X, Y)$ *sobre* $\mathfrak{A}$*. Um par* (B, B') *está em* $\mathcal{R}^{\phi,\mathfrak{A}}$ *sse* $B = \emptyset$ *e* B' *é o grafo de uma bijeção sobre uma partição do domínio* A *de* $\mathfrak{A}$*. Tal bijeção existe se e somente se* $\mathfrak{A}$ *tem cardinalidade par. A relação* $\mathcal{R}^{\phi,\mathfrak{A}}_{\text{INF}}$ *contem* $\mathcal{R}^{\phi,\mathfrak{A}}$ *e os pares* (B, B) *para todo* $B \subseteq A^2$*, tal que* $B \neq \emptyset$*. Se* A *tem cardinalidade ímpar, então qualquer* cadeia *sobre* $\mathcal{R}^{\phi,\mathfrak{A}}_{\text{INF}}$ *tem comprimento* 1 *e o único ponto fixo inflacionário inicial é* $\emptyset$*. Se* A *tem cardinalidade par, então qualquer* cadeia *de tamanho* 2 *alcança um ponto fixo inflacionário inicial que é o grafo de uma função bijetiva, como descrito acima. Segue que* $\bigcup \mathbf{rifp}(\mathcal{R}^{\phi,\mathfrak{A}})$ *é vazio se, e somente se,* A *tem cardinalidade ímpar. A sentença*

$$\psi_{\text{EVEN}} \;=\; \exists u\, \exists v\, \left[rifp_{X,Y}\phi(X,Y)\right](u,v) \tag{4.15}$$

é satisfeita por $\mathfrak{A}$ *se, e somente se,* $\bigcup \mathbf{rifp}(\mathcal{R}^{\phi,\mathfrak{A}})$ *é não-vazio, o que somente ocorre se* A *tem cardinalidade par. Portanto,* ψ_{EVEN} *define a* query EVEN*.*

O problema SAT, que definiremos a seguir, é completo para a classe NP.

Além disso, neste próximo exemplo, note como é o modo com que fazemos para defini-lo usando a lógica RIFP. A fórmula que exprime SAT descreve uma relação que codifica uma valoração para os literais, cláusulas e por fim a fórmula que é a instância do SAT (veja o esquema na Figura 4.2).

Exemplo 4.3.2 (Satisfatibilidade) *O problema* Satisfiability *ou* ***Satisfatibilidade****, o qual denotaremos por* SAT*, consiste do conjunto de fórmulas da lógica proposicional na forma normal conjuntiva (CNF) que são satisfatíveis. Uma fórmula α em CNF tem a forma*

$$\alpha = C_1 \wedge \ldots \wedge C_m, \tag{4.16}$$

onde cada C_i é uma cláusula, isto é, uma disjunção como

$$C_i = l_1 \vee \ldots \vee l_s \tag{4.17}$$

e cada l_j é um literal, isto é, ou é um símbolo proposicional p_k, e dizemos que p_k ocorre positivo em C_i, ou um símbolo proposicional negado $\neg p_k$, e dizemos que p_k ocorre negativo em C_i. Observe na Figura 4.2 o esquema da valoração para os símbolos proposicionais, em seguida, os valores verdades para as cláusulas e, finalmente, a valoração para a fórmula.

Como queremos usar uma fórmula para expressar o problema de decisão, temos que representar as entradas do problema como estruturas finitas. Seja $\mathfrak{A}_\alpha$ uma estrutura sobre o conjunto de símbolos $S = \{P, N\}$ onde P e N são relações binárias. Seja r o número de símbolos proposicionais que ocorrem em

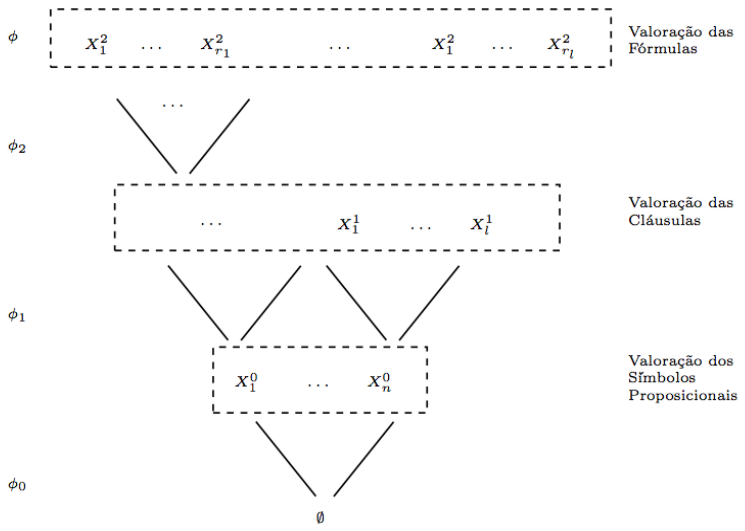

Figura 4.2: SAT

α *e* $A = \{1, \ldots, max\{m, r\}\}$. *Seja* $P^{\mathfrak{A}}$ *e* $N^{\mathfrak{A}}$ *definidos como:*

$$P^{\mathfrak{A}} = \{(i,j) \mid p_j \text{ ocorre positivo em } C_i\} \tag{4.18}$$

$$N^{\mathfrak{A}} = \{(i,j) \mid p_j \text{ ocorre negativo em } C_i\}. \tag{4.19}$$

A primeira posição de P *e* N *representa uma cláusula e a segunda posição um símbolo proposicional. Sem perda de generalidade, consideramos que qualquer fórmula tem pelo menos tantas cláusulas quantos símbolos proposicionais, o que pode ser obtido adicionando cláusulas do tipo* $(p_0 \vee \neg p_0)$*, o que não muda a satisfatibilidade da fórmula.*

Nós escreveremos uma RIFP*-fórmula que satisfaz estas estruturas* $\mathfrak{A}_\alpha$ *tal que* α *é satisfatível. Usaremos o operador* **rifp** *para construir uma relação*

ternária cujas tuplas representam os valores verdade dos símbolos proposicionais, cláusulas e a fórmula de acordo com alguma valoração dos símbolos proposicionais. Usamos triplas (a, b, c) *tal que a pode assumir os valores* 0, 1 *ou* 2*, caso se refira a um símbolo proposicional, uma cláusula ou uma fórmula, respectivamente, b é um elemento do domínio que representa o índice de uma cláusula ou símbolo proposicional e c pode ser* 0 *ou* 1*, para verdadeiro ou falso. Esta relação será construída em três estágios. Primeiro, colocamos as triplas que representam uma valoração. Seja* $PVAL(Y)$ *a fórmula*

$$PVAL(Y) = \forall x \forall y \forall z \left(Yxyz \rightarrow x = 0\right) \wedge \forall y \left((Y0y0 \vee Y0y1) \wedge \neg(Y0y0 \wedge Y0y1)\right) \tag{4.20}$$

e $\phi_0(X, Y)$ *a fórmula*

$$\phi_0(X, Y) = (X = \emptyset) \wedge PVAL(Y). \tag{4.21}$$

Duas relações $\mathbf{X}, \mathbf{Y}$ *satisfazem* $\phi_0(X, Y)$ *sse* $\mathbf{X} = \emptyset$ *e* $\mathbf{Y}$ *representa uma valoração, onde* $(0, i, 1)$ *significa que o símbolo proposicional i tem valor verdadeiro, e* $(0, i, 0)$ *significa que o símbolo proposicional i tem valor falso. No segundo passo, incluímos as triplas correspondentes aos valores verdade das cláusulas. Podemos considerar que* X *contém aquelas triplas que representam uma valoração e foram incluídas no primeiro passo. Seja* $CVAL(Y)$ *a fórmula*

$$CVAL(Y) = \forall x \forall y \forall z \left(Yxyz \rightarrow x = 1\right) \wedge \forall y \left((Y1y0 \vee Y1y1) \wedge \neg(Y1y0 \wedge Y1y1)\right) \tag{4.22}$$

e a fórmula $CL(X,Y)$

$$\begin{aligned} CL(X,Y) \;\; = \;\; & CVAL(Y) \wedge \\ & \forall y \left(Y1y1 \leftrightarrow \exists p \left((Pyp \wedge X0p1) \;\vee\; (Nyp \wedge X0p0) \right) \right) \end{aligned} \tag{4.23}$$

Seja $\phi_1(X,Y)$ *a fórmula*

$$\phi_1(X,Y) \;=\; PVAL(X) \;\wedge\; CL(X,Y). \tag{4.24}$$

Se $\mathbf{X}$ *e* $\mathbf{Y}$ *satisfazem* $\phi_1(X,Y)$ *então* $\mathbf{X}$ *corresponde a uma valoração de símbolos proposicionais. Além disso, a fórmula* CL *afirma que uma tripla* $(1,y,1)$ *está em* $\mathbf{Y}$ *se existe um símbolo proposicional* p *que ocorre positivo na cláusula* y *e tem valor verdadeiro de acordo com* $\mathbf{X}$ *ou* p *ocorre negativo na cláusula* y *e tem valor falso de acordo com* $\mathbf{X}$*. Dessa forma,* $\mathbf{Y}$ *representa os valores verdade das cláusulas de acordo com a valoração* $\mathbf{X}$*.*

O último passo calcula o valor verdade da fórmula baseada nos valores verdade de suas cláusulas. Seja $FVAL(Y)$ *a fórmula*

$$FVAL(Y) \;=\; \big((Y200 \vee Y201) \;\wedge\; \neg(Y200 \wedge Y201) \big). \tag{4.25}$$

A fórmula $FVAL(Y)$ *diz que* Y *ou tem a tupla* $(2,0,0)$*, indicando que a entrada do SAT não é satisfatível, ou a tupla* $(2,0,1)$ *caso contrário. Seja* $FOR(X,Y)$ *a fórmula*

$$FOR(X,Y) \;=\; Y201 \;\leftrightarrow\; \big(\forall y \, X1y1 \big) \tag{4.26}$$

e

$$\phi_2(X,Y) \;=\; FVAL(Y) \;\wedge\; FOR(X,Y). \tag{4.27}$$

A fórmula $FOR(X,Y)$ *afirma que a tripla* $(2,0,1)$ *está em* $\mathbf{Y}$ *se todas as cláusulas tiverem valor verdadeiro de acordo com* $\mathbf{X}$.

Finalmente, seja $\phi(X,Y) = \phi_0(X,Y) \vee \phi_1(X,Y) \vee \phi_2(X,Y)$. *Um par* $(\mathbf{X},\mathbf{Y})$ *satisfaz* $\phi(X,Y)$ *se, e somente se, satisfaz algum* $\phi_i(X,Y)$, $0 \leq i \leq 2$. *Em particular,* $(2,0,1) \in \mathbf{Y}$ *se, e somente se, a fórmula* α *(definida na fórmula 4.16) é satisfatível. Isso garante que a tripla* $(2,0,1)$ *ou* $(2,0,0)$ *estará em cada ponto fixo inflacionário inicial, de acordo com a fórmula* α *ser satisfatível ou não. Segue que* $(2,0,1) \in \bigcup \mathbf{rifp}(\mathcal{R}^{\phi,\mathfrak{A}_\alpha})$ *se, e somente se,* α *é satisfatível. Seja*

$$\psi_{\text{SAT}} \;=\; \left[rifp_{X,Y}\ \phi(X,Y)\right](201). \tag{4.28}$$

Então $\mathfrak{A}_\alpha \models \psi_{\text{SAT}}$ *se, e somente se,* α *é satisfatível.*

O exemplo que será apresentado a seguir trata-se de uma fórmula na lógica RIFP com operadores de **rifp** aninhados e com negação. Mostraremos como verificar se uma dada interpretação satisfaz a fórmula.

Exemplo 4.3.3 (Operadores rifp aninhados e com negação) *Considere a seguinte interpretação* $\mathfrak{I} = (\mathfrak{A},\beta)$, *onde :* $A = \{1,2,3,4,5\}$, $\beta(u) = 1$, $\beta(l) = 3$, $\beta(Z) = \{1,3,4\}$, $\beta(W) = \{3,4,5\}$. *Em seguida, considere a seguinte fórmula:*

$$\psi_1 \;=\; \left[rifp_{Z,W}\phi_1(Z,W)\right](l) \tag{4.29}$$

onde

$$\phi_1 \;=\; W \neq \emptyset \;\wedge\; (\neg\psi_2 \;\vee\; Wu) \tag{4.30}$$

$$\psi_2 \;=\; \left[rifp_{X,Y}\;\phi_2(X,Y,Z,W)\right](u) \tag{4.31}$$

$$\phi_2 \;=\; Y \neq \emptyset \;\wedge\; \left(X \subseteq Y \;\wedge\; Y \subseteq Z \cap W\right) \tag{4.32}$$

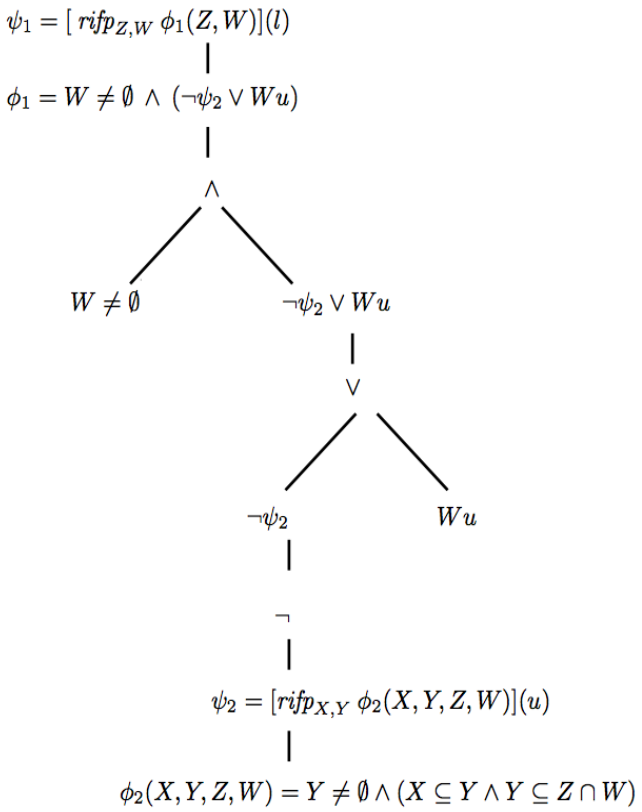

Figura 4.3: Operador **rifp** aninhado e com negação

Observe um esquema na Figura 4.3 das fórmulas aninhadas. Pela definição

de satisfatibilidade temos:

$$\mathfrak{I} \models \left[rifp_{Z,W}\phi_1(Z,W)\right](l) \quad sse \quad \mathfrak{I}(l) \in \bigcup \mathbf{rifp}(\mathcal{R}^{\phi_1,\mathfrak{I}}). \tag{4.33}$$

Como $\mathfrak{I}(l) = 3$ *temos que verificar se* $3 \in \bigcup \mathbf{rifp}(\mathcal{R}^{\phi_1,\mathfrak{I}})$. *Basta encontrarmos um ponto fixo inflacionário inicial que contenha o elemento* 3. *Vamos mostrar que* $\{3\}$ *é ponto fixo inflacionário inicial de* $\mathcal{R}^{\phi_1,\mathfrak{I}}$. *Para isso analisaremos a relação* $\mathcal{R}^{\phi_1,\mathfrak{I}}$. *Por definição,*

$$\mathcal{R}^{\phi_1,\mathfrak{I}} = \{(V_1, V_2) \in \mathcal{P}(A^k) \times \mathcal{P}(A^k) \mid \mathfrak{I}\frac{V_1, V_2}{Z, W} \models \phi_1(Z,W)\}.$$

Mostraremos que $(\emptyset, \{3\}) \in \mathcal{R}^{\phi_1,\mathfrak{I}}$, *ou seja*

$$\mathfrak{I}\frac{\emptyset, \{3\}}{Z, W} \models \phi_1(Z,W).$$

Temos que mostrar que

$$\mathfrak{I}\frac{\emptyset, \{3\}}{Z, W} \models (\neg\psi_2 \vee Wu).$$

Como $\mathfrak{I}\frac{\emptyset,\{3\}}{Z,W} \not\models Wu$, *temos que mostrar que* $\mathfrak{I}\frac{\emptyset,\{3\}}{Z,W} \models \neg\psi_2$, *ou seja,*

$$\mathfrak{I}\frac{\emptyset, \{3\}}{Z, W} \not\models \left[rifp_{X,Y}\ \phi_2(X,Y,Z,W)\right](u).$$

Basta mostrarmos que $\mathfrak{I}\frac{\emptyset,\{3\}}{Z,W}(u) = 1 \notin \mathbf{rifp}\big(\mathcal{R}^{\phi_2,\mathfrak{I}\frac{\emptyset,\{3\}}{Z,W}}\big)$. *Por definição, temos*

$$\mathcal{R}^{\phi_2,\mathfrak{I}\frac{\emptyset,\{3\}}{Z,W}} = \big\{(V_3,V_4) \in \mathcal{P}(A^k) \times \mathcal{P}(A^k) \mid \mathfrak{I}\frac{\emptyset,\{3\},V_3,V_4}{Z,W,X,Y} \models \phi_2(X,Y,Z,W)\big\}.$$

Note que na fórmula ϕ_2 *temos que* $Y \neq \emptyset$ *e* $Y \subseteq Z \cap W$. *Na interpretação* $\mathfrak{I}\frac{\emptyset,\{3\}}{Z,W}$ *temos que* $Z \cap W = \emptyset$. *Logo, não existe valor para* Y *que satisfaça a fórmula* ϕ_2. *Logo, a relação* $\mathcal{R}^{\phi_2,\mathfrak{I}\frac{\emptyset,\{3\}}{Z,W}} = \emptyset$. *Logo,* $(\emptyset,\emptyset) \in \mathcal{R}^{\phi_2,\mathfrak{I}\frac{\emptyset,\{3\}}{Z,W}}_{\text{INF}}$. *Logo,* $\emptyset$ *é o único ponto fixo inflacionário inicial de* $\mathcal{R}^{\phi_2,\mathfrak{I}\frac{\emptyset,\{3\}}{Z,W}}$. *Portanto,* $\mathfrak{I}\frac{\emptyset,\{3\}}{Z,W}(u) = 1 \notin \bigcup \mathbf{rifp}\big(\mathcal{R}^{\phi_2,\mathfrak{I}\frac{\emptyset,\{3\}}{Z,W}}\big) = \emptyset$. *Logo,*

$$\mathfrak{I}\frac{\emptyset,\{3\}}{Z,W} \models \phi_1(Z,W).$$

Portanto, $(\emptyset,\{3\}) \in \mathcal{R}^{\phi_1,\mathfrak{I}}$.

Basta mostrarmos que $(\{3\},\{3\}) \in \mathcal{R}^{\phi_1,\mathfrak{I}}$, *ou seja,*

$$\mathfrak{I}\frac{\{3\},\{3\}}{Z,W} \models \phi_1(Z,W).$$

Temos que mostrar que

$$\mathfrak{I}\frac{\{3\},\{3\}}{Z,W} \models (\neg\psi_2 \vee Wu).$$

Como $\mathfrak{I}\frac{\{3\},\{3\}}{Z,W} \not\models Wu$, *temos que mostrar que* $\mathfrak{I}\frac{\{3\},\{3\}}{Z,W} \models \neg\psi_2$, *ou seja,*

$$\mathfrak{I}\frac{\{3\},\{3\}}{Z,W} \not\models \big[rifp_{X,Y}\ \phi_2(X,Y,Z,W)\big]\,(u).$$

Basta mostrarmos que $\mathfrak{I}\frac{\{3\},\{3\}}{Z,W}(u) = 1 \notin \mathbf{rifp}\big(\mathcal{R}^{\phi_2,\mathfrak{I}\frac{\{3\},\{3\}}{Z,W}}\big)$. *Por definição, temos*

$$\mathcal{R}^{\phi_2,\mathfrak{I}\frac{\{3\},\{3\}}{Z,W}} = \big\{(V_3,V_4) \in \mathcal{P}(A^k)\times\mathcal{P}(A^k) \mid \mathfrak{I}\frac{\{3\},\{3\},V_3,V_4}{Z,W,X,Y} \models \phi_2(X,Y,Z,W)\big\}.$$

Similar ao caso anterior, temos que $\{3\}$ *é o único ponto fixo inflacionário inicial de* $\mathcal{R}^{\phi_2,\mathfrak{I}\frac{\{3\},\{3\}}{Z,W}}$. *Portanto,* $\mathfrak{I}\frac{\{3\},\{3\}}{Z,W}(u) = 1 \notin \bigcup \mathbf{rifp}\big(\mathcal{R}^{\phi_2,\mathfrak{I}\frac{\{3\},\{3\}}{Z,W}}\big) = \{3\}$. *Logo,*

$$\mathfrak{I}\frac{\{3\},\{3\}}{Z,W} \models \phi_1(Z,W).$$

Portanto, $(\{3\},\{3\}) \in \mathcal{R}^{\phi_1,\mathfrak{I}}$. *Logo,* $(\emptyset,\{3\})$ *e* $(\{3\},\{3\})$ *estão em* $\mathcal{R}^{\phi_1,\mathfrak{I}}_{\text{INF}}$. *Além disso, é fácil ver que* $\emptyset$ *não é ponto fixo de* $\mathcal{R}^{\phi_1,\mathfrak{I}}_{\text{INF}}$. *Portanto,* $\{3\}$ *é ponto fixo inflacionário inicial de* $\mathcal{R}^{\phi_1,\mathfrak{I}}$. *Logo,* $\mathfrak{I} \models \psi_1$.

A seguir, mostramos o exemplo do fecho transitivo expresso em RIFP.

Exemplo 4.3.4 (Fecho Transitivo) *A* query transitive closure *ou* ***fecho transitivo****, a qual denotaremos por* TC*, é uma* query *binária (veja a expressão em 4.34) define o conjunto dos pares de vértices em um grafo tais que existe um caminho entre eles.*

$$TC(G) \quad := \quad \{(a,b) \in V^2 \mid \textit{existe um caminho de } a \textit{ até } b\}. \tag{4.34}$$

Esta propriedade não pode ser expressa em FO. Mostraremos como expressá-la em RIFP.

Seja $G = (V,E)$ *um grafo e* $\mathfrak{I} = (G,\beta)$ *uma interpretação. Considere a*

seguinte fórmula ψ da lógica RIFP*:*

$$\psi \quad = \quad \left[rifp_{X,Y}\phi(X,Y)\right](u_1,u_2) \tag{4.35}$$

onde

$$\begin{aligned} \phi \quad = \quad & \exists xy\, Yxy \wedge \\ & \left[\left(X = \emptyset \wedge \forall xy\,(Exy \leftrightarrow Yxy)\right) \vee \right. \\ & \left. \forall xy\left((Xxy \vee \exists z\,(Xxz \wedge Xzy)) \leftrightarrow Yxy\right)\right] \end{aligned} \tag{4.36}$$

Para que as variáveis relacionais X e Y satisfaçam a fórmula $\phi(X,Y)$, é preciso que a relação $Y \neq \emptyset$, logo o par $(\emptyset,\emptyset)$ não pertence a $\mathcal{R}^{\phi,\mathfrak{I}}_{\text{INF}}$. Esta condição está expressa na subfórmula $\exists xy\, Yxy$.

Para satisfazer a subfórmula $(X = \emptyset \wedge \forall xy\,(Exy \leftrightarrow Yxy))$, a relação Y é a mesma relação E de arestas do grafo.

Para satisfazer a subfórmula $\forall xy\left((Xxy \vee \exists z\,(Xxz \wedge Xzy)) \leftrightarrow Yxy\right)$, Y deve conter X e todo par de vértices (a,b) para os quais existe um c tal que (a,c) e (c,b) estão em X. Logo, existe exatamente uma cadeia (Definição 4.1.3) $X_0 = \emptyset, X_1, \ldots, X_m$, tal que $(X_i, X_{i+1}) \in \mathcal{R}^{\phi,\mathfrak{I}}$ e X_m é exatamente o fecho transitivo. Portanto, o único ponto fixo inflacionário inicial de $\mathcal{R}^{\phi,\mathfrak{I}}$ é o fecho transitivo do grafo que será igual a $\bigcup \mathbf{rifp}(\mathcal{R}^{\phi,\mathfrak{I}})$. Logo, a fórmula ψ será satisfeita se $(\mathfrak{I}(u_1),\mathfrak{I}(u_2))$ pertencer ao fecho transitivo do grafo.

Exemplo 4.3.5 (Conectividade) *A* query CONECTIVIDADE *ou* CN *é uma*

query *booleana tal que verifica se um dado grafo é conexo ou não.*

$$CN(G) = \begin{cases} 1 & se \quad é\ conexo \\ 0 & caso\ contrário \end{cases} \tag{4.37}$$

Semelhante ao Exemplo 4.3.4 do Fecho Transitivo, faremos a consulta para verificar se existe um caminho entre qualquer par de vértices da estrutura. Basta verificar se todo par de vértices está no fecho transitivo.

Seja $\psi = \big[rifp_{X,Y}\phi(X,Y)\big]\,(x,y)$ *a fórmula que define o fecho transitivo do Exemplo 4.3.4. A query* CN *é expressa pela fórmula abaixo:*

$$\psi' \;=\; \forall x \forall y \big[\neg x = y \;\wedge\; \big[rifp_{X,Y}\phi(X,Y)\big]\,(x,y)\big].$$

A seguir, mostramos o exemplo do Caminho Hamiltoniano que é definido como um caminho que passa por todos os vértices de um grafo exatamente uma única vez.

Exemplo 4.3.6 (Caminho Hamiltoniano) *O problema do* ***Caminho Hamiltoniano*** *consiste em encontrar um caminho que passe por todos os vértices de um grafo uma única vez. Um grafo* $G = (V, E)$ *é Hamiltoniano se as seguintes condições são satisfeitas:*

1. *existe uma relação sucessor nos vértices do grafo, i.e.,*

$$(\exists T)(T \text{ é uma relação sucessor sobre } V) \tag{4.38}$$

onde T *é uma variável de segunda ordem de aridade 2;*

2. a relação sucessor está contida no conjunto E de arestas

$$(\forall x)(\forall y)(\text{ se } y \text{ é o sucessor de } x \text{ em } T \text{ então } E(x,y))). \tag{4.39}$$

Seja $G = (V, E)$ um grafo. Usaremos as seguintes fórmulas auxiliares. A fórmula 4.40 expressa que todo elemento x aparece na relação Y.

$$\Theta(Y,x) \;=\; \exists y\big(Yxy \;\vee\; Yyx\big) \tag{4.40}$$

A fórmula 4.41 expressa que o elemento x da relação Y não possui sucessor.

$$NS(Y,x) \;=\; \neg\exists y\big(Yxy\big) \tag{4.41}$$

Nós escreveremos a RIFP*-fórmula em 4.42 que será satisfeita se o grafo G for Hamiltoniano.*

Seja $\phi(X,Y) = \phi_1 \vee \phi_2 \vee \phi_3$, onde

$$\begin{aligned}
\phi_1(X,Y) &= (X = \emptyset \wedge \exists^{=1}xy(Exy \wedge Yxy) \wedge \exists^{=1}xy(Yxy)) \\
\phi_2(X,Y) &= \left(\begin{array}{l} X \subseteq Y \wedge \\ \exists x_1\big[\neg\Theta(X,x_1) \wedge \exists xy(Xxy \wedge NS(X,y) \wedge Eyx_1 \wedge Yyx_1)\big] \wedge \\ \exists^{x=1}(Yxy \wedge \neg Xxy) \end{array}\right) \\
\phi_3(X,Y) &= \forall x_2(\Theta(X,x_2)) \leftrightarrow Y11
\end{aligned}$$

Seja $\mathfrak{I} = (G, \beta)$ uma interpretação. Se uma cadeia $X_0 = \emptyset, X_1, \ldots, X_m$ é tal que $(X_i, X_{i+1}) \in \mathcal{R}^{\phi,\mathfrak{I}}$, então $X_i \subseteq X_{i+1}$ e X_i é uma relação sucessor em

algum subconjunto de V. Além disso, se $(a, b) \in X_i$, então (a, b) é uma aresta. O conjunto X_{i+1} é uma relação sucessor que possui apenas um elemento a mais que X_i (a fórmula ϕ_2 garante isso). Assim, essa cadeia constrói uma relação sucessor sobre todo o domínio incrementalmente. Se o grafo G for Hamiltoniano, então existirá uma cadeia tal que para um m suficientemente grande, X_m será uma relação sucessor sobre V.

A fórmula ϕ_1 afirma que a relação X é a relação $\emptyset$ e que Y possui exatamente um par de vértices que é uma aresta do grafo.

A fórmula ϕ_2 afirma que a relação $X \subseteq Y$ e existe um vértice x_1 que ainda não está em nenhum par da relação X; e existe um vértice y que é sucessor de algum vértice na relação X mas que não é predecessor de nenhum elemento da relação X (y é o último elemento da relação sucessor X) e $(y, x_1) \in E$. Assim, o par (y, x_1) entra na relação Y e apenas este par. O par (y, x_1) não pertence à relação X.

E, finalmente, a fórmula ϕ_3 afirma que o par $(1, 1)$ está na relação Y se, e somente se, todo vértice do grafo estiver em algum par da relação X.

A seguir, a fórmula ψ em RIFP *que será satisfeita se o grafo for Hamiltoniano,*

$$\psi \;=\; \left[rifp_{X,Y}\, \phi(X, Y)\right](1, 1). \tag{4.42}$$

Apresentamos o exemplo a seguir por ser definido usando conceitos de funções bijetivas, injetivas, etc. Usaremos a ideia de verificar a cardinalidade do domínio da estrutura para tratar o problema de decidir se um grafo é ou não um grafo Euleriano.

Exemplo 4.3.7 (Euleriano) *O problema de saber se um grafo é Euleriano é saber se existe um caminho fechado que atravessa cada aresta exatamente uma vez. Euler mostrou que essa propriedade vale se, e somente se, todo vértice possui grau par, i.e., um número par de vizinhos. Definiremos a* query EULERIAN *ou* EULERIANO *como a* query *que é verdadeira caso todos os vértices possuam grau par. Essa* query *não é definível em FO sobre a classe de todos os grafos finitos.*

Inicialmente apresentaremos algumas fórmulas auxiliares para definir a query EULERIANO.

1. *A fórmula $FUNC(Y, Z)$ afirma que a relação binária Y é o grafo de uma função.*

$$FUNC(Y,Z) \;=\; \forall x \forall y \forall z \big[Zx \wedge Zy \wedge Zz \rightarrow [Yxy \wedge Yxz \rightarrow y = z] \big] \quad (4.43)$$

2. *A fórmula $INJ(Y, Z)$ afirma que a relação Y é o grafo de uma função injetiva.*

$$INJ(Y,Z) \;=\; \forall x \forall y \forall z \left[Zx \wedge Zy \wedge Zz \rightarrow [Yxy \wedge Yzy \rightarrow x = z] \right] \quad (4.44)$$

3. *A fórmula $TOT(Y, Z)$ afirma que todo elemento de Z está no domínio ou na imagem de Y.*

$$TOT(Y,Z) \quad = \quad \big[\forall x \left[Zx \rightarrow \exists y (Yxy \vee Yyx) \right] \big] \quad (4.45)$$

4. *A fórmula SEP(Y, Z) afirma que nehum elemento de Z aparece na imagem e no domínio de Y.*

$$SEP(Y,Z) \quad = \quad \left[\forall x \left[Zx \rightarrow \neg\exists yz(Yxy \wedge Yzx)\right]\right] \tag{4.46}$$

5. *A fórmula Θ afirma que todo elemento que ocorre em algum par ordenado de Y é um elemento de Z.*

$$\Theta(Y,Z) \; = \; \forall x \forall y(Yxy \rightarrow Zx \wedge Zy) \tag{4.47}$$

6. *A fórmula $\phi(X, Y)$ garante que Y é uma bijeção cujo o domínio e a imagem correspondem a uma partição de Z.*

$$\phi(X,Y) = X = \emptyset \wedge FUNC(Y,Z) \wedge INJ(Y,Z) \wedge TOT(Y,Z) \wedge \Theta(Y,Z) \tag{4.48}$$

Seja $G = (V, E)$ um grafo e $\mathfrak{I} = (G, \beta)$ uma interpretação. Todo ponto fixo inflacionário inicial de $\mathcal{R}^{\phi,\mathfrak{I}}$ é alcançado em um passo a partir do vazio e é o grafo de uma bijeção cujo o domínio e a imagem formam uma partição de $\mathfrak{I}(Z)$. Isso só é possível se $\mathfrak{I}(Z)$ tiver cardinalidade par. Caso contrário, o único ponto fixo inflacionário inicial de $\mathcal{R}^{\phi,\mathfrak{I}}$ é o $\emptyset$.

7. *A fórmula EVEN(Z) é satisfeita se, e somente se, Z tiver cardinalidade par.*

$$EVEN(Z) \; = \; Z \; = \; \emptyset \; \vee \; \exists z_1 \exists z_2 \left[rifp_{X,Y} \; \phi(X,Y)\right](z_1, z_2) \tag{4.49}$$

Temos que checar se o conjunto dos vizinhos de cada vértice é par.

8. *A fórmula VIZ(w) define o conjunto dos vizinhos de w, ou seja, y satisfaz a fórmula se, e somente se, for vizinho de w.*

$$VIZ(w) \;=\; Ewy \wedge \neg(w = y) \tag{4.50}$$

Utilizaremos a fórmula VIZ(w) no lugar de Z na fórmula EVEN(Z) para afirmar que o conjunto dos vizinhos de w é par.

9. *A fórmula ψ afirma que todo elemento do grafo possui quantidade par de vizinhos.*

$$\psi \;=\; \forall w \; EVEN(\, VIZ(w)) \tag{4.51}$$

Portanto, a estrutura satisfaz a fórmula ψ se for Euleriana.

O problema Clique é completo para a classe NP. A seguir, apresentaremos a fórmula para Clique em RIFP.

Exemplo 4.3.8 (Clique) *A* query Clique *verifica, para um dado grafo não-direcionado e um valor $k \in \mathbb{N}$, se o grafo tem um subgrafo completo com k vértices. A entrada da query* Clique *serão estruturas do tipo $G = (V, E, K)$, onde (V, E) é um grafo e K é um conjunto de vértices de cardinalidade k. Verificaremos se existe uma bijeção entre K e algum subgrafo completo de (V, E).*

Apresentaremos a seguir fórmulas auxiliares. A relação Y usada nas fórmulas abaixo será interpretada como o grafo de uma função injetiva cujo domínio é K.

1. *A fórmula IM verifica se determinado elemento y está na imagem da relação Y.*

$$IM(Y, y) \;=\; \exists x\, Yxy \tag{4.52}$$

2. *A fórmula Θ afirma que entre quaisquer dois elementos da imagem de Y existe uma aresta.*

$$\Theta(Y) \;=\; \forall x \forall y \big(\, IM(Y, x) \;\wedge\; IM(Y, y) \;\rightarrow\; Exy\big) \tag{4.53}$$

3. *A fórmula TOT' afirma que a relação K está contida no domínio da relação Y.*

$$TOT'(Y, K) \;=\; \forall x \left[Kx \;\rightarrow\; \exists y \,(Yxy)\right] \tag{4.54}$$

4. *A fórmula FUNC', similar à do Exemplo 4.3.7, verifica se a relação Y é uma função.*

$$FUNC'(Y) \;=\; \forall x \forall y \forall z \left[Yxy \;\wedge\; Yxz \;\rightarrow\; y = z\right] \tag{4.55}$$

5. *A fórmula INJ', similar à do Exemplo 4.3.7, verifica se a relação Y é uma função injetiva.*

$$INJ'(Y) \;=\; \forall x \forall y \forall z \left[Yxy \;\wedge\; Yzy \;\rightarrow\; x = z\right] \tag{4.56}$$

6. *A fómula ϕ é satisfeita se a relação Y for uma função injetiva, que contém*

K no domínio e cuja imagem é um grafo completo.

$$\phi(X,Y) \;=\; \mathit{FUNC'}(Y) \;\wedge\; \mathit{INJ'}(Y) \;\wedge\; \mathit{TOT'}(Y,K) \;\wedge\; \Theta(Y) \qquad (4.57)$$

Seja $G = (V, E, K)$ um grafo e $\mathfrak{I} = (G, \beta)$ uma interpretação. Todo ponto fixo inflacionário inicial de $\mathcal{R}^{\phi,\mathfrak{I}}$ é alcançado em um passo a partir do vazio e é o grafo de uma função injetiva cujo o domínio contém K, portanto, tem cardinalidade maior ou igual à cardinalidade de K, e cuja imagem é um grafo completo. Isso só é possível se G contém um subgrafo completo de cardinalidade k. Caso contrário, o único ponto fixo inflacionário inicial de $\mathcal{R}^{\phi,\mathfrak{I}}$ é o $\emptyset$.

7. *A fórmula* Clique *é satisfeita se o grafo G contém um subgrafo completo de cardinalidade k.*

$$\textsc{Clique} \;=\; \exists x \exists y \left[\mathit{rifp}_{X,Y}\, \phi(X,Y)\right](x,y) \;\vee\; K = \emptyset \qquad (4.58)$$

No exemplo a seguir, mostraremos uma fórmula $\phi(X, Y)$ que define uma relação cujos pontos fixos inflacionários iniciais são ordens lineares.

Exemplo 4.3.9 (Ordem Linear) *Considere as seguintes fórmulas $\mathit{TRAN}(Y)$, $\mathit{IRRE}(Y)$ e $\mathit{TRIC}(Y)$ que afirmam que Y é uma relação binária transitiva, irreflexiva e que possui tricotomia.*

1. *trasitividade:*

$$\mathit{TRAN}(Y) \;=\; \forall x \forall y \forall z \left(Yxy \;\wedge\; Yyz \;\rightarrow\; Yxy\right) \qquad (4.59)$$

2. irreflexividade:

$$IRRE(Y) \;=\; \forall x \neg Y x x \tag{4.60}$$

3. tricotomia:

$$TRIC(Y) \;=\; \forall x \forall y \left(x = y \;\vee\; Y x y \;\vee\; Y y x\right) \tag{4.61}$$

4. Fazendo a conjunção das propriedades acima, segue a fórmula ϕ,

$$\phi(X, Y) \;=\; X = \emptyset \;\wedge\; TRAN(Y) \;\wedge\; IRRE(Y) \;\wedge\; TRIC(Y) \tag{4.62}$$

A fórmula ϕ é satisfeita apenas no caso em que $X = \emptyset$ e Y é uma ordem linear no domínio da estrutura. Seja $\mathfrak{A}$ uma estrututa e $\mathfrak{I} = (\mathfrak{A}, \beta)$ uma interpretação. A relação $\mathcal{R}^{\phi,\mathfrak{I}}$ tem como pontos fixos inflacionários iniciais exatamente as ordenações lineares do domínio de $\mathfrak{A}$.

Capítulo 5

Complexidade Descritiva de RIFP

Neste capítulo, mostraremos que o poder expressivo de RIFP é equivalente ao da lógica SO. Consequentemente RIFP captura a hierarquia polinomial (PH). Também apresentaremos um fragmento de RIFP o qual chamaremos RIFP_1. Em seguida, provaremos que o tal fragmento captura a classe NP. Isso será provado de duas maneiras, primeiro apresentando uma tradução para o fragmento existencial da lógica de segunda ordem e, depois, através de uma prova direta utilizando o modelo computacional de máquinas de Turing não-determinísticas.

5.1 RIFP captura PH

Para provar que RIFP captura PH mostraremos que RIFP é equivalente a SO. Para mostrar que o poder expressivo de RIFP é o mesmo que o de SO, apresentaremos uma tradução entre as duas linguagens. Considere a seguinte tradução de SO para RIFP:

$$
\begin{array}{lcl}
Tr(t_0 \equiv t_1) & = & t_0 \equiv t_1 \\
Tr(Rt_1 \ldots t_k) & = & Rt_1 \ldots t_k \\
Tr(Xt_1 \ldots t_k) & = & Xt_1 \ldots t_k \\
Tr(\neg\phi) & = & \neg Tr(\phi) \\
Tr(\phi_1 \wedge \phi_2) & = & Tr(\phi_1) \wedge Tr(\phi_2) \\
Tr(\exists x \phi) & = & \exists x Tr(\phi) \\
Tr(\exists Y \phi(Y)) & = & \exists \overline{x}\big[\mathit{rifp}_{X,Y}\psi(X,Y)\big]\,(\overline{x}) \vee Tr(\phi(\emptyset))
\end{array}
$$

onde ψ é a fórmula

$$\psi(X,Y) = \big(X = \emptyset \wedge Tr(\phi(Y))\big).$$

Lema 5.1.1 *Seja $\alpha \in SO$. Então $\alpha \equiv Tr(\alpha)$.*

O lema acima segue por indução em $\alpha \in$ SO. A prova é feita mostrando que para toda interpretação $\mathfrak{I}$, $\mathfrak{I} \models \alpha$ se, e somente se, $\mathfrak{I} \models Tr(\alpha)$. O único caso interessante é o caso em que $\alpha = \exists Y \phi(Y)$. Por hipótese indutiva, $\mathfrak{I} \models \phi$ se, e somente se, $\mathfrak{I} \models Tr(\phi)$. Seja $\mathfrak{I}$ tal que $\mathfrak{I} \models \alpha$. Logo, $\mathfrak{I} \models \exists Y \phi(Y)$. Pela definição de satisfatibilidade, existe $\mathbf{Y} \subset A^k$ tal que $\mathfrak{I}^{\mathbf{Y}}_{Y} \models \phi(Y)$, onde A é o domínio da estrutura e k é a aridade da relação Y. Analisemos o caso em que $\mathbf{Y} = \emptyset$, nesse caso temos $\mathfrak{I}^{\emptyset}_{Y} \models \phi(\emptyset)$. Logo, pela hipótese indutiva, $\mathfrak{I}^{\emptyset}_{Y} \models Tr(\phi(\emptyset))$. Logo, $\mathfrak{I}^{\emptyset}_{Y} \models \exists \overline{x}\big[\mathit{rifp}_{X,Y}\psi(X,Y)\big]\,(\overline{x}) \vee Tr(\phi(\emptyset))$. Logo $\mathfrak{I}^{\emptyset}_{Y} \models Tr(\alpha)$. Agora, analisemos o caso em que $\mathbf{Y} \neq \emptyset$, $\mathfrak{I}^{\mathbf{Y}}_{Y} \models \phi(Y)$, onde A é o domínio da estrutura e k é a aridade da relação Y. Basta mostrar que $\mathfrak{I}^{\mathbf{Y}}_{Y} \models \exists \overline{x}\big[\mathit{rifp}_{X,Y}\psi(X,Y)\big](\overline{x})$. Pela definição de satisfatibilidade, $\mathfrak{I} \models \exists \overline{x}\big[\mathit{rifp}_{X,Y}\psi(X,Y)\big](\overline{x})$ se, e somente se, existe $\overline{\mathbf{x}} \in A^k$ tal que $\mathfrak{I}^{\overline{\mathbf{x}}}_{\overline{x}} \models \big[\mathit{rifp}_{X,Y}\psi(X,Y)\big](\overline{x})$. Pela definição de satisfatibilidade, $\mathfrak{I}^{\overline{\mathbf{x}}}_{\overline{x}} \models \big[\mathit{rifp}_{X,Y}\psi(X,Y)\big](\overline{x})$ se, e somente se, $\mathfrak{I}^{\overline{\mathbf{x}}}_{\overline{x}}(\overline{x}) \in \bigcup \mathbf{rifp}\big(\mathcal{R}^{\psi,\mathfrak{I}^{\overline{\mathbf{x}}}_{\overline{x}}}\big)$. Como $\overline{x}$ não ocorre livre em $\psi(X,Y)$, temos $\mathcal{R}^{\psi,\mathfrak{I}^{\overline{\mathbf{x}}}_{\overline{x}}} = \mathcal{R}^{\psi,\mathfrak{I}}$. Por definição, temos que $\mathcal{R}^{\psi,\mathfrak{I}} = \big\{(\mathbf{V_1},\mathbf{V_2}) \in \mathcal{P}(A^k) \times \mathcal{P}(A^k) \mid \mathfrak{I}^{\mathbf{V_1},\mathbf{V_2}}_{X,Y} \models \psi(X,Y)\big\}$. Como $\psi(X,Y) = \big(X = \emptyset \wedge Tr(\phi(Y))\big)$, temos que $\mathbf{V_1} = \emptyset$ e $\mathfrak{I}^{\mathbf{V_2}}_{Y} \models Tr(\phi(Y))$. Mas, como $\mathfrak{I}^{\mathbf{Y}}_{Y} \models \phi(Y)$, pela hipótese indutiva, $\mathfrak{I}^{\mathbf{Y}}_{Y} \models Tr(\phi(Y))$. Pela definição de $\mathcal{R}^{\psi,\mathfrak{I}}$, note que $\mathcal{R}^{\psi,\mathfrak{I}} = \mathcal{R}^{\psi,\mathfrak{I}}_{\text{INF}}$. Além disso, toda cadeia em $\mathcal{R}^{\psi,\mathfrak{I}}$ tem tamanho 2, e, portanto, os pontos fixos inflacionários iniciais de $\mathcal{R}^{\psi,\mathfrak{I}}$ são exatamente os conjuntos $C \subseteq A^k$ tais que $\mathfrak{I}^{C}_{Y} \models \phi(Y)$. Logo, $\mathbf{Y}$ é ponto fixo inflacionário inicial de $\mathcal{R}^{\psi,\mathfrak{I}}$ e, como $\mathbf{Y} \neq \emptyset$, então $\bigcup \mathbf{rifp}\big(\mathcal{R}^{\psi,\mathfrak{I}}\big) \neq \emptyset$. Logo, $\mathfrak{I} \models \exists \overline{x}\big[\mathit{rifp}_{X,Y}\psi(X,Y)\big](\overline{x})$ e, portanto $\mathfrak{I} \models Tr(\alpha)$. De maneira análoga, é

possível mostrar que se $\mathfrak{I} \models Tr(\alpha)$ então $\mathfrak{I} \models \alpha$. Logo, vale o lema.

No outro sentido, isto é, de RIFP para SO, temos que reconstruir as cadeias (Definição 4.1.3) que dão origem aos pontos fixos inflacionários iniciais utilizando a lógica de segunda ordem. O fato fundamental é a propriedade inflacionária da relação $\mathcal{R}^{\phi,\mathfrak{I}}_{\text{INF}}$ que garante que qualquer ponto fixo inicial pode ser alcançado através de uma sequência de comprimento suficientemente curto.

Seja $\phi(X,Y)$ a fórmula com X e Y variáveis relacionais de aridade k, e $\mathfrak{I}$ uma interpretação cujo domínio tem cardinalidade n. Seja $X_0, X_1, \ldots, X_m$ uma cadeia em $\mathcal{R}^{\phi,\mathfrak{I}}_{\text{INF}}$ e X_m um ponto fixo inicial. Como $\mathcal{R}^{\phi,\mathfrak{I}}_{\text{INF}}$ é inflacionário, m é no máximo n^k. Suponha que $<$ é uma relação de aridade $2k$ sobre o domínio de $\mathfrak{I}$ que é uma ordem sobre as k-tuplas de elementos em $\mathfrak{I}$, e, para $1 \leq i \leq n^k$, seja $\overline{t_i}$ a i-ésima k-tupla com relação a $<$. Seja X' uma relação de aridade $2k$ definida como:

$$X' = \{(\overline{t_i}, \overline{a}) \in A^{2k} \mid \overline{a} \in X_i\} \tag{5.1}$$

Usaremos as primeiras k posições de uma tupla em X' para indicar o índice i de uma relação X_i na cadeia e as outras k posições representam uma k-tupla em X_i. Podemos usar a quantificação existencial para checar a existência de uma relação como X' que testemunha a existência de uma cadeia que alcança um ponto fixo inicial. Precisamos garantir que relações sucessivas na cadeia são pares em $\mathcal{R}^{\phi,\mathfrak{I}}_{\text{INF}}$. Considere a seguinte fórmula:

$$\begin{aligned} \text{COMPUTE}(X', \overline{t}) \quad = \quad & \exists \overline{t}' \exists X \, \exists Y \big(\overline{t}' = \overline{t} + 1 \wedge \\ & \phi(X,Y) \wedge \neg\phi(X,X) \wedge \\ & \forall \overline{a} (X' \overline{t}, \overline{a} \; \leftrightarrow \; X\overline{a}) \wedge \\ & \forall \overline{b} (X' \overline{t}', \overline{b} \; \leftrightarrow \; Y\overline{b} \vee X\overline{b}) \big) \end{aligned} \tag{5.2}$$

A fórmula COMPUTE$(X', \overline{t})$ diz que $\overline{t}$ e $\overline{t}'$ são consecutivos com respeito a $<$ e que a projeção de X' em $\overline{t}$ e $\overline{t}'$ forma um par em $\mathcal{R}^{\phi,\mathfrak{I}}_{\text{INF}}$ e a projeção em t não é um ponto fixo inflacionário inicial.

A fórmula START(X') afirma que a relação X_1 da cadeia é a projeção de X' em $\overline{t_1}$ (lembre-se de que $X_0 = \emptyset$ em uma cadeia):

$$\text{START}(X') = \exists Y (\phi(\emptyset, Y) \wedge \forall \overline{a} (X' \overline{t_1} \overline{a} \leftrightarrow Y\overline{a})). \tag{5.3}$$

Finalmente, a fórmula

$$\mathrm{END}(X', \bar{t}_{\mathrm{pf}}, \bar{z}) \quad = \quad \exists X \big(\forall \bar{a}(X' \bar{t}_{fp} \bar{a} \leftrightarrow X\bar{a}) \wedge \phi(X, X)\big) \wedge X \bar{t}_{fp} \bar{z} \qquad (5.4)$$

afirma que a projeção de X' sobre $\bar{t}_{fp}$ é um ponto fixo e que $\bar{z}$ está nesta projeção.

Seja $Ord(<)$ uma fórmula de primeira ordem que afirma que $<$ é uma ordem linear sobre as k-tuples. Considere a seguinte tradução de RIFP para SO:

$$\begin{array}{lcl}
Tr'(t_0 \equiv t_1) & = & t_0 \equiv t_1 \\
Tr'(Rt_1 \ldots t_k) & = & Rt_1 \ldots t_k \\
Tr'(Xt_1 \ldots t_k) & = & Xt_1 \ldots t_k \\
Tr'(\neg\phi) & = & \neg Tr'(\phi) \\
Tr'(\phi_1 \wedge \phi_2) & = & Tr'(\phi_1) \wedge Tr'(\phi_2) \\
Tr'(\exists x \phi) & = & \exists x Tr'(\phi) \\
Tr'(\big[rifp_{X,Y} \phi(X, Y)\big](\overline{u})) & = & \exists < \exists X' \, \exists \bar{t}_{\mathrm{fp}} (Ord(<) \wedge \mathrm{START'}(X) \wedge \\
 & & \forall \bar{t}(\bar{t} < \bar{t}_{pf} \rightarrow \mathrm{COMPUTE'}(X, \bar{t})) \wedge \\
 & & \mathrm{END'}(X, \bar{t}_{\mathrm{pf}}, \overline{u})).
\end{array}$$

onde COMPUTE'$(X, \bar{t})$ é obtido de COMPUTE$(X, \bar{t})$ substituindo ϕ por $Tr'(\phi)$, similar para START'(X) e END'$(X, \bar{t}_{\mathrm{pf}}, \overline{u})$.

Lema 5.1.2 *Seja $\alpha \in$ RIFP. Então $\alpha \equiv Tr'(\alpha)$.*

Teorema 5.1.3 (RIFP = SO) *RIFP e SO têm o mesmo poder expressivo.*

Segue imediatamente do Teorema 5.1.3 e do fato que SO captura a hierarquia polinomial [Sto76] que:

Corolário 5.1.4 (RIFP = PH) *RIFP captura a hierarquia polinomial PH.*

5.2 Complexidade Descritiva de RIFP_1

Mostraremos nesta seção que o fragmento RIFP_1 captura a classe de complexidade NP. Apresentaremos este resultado de duas maneiras. Uma dessas maneiras será apresentando uma tradução de RIFP_1 para SO e vice-versa. A outra maneira será mostrando a prova do resultado similar ao Teorema de Fagin, i.e., mostrando uma máquina de Turing que simula a lógica RIFP_1 e a fórmula em RIFP_1 que descreve uma máquina de Turing não-determinística.

5.2.1 Equivalência entre RIFP_1 e $\exists$SO

A lógica RIFP é equivalente à lógica SO. O Teorema de Fagin mostra que o fragmento existencial de SO captura NP. O fragmento existencial de SO é composto de fórmulas da forma

$$\exists X_1 \ldots \exists X_q \phi \tag{5.5}$$

onde ϕ é uma fórmula de primeira ordem. Há um fragmento definido sintaticamente de RIFP que corresponde ao fragmento existencial de SO. Chamaremos de RIFP_1 o fragmento de RIFP consistindo de fórmulas da forma

$$\exists x_1 \ldots \exists x_k \left[rifp_{X,Y} \phi(X,Y) \right] (\overline{u}) \tag{5.6}$$

onde $\phi(X,Y)$ está em primeira ordem.

A tradução entre RIFP e SO apresentada na Seção 5.1 não nos dá diretamente o resultado que queremos, porque não mapeia o fragmento existencial de SO em RIFP, e vice-versa. Vamos considerar a tradução Tr de SO para RIFP. De acordo com a tradução, cada quantificador existencial será transformado em um construtor **rifp**. Assim, uma fórmula existencial deverá ser mapeada em uma RIFP-fórmula com vários operadores de **rifp** aninhados. Vamos considerar então fórmulas de $\exists$SO com apenas um quantificador existencial de segunda ordem. Então, cada fórmula existencial é equivalente a uma RIFP-fórmula com apenas um operador **rifp**. Por definição, temos:

$$Tr(\exists Y \phi(Y)) = \exists \overline{x} \big[rifp_{X,Y} \psi(X,Y) \big] (\overline{x}) \ \vee Tr(\phi(\emptyset)), \tag{5.7}$$

o que não é a forma exata de uma RIFP_1-fórmula. Mas podemos ver que $Tr(\exists Y \phi(Y))$ é equivalente a

$$\exists \overline{x} \big[rifp_{X,Y} \psi'(X,Y) \big] \, (\overline{x}), \tag{5.8}$$

onde

$$\psi'(X,Y) = \big(X = \emptyset \wedge Y \neq \emptyset \wedge Tr(\phi(Y)) \big) \vee \big(Tr(\phi(\emptyset)) \wedge Y \overline{x} \big) \tag{5.9}$$

Lema 5.2.1 ($\exists$SO $\subset$ RIFP$_1$) *A lógica existencial de segunda ordem está contida em RIFP$_1$.*

Por outro lado, a tradução Tr' de RIFP para SO quando aplicada a fórmulas em RIFP_1, também não nos dá fórmulas existenciais. Obviamente, isso pode ser colocado na forma prenex, mas, se considerarmos a última cláusula da definição de Tr'

$$\begin{aligned} Tr'(\big[rifp_{X,Y} \phi(X,Y) \big](\overline{u})) \quad = \quad & \exists < \exists X' \, \exists \bar{t}_{\mathrm{fp}} (Ord(<) \ \wedge \mathrm{START'}(X) \wedge \\ & \forall \bar{t} (\bar{t} < \bar{t}_{pf} \rightarrow \mathrm{COMPUTE'}(X, \bar{t})) \wedge \\ & \mathrm{END'}(X, \bar{t}_{\mathrm{pf}}, \overline{u})), \end{aligned} \tag{5.10}$$

a subfórmula $\forall \bar{t} (\bar{t} < \bar{t}_{pf} \rightarrow \mathrm{COMPUTE'}(X, \bar{t}))$ implica que teremos um quantificador universal de primeira ordem antes da fórmula existencial de segunda ordem em COMPUTE'. No entanto, quantificadores de primeira ordem podem ser internalizados na parte da fórmula de primeira ordem. Por exemplo, a fórmula $\forall x \exists X \phi$, onde X tem aridade k, é equivalente a $\exists X' (\forall x \phi')$, onde X' tem aridade $k+1$ e ϕ' é obtido de ϕ substituindo qualquer subfórmula atômica $X\bar{t}$ por $X' x \bar{t}$. Portanto, a fórmula dada pela tradução Tr' é equivalente a uma fórmula em RIFP_1.

Lema 5.2.2 (RIFP$_1$ $\subset$ $\exists$SO) *RIFP$_1$ está contido na lógica existencial de segunda ordem.*

Teorema 5.2.3 (RIFP$_1$ = $\exists$SO) *RIFP$_1$ tem o mesmo poder expressivo que a lógica existencial de segunda ordem.*

Na próxima seção, apresentaremos a prova direta que RIFP_1 captura NP.

5.2.2 $\mathbf{RIFP_1}$ captura NP

A seguir, vamos esboçar uma prova direta de que RIFP_1 captura NP, como na prova do Teorema de Fagin [Fag74]. Para tornar a prova mais fácil, vamos supor que as estruturas são ordenadas. Esta condição pode ser removida, mas as fórmulas envolvidas são mais complexas. A última parte mostra como eliminar a relação de ordem.

Teorema 5.2.4 ($\mathbf{RIFP_1 = NP}$) *$RIFP_1$ captura NP.*

Abaixo provaremos esse teorema.

Lema 5.2.5 ($\mathbf{RIFP_1 \subseteq NP}$) *Seja $\psi(x_1, \ldots, x_l)$ uma fórmula de $RIFP_1[S]$, com variáveis livre em $x_1, \ldots, x_l$. Então a classe de estruturas $\mathcal{K} = \big\{(\mathfrak{A}, a_1, \ldots, a_l) \mid \mathfrak{A} \models \psi(x_1, \ldots, x_l)[a_1, \ldots, a_l]\big\}$ está em NP.*

Prova.

Nós vamos mostrar que para toda sentença $\psi(x_1, \ldots, x_l)[a_1, \ldots, a_l] \in \mathrm{RIFP}_1$ existe uma máquina de Turing $N_\psi \in \mathrm{NP}$ tal que para toda estrutura $\mathfrak{A}$ temos que:

$$\mathfrak{A} \models \psi \Leftrightarrow N_\psi \text{ aceita a codificação binária de } (\mathfrak{A}) \tag{5.11}$$

Considere $\psi \in \mathrm{RIFP}_1$. Logo, existem duas possibilidade para analisarmos a fórmula ψ,

i. $\psi \in \mathrm{FO}$

ii. $\psi = \exists v_1 \ldots \exists v_m \big[\mathit{rifp}_{X,Y}\phi(X,Y)\big](\bar{t}_r)$

Caso (i): Se $\psi \in \mathrm{FO}$, então pelo Teorema 2.5.4, o conjunto das consultas booleanas em primeira ordem está contido no conjunto das consultas booleanas computáveis em uma máquina determinística logspace, i.e., existe uma máquina de Turing na classe de complexidade L. Como $\mathrm{L} \subseteq \mathrm{P}$ e $\mathrm{P} \subseteq \mathrm{NP}$, o resultado segue.

Caso (ii): Se $\psi := \exists v_1 \ldots \exists v_m \big[\mathit{rifp}_{X,Y}\phi(X,Y)\big](\bar{t}_r)$, vamos mostrar como construir uma máquina de Turing não-determinística N_ψ que decide se $(\mathfrak{A}, \overline{a}_l) \models \psi$.

Seja $\mathfrak{I}$ uma interpretação tal que $\mathfrak{I}(x_i) = a_i$, $1 \leq i \leq l$. Por definição, temos que

$$\begin{array}{lll} (\mathfrak{A}, \overline{a}_l) \models \psi & sse & \mathfrak{I} \models \exists v_1 \ldots \exists v_m \big[rifp_{X,Y}\phi(X,Y) \big](\overline{t}_r) \\ & sse & \text{existe } b_i \in A \text{ para todo } 1 \leq i \leq m \\ & & \text{ tal que } \mathfrak{I}\frac{b_1 \ldots b_m}{v_1 \ldots v_m} \models \big[rifp_{X,Y}\phi(X,Y) \big](\overline{t}_r) \\ & sse & \mathfrak{I}(\overline{t}_r) \in \bigcup \mathbf{rifp}(\mathcal{R}^{\varphi(X,Y),\mathfrak{I}}). \end{array} \tag{5.12}$$

A máquina N_ψ funcionará da seguinte maneira. A cada instante, a fita conterá a codificação binária de $\mathfrak{A}$ seguida da codificação binária da tupla de elementos $\overline{a}_l$ e de duas relações V_1 e V_2 que são interpretações de X e Y. Inicialmente a máquina N_ψ adivinhará $b_1, \ldots, b_l$ não-determinísticamente o que pode ser feito em $\mathrm{O}(\log n)$ passos. Pelo Teorema 2.5.4, existe uma máquina de Turing $N_\phi \in \mathrm{L}$ que avalia a fórmula $\phi(X,Y)$ em $(\mathfrak{A}, V_1, V_2)$. A máquina N_ψ simula a máquina N_ϕ a fim de verificar se o par (V_1, V_2) está na relação $\mathcal{R}^{\phi,\mathfrak{I}}$.

(1) Se a interpretação de X for um ponto fixo de $\mathcal{R}^{\phi,\mathfrak{I}}$, isto é, caso o par (V_1, V_1) esteja em $\mathcal{R}^{\phi,\mathfrak{I}}$, a máquina N_ψ verifica se a tupla de elementos $\mathfrak{I}(\overline{t}_r)$ está em V_1, terminando em estado de aceitação caso afirmativo e em estado de rejeição caso contrário. (2) Se (V_1, V_1) não estiver em $\mathcal{R}^{\phi,\mathfrak{I}}$, a máquina N_ψ adivinha um valor para V_2 não-deterministicamente o que pode ser feito com uma quantidade polinomial de passos e verifica se o par (V_1, V_2) está em $\mathcal{R}^{\phi,\mathfrak{I}}$. Caso negativo a execução é encerrada em estado de rejeição. Caso positivo, a codificação de V_1 é substituída pela de $V_1 \cup V_2$. Neste ponto, N_ψ volta ao ponto (1) em que verifica se a interpretação de $V_1 \cup V_2$ é ponto fixo de $\mathcal{R}^{\phi,\mathfrak{I}}$ e repete o procedimento.

∎

O seguinte algoritmo descreve sucintamente o funcionamento da máquina N_ψ:

(1) Escolhe $\overline{b}$ não-deterministicamente
(2) $V_1 \leftarrow \emptyset$
(3) **Se** $\mathfrak{A} \vDash \phi(X, X)[V_1, V_1]$ **retorna** s_-
(4) Escolhe V_2 não-deterministicamente
(5) **Se** $\mathfrak{A} \vDash \phi(X, Y)[V_1, V_2]$
(6) **então** **Se** $\overline{a} \in V_2$
(7) **então retorna** s_+
(8) **senão** $V_1 \leftarrow V_1 \cup V_2$ e GO TO (3)
(9) **senão retorna** s_-

Lema 5.2.6 (NP $\subseteq$ RIFP$_1$) *Seja $U \subseteq \mathcal{O}[S]$ uma classe de estruturas ordenadas. Mostraremos que, se U está em NP, então U é axiomatizável em RIFP$_1$.*

Prova. Semelhante à prova do Teorema 2.5.1 podemos assumir que N_ψ possui um limite de passos de n^d para um d adequado.

Seja $N = (Q, \Sigma, q_0, F^+, F^-, \Delta)$ uma máquina de Turing não-determinística que computa U, onde Σ é o alfabeto, Q é o conjunto de estados, $\Delta \subseteq (Q \times \Sigma) \times (Q \times \Sigma \times \{-1, 0, +1\})$ é a tabela de transição, q_0 é o estado inicial e $F = F^+ \cup F^-$ é o conjunto de estados finais. Denotamos por $\Delta(q, \sigma)$ o conjunto de triplas (q', σ', m) tal que $(q, \sigma, q', \sigma', m) \in \Delta$. Podemos assumir que qualquer execução não-determinística de N tem tamanho no máximo n^d para algum d.

Representaremos todas as informações referentes a uma configuração utilizando apenas uma relação, isto é, conteúdo da fita, estado atual e posição da cabeça de leitura e escrita. Além disso, as relações consideradas armazenarão uma sequência de configurações sucessivas da máquina de Turing não-determinística. Sem perda de generalidade, assumiremos que os estados da máquina correspondem aos elementos $Q = \{0, \ldots, e\}$ da estrutura e o alfabeto corresponde aos elementos $\Sigma = \{0, \ldots, l\}$ da estrutura.

As relações representando as configurações terão aridade $2d + 2$. Considere a seguinte tupla $\overline{t}zy\overline{x}$ de tamanho $2d + 2$. As primeiras d variáveis $\overline{t}$ representam o tempo, as duas posições seguintes zy indicam o tipo de informação que está sendo representada e as variáveis $\overline{x}$ restantes representam uma posição na fita. Temos três casos:

1. $z = 0$: a informação y armazenada pela tupla é o estado da máquina no tempo $\bar{t}$, neste caso, y é o estado atual e $\overline{x}$ é $\overline{0}$.
2. $z = 1$: a informação y armazenada é o símbolo escrito na posição $\overline{x}$ da fita no tempo $\bar{t}$.
3. $z = 2$: a informação y representada é a posição $\overline{x}$ da cabeça no tempo $\bar{t}$. Neste caso, o valor de y é 0.

Construiremos a fórmula ψ_N da forma

$$\psi_N \quad = \quad \exists \bar{t} \left[rifp_{X,Y} \phi(X,Y) \right] (\bar{t}, 1, s_+, \overline{0}) \tag{5.13}$$

onde s_+ representa um estado de aceitação. A fórmula ψ_N deverá ser satisfeita por uma estrutura $\mathfrak{A}$ se $code(\mathfrak{A}, <)$ for aceito pela máquina N.

Agora, descreveremos a fórmula $\phi(X,Y)$ que define as duas condições para que a relação Y seja a relação sucessora da relação X. As condições são as seguintes:

(i.) X está vazio e Y representa a configuração inicial;

(ii.) X contém todas as configurações até algum tempo $\bar{t}$ e Y conterá X e a próxima configuração correspondente ao tempo $\bar{t} + 1$.

As condições descritas acima estão expressas na fórmula (5.14).

$$\begin{array}{rcl} \phi(X,Y) & = & \left[(X = \emptyset) \wedge \psi_1(Y) \right] \vee \\ & & \left[(X \neq \emptyset) \wedge \psi_2(X,Y) \right] \end{array} \tag{5.14}$$

onde $\psi_1(Y)$ afirma que Y é a configuração inicial e $\psi_2(X,Y)$ expressa o fato que Y contém a próxima configuração após a última configuração em X.

A subfórmula ψ_1 que ocorre na fómula (5.14) é definida na fórmula (5.15), e afirma que as configurações que compõem Y são aquelas que contêm a configuração inicial C_0.

$$\begin{array}{rcll} \psi_1(Y) & = & \forall \bar{t}_d \forall \overline{x}_{d+2} & \Big[\\ & & & Y \bar{t} \overline{x} \leftrightarrow \\ & & & \left(\bar{t} = \overline{0} \wedge \text{START} \right) \\ & & & \Big] \end{array} \tag{5.15}$$

onde a fórmula START descreve o estado inicial da máquina:

$$\begin{aligned}\text{START} \quad = \quad & Y(\overline{0},0,q_0,\overline{0}) \wedge Y(\overline{0},2,0,\overline{0}) \wedge \bigwedge_{\sigma\in\Sigma}(\beta_\sigma(\overline{x}) \rightarrow Y(\overline{0},1,\sigma,\overline{x})) \\ & \wedge \textstyle\bigwedge_{q\neq q_0\in Q} \neg Y(\overline{0},0,q,\overline{0}) \\ & \wedge \forall\overline{x}\left(\textstyle\bigwedge_{\sigma\neq\sigma'\in\Sigma} \neg(Y(\overline{0},1,\sigma,\overline{x}) \wedge Y(\overline{0},1,\sigma',\overline{x})\right) \\ & \wedge \neg\exists\overline{x}(\overline{x}\neq\overline{0} \wedge Y(\overline{0},2,0,\overline{x}))\end{aligned} \tag{5.16}$$

A primeira linha da fórmula START afirma que o estado no tempo $\overline{0}$ é q_0, a posição da cabeça de leitura e escrita é $\overline{0}$ e que a codificação da estrutura está escrita na fita. A fórmula $\beta_\sigma(\overline{x})$ (Definição 2.5.3) é satisfeita se o símbolo σ está na posição $\overline{x}$ no instante inicial. Em particular, $\beta_1(\overline{x})$ é satisfeita se, e somente se, $code(\mathfrak{A},<)$ possui 1 na posição $\overline{x}$ e $\beta_0(\overline{x})$ é satisfeita se, e somente se, $code(\mathfrak{A},<)$ possui 0 na posição $\overline{x}$. As últimas três linhas da fórmula START garantem a consistência da relação como representação do estado inicial, ou seja, afirma que no instante $\overline{0}$ o estado atual é único, não há mais de um símbolo na mesma posição da fita, a cabeça de leitura e escrita não está em mais de um lugar ao mesmo tempo.

A subfórmula $\psi_2(X,Y)$ da fórmula (5.14) verifica três condições que devem ser satisfeitas por Y. Essas condições são:

(i.) Todas as configurações de X estão em Y;

(ii.) Y contém uma configuração sucessora, digamos C, após a última configuração em X;

(iii.) Y não contém nenhuma configuração após C.

A fórmula $\psi_2(X,Y)$ tem a seguinte forma:

$$\begin{aligned}\psi_2(X,Y) \quad = \quad & \forall\bar{t}\big[(\,C_1(\bar{t},X) \;\rightarrow\; \gamma_1(\bar{t},X,Y)\,) \wedge \\ & (\,C_2(\bar{t},X) \;\rightarrow\; \gamma_2(\bar{t},X,Y)) \wedge \\ & \big((\,\neg C_1(\bar{t},X) \;\wedge\; \neg C_2(\bar{t},X)\,) \;\rightarrow\; \gamma_3(\bar{t},X,Y)\big)\big].\end{aligned} \tag{5.17}$$

onde a expressão $\forall\bar{t}\big[C_1(\,\bar{t},X) \;\rightarrow\; \gamma_1(\bar{t},X,Y)\,)\big]$ significa que todas as configurações em X devem estar em Y, a expressão $\forall\bar{t}\big[(\,C_2(\bar{t},X) \;\rightarrow\; \gamma_2(\bar{t},X,Y))\big]$ afirma a condição (ii) que Y contém uma configuração imediatamente sucessora

da última configuração em X, e, finalmente, a expressão $\forall \bar{t}\big[\big(\ (\ \neg C_1(\bar{t}, X)\ \wedge\ \neg C_2(\bar{t}, X)\ )\ \rightarrow\ \gamma_3(\bar{t}, X, Y)\big)\big]$ afirma a condição (iii), i.e., que se não for o caso (i) ou o caso (ii) Y não deve conter qualquer outra configuração.

A subfórmula $C_1(\bar{t}, X) \rightarrow\ \gamma_1(\bar{t}, X, Y)\ )$ expressa a condição (i), onde

$$C_1(\bar{t}, X) = (\exists \overline{x}\ X\bar{t}\overline{x}) \tag{5.18}$$

e

$$\gamma_1(\bar{t}, X, Y) = \forall z \forall y \forall \overline{x}\ (X\bar{t}zy\overline{x}\ \leftrightarrow Y\ \bar{t}zy\overline{x}). \tag{5.19}$$

A condição (ii) está expressa na fórmula $C_2(\bar{t}, X) \rightarrow\ \gamma_2(\bar{t}, X, Y)$ onde

$$C_2(\bar{t}, X) = \big[(\neg\, \exists z \exists y \exists \overline{x}\, (X\bar{t}zy\overline{x})\ ) \wedge (\ \exists \overline{u}\, (Suc^d\, \overline{u}\bar{t}) \wedge \exists z \exists y \exists \overline{x}\, (X\overline{u}zy\overline{x})\ )\big], \tag{5.20}$$

dizemos que $\bar{t}$ é o instante de tempo correspondente à próxima configuração após a última configuração em X. $Suc^d \overline{u}\bar{t}$ diz que a tupla $\bar{t}$ é o sucessor de $\overline{u}$ na ordem lexicográfica de d-tuplas induzidas pela ordem da estrutura de entrada (lembre-se de que estamos considerando estruturas ordenadas por simplicidade). Seja

$$\gamma_2(\bar{t}, X, Y)\ =\ \exists \overline{u}\ \big(\ Suc^d \overline{u}\bar{t}\ \wedge \text{COMPUTE}), \tag{5.21}$$

onde COMPUTE é definida como o fecho (de primeira ordem) universal da conjunção NOCHANGE $\wedge$ CHANGE, quantificando as variáveis livres de NOCHANGE e CHANGE diferentes de $\bar{t}$ e $\overline{u}$,

$$\text{NOCHANGE} = \bigwedge_{\sigma \in \Sigma} \big[\big(X(\overline{u}, 2, 0, \overline{w}) \wedge X(\overline{u}, 1, \sigma, \overline{x}) \wedge (\overline{w} \neq \overline{x})\big) \leftrightarrow Y(\bar{t}, 1, \sigma, \overline{x})\big] \tag{5.22}$$

e

$$\text{CHANGE} = \bigvee_{q \in Q,\ \sigma \in \Sigma, (q', \sigma', m) \in \delta(q, \sigma)} (\text{PRE}[q, \sigma] \wedge \text{POST}[q', \sigma', m])\,, \tag{5.23}$$

onde

$$\mathrm{PRE}[q,\sigma] = X(\overline{u},0,q,\overline{0}) \ \wedge \ X(\overline{u},2,0,\overline{x}) \ \wedge \ X(\overline{u},1,\sigma,\overline{x}) \tag{5.24}$$

e

$$\begin{array}{rcl} \mathrm{POST}[q',\sigma',m] & = & Y(\overline{t},0,q',\overline{0}) \ \wedge \ Y(\overline{t},1,\sigma',\overline{x}) \wedge \\ & & \exists \overline{y} \Big((\overline{x} \ + \ m \ = \ \overline{y}) \ \wedge \ Y(\overline{t},2,0,\overline{y}) \wedge \\ & & \forall \overline{x'}((\overline{x'} \neq \overline{y}) \rightarrow \neg Y(\overline{t},2,0,\overline{x'})) \Big) \wedge \\ & & \bigwedge_{q'' \neq q' \in Q} \neg Y(\overline{t},0,q'',\overline{0}) \wedge \\ & & \bigwedge_{\sigma'' \neq \sigma' \in \Sigma} \neg Y(\overline{t},1,\sigma'',\overline{x}). \end{array} \tag{5.25}$$

A fórmula NOCHANGE descreve o fato de que o conteúdo da fita permanece o mesmo nas posições onde a cabeça de leitura e escrita não se encontra, condição expressa pela subfórmula $(\overline{w} \neq \overline{x}) \wedge X(\overline{u},2,0,\overline{w})$. A fórmula CHANGE contempla o caso em que a cabeça de leitura e escrita está na posição $\overline{x}$ e expressa as alterações do estado, do conteúdo da fita e da posição da cabeça de leitura e escrita. Cada cláusula da disjunção corresponde a uma instrução da fita, isto é, um possível estado atual q e símbolo lido σ e uma possível escolha não-determinística do próximo estado q', novo símbolo escrito σ' e um movimento da cabeça de leitura escrita. As relações X e Y satisfazem as fórmulas $\mathrm{PRE}[q,\sigma]$ e $\mathrm{POST}[q',\sigma',m]$ se forem configurações consecutivas. Note que $\overline{u}$ e $\overline{t}$ representam instantes de tempo sucessivos. A fórmula $\mathrm{PRE}[q,\sigma]$ indica que no tempo $\overline{u}$ a cabeça está na posição $\overline{x}$ lendo o símbolo σ. A fórmula $\mathrm{POST}[q',\sigma',m]$ exprime o fato de que, após escolher não-deterministicamente $(q',\sigma',m) \in \delta(q,\sigma)$, a máquina escreve σ' na posição atual, vai para o estado q' e move a cabeça de leitura de acordo. Além disso, a fórmula também proíbe a entrada de tuplas na relação Y que não sejam aquelas obtidas via aplicação de uma instrução, por exemplo, uma tupla indicando que a cabeça esteja em outra posição que não seja aquela obtida a partir da posição anterior e do movimento m. A fórmula $\overline{x} + m = \overline{y}$ abrevia uma fórmula de primeira ordem que afirma que $\overline{y}$ é a nova posição da cabeça após o movimento m.

A última condição (iii) é expressa pela fórmula

$$\neg \left(C_1(\overline{t},X) \ \vee \ C_2(\overline{t},X) \right) \rightarrow \gamma_3 \left(\overline{t},X,Y \right) \tag{5.26}$$

onde

$$\gamma_3(\bar{t}, X, Y) = \neg\, \exists z \exists y \exists \overline{x}\, (Y\bar{t}\overline{x}). \tag{5.27}$$

∎

Mostraremos como estender o resultado acima para estruturas não-ordenadas. Seja

$$\psi = [rifp_{X,Y}\phi(<, X, Y)](\overline{u}),$$

onde X e Y tem aridade $k \geq 2$, e $\phi(X, Y)$ usa uma relação de ordem, digamos, $<$. Aumentamos a aridade de X e Y e usamos para armazenar uma ordem. Usamos as posições extras como um índice para uma posição na ordem. Seja X' e Y' variáveis relacionais de aridade $k+1$ e $\overline{0}$ uma tupla de $k-1$ zeros. Representaremos a fórmula $x < y$ como $Y'\overline{0}xy$. As fórmulas da forma $X\bar{t}$ e $Y\bar{t}$ serão representadas por $X'1\bar{t}$ e $Y'\bar{t}$. Seja $\phi'(X, Y)$ a seguinte fórmula

$$\phi'(X', Y') = \phi(<, X, Y)[< /Y'\overline{0}\_; /X'1\_/X\_; Y'1\_/Y\_], \tag{5.28}$$

obtida de $\phi(<, X, Y)$ substituindo cada fórmula da forma $t < t'$ por $Y'\overline{0}tt'$, $X\bar{t}$ por $X'1\bar{t}$ e $Y\bar{t}$ por $Y'1\bar{t}$. Note que ϕ' não contém a relação de ordem. Seja $Ord(<)$ uma fórmula de primeira ordem que afirma que $<$ é uma relação de ordem. Seja $Ord(X')$ (respectivamente $Ord(Y')$ obtido de $Ord(<)$ substituindo $t < t'$ por $X'\overline{0}tt'$ (respectivamente $Y'\overline{0}tt'$).

Seja $\phi''(X', Y')$ a fórmula

$$\phi''(X', Y') = Ord(Y') \wedge \phi'(X', Y'), \tag{5.29}$$

e

$$\psi' = [\mathbf{rifp}_{X',Y'}\phi''(X', Y')](1\overline{u}). \tag{5.30}$$

Temos que

$$\exists < (Ord(<) \wedge \psi) \equiv \psi'. \tag{5.31}$$

Esta técnica pode ser usada para eliminar a relação de ordem da fórmula ψ_N do Lema 5.2.6, uma vez que esta fórmula é ordem invariante, i.e., não depende de uma ordem explícita da relação de ordem atual, mas simplesmente do fato

que é uma relação de ordem. Precisamente, se $<$ e $<'$ são ordenações sobre $\mathfrak{A}$, então $(\mathfrak{A}, <) \models \psi_N$ se, e somente se, $(\mathfrak{A}, <') \models \psi_N$.

O Teorema 5.2.3 segue imediatamente dos Lemas 5.2.5 e 5.2.6 e da eliminação da relação de ordem exposta acima.

Capítulo 6

$\mathbf{RIFP}_1$ *versus* NIFP

Neste capítulo veremos a lógica de ponto fixo não-determinístico inflacionário (NIFP) que foi apresentada em [AVV97], para compará-la com a lógica RIFP_1. Mostraremos uma tradução da lógica NIFP para a lógica RIFP. Depois, mostraremos, usando um resultado da lógica infinitária $L^{\omega}_{\infty\omega}$, que RIFP_1 não é equivalente a NIFP.

6.1 Teoria do Ponto Fixo Não-determinístico

As lógicas de ponto fixo funcionais apresentadas até aqui possuem iteração sequencial e determinística. Porém, existem problemas que não podem ser expressos por tais lógicas. Nessa seção apresentaremos o modo não-determinístico do operador de ponto fixo inflacionário. Aqui todos os operadores considerados são definidos sobre o conjunto das partes de conjuntos finitos.

Para tanto, sejam $\Phi_1, \Phi_2 : \mathcal{P}(A^k) \to \mathcal{P}(A^k)$ dois operadores inflacionários, onde A é um conjunto finito. Esse par de operadores gera sequências convergentes de estágios que são obtidos por sucessivas aplicações desses operadores até

a convergência ser atingida por ambos. Isto é, o par gera sequências da forma,

$$S_0,\ S_1,\ \ldots,\ S_m \quad \text{onde} \quad \begin{array}{lcll} S_0 & = & \emptyset; & \\ S_{i+1} & = & \Phi_1(S_i) & \text{ou} \\ S_{i+1} & = & \Phi_2(S_i) & \end{array} \tag{6.1}$$

onde $S_i \subseteq S_{i+1}$, e

$$\Phi_1(S_m) \quad = \quad \Phi_2(S_m) \quad = \quad S_m. \tag{6.2}$$

Definição 6.1.1 (Ponto fixo não-determinístico local) *Um conjunto $S \subseteq A^k$ é um* **ponto fixo não-determinístico local** *de Φ_1, Φ_2 se, e somente se, existe uma sequência da forma descrita em 6.1 acima tal que $S = S_m$. Denotaremos por* $\mathbf{nifp}(\Phi_1, \Phi_2)$ *o conjunto dos pontos fixos não-determinísticos locais de Φ_1, Φ_2.*

Definição 6.1.2 (Ponto fixo não-determinístico) *Definiremos o* **ponto fixo não-determinístico** *do par de operadores Φ_1 e Φ_2 como a união de todos os pontos fixos não-determinísticos locais do par de operadores Φ_1 e Φ_2,*

$$\bigcup \mathbf{nifp}(\Phi_1, \Phi_2) \tag{6.3}$$

[AVV97]. □

6.2 Lógica de Ponto Fixo Não-determinístico

A lógica de ponto fixo não-determinístico inflacionário é a lógica de primeira ordem estendida através de um operador que define o ponto fixo não-determinístico de dois operadores inflacionários de primeira ordem.

O não-determinismo descrito em [AVV97] consiste na escolha do operador

que será aplicado em cada estágio da sequência 6.1.

Definiremos a seguir a linguagem da lógica NIFP.

Definição 6.2.1 (Fórmulas) *As fórmulas em NIFP são fórmulas de FO ou fórmulas da forma*

$$\left[nifp_{S,\overline{x}}\left(\phi_1(S,\overline{x}),\phi_2(S,\overline{x})\right)\right](\overline{t}) \tag{6.4}$$

onde $\phi_1(S,\overline{x})$ e $\phi_2(S,\overline{x})$ são fórmulas de primeira ordem, com $\overline{x} = x_1,\ldots,x_k$, $\overline{t} = t_1,\ldots,t_k$ e S é um símbolo relacional com aridade k.

□

A semântica de NIFP é definida como

$$\begin{array}{c} \mathfrak{I} \models \left[nifp_{S,\overline{x}}\phi_1(S,\overline{x}),\phi_2(S,\overline{x})\right](\overline{t}) \\ \text{sse} \\ (t_1^{\mathfrak{I}},\ldots,t_k^{\mathfrak{I}}) \in \bigcup \mathbf{nifp}(\phi_1^{\mathfrak{I}},\phi_2^{\mathfrak{I}}). \end{array} \tag{6.5}$$

Na próxima seção, mostraremos a tradução de NIFP para RIFP.

6.3 RIFP$_1$ *versus* NIFP

Em [AVV97], é mostrado que NIFP captura NP sobre estruturas ordenadas, então, usando o Teorema 5.2.3 e o Teorema de Fagin, temos que NIFP tem o mesmo poder expressivo de RIFP$_1$ sobre estruturas ordenadas. Na sequência, apresentaremos uma tradução de NIFP para RIFP$_1$.

Sejam $\Phi_1,\Phi_2 : \mathcal{P}(A^k) \to \mathcal{P}(A^k)$ dois operadores. Eles dão origem à seguinte relação binária,

$$\mathcal{R}_{\Phi_1,\Phi_2} \quad = \quad \{(X,\Phi_1(X)) \mid X \in \mathcal{P}(A^k)\} \cup \{(X,\Phi_2(X)) \mid X \in \mathcal{P}(A^k)\}. \tag{6.6}$$

Usaremos a relação $\mathcal{R}_{\Phi_1,\Phi_2}$ e o operador de RIFP para definir o ponto fixo não-determinístico de Φ_1,Φ_2.

Um ponto fixo local X do par Φ_1, Φ_2 é um ponto fixo de $\mathcal{R}_{\Phi_1,\Phi_2}$, pois $\Phi_1(X) = \Phi_2(X) = X$, mas, se $X = \Phi_1(X) \neq \Phi_2(X)$ ou $X = \Phi_2(X) \neq \Phi_1(X)$, então X é um ponto fixo de $\mathcal{R}_{\Phi_1,\Phi_2}$ mas não é um ponto fixo local de Φ_1, Φ_2. Temos os seguintes casos:

i. Caso 1: $X = \Phi_1(X) = \Phi_2(X)$ é um ponto fixo local de Φ_1, Φ_2 e um ponto fixo de $\mathcal{R}_{\Phi_1,\Phi_2}$;

ii. Caso 2: $X = \Phi_1(X)$ e $X \neq \Phi_2(X)$ é um ponto fixo de $\mathcal{R}_{\Phi_1,\Phi_2}$, mas não é ponto fixo local de Φ_1, Φ_2;

iii. Caso 3: $X \neq \Phi_1(X)$ e $X = \Phi_2(X)$, similar ao caso anterior;

iv. Caso 4: $X \neq \Phi_1(X)$ e $X \neq \Phi_2(X)$ não é nem ponto fixo local de Φ_1, Φ_2 nem um ponto fixo de $\mathcal{R}_{\Phi_1,\Phi_2}$.

Temos que definir uma relação que coincida com $\mathcal{R}_{\Phi_1,\Phi_2}$, exceto que não tenha os pontos fixos correspondentes aos casos 2 e 3 acima. Definiremos a fórmula C_1 que expressa o fato de que X é um ponto fixo de Φ_1 e a fórmula C_2 para o caso em que X é um ponto fixo de Φ_2.

$$C_1(X) \quad = \quad \forall \overline{x}(X\overline{x} \leftrightarrow \phi_1(\overline{x}, X)) \tag{6.7}$$

$$C_2(X) \quad = \quad \forall \overline{x}(X\overline{x} \leftrightarrow \phi_2(\overline{x}, X)) \tag{6.8}$$

onde ϕ_1 e ϕ_2 são fórmulas de primeira ordem que definem os operadores Φ_1 e Φ_2.

Podemos usar essas fórmulas para incluir o par $(X, \Phi_i(X))$ se X não for um ponto fixo de Φ_i, $i = 1, 2$, ou se for um ponto fixo de ambos Φ_1 e Φ_2. Seja $\psi(X, Y)$ a fórmula a seguir:

$$\begin{array}{rcl} \psi(X,Y) & = & \big[\neg C_1(X) \wedge \forall \overline{x} Y\overline{x} \ \leftrightarrow \ \phi_1(\overline{x}, X)\big] \vee \\ & & \big[\neg C_2(X) \wedge \forall \overline{x} Y\overline{x} \ \leftrightarrow \ \phi_2(\overline{x}, X)\big] \vee \\ & & \big[C_1(X) \wedge C_2(X) \wedge \forall \overline{x} X\overline{x} \ \leftrightarrow \ Y\overline{x}\big]. \end{array} \tag{6.9}$$

onde a expressão $\big[\neg C_1(X) \wedge \forall \overline{x} Y\overline{x} \leftrightarrow \phi_1(\overline{x}, X)\big]$ afirma que X é não é ponto fixo local (negando o fato de X ser um ponto fixo de Φ_1) e Y contém as tuplas que satisfazem a fórmula ϕ_1; similar ao caso anterior, a expressão $\big[\neg C_2(X) \wedge \forall \overline{x} Y\overline{x} \leftrightarrow$

$\phi_2(\overline{x}, X)\big]$ afirma que X não é ponto fixo local (negando o fato de X ser ponto fixo de Φ_2) e Y contém as tuplas que satisfazem a fórmula ϕ_2; e, finalmente, a expressão $\big[C_1(X) \wedge C_2(X) \wedge \forall\overline{x} X\overline{x} \leftrightarrow Y\overline{x}\big]$ garante que X é ponto fixo local de Φ_1, Φ_2 e que $X = Y$.

Um conjunto X é um ponto fixo inflacionário inicial de $\mathcal{R}^{\psi,\mathfrak{I}}$ se, e somente se, é um ponto fixo local de Φ_1, Φ_2. A fórmula

$$\phi = \big[\mathit{nifp}_{S,\overline{x}}\, \phi_1(S, \overline{x}), \phi_2(S, \overline{x})\,\big]\,(\overline{t}) \tag{6.10}$$

de NIFP é equivalente à fórmula

$$\psi = \big[\mathit{rifp}_{X,Y} \psi(X, Y)\big]\,(\overline{t}) \tag{6.11}$$

de RIFP. Como ϕ_1 e ϕ_2 são fórmulas de primeira ordem, então ψ é uma RIFP$_1$-fórmula.

O sentido contrário da tradução, i.e., de RIFP$_1$ para NIFP, não é possível, em geral, porque a equivalência entre RIFP$_1$ e NIFP vale somente para o caso das estruturas serem ordenadas. Para mostrarmos que NIFP é menos expressiva que RIFP$_1$ usaremos o fato de que a *query* EVEN (veja o Exemplo 4.3.1) é RIFP$_1$-definível, mas não é definível na lógica infinitária $L^{\omega}_{\infty\omega}$. Mostraremos que podemos traduzir a lógica NIFP em $L^{\omega}_{\infty\omega}$. E, portanto, temos que RIFP$_1$ $\not\subseteq$ NIFP.

Definiremos a seguir a lógica $L^{\omega}_{\infty\omega}$.

Definição 6.3.1 (Lógica Infinitária) *A lógica infinitária $L_{\infty\omega}$ permite disjunções de cardinalidade arbitrária. Seja S um vocabulário. A classe de $L_{\infty\omega}$ sobre S é dada pelas seguintes cláusulas:*

- *contém todas as fórmulas atômicas de primeira ordem sobre S*
- *se ϕ é uma fórmula então $\neg\phi$ também é fórmula*
- *se ϕ é uma fórmula e x uma variável então $\exists x\phi$ é uma fórmula*

- *se Ψ é um conjunto de fórmulas então $\bigvee \Psi$ é uma fórmula.*

A semântica de $L_{\infty\omega}$ é uma extensão direta da semântica de lógica de primeira ordem com $\bigvee \Psi$ sendo interpretada como uma disjunção sobre todas as fórmulas em Ψ; logo,

$$\mathfrak{I} \models \bigvee \Psi \text{ sse para algum } \psi \in \Psi, \mathfrak{I} \models \psi \tag{6.12}$$

A lógica $L^{\omega}_{\infty\omega}$ corresponde às fórmulas de $L_{\infty\omega}$ onde cada fórmula contém uma quantidade finita de variáveis. [EF99].

Teorema 6.3.1 ([GKL$^+$07]) *A consulta* EVEN *não é $L^{\omega}_{\infty\omega}$-definível sobre $\mathcal{G}$, onde $\mathcal{G}$ é classe de todos os grafos finitos.*

Mostraremos que a lógica NIFP está contida na lógica $L^{\omega}_{\infty\omega}$. Para isso, mostraremos como definir os pontos fixos não-determinísticos locais de operadores inflacionários de primeira ordem na lógica $L^{\omega}_{\infty\omega}$.

Uma **sequência de aplicação** é uma sequência infinita $w = w_1, w_2, \ldots$ tal que $w_i \in \{1, 2\}$, para $i \in \mathbb{N}^*$, onde tanto o número 1 quanto o número 2 aparecem infinitas vezes. Sejam $\Phi_1, \Phi_2 : \mathcal{P}(A^k) \to \mathcal{P}(A^k)$ operadores inflacionários e w uma sequência de aplicação. A **sequência de estágios de Φ_1, Φ_2 baseada em** w é a sequência infinita $(\Phi_1, \Phi_2)^w = S_0, S_1, \ldots$ de subconjuntos de A^k tais que $S_0 = \emptyset$ e $S_{i+1} = \Phi_{w_{i+1}}(S_i)$. Chamamos o conjunto S_i de **i-ésimo estágio da sequência** $(\Phi_1, \Phi_2)^w$ e denotaremos por $(\Phi_1, \Phi_2)^w_i$.

Uma sequência de estágios sempre alcança um ponto fixo não-determinístico local de Φ_1, Φ_2 em algum estágio $(\Phi_1, \Phi_2)^w_m$.

Lema 6.3.2 *Sejam $\Phi_1, \Phi_2 : \mathcal{P}(A^k) \to \mathcal{P}(A^k)$ operadores inflacionários. Para toda sequência de aplicação w existe um $m \in \mathbb{N}$ tal que $(\Phi_1, \Phi_2)^w_m$ é ponto fixo não-determinístico local de Φ_1, Φ_2 e, para todo ponto fixo não-determinístico*

local S, existe uma sequência de aplicação w tal que $S = (\Phi_1, \Phi_2)^w_m$ para algum m.

Prova. Seja w uma sequência de aplicação e seja $(\Phi_1, \Phi_2)^w$ a sequência de estágios baseada em w. Como A é finito, existe um $m \in \mathbb{N}$ tal que $(\Phi_1, \Phi_2)^w_n = (\Phi_1, \Phi_2)^w_m$ para todo $n \geq m$. Como em uma sequência de aplicação os números 1 e 2 ocorrem infinitas vezes, temos que $\Phi_1\big((\Phi_1, \Phi_2)^w_m\big) = \Phi_2\big((\Phi_1, \Phi_2)^w_m\big)$. Logo, $(\Phi_1, \Phi_2)^w_m$ é um ponto fixo não-determinístico local.

Seja S um ponto fixo não-determinístico local de Φ_1, Φ_2. Logo, existe uma sequência $S_0, \ldots, S_m$ tal que $S_0 = \emptyset$, $S_{i+1} = \Phi_1(S_i)$ ou $S_{i+1} = \Phi_2(S_i)$ e $S_m = \Phi_1(S_m) = \Phi_2(S_m) = S$. Seja w uma sequência de aplicação tal que $S_{i+1} = \Phi_{w_{i+1}}(S_i)$. Logo, $S = (\Phi_1, \Phi_2)^w_m$. ∎

Sejam $\phi_1(X, \overline{x})$ e $\phi_2(X, \overline{x})$ fórmulas de primeira ordem e seja w uma sequência de aplicação. Mostraremos que cada estágio de uma sequência de estágios pode ser definido por uma fórmula de primeira ordem.

Lema 6.3.3 *Sejam $\phi_1(X, \overline{x})$ e $\phi_2(X, \overline{x})$ fórmulas de primeira ordem e seja w uma sequência de aplicação. Para todo $i \in \mathbb{N}$, existe uma fórmula de primeira ordem $\psi^w_i(\overline{x})$ tal que para toda interpretação $\mathfrak{I}$, $\psi^w_i(\overline{x})$ define o i-ésimo estágio $(\phi_1^{\mathfrak{I}}, \phi_2^{\mathfrak{I}})^w_i$ da sequência de estágios $(\phi_1^{\mathfrak{I}}, \phi_2^{\mathfrak{I}})^w$ baseada na sequência de aplicação w.*

Prova. Sejam $\phi_1(X, \overline{x})$ e $\phi_2(X, \overline{x})$ fórmulas cujas variáveis de primeira ordem estão entre $v_1, \ldots, v_s$. Seja $y_1 = v_{s+1}, \ldots, y_k = v_{k+s}$.
Então $\psi^w_i(\overline{x})$ pode ser definido indutivamente como

$$\psi^w_0(\overline{x}) = \neg x_1 = x_1 \tag{6.13}$$

$$\psi^w_{i+1}(\overline{x}) \;=\; \phi_{w_{i+1}}(X, \overline{x})\frac{\alpha_i(\_)}{X_\_} \tag{6.14}$$

onde a fórmula $\alpha_i(\overline{x})$ é

$$\alpha_i(\overline{x}) = \exists \overline{y}\big(\overline{y} = \overline{x} \wedge \exists \overline{x}(\overline{x} = \overline{y} \wedge \psi^w_i(\overline{x}))\big)$$

e $\psi^w_{i+1}(\overline{x})$ é obtida a partir de $\phi_{w_{i+1}}$ substituindo toda subfórmula da forma $X\bar{t}$ por $\alpha_i(\bar{t})$, para toda tupla de termos $\bar{t}$.

É fácil ver que

$$\mathfrak{I} \models \phi_{w_{i+1}}(X,\overline{x})\frac{\alpha_i(\_)}{X\_} \quad \text{sse} \quad \mathfrak{I}\frac{\alpha_i(\overline{x})^{\mathfrak{I}}}{X} \models \phi_{w_{i+1}}(X,\overline{x}). \tag{6.15}$$

Note que a fórmula $\alpha_i(\overline{x})$ é equivalente a $\psi^w_i(\overline{x})$. Logo, a relação $\alpha_i(\overline{x})^{\mathfrak{I}}$ definida por $\alpha_i(\overline{x})$ em uma interpretação $\mathfrak{I}$ é a mesma relação $\psi^w_i(\overline{x})^{\mathfrak{I}}$ definida por $\psi^w_i(\overline{x})$ em $\mathfrak{I}$. A fórmula $\alpha_i(\overline{x})$ é definida dessa forma para reaproveitar variáveis e manter o número de variáveis usadas menor ou igual a $k+s$.

Mostraremos que estágio $(\phi^{\mathfrak{I}}_1,\phi^{\mathfrak{I}}_2)^w_i$ é exatamente o conjunto

$$\psi^w_i(\overline{x})^{\mathfrak{I}} = \{\overline{a} \in A^k | \mathfrak{I}\frac{\overline{a}}{\overline{x}} \models \psi^w_i(\overline{x})\}$$

definido por $\psi^w_i(\overline{x})$ em $\mathfrak{I}$. Por indução em i. Para $i=0$ temos que $\psi^w_0(\overline{x})^{\mathfrak{I}} = \emptyset = (\phi^{\mathfrak{I}}_1,\phi^{\mathfrak{I}}_2)^w_0$. Por hipótese indutiva, suponha que $(\phi^{\mathfrak{I}}_1,\phi^{\mathfrak{I}}_2)^w_i = \psi^w_i(\overline{x})^{\mathfrak{I}}$. Por definição, temos que $(\phi^{\mathfrak{I}}_1,\phi^{\mathfrak{I}}_2)^w_{i+1} = \phi^{\mathfrak{I}}_{w_{i+1}}\big((\phi^{\mathfrak{I}}_1,\phi^{\mathfrak{I}}_2)^w_i\big)$. Temos também, por definição, que

$$\phi^{\mathfrak{I}}_{w_{i+1}}(\mathbf{X}) \;=\; \big\{\overline{a} \in A^k \mid \mathfrak{I}\frac{\mathbf{X},\overline{a}}{X,\overline{x}} \models \phi_{w_{i+1}}(X,\overline{x})\big\}. \tag{6.16}$$

Logo,

$$\phi^{\mathfrak{I}}_{w_{i+1}}\big((\phi^{\mathfrak{I}}_1,\phi^{\mathfrak{I}}_2)^w_i\big) \;=\; \big\{\overline{a} \in A^k \mid \mathfrak{I}\frac{(\phi^{\mathfrak{I}}_1,\phi^{\mathfrak{I}}_2)^w_i,\overline{a}}{X,\overline{x}} \models \phi_{w_{i+1}}(X,\overline{x})\big\}. \tag{6.17}$$

Mas, pela hipótese indutiva, temos que $(\phi^{\mathfrak{I}}_1,\phi^{\mathfrak{I}}_2)^w_i = \psi^w_i(\overline{x})^{\mathfrak{I}}$. Como $\alpha_i(\overline{x})$ é equivalente a $\psi^w_i(\overline{x})$, temos $\alpha_i(\overline{x})^{\mathfrak{I}} = (\phi^{\mathfrak{I}}_1,\phi^{\mathfrak{I}}_2)^w_i$. Logo

$$\phi^{\mathfrak{I}}_{w_{i+1}}\big(\alpha_i(\overline{x})^{\mathfrak{I}}\big) \;=\; \big\{\overline{a} \in A^k \mid \mathfrak{I}\frac{\alpha_i(\overline{x})^{\mathfrak{I}},\overline{a}}{X,\overline{x}} \models \phi_{w_{i+1}}(X,\overline{x})\big\}. \tag{6.18}$$

Logo,

$$\phi^{\mathfrak{I}}_{w_{i+1}}\big(\alpha_i(\overline{x})^{\mathfrak{I}}\big) \;=\; \big\{\overline{a} \in A^k \mid \mathfrak{I}\frac{\overline{a}}{\overline{x}} \models \phi_{w_{i+1}}(X,\overline{x})\frac{\alpha_i(\_)}{X\_}\big\}. \tag{6.19}$$

Logo,

$$\phi^{\mathfrak{J}}_{w_{i+1}}\big((\phi^{\mathfrak{J}}_1,\phi^{\mathfrak{J}}_2)^w_i\big) \;=\; \big\{\overline{a}\in A^k \mid \mathfrak{J}\frac{\overline{a}}{\overline{x}} \models \psi^w_{i+1}(\overline{x})\big\}. \tag{6.20}$$

Portanto,

$$(\phi^{\mathfrak{J}}_1,\phi^{\mathfrak{J}}_2)^w_{i+1} \;=\; \psi^w_{i+1}(\overline{x})^{\mathfrak{J}}. \tag{6.21}$$

∎

Pelo Lema 6.3.2, toda sequência de aplicação leva a um ponto fixo não-determinístico local de um par de operadores inflacionários. Usaremos a lógica $L^{\omega}_{\infty\omega}$ para construir uma fórmula que expressa o ponto fixo não-determinístico local de um par de operadores inflacionários de primeira ordem.

Lema 6.3.4 *Sejam $\phi_1(X,\overline{x})$ e $\phi_2(X,\overline{x})$ fórmulas de primeira ordem e seja w uma sequência de aplicação. Existe uma fórmula $\psi^w(\overline{x})$ em $L^{\omega}_{\infty\omega}$ tal que para toda interpretação $\mathfrak{J}$, $\psi^w(\overline{x})$ define o ponto fixo não-determinístico local atingido pela sequência de estágios $(\phi^{\mathfrak{J}}_1,\phi^{\mathfrak{J}}_2)^w$ baseada na sequência de aplicação w.*

Prova. Seja $\psi^w(\overline{x})$ definida como

$$\psi^w(\overline{x}) \;=\; \bigvee\big\{\psi^w_i(\overline{x}) \mid i\in\mathbb{N}\big\} \tag{6.22}$$

Note que cada fórmula ψ^w_i possui variáveis entre $v_1,\ldots,v_{s+k}$, onde a aridade de X é k e as variáveis de ϕ_1 e ϕ_2 estão entre $v_1,\ldots,v_s$. Logo, a fórmula ψ^w possui $k+s$ variáveis.

Seja S o ponto fixo não-determinístico local alcançado pela sequência de estágios $(\phi^{\mathfrak{J}}_1,\phi^{\mathfrak{J}}_2)^w$. Como $\phi^{\mathfrak{J}}_1$ e $\phi^{\mathfrak{J}}_2$ são operadores inflacionários, a sequência $(\phi^{\mathfrak{J}}_1,\phi^{\mathfrak{J}}_2)^w$ é crescente, e, portanto S pertence à união de todos os estágios, i.e.,

$$S=\bigcup\big\{(\phi^{\mathfrak{J}}_1,\phi^{\mathfrak{J}}_2)^w_i \mid i\in\mathbb{N}\big\}. \tag{6.23}$$

Mostraremos que o conjunto S é igual ao conjunto $\psi^w(\overline{x})^{\mathfrak{J}}$ definido por $\psi^w(\overline{x})$ em $\mathfrak{J}$.

Seja $\overline{a} \in A^k$. Temos que $\overline{a} \in S$ se, e somente se, $\overline{a}$ pertence a $(\phi_1^{\mathfrak{J}}, \phi_2^{\mathfrak{J}})_i^w$ para algum i se, e somente se, $\mathfrak{I}\frac{\overline{a}}{\overline{x}} \models \psi_i^w(\overline{x})$ para algum i se, e somente se, $\mathfrak{I}\frac{\overline{a}}{\overline{x}} \models \psi^w(\overline{x})$ se, e somente se, $\overline{a} \in \psi^w(\overline{x})^{\mathfrak{J}}$. Portanto $S = \psi^w(\overline{x})^{\mathfrak{J}}$.

∎

Analogamente, existe uma fórmula ψ^{ϕ_1,ϕ_2} em $L^\omega_{\infty\omega}$ que define o ponto fixo não-determinístico dos operadores definidos por ϕ_1 e ϕ_2.

Lema 6.3.5 *Sejam $\phi_1(X, \overline{x})$ e $\phi_2(X, \overline{x})$ fórmulas de primeira ordem. Existe uma fórmula ψ^{ϕ_1,ϕ_2} em $L^\omega_{\infty\omega}$ tal que para toda interpretação $\mathfrak{I}$, ψ^{ϕ_1,ϕ_2} define o ponto fixo não-determinístico de $\phi_1^{\mathfrak{J}}$ e $\phi_2^{\mathfrak{J}}$.*

Prova. Seja ψ^{ϕ_1,ϕ_2} definida como

$$\psi^{\phi_1,\phi_2}(\overline{x}) \;=\; \bigvee\{\psi^w(\overline{x}) \mid \text{ para toda sequência de aplicação } w\} \tag{6.24}$$

Note que cada fórmula $\psi^w(\overline{x})$ possui variáveis entre $v_1, \ldots, v_{s+k}$. Logo, a fórmula $\psi^{\phi_1,\phi_2}(\overline{x})$ possui $k + s$ variáveis.

Seja $\bigcup \mathbf{nifp}(\phi_1^{\mathfrak{J}}, \phi_2^{\mathfrak{J}})$ a união dos pontos fixos não-determinísticos locais dos operadores $\phi_1^{\mathfrak{J}}$ e $\phi_2^{\mathfrak{J}}$. Temos que $\overline{a} \in \bigcup \mathbf{nifp}(\phi_1^{\mathfrak{J}}, \phi_2^{\mathfrak{J}})$ se, e somente se, pelo Lema 6.3.2, existe uma sequência de aplicação w e um $m \in \mathbb{N}$ tal que $(\phi_1^{\mathfrak{J}}, \phi_2^{\mathfrak{J}})_m^w$ é um ponto fixo não-determinístico local e $\overline{a} \in (\phi_1^{\mathfrak{J}}, \phi_2^{\mathfrak{J}})_m^w$ se, esomente se, pelo Lema 6.3.4, temos que $\mathfrak{I}\frac{\overline{a}}{\overline{x}} \models \psi^w(\overline{x})$ se, e somente se, $\mathfrak{I}\frac{\overline{a}}{\overline{x}} \models \psi^{\phi_1,\phi_2}(\overline{x})$ se, e somente se, $\overline{a} \in \psi^{\phi_1,\phi_2}(\overline{x})^{\mathfrak{J}}$. Portanto, $\bigcup \mathbf{nifp}(\phi_1^{\mathfrak{J}}, \phi_2^{\mathfrak{J}}) \;=\; \psi^{\phi_1,\phi_2}(\overline{x})^{\mathfrak{J}}$.

∎

Teorema 6.3.6 (NIFP $\subseteq L^\omega_{\infty\omega}$) *Toda fórmula de NIFP possui uma fórmula equivalente em $L^\omega_{\infty\omega}$.*

Prova. Seja α uma fórmula de NIFP. Temos dois casos: (i) α é de primeira ordem ou (ii) α é da forma $[nifp_{X,\overline{x}}\phi_1(X, \overline{x}), \phi_2(X, \overline{x})](\overline{t})$. Se α é de primeira ordem, então α pertence a $L^\omega_{\infty\omega}$. No caso (ii), temos que

$$\alpha \equiv \psi^{\phi_1,\phi_2}(\overline{t}). \tag{6.25}$$

Corolário 6.3.7 *A query* EVEN *não é NIFP-definível sobre $\mathcal{G}$, onde $\mathcal{G}$ é classe de todos os grafos finitos.*

Prova. Segue imediatamente dos Teoremas 6.3.1 e 6.3.6. ∎

Capítulo 7

Conclusões e Trabalhos Futuros

Nós investigamos a lógica de primeira ordem com o operador de ponto fixo relacional cuja semântica difere da semântica clássica das lógicas de ponto fixo usando relações no lugar de operadores. Chamamos de Lógica de Ponto Fixo Relacional Inflacionário (RIFP). Nossa abordagem generaliza as lógicas de ponto fixo tradicionais e leva a um acréscimo de poder expressivo.

Caracterizamos o poder expressivo de RIFP mostrando que RIFP é equivalente a SO, isto é, cada fórmula de RIFP tem uma fórmula equivalente em SO, e vice-versa. Isso mostra que RIFP captura a hierarquia polinomial (PH). Apresentamos as traduções que mapeiam fórmulas em uma linguagem para outra linguagem. Também investigamos o fragmento RIFP_1 de fórmulas da forma

$$\exists \overline{z}[\mathbf{rifp}_{X,Y}\phi(X,Y)](\overline{t}),$$

onde $\phi(X,Y)$ está em primeira ordem. Pequenas modificações na tradução nos dão que RIFP_1 é equivalente ao fragmento existencial de segunda ordem e, logo, captura NP, a classe de problemas resolvidos em tempo não-determinístico

polinomial.

Também comparamos nossa lógica com a lógica de ponto fixo não-determinístico inflacionário (NIFP). NIFP generaliza a lógica de ponto fixo tradicional usando um par de operadores para construir pontos fixos através de sequências obtidas por sucessivas aplicações dos operadores. NIFP captura NP sobre estruturas ordenadas [AVV97] e portanto tem o mesmo poder expressivo de RIFP_1 sobre a classe de estruturas ordenadas. Mostramos como construir uma RIFP_1-fórmula equivalente a uma NIFP-fórmula. Mostramos também que o contrário não vale e, portanto, RIFP_1 é estritamente mais expressiva que NIFP. Fizemos isso usando o fato que a consulta Even não pode ser expressa na lógica NIFP e que NIFP $\subseteq L^{\omega}_{\infty\omega}$, i.e., uma fórmula de NIFP pode ser definida na lógica infinitária $L^{\omega}_{\infty\omega}$.

Apresentamos os resultados deste trabalho em [FMF15] e tivemos uma artigo aceito referente aos nossos resultados mais recentes que será publicado em [FMF16].

A abordagem apresentada pode ser usada para explorar outras formas de lógicas de ponto fixo relacional. Da mesma forma que novas lógicas de ponto fixo são definidas impondo algumas condições sobre o operador usado, podemos definir outras lógicas de ponto fixo relacional restringindo as relações consideradas. Usando esta abordagem, várias lógicas podem ser definidas de uma maneira uniforme. Alguns próximos passos naturais nessa linha de pesquisa são:

- Definir uma versão de ponto fixo parcial para o operador relacional: a equivalência entre RIFP_1 e SO depende fortemente do fato que as relações sejam inflacionárias. Removendo esta condição podemos ter um acréscimo no poder expressivo.

- Investigar a restrição do fragmento monádico da lógica RIFP: o fragmento monádico de RIFP é obtido restringindo o operador **rifp** nas fórmulas $\phi(X, Y)$ onde X e Y são relações monádicas. A tradução de SO para RIFP mapeia fórmulas monádicas de SO para fórmulas monádicas de RIFP. A tradução de RIFP para SO não preserva a aridade das relações envolvidas.

- Analisar a aplicação dos operadores relacionais para lógicas de ordem superior além de SO.

Referências Bibliográficas

[AB09] S. Arora and B. Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[AHV95] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley Publishing Co., Reading, Massachusetts, 1995.

[AU79] A. V. Aho and J. D. Ullman. Universality of data retrieval languages. In *Proceedings of the 6th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 110–119. ACM, 1979.

[AV91] S. Abiteboul and V. Vianu. Datalog extensions for database queries and updates. *Journal of Computer and System Sciences*, 43(1):62–124, 1991.

[AVV97] S. Abiteboul, M. Vardi, and V. Vianu. Fixpoint logics, relational machines, and computational complexity. *Journal of the ACM*, 44 - 1:30–56, 1997.

[Büc60] J. R. Büchi. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6):66–92, 1960.

[CH80] A. K. Chandra and D. Harel. Structure and complexity of relational queries. *Foundations of Computer Science 21st Annual Symposium on*, 1980.

[CK90] C. C. Chang and H. J. Keisler. *Model Theory*. North-Holland, Amsterdam, 3rd edition, 1990.

[Cod70] E. F. Codd. A relational model for large shared data banks. *Communications of the ACM*, 13(6):377–387, June 1970.

[Ebb94] H. D. Ebbinghaus. *Mathematical logic*. Springer, 1994.

[EF99] H. D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 1999.

[End01] H. B. Enderton. *A mathematical introduction to logic*. Academic press, 2001.

[Fag74] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. *SLAM-AMS Proceedings*, 7:43–73, 1974.

[FMF15] M. R. F. Farias, A. T. C. Martins, and F. M. Ferreira. Logics with relational fixed-point operators. *Apresentado em 2nd FILOMENA: Philosophy, Logic and Analytical Metaphysics*, 2015.

[FMF16] M. R. F. Farias, A. T. C. Martins, and F. M. Ferreira. The descriptive complexity of logics with relational fixed-point. *Aceito para publicação LSFA*, 2016.

[GKL+07] E. Grädel, P. G. Kolaitis, L. Libkin, M. Marx, J. Spencer, M. Y. Vardi, Y. Venema, and S. Weinstein. *Finite Model Theory and its applications (Vol.2)*. Heidelberg: Springer, 2007.

[GS86] Y. Gurevich and S. Shelah. Fixed-point extensions of first-order logic. *Annals of pure and applied logic*, 32:265–280, 1986.

[Hod93] W. Hodges. *Model Theory*, volume 42 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, 1993.

[Imm82] N. Immerman. Relational queries computable in polynomial time. *Proceedings of the 14th Annual ACM Symposium on Theory of Computing*, 1982.

[Imm99] N. Immerman. *Descriptive Complexity*. Springer, 1999.

[Kre02] S. Kreutzer. *Pure and applied fixed-point logics*. PhD thesis, Bibliothek der RWTH Aachen, 2002.

[KV96] P. G. Kolaitis and M. Y. Vardi. On the expressive power of variable-confined logics. In *Logic in Computer Science, 1996. LICS'96. Proceedings., Eleventh Annual IEEE Symposium on*, pages 348–359. IEEE, 1996.

[Lib12] L. Libkin. *Elements of Finite Model Theory*. Springer, 2012.

[LP97] H. R. Lewis and C. H. Papadimitriou. *Elements of the Theory of Computation*. Prentice Hall PTR, 1997.

[Pap94] C. H. Papadimitriou. *Computational Complexity*. Addison Wesley Longman, 1994.

[Sto76] L. J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.

[Var82] M. Y. Vardi. The complexity of relational query languages. *Proceedings of the 14th Annual ACM Symposium on Theory of Computing*, pages 137–146, 1982.