Lógicas Modais e Complexidade Descritiva*

Cibele Matos Freire[1], **Ana Teresa Martins**[1]

[1]Departamento de Computação – Universidade Federal do Ceará (UFC)
Campus do Pici - Bloco 910 – 60.455-760 – Fortaleza – CE – Brasil

{cibelemf,ana}@lia.ufc.br

Abstract. *In the theory of Computational Complexity, we usually analyse the efficiency of an algorithm through the time and space needed to compute it. Such metrics induce the definition of classes of problems. In 1974, Fagin proved that the class NP is captured by existential second-order logic. This results showed that the complexity of a problem may have a machine-independent characterization, through the expressiveness of a language able to define a class which this problem belongs to. This theorem launched the area of Descriptive Complexity. Here, we are particularly interested in the role of logical modal operators from K, S4, S5, CTL and CTL* in the expression of decision problems and in the characterization of classes of problems.*

Resumo. *Na Complexidade Computacional, medimos a eficiência de um algoritmo pelo tempo e espaço dispendido em sua execução, induzindo a definição de classes de problemas. Em 1974, Fagin demonstrou que a classe NP é aquela cujos problemas são descritos pela lógica de segunda-ordem existencial. Este resultado mostrou que a complexidade de um problema pode ser caracterizado por um viés independente da máquina, através da expressividade de uma linguagem capaz de definir uma classe a qual este problema pertence, e lançou a área de Complexidade Descritiva. Estamos aqui interessados no papel dos operadores modais das lógicas K, S4, S5, CTL e CTL* na expressão de problemas de decisão e caracterização de classes de problemas.*

Palavras Chave*: Representação do conhecimento, Expressividade de lógicas, Complexidade Computacional, Complexidade Descritiva.*

1. Introdução

Ao investigarmos os fundamentos da Ciência da Computação nos deparamos com três questões centrais: (1) O que é computar uma função, ou seja, como definir a noção de computação? (2) Quais são os limites da computação, isto é, o que dá e o que não dá para ser computado? (3) Como medir a eficiência de um algoritmo para solucionar um problema? enquanto a terceira pergunta é de interesse da área de Complexidade Computacional [Papadimitriou 1994]. As duas primeiras questões são respondidas no âmbito da Computabilidade [Davis et al. 1994]

Em geral, a medida de eficiência de um algoritmo é baseada no tempo e espaço dispendido em sua execução. A investigação sobre a eficiência de um algoritmo permite-nos a definição de uma hierarquia de classes de problemas. Quando esta hierarquia é baseada no tempo e espaço, identificamos classes importantes como a classe P, ou PTIME,

*Patrocinado por Bolsa de Mestrado do CNPq e projetos de pesquisa CNPq/PQ, CNPq/Universal 2008, CNPq/"Casadinho" 2008.

que contém todos os problemas para os quais existem algoritmos que os solucionam em tempo polinomial através de máquinas de Turing determinísticas, e a classe NP dos problemas solúveis em máquinas de Turing não determinísticas em tempo polinomial.

As medidas de tempo e espaço da complexidade computacional, embora sejam naturais do ponto de vista da engenharia, pois falam de recursos físicos necessários à computação, são pouco intuitivas na análise da complexidade inerente ao problema do ponto de vista matemático.

Em 1974, Fagin demonstrou que a classe de complexidade NP é exatamente aquela cujos problemas são descritos pelo fragmento existencial da lógica de segunda ordem ($\exists$SO) [Fagin 1974]. Este resultado é bastante importante pois mostrou que a complexidade de um problema pode ser caracterizada por um viés independente da máquina, através da expressividade de uma linguagem lógica necessária para especificar todos os problemas, e apenas aqueles, de uma classe.

Após o resultado inicial de Fagin, muitos outros foram obtidos levando-nos a crer que toda classe de complexidade computacional pode ser capturada precisamente por uma linguagem lógica. Esta área passou a ser conhecida como Complexidade Descritiva [Immerman 1999].

Uma imediata aplicação da Complexidade Descritiva é na teoria de Banco de Dados. Um banco de dados relacional pode ser entendido como uma estrutura finita e suas linguagens de consulta como extensões da lógica de primeira ordem. Sendo assim, muitas questões que envolvem a expressividade de linguagens de consulta em banco de dados e a eficiencia de sua avaliação podem ser analisadas pelas ferramentas da Complexidade Descritiva.

Na definição das classes de complexidade computacional, os problemas de decisão desempenham um papel primordial. Apropriando-se da terminologia de banco de dados, a área de Complexidade Descritiva usualmente refere-se aos problemas de decisão como consultas booleanas (*boolean queries*, em inglês). Uma consulta booleana importante é o REACHABILITY, ou simplesmente REACH, assim definido: dado um grafo G e dois vértices **s** e **t**, existe um caminho entre **s** e **t** em G? Como a execução de um algoritmo pode ser modelada através de um grafo que representa o espaço de busca do problema, a consulta booleana REACH, e suas variantes, são consideradas representativas de várias classes de complexidade [Papadimitriou 1994].

Na relação entre complexidade descritiva e computacional, buscamos identificar padrões entre a linguagem lógica e os recursos computacionais. Alguns destes padrões já foram identificados. Por exemplo, há uma estreita relação entre o tempo necessário para computar uma consulta em um certo computador paralelo e a quantidade de (blocos de) quantificadores utilizados para descrever tal consulta [Immerman 1999]. Há também uma correspondência direta entre a quantidade de *hardware* utilizado, em termos de memória e processadores, e o número de variáveis, livres e ligadas, necessárias para a expressão de uma consulta [Immerman 1999].

Estamos aqui particularmente interessados em investigar o poder expressivo dos operadores modais na especificação de consultas, em particular, os operadores modais de necessidade e possibilidade das lógicas modais K, S4 e S5, e os operadores temporais das lógicas CTL e CTL*. Gostaríamos também de identificar quais classes de complexidade

computacional tais lógicas capturam (ou estão, conforme Definição 2.1). Mais ainda, dado que a relação de acessibilidade das lógicas modais pode ser vista como uma noção de alcançabilidade em grafos, pretendemos investigar se tais lógicas permitem expressar a consulta booleana REACH. Trataremos destas questões nas seções 3 e 4. Na seção 2, apresentaremos os principais conceitos e resultados necessários sobre Complexidade Descritiva. Na conclusão, resumiremos nossas contribuições e apontaremos os trabalhos futuros. Os lemas e teoremas que estabelecemos neste artigo são aqueles que estão sem indicação de sua referência bibliográfica.

2. Complexidade Descritiva

Nesta seção, introduziremos algumas definições e teoremas importantes da área de Complexidade Descritiva. As noções básicas de Computabilidade, Complexidade Computacional e Lógica usadas neste texto estão em [Davis et al. 1994], [Papadimitriou 1994] e [Ebbinghaus et al. 1994], respectivamente.

O vocabulário relacional σ do alfabeto $\mathcal{A}_\sigma$ das linguagens de primeira-ordem $\mathcal{L}_\sigma$ que lidaremos é um conjunto finito de símbolos relacionais e constantes. A ausência de funções não é uma restrição significativa dado que podemos tratar funções k-árias como relações k+1-árias. As σ-estruturas $\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \ldots, R_r^{\mathfrak{A}}, c_1^{\mathfrak{A}}, \ldots, c_s^{\mathfrak{A}})$ aqui consideradas serão estruturas finitas, ou seja, com domínio A finito. Chamaremos de $STRUC[\sigma]$ o conjunto das σ-estruturas finitas. Estas restrições de finitude no domínio da estrutura e no número de símbolos relacionais e constantes dá-se por estarmos tratando de problemas computacionais nos quais a entrada é uma estrutura finita.

Conforme dissemos na seção anterior, um problema de decisão será aqui denominado de consulta booleana. Uma consulta booleana é qualquer mapeamento $q : STRUC[\sigma] \rightarrow \{0, 1\}$. Tal consulta pode ser pensada como um subconjunto de $STRUC[\sigma]$, ou seja, o conjunto de σ-estruturas $\mathfrak{A}$ tal que $q(\mathfrak{A}) = 1$.

As classes de complexidade computacional são usualmente definidas como classes de linguagens decididas por máquinas de Turing. Podemos pensar que as máquinas de Turing computam consultas sobre strings binárias. Portanto, se quisermos falar de complexidade descritiva em correspondência à complexidade computacional, precisaremos codificar estruturas relacionais finitas como strings binárias. Se o domínio A da estrutura $\mathfrak{A}$ tem cardinalidade n, identificaremos os elementos do universo com os números naturais, ou seja, $A = \{0, \ldots, n\text{-}1\}$. Cada relação $R_i^{\mathfrak{A}}$ será codificada como um string binário de tamanho n^{a_i}, onde a_i é a aridade da relação $R_i^{\mathfrak{A}}$. Nesta codificação, o "1" em uma dada posição indica que a tupla correspondente, considerando uma ordenação lexicográfica das a_i-tuplas, está em $R_i^{\mathfrak{A}}$. Similarmente, cada constante $c_i^{\mathfrak{A}}$ será codificada como uma string binária de tamanho sendo igual a $\lceil \log n \rceil$. A codificação $bin_\sigma(\mathfrak{A})$ da estrutura $\mathfrak{A}$ será a concatenação das codificações de suas relações e constantes [Immerman 1999].

Definição 2.1. *($\mathcal{L}$ **está em** $\mathcal{C}$) Seja $\mathcal{C}$ uma classe de complexidade computacional e $\mathcal{L}$ uma lógica. Dizemos que $\mathcal{L}$ está em $\mathcal{C}$ se, para todo vocabulário σ e toda sentença φ em $\mathcal{L}_\sigma$, a correspondente linguagem $\{bin_\sigma(\mathfrak{A}) : \mathfrak{A} \in STRUC[\sigma]$ e $\mathfrak{A} \models \varphi\}$ está em $\mathcal{C}$.* □

Definição 2.2. *(q **está em** $\mathcal{C}$) Dizemos que uma consulta booleana q sobre o vocabulário σ*

está em $\mathcal{C}$ se ela pode ser testada com complexidade $\mathcal{C}$, ou seja, se a linguagem $\{bin_\sigma(\mathfrak{A}) : \mathfrak{A} \in STRUC[\sigma]$ e $q(\mathfrak{A}) = 1\}$ pertence à $\mathcal{C}$. □

Note que qualquer sentença de primeira-ordem $\varphi \in \mathcal{L}_\sigma$ define uma consulta booleana q_φ sobre $STRUC[\sigma]$ tal que $q_\varphi(\mathfrak{A}) = 1$ sse $\mathfrak{A} \models \varphi$. Assim, usaremos FO para representar tanto a linguagem de primeira ordem quanto as consultas booleanas definíveis em primeira ordem. Vamos agora entender o que significa uma classe de complexidade computacional $\mathcal{C}$ ser capturada por uma linguagem lógica $\mathcal{L}$.

Definição 2.3. *($\mathcal{L}$ = $\mathcal{C}$) Dizemos que $\mathcal{L}$ captura $\mathcal{C}$, notação $\mathcal{L}$=$\mathcal{C}$, se o seguinte vale:*

1. *A complexidade de $\mathcal{L}$ está em $\mathcal{C}$.*
2. *Para toda consulta booleana q na classe de complexidade $\mathcal{C}$, existe uma sentença $\varphi_q \in \mathcal{L}$ tal que $\mathfrak{A} \models \varphi_q$ sse $q(\mathfrak{A}) = 1$.* □

Para exemplificar esta definição, podemos citar o teorema de Fagin, mencionado na seção anterior, o qual prova que ∃SO=NP. Um outro resultado importante diz respeito ao conjunto FO e à classe correspondente à hierarquia de tempo logarítmico, LH.

Teorema 2.4. *[Immerman 1999]* $\mathrm{FO} = \mathrm{LH}$.

Prova. Por falta de espaço apresentaremos apenas o esboço da prova. Esta demonstração permitirá ao leitor entender melhor como se estabelece a relação entre complexidade computacional e lógica. Para provar que FO ⊆ LH, para cada fórmula $\varphi \equiv (\exists x_1)(\forall x_2)\ldots(Q_k x_k)\alpha(\overline{x})$ em $\mathcal{L}_\sigma$ que define uma consulta booleana $q_\varphi : STRUC[\sigma] \rightarrow \{0, 1\}$, devemos construir uma máquina de Turing M que execute em tempo logarítmico tal que, para toda estrutura $\mathfrak{A} \in STRUC[\sigma]$, $\mathfrak{A}$ satisfaz φ sse M aceita a codificação binária de $\mathfrak{A}$. Em símbolos: $\mathfrak{A} \models \varphi$ sse M$(bin_\sigma(\mathfrak{A}))\downarrow$. Esta máquina é construída por indução em k. Para provar que LH ⊆ FO, devemos encontrar uma consulta booleana q que seja completa para LH via reduções de primeira-ordem, mostrar que FO é fechada para reduções de primeira-ordem e finalizar expressando q em FO. □

Conforme dissemos na Introdução, estamos particularmente interessados em investigar lógicas que possam expressar a consulta booleana REACH já que REACH, e suas variantes, são problemas completos para várias classes de complexidade. Relembre que REACH verifica se, em um dado grafo direcionado G e dois vértices **s** e **t** em G, existe um caminho de **s** para **t**. Será que REACH pode ser expresso na linguagem de primeira-ordem da lógica clássica, ou seja, será que REACH ∈ FO? A resposta a esta pergunta é negativa. A consulta booleana REACH fala da existência de um caminho entre dois vértices. Para um dado número natural n, é possível escrevermos, em primeira-ordem, a noção de caminho de tamanho n. Entretanto, a noção de caminho, para qualquer n, não é expressa em lógica de primeira-ordem, pois exige a noção de fecho transitivo. A prova da não expressividade deste conceito em lógica de primeira-ordem exige técnicas da Teoria de Modelos Finitos (veja Capítulo 4 em [Libkin 2004]).

Seja TC um operador que toma o fecho reflexivo-transitivo de uma relação binária, e FO(TC) o fechamento da lógica de primeira-ordem com ocorrências arbitrárias de TC. Temos que REACH pode ser expresso em FO(TC) como segue: REACH ≡ (TC$_{xy}$(E(x,y)))(s,t), onde E é a relação binária que representa uma aresta em G [Immerman 1999].

Seja $\mathcal{L}^k$ a restrição da linguagem de primeira-ordem onde somente variáveis $x_1, \ldots, x_k$ ocorrem nas fórmulas. Se considerarmos fórmulas que exprimem a cardinalidade do domínio de uma estrutura, facilmente verificamos que a linguagem $\mathcal{L}^k$ é menos expressiva que $\mathcal{L}$. Entretanto, conforme dissemos na Introdução, há uma relação entre o número de variáveis de uma fórmula e a quantidade de memória e processadores utilizados no processamento de uma consulta expressa por esta fórmula. Sendo assim, é interessante que investiguemos qual o menor número de variáveis necessário para expressar uma consulta booleana. Denominaremos de FO^k o conjunto das consultas booleanas de primeira-ordem expressas em $\mathcal{L}^k$. Desta forma podemos dizer mais precisamente que REACH $\in$ FO^2(TC). Nas seções seguintes, investigaremos o poder das lógicas modais na expressão de consultas booleanas, em especial a consulta REACH. Adicionalmente, tentaremos definir em quais classes de complexidade computacional estas lógicas estão.

3. As lógicas K, S4 e S5

Sabemos que as lógicas modais são um instrumento poderoso para descrever propriedades de estruturas relacionais. Assumimos que o leitor tem conhecimentos básicos em lógica modal. Para maiores detalhes ver [Blackburn et al. 2001]. A seguir, apresentaremos algumas definições e teoremas importantes.

Definição 3.1. *A linguagem modal básica é definida por um conjunto Φ de letras proposicionais e um operador unário $\Diamond$ (diamond). As fórmulas bem formadas φ são dadas pela regra BNF:*

$\varphi \equiv p| \perp |\neg\varphi|\varphi \wedge \varphi|\Diamond\varphi$*, onde* $p \in \Phi$. □

O operador $\Box$ (box) é o dual do $\Diamond$ (diamond), ou seja, $\Box\varphi \equiv \neg\Diamond\neg\varphi$. Iremos definir a relação de satisfação entre uma fórmula modal e um modelo.

Definição 3.2. ***(Modelo)*** *Um modelo é uma tripla (W, R, V) tal que:*

1. *W é um conjunto não-vazio.*
2. *$R \subseteq W \times W$, é uma relação binária e representa a acessibilidade entre os elementos de W.*
3. *$V : \Phi \to 2^W$, é uma função de valoração que indica em que elementos de W uma letra proposicional assume valor verdade.* □

Definição 3.3. ***(Satisfação)*** *Seja um modelo $M = (W, R, V)$ e $w \in W$. A relação de satisfação $\models$ é dada como segue:*

1. *$M, w \models p$ sse $w \in V(p)$, onde $p \in \Phi$.*
2. *$M, w \not\models \perp$.*
3. *$M, w \models \neg\varphi$ sse $M, w \not\models \varphi$.*
4. *$M, w \models \varphi \wedge \psi$ sse $M, w \models \varphi$ e $M, w \models \psi$.*
5. *$M, w \models \Diamond\varphi$ sse existe $v \in W$ tal que $R(w, v)$ e $M, v \models \varphi$.*

Uma fórmula φ é satisfatível em um modelo M se existe $w \in M$ tal que $M, w \models \varphi$. □

Definição 3.4. ***(Satisfação global)*** *Uma fórmula φ é globalmente satisfeita em um modelo M, notação $M \models \varphi$, se para todo $w \in W$, $M, w \models \varphi$.* □

Definição 3.5. ***(Validade)*** *Uma fórmula φ é válida, notação $\models \varphi$, se para todo modelo M, $M \models \varphi$. Dizemos também que φ é válida com relação a uma classe de modelos $\mathfrak{M}$ se para todo $M \in \mathfrak{M}$, $M \models \varphi$.* □

Definição 3.6. ***(Tradução Padrão)*** *Seja x uma variável de primeira ordem. A tradução ST_x entre fórmulas modais e de primeira-ordem é definida como segue [Blackburn et al. 2001]:*

1. $ST_x(p) = P(x)$.
2. $ST_x(\bot) = x \neq x$.
3. $ST_x(\neg\varphi) = \neg ST_x(\varphi)$.
4. $ST_x(\varphi \wedge \psi) = ST_x(\varphi) \wedge ST_x(\psi)$.
5. $ST_x(\Diamond\varphi) = \exists y(Rxy \wedge ST_y(\varphi))$, *onde* y *é uma variável que ainda não foi usada na tradução.* □

Lema 3.7. *[Blackburn et al. 2001] Seja* φ *uma fórmula da linguagem modal básica. Temos que:*

1. *Para todo modelo* M *e todo* w *de* M*:* $M, w \models \varphi$ *sse* $M \models ST_x(\varphi)[w]$.
2. *Para todo modelo* M*:* $M \models \varphi$ *sse* $M \models \forall x ST_x(\varphi)$.

Teorema 3.8. *[Blackburn et al. 2001] Toda fórmula modal básica* φ *é equivalente a uma fórmula de primeira ordem com no máximo duas variáveis.*

Agora vamos apresentar algumas lógicas modais e analisar seu poder expressivo, em particular com relação ao REACH. Além disto, iremos identificar em que classe de complexidade tais lógicas estão.

Definição 3.9. ***(Lógica* K*)*** *A lógica* K *é o menor conjunto de fórmulas que contém todas as tautologias, o axioma* K $\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)$*, e é fechado para modus ponens, susbtituição uniforme e generalização.* □

Teorema 3.10. *Toda fórmula de* K *pode ser traduzida para uma fórmula em* FO^2.

Prova. Pela Definição 3.6 de tradução padrão e como a linguagem de K é a modal básica, o resultado segue direto pelo Teorema 3.8. □

Conforme dito na seção anterior, sabemos que REACH é expresso em FO(TC), mas não pode ser expresso em FO. Assim, pelo Teorema 3.10, fica claro que REACH não pode ser expresso em K. Vamos agora investigar o poder expressivo das lógicas S4 e S5.

Definição 3.11. ***(Lógica* S4*)*** *A lógica* S4 *é o menor conjunto de fórmulas que contém todas as tautologias, o axioma* K*, o axioma* T $p \rightarrow \Diamond p$*, o axioma* 4 $\Diamond\Diamond p \rightarrow \Diamond p$*, e é fechado para modus ponens, substituição uniforme e generalização.* □

Definição 3.12. ***(Lógica* S5*)*** *A lógica* S5 *é o menor conjunto de fórmulas que contém todas as tautologias, o axioma* K*, o axioma* T*, o axioma* 4*, o axioma* B $p \rightarrow \Box\Diamond p$*, e é fechado para modus ponens, substituição uniforme e generalização.* □

Os teoremas de S4 são válidos na classe dos modelos que tem como relação de acessibilidade uma relação reflexiva e transitiva imposta pelos axiomas T e 4, respectivamente. No caso de S5, os teoremas são válidos na classe de modelos reflexivos, transitivos e simétricos e a simetria é consequência do axioma B. Chamaremos as classe de modelos citadas acima de $\mathfrak{M}_{S4}$ e $\mathfrak{M}_{S5}$, respectivamente.

Utilizando a tradução padrão para verificar se S4 e S5 podem ser expressos em FO^2, observamos que não é possível preservar a validade nos modelos , ou seja, algum modelo que não está em $\mathfrak{M}_{S4}$ ou $\mathfrak{M}_{S5}$ poderia validar um teorema de S4 ou S5. No caso da lógica K, o Teorema 3.7 mostra que a validade é preservada. Dessa forma, precisamos definir uma tradução para S4 e S5 que preserve a validade na classe de modelos $\mathfrak{M}_{S4}$ e $\mathfrak{M}_{S5}$.

Definição 3.13. ***(Tradução para* S4*)*** *A tradução é igual à tradução padrão da Definição 3.6 alterando apenas o seguinte caso:*

1. $ST^4_x(\Diamond\varphi) = \exists y(Rxy \wedge ST^4_y(\varphi)) \wedge (\forall x Rxx) \wedge (\forall x \forall y \forall z(Rxy \wedge Ryz \rightarrow Rxz))$, *onde* y *é uma variável que ainda não foi usada na tradução.* □

Lema 3.14. *Seja* φ *uma fórmula de* S4. *Temos que:*

1. *Para todo modelo* M *e todo* w *em* M: $M \in \mathfrak{M}_{S4}$ *e* $M, w \models \varphi$ *sse* $M \models ST^4_x(\varphi)[w]$.
2. *Para todo modelo* M: $M \in \mathfrak{M}_{S4}$ *e* $M \models \varphi$ *sse* $M \models \forall x ST^4_x(\varphi)$.

Prova. A prova deste lema é semelhante à prova do Lema 3.7 apresentada em [Blackburn et al. 2001]. Precisamos só garantir que os modelos que satisfazem as fórmulas traduzidas sejam reflexivos e transitivos e isto é garantido adicionando-se à tradução padrão as fórmulas que impõem a reflexividade e transitividade de R. □

Teorema 3.15. *Toda fórmula de* S4 *pode ser traduzida para uma fórmula de* FO^3.

Prova. Na tradução da Definição 3.13, expressamos a propriedade de um modelo ser reflexivo e transitivo. Para expressar reflexividade basta uma variável, mas para expressar a transitividade são necessárias três variáveis. Como apenas isto diferencia esta tradução da tradução padrão da Definição 3.6 e, pelo Teorema 3.8, temos que só precisamos de três variáveis para expressar S4. □

Para S5 a situação é diferente, pois podemos assumir duas classes de modelos para S5: os modelos cuja relação de acessibilidade é de equivalência e os modelos cuja relação é universal [Chellas 1980].

Definição 3.16. ***(Tradução para* S5 *com relação de equivalência)*** *A tradução é igual à tradução padrão da Definição 3.6 alterando apenas o seguinte caso:*

1. $ST^{5.1}_x(\Diamond\varphi) = \exists y(Rxy \wedge ST^{5.1}_y(\varphi)) \wedge (\forall x Rxx) \wedge (\forall x \forall y \forall z(Rxy \wedge Ryz \rightarrow Rxz)) \wedge (\forall x \forall y(Rxy \rightarrow Ryx))$, *onde* y *é uma variável que ainda não foi usada na tradução.* □

Lema 3.17. *Seja* φ *uma fórmula de* S5. *Temos que:*

1. Para todo modelo M *e todo* w *em* M: $M \in \mathfrak{M}_{S5}$ *e* $M, w \models \varphi$ *sse* $M \models ST^{5.1}_x(\varphi)[w]$.

2. Para todo modelo M: $M \in \mathfrak{M}_{S5}$ *e* $M \models \varphi$ *sse* $M \models \forall x ST^{5.1}_x(\varphi)$.

Prova. Semelhante à prova do Lema 3.14. □

Teorema 3.18. *Toda fórmula de* S5 *pode ser traduzida para uma fórmula de* FO^3.

Prova. Semelhante à prova do Teorema 3.15. □

Definição 3.19. ***(Tradução para* S5 *com relação universal)*** *A tradução é igual à tradução padrão da Definição 3.6 alterando apenas o seguinte caso:*

1. $ST^{5.2}_x(\Diamond\varphi) = \exists x ST^{5.2}_x(\varphi)$. □

Lema 3.20. *Seja* φ *uma fórmula de* S5 *e* $\mathfrak{M}_U$ *a classe de modelos com relação de acessibilidade universal. Temos que:*

1. Para todo modelo M *e todo* w *em* M: $M \in \mathfrak{M}_U$ *e* $M, w \models \varphi$ *sse* $M \models ST^{5.2}_x(\varphi)[w]$

2. Para todo modelo M: $M \in \mathfrak{M}_U$ *e* $M \models \varphi$ *sse* $M \models \forall x ST^{5.2}_x(\varphi)$

Prova. A prova deste lema é semelhante à prova para o Lema 3.7 apresentada em [Blackburn et al. 2001]. Basta ver que, ao varrer o domínio W com o quantificador existencial, qualquer elemento que garantir a satisfação de φ vai estar relacionado com x, pois a relação é universal. Isto é garantido pela tradução da Definição 3.19. □

Teorema 3.21. *Toda fórmula de* $\mathrm{S5}$ *pode ser traduzida para uma fórmula de* FO^1.

Prova. Como a relação de acessibilidade é universal, não é necessário considerá-la explicitamente na tradução. Dessa forma, só precisamos de uma variável. □

Teorema 3.22. K, $\mathrm{S4}$ *e* $\mathrm{S5}$ *estão em* LH.

Prova. Como FO^k é menos expressivo que FO, pelos Teoremas 3.10, 3.15, 3.18 e 3.21 K, S4 e S5 podem ser expressos em FO. Logo, pelo Teorema 2.4, provamos nosso resultado. □

Como estamos interessados em expressar REACH, com os resultados acima concluímos que esta consulta também não pode ser expressa em S4 e S5. Dessa forma, precisamos investigar lógicas modais com poder expressivo maior, como é o caso de CTL e CTL*.

4. As lógicas CTL* e CTL

CTL* (*Computation Tree Logic*) é uma lógica temporal que pode ser usada para especificar propriedades de sistemas de transição de estados tais como os sistemas reativos. Tais sistemas estão continuamente interagindo com o ambiente podendo, inclusive, serem programados para não parar. CTL* permite modelar sequências de computação que serão representadas por árvores de computação obtidas a partir de uma estrutura de Kripke.

Definição 4.1. *Uma estrutura de Kripke M é um modelo conforme a Definição 3.2 onde R é uma relação de transição total, ou seja, para qualquer $s \in W$ existe $s' \in W$ tal que Rss'. Chamaremos W de conjunto de estados.* □

Construímos a árvore de computação elegendo um estado para ser o inicial, a raiz, e obtendo os ramos infinitos da árvore que representam todos os caminhos possíveis a partir da raiz. As fórmulas de CTL* são de dois tipos: fórmulas de estado, que descrevem propriedades de um estado, e fórmulas de caminho, que descrevem propriedades de um caminho (veja Definição 4.3). CTL* é o conjunto das fórmulas de estado geradas pela seguinte sintaxe:

Definição 4.2. ***(Sintaxe de*** CTL^****)*** *Seja Φ um conjunto de letras proposicionais, $\mathcal{S}$ um conjunto das fórmulas de estado e $\mathcal{P}$ um conjunto das fórmulas de caminho que satisfazem o seguinte:*

1. *Se $p \in \Phi$ então $p \in \mathcal{S}$.*
2. *Se φ e $\psi \in \mathcal{S}$ então $\neg\varphi, \varphi \wedge \psi \in \mathcal{S}$.*
3. *Se $\varphi \in \mathcal{P}$ então $E\varphi \in \mathcal{S}$.*
4. *Se $\varphi \in \mathcal{S}$ então $\varphi \in \mathcal{P}$.*
5. *Se φ e $\psi \in \mathcal{P}$ então $\neg\varphi, \varphi \wedge \psi, X\varphi, \varphi U \psi \in \mathcal{P}$.* □

Na definição da sintaxe de CTL*, temos operadores temporais (modais) e quantificadores de caminho. Os operadores temporais são: $X\varphi$, *next*, que intuitivamente

significa que φ é verdade no próximo estado, e $\varphi U \psi$, *until*, que significa que existe um estado no qual ψ é verdade e, do estado atual até este estado, φ é verdade. Agora precisamos definir a semântica de CTL* com respeito a uma estrutura de Kripke e, para isto, precisamos definir o que é um caminho nesta estrutura.

Definição 4.3. *Um caminho em uma estrutura de Kripke M é uma sequência infinita de estados $\pi = s_0 s_1 s_2 ...$ tal que, para todo $i \geq 0$, $Rs_i s_{i+1}$. Denotamos por π^i o sufixo de π começando em s_i.* □

Se $\varphi \in \mathcal{S}$ usamos a notação $M, s \models \varphi$, que significa que φ vale no estado s do modelo M, e se $\varphi \in \mathcal{P}$ usamos $M, \pi \models \varphi$, que significa que φ vale ao longo do caminho π do modelo M. A relação $\models$ é definida indutivamente a seguir.

Definição 4.4. ***(Semântica de* CTL**)** *Assuma que $\varphi_1, \varphi_2 \in \mathcal{S}$ e $\psi_1, \psi_2 \in \mathcal{P}$.*

1. $M, s \models p$	*sse*	$p \in \Phi$.
2. $M, s \models \neg\varphi_1$	*sse*	$M, s \not\models \varphi_1$.
3. $M, s \models \varphi_1 \wedge \varphi_2$	*sse*	$M, s \models \varphi_1$ *e* $M, s \models \varphi_2$.
4. $M, s \models E\psi_1$	*sse*	*existe* π *partindo de* s *tal que* $M, \pi \models \psi_1$.
5. $M, \pi \models \varphi_1$	*sse*	$\pi = s\pi'$ *e* $M, s \models \varphi_1$.
6. $M, \pi \models \neg\psi_1$	*sse*	$M, \pi \not\models \psi_1$.
7. $M, \pi \models \psi_1 \wedge \psi_2$	*sse*	$M, \pi \models \psi_1$ *e* $M, \pi \models \psi_2$.
8. $M, \pi \models X\psi_1$	*sse*	$M, \pi^1 \models \psi_1$.
9. $M, \pi \models \psi_1 U \psi_2$	*sse*	*existe* $k \geq 0$ *tal que* $M, \pi^k \models \psi_2$ *e, para todo* $0 \leq j \leq k, M, \pi^j \models \psi_1$.

□

CTL é uma sublinguagem de CTL* na qual toda ocorrência de um operador temporal é precedida por um quantificador de caminho. Alguns resultados desta linguagem serão exibidos a seguir.

Teorema 4.5. *[Immerman 1999] Toda fórmula em* CTL *pode ser traduzida para uma fórmula em* $\mathrm{FO}^2(\mathrm{TC})$.

Sabendo que REACH pode ser expresso em FO(TC), iremos avaliar se REACH pode ser expresso em CTL.

Teorema 4.6. REACH *pode ser expresso em* CTL *e* CTL*.

Prova. REACH é o problema de decidir se existe um caminho entre dois vértices específicos de um grafo direcionado. Seja M uma estrutura de Kripke modificada tal que $M = (W, R, R_s, R_t, V)$, onde W representa o conjunto de vértices do grafo, R é a relação binária que representa as arestas entre os vértices, R_s e R_t são relações unárias que dizem quem é **s** e **t** respectivamente e V é a função de valoração. R_s e R_t definem dois operadores modais 0-ários, $\Diamond_s$ e $\Diamond_t$, tais que $M, w \models \Diamond_s$ sse $w \in R_s$, ou seja, w é **s** e $M, w \models \Diamond_t$ sse $w \in R_t$, ou seja, w é **t**. REACH pode ser definido com a fórmula $EF(\Diamond_s \wedge EF\Diamond_t)$, onde $F\varphi \equiv trueU\varphi$. Pela Definição 4.4, $M \in$ REACH sse $M, w_0 \models EF(\Diamond_s \wedge EF\Diamond_t)$, onde w_0 é a raiz. Logo REACH pode ser expresso em CTL, e em CTL*, já que a linguagem de CTL também é de CTL*. □

Teorema 4.7. CTL *está em* NL *(Nondeterministic Logspace) e não está em* LH.

Prova. Como NL=FO(TC) [Immerman 1999] e FO^2(TC) é menos expressivo que FO(TC), segue direto do Teorema 4.5 que CTL está em NL. Porém, como FO é menos expressivo que FO(TC), pelo Teorema 2.4 temos que CTL não está em LH. □

5. Conclusões e Trabalhos Futuros

A área de Complexidade Descritiva visa caracterizar a complexidade de um problema por um viés independente da máquina, através da expressividade de uma linguagem capaz de definir uma classe de problemas. Neste artigo, investigamos o papel dos operadores modais das lógicas K, S4, S5, CTL e CTL* na expressão da consulta boolena REACH e na investigação de que classes de problemas estas lógicas estão. Verificamos que as lógicas modais K, S4 e S5 podem ser traduzidas para FO^3 e, portanto, estão em LH. Além disto, apesar de S4 e S5 serem semanticamente definidas com uma relação de acessibilidade transitiva e reflexiva, elas não são capazes de expressar a consulta booleana REACH, pois não conseguem expressar a noção de fecho reflexivo-transitivo. No caso das lógicas temporais CTL e CTL*, vimos que tais lógicas podem ser traduzidas para FO^2(TC) e, portanto, pertencem à NL. Além disso, podemos expressar o REACH em CTL. Como trabalhos futuros, pretendemos investigar o papel expressivo de outras lógicas modais no âmbito da Complexidade Descritiva.

Agradecemos a colaboração de Francicleber Martins Ferreira na revisão deste artigo.

Referências

Blackburn, P., de Rijke, M., and Venema, Y. (2001). *Modal Logic*. Cambridge University Press.

Chellas, B. F. (1980). *Modal logic: an introduction*. Cambridge University Press.

Davis, M. D., Sigal, R., and Weyuker, E. J. (1994). *Computability, complexity, and languages: fundamentals of theoretical computer science*. Academic Press, 2nd edition.

Ebbinghaus, H.-D., Flum, J., and Thomas, W. (1994). *Mathematical Logic*. Springer, 2nd edition.

Fagin, R. (1974). Generalized first-order spectra and polynomial-time recognizable sets. In *Complexity of computation*, volume 7, pages 43–73. AMS Bookstore.

Immerman, N. (1999). *Descriptive Complexity*. Springer.

Libkin, L. (2004). *Elements of finite model theory*. Springer.

Papadimitriou, C. H. (1994). *Computational Complexity*. Addison Wesley Logman.