1
Introdução

1.1
Motivação

1.1.1 Heurísticas. Duas conceituações de heurísticas encontradas na literatura são

> "Heuristics are criteria, methods, or principles for deciding which among several alternative courses of action promises to be the most effective in order to achieve some goal." [1]

> "A *heuristic* is a technique which seeks good (i.e. near-optimal) solutions at a reasonable computational cost without being able to guarantee either feasibility or optimality, or even in many cases to state how close to optimality a particular feasible solution is." [2]

Alguns aspectos relevantes de diferentes heurísticas são descritos a seguir ([3]):

- Heurísticas *construtivas* tentam "montar", passo-a-passo, uma resposta para uma instância de um problema, tomando uma sequência de decisões "locais", adicionando componentes individuais (nós, arestas, variáveis etc.) um de cada vez. Enquanto a construção não termina, existe apenas uma resposta "parcial". Um exemplo seria a heurística do vizinho mais próximo para o problema do Caixeiro Viajante: escolher um nó inicial e acrescentar o nó mais próximo do último nó escolhido até completar a turnê;
- Heurísticas de *transformação* iniciam com uma resposta viável completa (mas geralmente de baixa qualidade) e tentam refiná-la até obter uma resposta

satisfatória. Um exemplo seria resolver o problema do Caixeiro Viajante tomando uma turnê inicial (construída de alguma forma) e alterá-la repetidas vezes para tentar melhorar o seu custo. Heurísticas de "busca local" e "*heuristic repair*" [4] estão incluídas nesta classe;

- Heurísticas de *decomposição* resolvem um problema transformando-o em uma sequência de problemas menores, que são então resolvidos um de cada vez, em série, com os resultados de um servindo de entrada para o seguinte;
- Heurísticas de *partição* são semelhantes às de decomposição na medida em que o problema original é transformado em um conjunto de problemas menores. A diferença está no fato de que agora os problemas menores são resolvidos independentemente uns dos outros;
- Heurísticas de *restrição* transformam o problema a resolver em um problema cujo espaço de respostas é uma restrição do espaço de respostas do problema original. A idéia é considerar apenas as respostas que satisfazem alguma propriedade especial; muitas vezes, o problema restrito é mais fácil de resolver do que o original;
- Heurísticas de *relaxação* fazem o oposto das de restrição, aumentando o espaço de respostas, buscando uma resposta para o problema relaxado e transformando a resposta achada em uma resposta viável para o problema original;
- As chamadas *técnicas heurísticas modernas* [5] costumam acrescentar características probabilísticas a heurísticas de construção ou de transformação, produzindo estratégias como *simulated annealing*, busca tabu, algoritmos genéticos etc. Tais técnicas são, às vezes, chamadas de *meta-heurísticas*, já que são um pouco mais independentes do problema a resolver do que outras estratégias, e frequentemente requerem o uso de uma combinação de heurísticas mais específicas durante sua execução. Neste caso, são ainda chamadas de *meta-heurísticas híbridas*.

1.1.2 Modelos formais para heurísticas.

Embora existam resultados matemáticos que garantem a eficácia de certas heurísticas para certos tipos de problemas (por exemplo, heurísticas gulosas – um tipo de heurística construtiva – sempre resolverão de forma ótima problemas com estrutura de matróide [6]), a escolha de uma heurística para resolver um problema costuma ser guiada por diversos fatores informais, inclusive a experimentação por tentativa e erro.

Parte da comunidade científica trabalha com a *geração automática de heurísticas* ([7]), que requer um grau maior de formalização dos problemas e estratégias

envolvidos. No entanto, cada trabalho tende a abordar um tipo bem específico de problema e/ou estratégia, oferecendo pouca oportunidade para generalização.

Um modelo formal geral o bastante para problemas e heurísticas poderia auxiliar na escolha, na comparação e na combinação de estratégias para atacar problemas de qualquer tipo. Em particular, um cálculo lógico que nos permitisse raciocinar (de forma automática ou semi-automática) sobre heurísticas, suas características e seus relacionamentos seria uma ferramenta valiosa para esta área de estudo e aplicações.

Na seção 1.5, abaixo, comentamos alguns trabalhos relacionados desenvolvidos por outros pesquisadores. Antes disso, porém, devemos ressaltar que nosso objetivo de alcançar uma generalidade maior nos leva a adotar técnicas e ferramentas teóricas capazes de uma abstração maior do que as que costumam ser empregadas nesta área. É neste ponto que a *Teoria das Categorias* se mostra adequada.

1.2 Teoria das Categorias

1.2.1 Objetivos desta seção. Esta seção apresenta uma brevíssima discussão sobre alguns tópicos de Teoria das Categorias utilizados em nosso trabalho: subcategorias (co-)reflectivas, *topoi* e teoria local dos conjuntos.

Referenciamos, aqui, termos comumente usados em Teoria das Categorias, mas não fornecemos definições detalhadas. Em especial, as definições dos seguintes termos podem ser encontradas nas referências citadas mais adiante:

- categoria;
- objeto;
- morfismo;
- subcategoria;
- subcategoria reflectiva;
- subcategoria co-reflectiva;
- categoria oposta;
- funtor;
- diagrama;
- funtor adjunto; adjunção;

- funtor de inclusão;
- funtor fidedigno;
- funtor pleno;
- imersão;
- *topos*.

O leitor que desejar se aprofundar ou buscar referências abrangentes deve consultar [8] (didático, voltado para Ciência da Computação e em português); [9] (didático, mais completo, voltado para Lógica e *topoi*); [10] (detalhado, com muitos exemplos e resultados úteis); [11] (matemático, mas com alguns exemplos de Computação e Controle); [12] (mais matemático); [13] (que discute em mais detalhes a lógica interna de *topoi*).

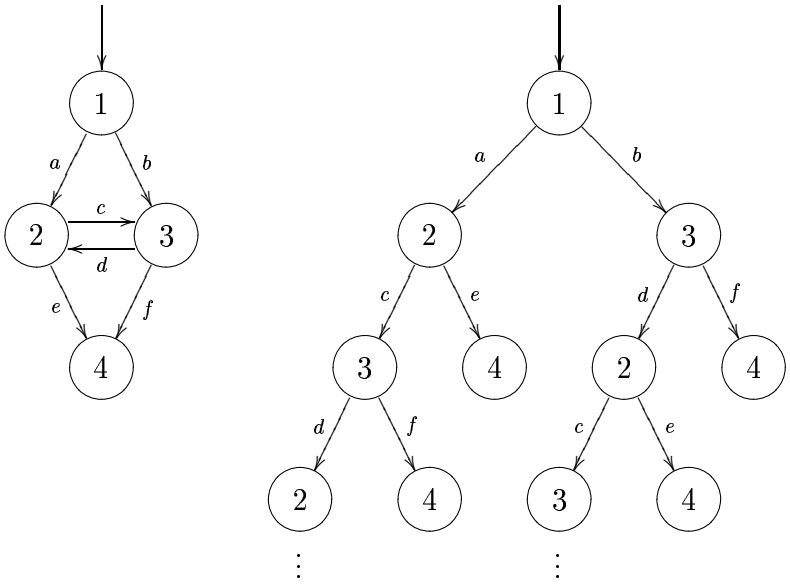
1.2.2 Exemplos de subcategorias (co-)reflectivas. Diversos exemplos de subcategorias reflectivas e subcategorias co-reflectivas ocorrem em Ciência da Computação. Frequentemente, subcategorias reflectivas correspondem a "completações", "quocientes" e certas "modificações" na estrutura dos objetos de uma categoria. Já subcategorias co-reflectivas podem corresponder a uma escolha padronizada de subobjetos interessantes dos objetos de uma categoria. Os exemplos seguintes envolvem subcategorias reflectivas e co-reflectivas.

1.2.3 Autômatos finitos determinísticos (AFDs) com estados alcançáveis e autômatos finitos determinísticos mínimos. Duas operações comuns sobre AFDs são (1) a eliminação de estados não-alcançáveis a partir do estado inicial e (2) a minimização do número de estados, gerando um AFD mínimo cujos estados são classes de equivalências do conjunto de estados do AFD original (ver [14], por exemplo).

A eliminação de estados não-alcançáveis corresponde a um funtor $E : \mathbf{AFD}_\Sigma \to \mathbf{AFD}_\Sigma^E$, onde $\mathbf{AFD}_\Sigma^E$ é a subcategoria plena de $\mathbf{AFD}_\Sigma$ cujos objetos são exatamente os AFDs que possuem apenas estados alcançáveis. Como o funtor E é adjunto à direita do funtor de inclusão, a subcategoria $\mathbf{AFD}_\Sigma^E$ é co-reflectiva em $\mathbf{AFD}_\Sigma$. Este é um caso em que uma subcategoria co-reflectiva corresponde à escolha de subobjetos interessantes (sub-AFDs com estados alcançáveis) dos objetos da categoria maior.

A minimização de um AFD que contém apenas estados alcançáveis corresponde a um funtor $M : \mathbf{AFD}^E_\Sigma \to \mathbf{AFD}^M_\Sigma$, onde $\mathbf{AFD}^M_\Sigma$ é a subcategoria plena de $\mathbf{AFD}^E_\Sigma$ cujos objetos são exatamente os AFDs mínimos. O funtor M é adjunto à esquerda do funtor de inclusão, o que mostra que a subcategoria $\mathbf{AFD}^M_\Sigma$ é reflectiva em $\mathbf{AFD}^E_\Sigma$. Este exemplo de subcategoria reflectiva corresponde à formação de quocientes.

1.2.4 Árvores de sincronização e sistemas de transições. Dado um sistema de transições rotuladas S com um estado inicial, podemos construir uma árvore T cujos nós são (cópias d)os nós de S: a raiz de T é o estado inicial de S. O segundo nível de T é formado pelos nós de S que são acessíveis a partir do estado inicial através de uma transição. Cada nó n do segundo nível tem como filhos os nós de S acessíveis a partir de n através de uma transição, e assim por diante. Caso um nó n de S seja acessível a partir do estado inicial através de vários caminhos, haverá em T várias cópias de n, uma para cada caminho. A ilustração abaixo fornece um exemplo. A árvore de sincronização à direita foi construída a partir do sistema de transições à esquerda:



Árvores de sincronização são usadas, por exemplo, como semântica de sistemas concorrentes ([15]).

Sistemas de transições rotuladas formam uma categoria **ST**. Árvores de sincronização formam uma subcategoria **Arv** de **ST**, pois uma árvore de sincronização é um tipo particular de sistema de transições. O funtor A, que leva cada sistema de transição na árvore de sincronização correspondente, é adjunto à direita do funtor

inclusão, tornando **Arv** uma subcategoria co-reflectiva de **ST**.

1.2.5 O significado de subcategorias (co-)reflectivas. Falando informalmente, o primeiro exemplo significa que, se estivermos dispostos a ignorar certas características de AFDs (quantidade de estados, existência de estados equivalentes, estados não-alcançáveis etc.), podemos muito bem trabalhar na categoria $\mathbf{AFD}_\Sigma^M$ de AFDs mínimos sem que qualquer informação importante (do nosso ponto de vista) seja perdida. De certa forma, a informação relevante contida em $\mathbf{AFD}_\Sigma$, a categoria de todos os AFDs, também está presente em $\mathbf{AFD}_\Sigma^M$. Em outras palavras, $\mathbf{AFD}_\Sigma^M$ pode ser vista como a semântica de $\mathbf{AFD}_\Sigma$.

De maneira semelhante, no segundo exemplo, árvores de sincronização podem ser vistas como dando a semântica de sistemas de transições rotuladas.

1.2.6 *Topoi*. Um *topos* é uma categoria satisfazendo certas condições que lhe conferem algumas propriedades interessantes também possuídas pela categoria **Set** de conjuntos e funções. Na verdade, um *topos* pode ser visto como um universo generalizado de Teoria dos Conjuntos. Mais precisamente, um *topos* é uma categoria **C** tal que

- **C** possui um objeto terminal 1 tal que, para todo objeto C de **C**, existe um único morfismo $C \xrightarrow{!_C} 1$. Objetos terminais são interessantes porque qualquer morfismo $1 \xrightarrow{f} C$ (quando existente) serve para escolher um "elemento" do objeto C.

  Em **Set**, qualquer conjunto unitário é um objeto terminal.

- Todo diagrama finito em **C** possui um limite. O limite de um diagrama é um objeto especial L e uma coleção de morfismos de L para os objetos do diagrama satisfazendo certas propriedades.

  Em **Set**, o produto cartesiano $A \times B$ e as projeções $\pi_A : A \times B \to A$ e $\pi_B : A \times B \to B$ de dois conjuntos compõem o limite do diagrama

  $$A \qquad B$$

  (sem nenhum morfismo entre os objetos).

Ainda em **Set**, o diagrama

$$\begin{array}{ccc} & & A \\ & & \downarrow f \\ B & \xrightarrow[g]{} & C \end{array}$$

possui como limite o conjunto

$$A \times_C B = \{(a, b) \in A \times B \mid f(a) = g(b)\}$$

acompanhado das inclusões $f' : A \times_C B \hookrightarrow A$ e $g' : A \times_C B \hookrightarrow B$. O conjunto $A \times_C B$ é chamado de *produto fibrado* (ou *pullback*) do diagrama.

Informalmente falando, o limite de um diagrama denota a "melhor" solução do sistema de restrições representado pelo diagrama.

- Para cada par de objetos A, B de $\mathbf{C}$, existe o *objeto exponencial* B^A que armazena informações sobre todos os morfismos de A para B em $\mathbf{C}$.

  Em **Set**, o exponencial B^A de dois conjuntos A e B é o conjunto de todas as funções de A para B.

- $\mathbf{C}$ possui um objeto Ω, chamado *objeto de valores-verdade*, e um morfismo $1 \xrightarrow{\top} \Omega$ que denota o elemento $\top$ (verdadeiro) de Ω. O objeto de valores-verdade mais o morfismo $\top$ compõem o *classificador de sub-objetos* do *topos*. Basicamente, o classificador funciona de tal modo que qualquer morfismo $A \xrightarrow{f} \Omega$ equivale a um sub-objeto B de A.

  Em **Set**, o classificador consiste de $\Omega = \{0, 1\}$ e $\top : \star \mapsto 1$.

1.2.7 Exemplos de *topoi*. Além de **Set**, satisfazem as condições acima diversas outras categorias de uso frequente em Teoria da Computação:

- A categoria de grafos direcionados e homomorfismos;
- A categoria de grafos direcionados reflexivos e homomorfismos;
- A categoria de árvores e homomorfismos;
- A categoria de florestas e homomorfismos;
- Toda categoria da forma $\mathbf{Set}^{\mathbf{C}}$, cujos objetos são todos os funtores de uma categoria pequena $\mathbf{C}$ para **Set** (na verdade, todos os *topoi* dos itens acima caem neste caso).

1.2.8 A lógica interna de um *topos*. As condições para que uma categoria seja um *topos* possuem consequências interessantes. Entre elas, a possibilidade de "fazer lógica" no *topos*, usando seus objetos e morfismos. Cada *topos* possui uma lógica própria, chamada sua *lógica interna*, cuja natureza depende da estrutura do *topos*. A lógica interna de **Set**, por exemplo, é a lógica clássica. As lógicas internas de outros *topoi* diferem em outros aspectos: a lógica do *topos* de grafos direcionados, por exemplo, possui 3 valores-verdade. A lógica interna de todo *topos*, no entanto, contém a lógica intuicionista.

1.2.9 Teoria Local dos Conjuntos. A lógica interna de um *topos* pode ser manipulada através de uma linguagem análoga à de Teoria dos Conjuntos. Trata-se de uma linguagem tipada, de alta ordem, cujos tipos são interpretados como objetos do *topos*, e cujos termos denotam morfismos entre objetos, elementos e sub-objetos dos objetos do *topos*. A linguagem é denominada "local" porque certas operações só são possíveis entre termos do mesmo tipo (i.e., localmente).

A todo *topos* está associada uma teoria nesta linguagem; esta teoria é a lógica interna do *topos*. Reciprocamente, toda teoria nesta linguagem define uma classe contendo os *topoi* que satisfazem as sentenças da teoria.

1.2.10 Por que categorias? Dado o exposto acima, podemos justificar o uso de Teoria das Categorias e de Teoria de *Topoi* neste trabalho observando que

- Ao definirmos uma categoria de problemas e reduções, e funtores que têm como domínio esta categoria (ou uma subcategoria desta categoria) estaremos enfatizando igualmente os objetos (problemas) e os morfismos (reduções); se, em vez de Teoria das Categorias, usássemos Teoria dos Conjuntos, por exemplo, estaríamos enfatizando os problemas em detrimento das reduções;
- Ainda usando reduções como morfismos, podemos estudar a estrutura de problemas complexos, compostos a partir de problemas mais simples. Em uma categoria, objetos compostos típicos são construídos através de morfismos relacionados a noções como produto, co-produto, equalizador, produto fibrado, etc.;
- Ao definirmos um *topos* de estratégias de construção de espaços de busca (no cap. 3), teremos à nossa disposição a lógica interna do *topos* e a linguagem de Teoria Local de Conjuntos para raciocinar sobre tais estratégias e outras entidades que habitam o mesmo universo de discurso. Na verdade, a especificação

de estratégias de busca e heurísticas se dará inteiramente na lógica do *topos* definido.

1.3 Objetivo da Tese

Este trabalho tem como principal objetivo a apresentação de um modelo matemático, baseado em teoria de *topoi*, para heurísticas.

Conjecturamos que o modelo apresentado é abrangente o bastante para incluir todas as conceituações de heurísticas baseadas em espaços de busca e estratégias sequenciais de busca.

Apresentamos, também, vários exemplos para dar apoio à tese de que nosso modelo é, de fato, adequado.

1.4 Estrutura da Tese

Cap. 2 – Teoria dos Problemas – Visão Categórica. Nosso trabalho se inicia com a definição de uma categoria de problemas e reduções, formalizando estas duas noções. Em particular, torna-se necessário escolher uma definição de redução entre problemas. Diferentes alternativas são estudadas, em especial reduções unárias e reduções binárias. A decisão final é justificada por um teorema que determina que a categoria dos problemas com reduções unárias é uma subcategoria reflectiva da categoria dos problemas com reduções binárias.

Cap. 3 – Construção de Espaços de Busca. Existem estratégias de resolução de problemas que não incluem a definição explícita de um espaço de busca. No entanto, partimos da premissa de que mesmo essas estratégias podem ser modeladas adequadamente através da definição de *florestas de respostas*. Isto nos permite modelar de maneira uniforme as mais variadas estratégias de resolução de problemas.

Cap. 4 – Estrutura Lógica do *Topos*. A definição categórica de heurísticas se dá no contexto de um *topos*. Todo *topos* é um universo de discurso matemático, e a todo *topos* está associada uma lógica interna e uma linguagem de Teoria Local dos Conjuntos para raciocinar sobre as construções deste universo.

Dito de outra maneira, nosso *topos* serve como um modelo matemático de estratégias e heurísticas em geral, um modelo estruturado, que já vem equipado com uma lógica e uma linguagem próprias. Esta situação apresenta grande vantagem sobre a alternativa de definir uma lógica e um modelo separadamente e demonstrar a adequação daquela em relação a este, pois toda a estrutura do modelo precisaria ser dada pela lógica.

A semântica desta lógica está intimamente relacionada com a estrutura do *topos* que lhe deu origem: a quantidade de elementos do classificador de subobjetos, a existência de elementos globais de certos objetos, de um objeto de números naturais, etc.

Cap. 5 – Estratégias de Busca em Teoria Local dos Conjuntos. Uma vez estabelecida a lógica, passamos a definir termos e predicados úteis para a especificação de heurísticas no nosso modelo. Como o capítulo 3 abordou somente a construção de espaços de busca (florestas de respostas), torna-se necessário definir estratégias de busca para percorrer esses espaços. Esta definição é feita na própria linguagem interna do *topos* adotado. Assim, uma heurística é definida como uma estratégia de construção de espaços de busca, acoplada com uma estratégia de busca nestes espaços.

Cap. 6 – Exemplos e Aplicações. O modelo desenvolvido nos capítulos anteriores é usado para especificar algumas heurísticas diferentes, inclusive metaheurísticas de caráter estocástico. Além disso, um conhecido resultado sobre a otimalidade de heurísticas gulosas na resolução de problemas com estrutura de matróide é examinado à luz do nosso modelo.

Apêndices. O apêndice A traz provas detalhadas dos teoremas e proposições enunciados no texto; o apêndice B apresenta uma lista com o nome e uma breve descrição de cada categoria referenciada neste trabalho.

1.5 Trabalhos Relacionados

Modelos, classificações e taxonomias de heurísticas têm aparecido na literatura desde os anos 70:

- Um artigo interessante e rigoroso sobre representações de problemas, formulação de espaços de estados e relações entre problemas é [16], que, no entanto, não aborda estratégias de busca. É curioso notar que algumas idéias de Teoria

das Categorias são mencionadas no texto, mas o autor, apesar de considerá-las promissoras, não as explora mais a fundo. Outra possibilidade atraente discutida em [16] é a automação parcial das transformações de problemas.

- Outro exemplo de uma abordagem formal bem-fundamentada é o trabalho apresentado em [17] sobre um modelo de algoritmos de busca *branch-and-bound* generalizada aplicados a problemas de otimização. Trabalhos posteriores [18, 19] dos mesmos autores tentam definir um grande modelo unificador de busca heurística, *branch-and-bound* e programação dinâmica.
- O artigo [20] apresenta um modelo unificado para Busca Tabu, *Simulated Annealing* e Algoritmos Genéticos, mas a exposição, ainda que instrutiva, é bem informal e imprecisa.
- Taxonomias como [21] e classificações como [3] tendem a ser ainda mais imprecisas ou orientadas a aplicações, não podendo ser realmente consideradas modelos formais.
- Mais recentemente, a comunidade de heurísticas tem se interessado por modelos formais de estratégias de busca com o objetivo de facilitar a definição e a instanciação de *frameworks*, bem como o desenvolvimento e a implementação de algoritmos orientados a objetos para a resolução de problemas de otimização, como exposto em [22, 23].

  Os modelos formais definidos nesses trabalhos costumam recair em uma das seguintes categorias:

  - Bibliotecas de classes e componentes em linguagens OO;
  - *Frameworks* em linguagens OO;
  - Linguagens de modelagem (*modeling languages*), possivelmente acopladas a resolvedores automáticos ou semi-automáticos de problemas de otimização ou de problemas de satisfação de restrições.

  Uma vez que o modelo proposto neste projeto esteja consolidado, possíveis aplicações relacionadas às abordagens dos trabalhos acima são a definição de *frameworks* para heurísticas e meta-heurísticas e/ou a geração de código em uma linguagem de modelagem (ver Seção 1.3).
- Aplicações significativas de Teoria das Categorias ao estudo de problemas, reduções, algoritmos aproximativos e heurísticas de que temos conhecimento são [24] e [25]. Nestes trabalhos, são definidas uma categoria de problemas de otimização e uma categoria de problemas de otimização solúveis em tempo polinomial; esta última é vista como uma aproximação da primeira (em um sentido preciso, definido em Teoria Categórica das Formas) com o objetivo de estudar uma hierarquia de algoritmos aproximativos.

Até onde pudemos constatar, a presente tese descreve uma aplicação completamente original de Teoria das Categorias e Teoria dos *Topoi* à construção de um modelo formal para problemas, reduções, espaços de busca e heurísticas sequenciais.