Uma Modelagem em Teoria das Categorias de Interações de Caminhos para Path Planning Multi-Agente

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

Abstract. *Using basic mathematic structures like Graphs and Partially Ordered Sets we can formalize uni and multi-agent Path Planning systems by the Category Theory, creating consistent and highly generic results, revealing the most diverse possibilities of use. The focus of this work is practical, not theoretical: we are interested in the ideas that the Theory tools bring us, and how we can combine each other generating results in the context of Multi-Agent Path Planning. Mathematical proofs and mathematical implementation details are omitted, but the reader is pointed for reference material.*

Resumo. *Utilizando estruturas matemáticas básicas como Grafos e Conjuntos Parcialmente Ordenados podemos formalizar sistemas de Path Planning uni e multi-agente pela Teoria das Categorias, gerando resultados consistentes e altamente genéricos, revelando as mais diversas possibildades de utilização. O enfoque desse trabalho é prático, e não teórico: nos interessamos pelas ideias que as ferramentas da Teoria nos proporcionam, e como podemos combinar umas com as outras na geração de resultados no contexto de Path Planning Multi-Agente. Provas e detalhes de implementação matemáticos são omitidos, mas o leitor é apontado para material de referência.*

1. Introdução

Nesse paper, apresentarei e descreverei uma proposta de modelagem formal baseada na Teoria das Categorias para tratar problemas de Path-Planning Multi-Agente, assim como ideias que podem gerar futuros trabalhos.

O Path Planning é uma classe de problemas comum na Computação. Path Planning define-se, como o nome diz, no planejamento de um caminho, dado um certo ambiente, 2D ou 3D, com uma configuração de obstáculos, ou num caso mais avançado, tipos de ”terreno”. O interesse aqui é em ter um algoritmo que resolve automaticamente o problema de mover um (single-agent) ou mais agentes (multi-agent) de uma coleção de pontos no mapa a outra coleção, tendo em vista as limitações e vantagens do ambiente. As aplicações dos estudos na área são diversas ([Kuffner 2000]), como: robótica, montagem de componentes eletrônicos ou mecânicos, prototipagem virtual de modelos, design de drogas (farmacêutica), manufaturação, e até animação computacional.

Para o contexto desse paper, trabalharemos com o caso de já termos as possíveis configuração de caminhos que nossos agentes podem percorrer, o que será a entrada do nosso sistema. A partir daí faremos análises de diferentes formas como esses caminhos

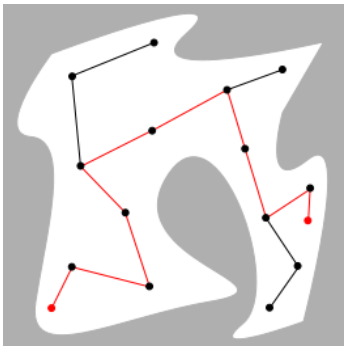

Figura 1. Exemplo de caminho solução da aplicação de um algoritmo de Path Planning.

podem interagir entre si, além de modificações em suas estruturas, gerando ou filtrando detalhes do movimento. Um rápido exemplo de aplicação poderia ser a coordenação de uma equipe de resgate de robôs, que em certo momento devem executar ações em conjunto, ou devem prevenir de estarem próximos uns dos outros.

Utilizaremos a Teoria das Categorias para modelar matematicamente nossas ideias. Ela foi introduzida pela primeira vez por S. Eilenberg e S. Mac Lane em 1945 em [Eilenberg 1945]. Ela foi primeiramente criada pra transformar complicados problemas de Topologia em problemas mais simples de Álgebra (Divisão e Conquista?). Para isso, os autores formalizaram algo que podemos entender como o estudo da essência subjacente às estruturas e suas relações. Assim, puderam relacionar dois campos aparentemente distintos da Matemática em um único estudo. Uma pesquisa rápida por material acadêmico relacionado ao uso da Teoria das Categorias hoje em dia revela estudos em campos do conhecimento altamente interdisciplinares, como: modelagem de Redes Neurais Artificais [Healy 2000], Sistemas Evolutivos com Memória [VANBREMEERSCH 1999], Sistemas de Vida [Kainen 2005].

Nesse trabalho, apresentarei as ideias supondo que o leitor tenha um certo conhecimento da Teoria das Categorias, ou seja, que a tenha estudado de maneira que entenda suas construções e operações básicas. As aplicações do diferentes campos da Teoria foram divididas em tópicos, então o leitor que julgue não ter conhecimento suficiente pode tentar entender e estudar cada parte em separado. Para uma introdução às Categorias, recomendo a leitura de [Goguen 1991], onde o autor explicita a importância e as aplicabilidades no contexto da Computação. Para um estudo aprofundado, com um material didático para o aprendizado ou para referências, recomenda-se os livros [Menezes 2006] e [Longo 1991], ambos disponíveis gratuitamente na Internet.

Serão omitidas as provas de **Gr** e **Poset**, tanto de suas naturezas como categorias quanto das operações definidas sobre elas. Tentarei me ater mais à prática e menos à teoria, de forma que enxerguemos mais as ideias das aplicações e menos a forma como elas são implementadas. Para referência das provas e detalhes extras, sugiro recorrer a livros-texto como os supracitados.

2. Fundamentos: modelando um caminho como Categoria

Definiremos um *caminho* como um grafo direcionado, onde cada nodo tem como chave uma n-upla que representa a posição de um ponto em um espaço n-dimensional, e as arestas são relações entre os pontos.

Informalmente, o que temos é uma série de pontos no espaço ligados entre si de forma que crie a noção de sequencialidade no movimento por este caminho. Esta é a forma como algoritmos de Path Planning normalmente trabalham sua solução. Vale notar que estaremos, então, levando em conta uma discretização de um possível domínio contínuo, como é no caso de simulações reais. Eventualmente, cada ponto representará um estado, uma posição em um tempo t, do agente que deseja se mover no ambiente. Se for de nosso interesse, poderemos mais tarde adicionar ao grafo endoarcos representam um agente que não se move em um tempo t (como se ele executasse uma operação NOP de Assembler).

Pode ser do interesse haver uma malha onde se tenham registrados mais ou menos pontos por unidade de medida, o que influencia na velocidade de iteração, ao longo do caminho, do controle do movimento do agente. Essa possibilidade não nos interessa, desde que a resolução da malha não tenda ao infinito, ou seja, que o caminho se torne uma configuração em um espaço contínuo.

Dependendo da aplicação desejada, poderemos estar implementando caminhos de diferentes maneiras. Descreverei aqui as implementações para, posteriormente, apenas me referenciar a elas.

2.1. Caminho como subcategoria de Gr

A categoria **Gr** dos grafos é tal que possui todos os grafos como objetos e todos os homomorfismos de grafos como setas. A composição de homomorfismos de grafos é definida a partir da composição das funções componentes [Menezes 2006].
Pode tanto nos interessar ter um grafo para cada caminho possível de um agente como podemos querer grafos que juntem vários caminhos, inclusive de diferentes agentes. Inicialmente teremos cada caminho como um grafo. Logo mostrarei como concatená-los.

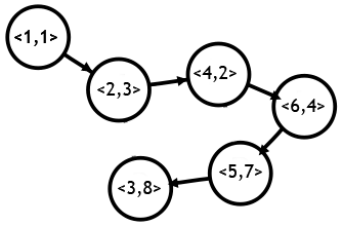

Figura 2. Caminho como subcategoria de Gr

2.2. Caminho como subcategoria de Poset

A categoria **Poset** é tal que possui todos os conjuntos parcialmente ordenados como objetos, todas as funções monotônicas (que preservam a ordem) como morfismos, a composição de funções como a operação de composição categorial e a identidade de cada objeto dada pela correspondente função identidade [Menezes 2006].
Esta é uma forma um pouco diferente de enxergar caminhos. Seja $Po =< A, \leq >$ um conjunto parcialmente ordenado e $G =< V, T, delta0, delta1 >$ um caminho que será "transformado". Mapearemos um conjunto livremente gerado das chaves dos nodos do conjunto V para o conjunto A, e a forma como os pontos se relacionam, ou seja, a relação de ordem parcial $\leq$ definida para o conjunto A deverá representar os arcos T com suas fontes delta0 e destinos delta1.

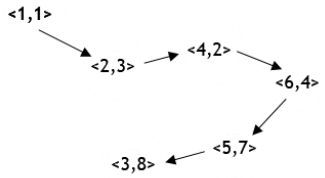

Figura 3. Caminho como subcategoria de Poset. É essencialmente a mesma coisa que em Gr.

3. Coproduto como União Disjunta de caminhos

Operando um coproduto tanto no caso de termos nossos caminhos em Gr quanto em Poset teremos uma mesma interpretação de Combinação de Sistemas: a inclusão de caminhos dentro de um mesmo objeto. Esta é uma operação essencial para o caso de querermos futuramente realizar operações sobre mais de um caminho ao mesmo tempo.

4. Produto Fibrado como sincronização de agentes

O produto fibrado é uma operação altamente genérica e bastante poderosa que, como uma combinadora de sistemas, pode ser vista como a sincronização de estados de um certo caminho. Essa não é uma construção trivial para o caso de path plannings, já que raramente se deseja que mais de um agente tente ficar numa mesma posição ao mesmo tempo, mas podemos pensar no caso de robôs que atuem juntos em uma determinada tarefa que envolva um mesmo objeto do ambiente.
Mas há ainda uma aplicação bastante interessante: facilmente podemos elaborar um algoritmo simples que detecte e nos identifique estados de caminhos diferentes estão próximos demais. Será o caso em que sincronizaremos caminhos por onde haja colisão dos agentes. Este caso será melhor abordado logo a seguir. Por enquanto, basta entendermos que tal sincronização "colará"os caminhos por onde eles se choquem, facilitando a posterior identificação.

5. Produto e Coproduto Binários como detectores de colisões

Este é o caso em que será de nosso interesse enxergar um caminho como um conjunto parcialmente ordenado. O resultado de um produto binário entre dois elementos em

um conjunto p.o. equivale a encontrarmos o Maior Limitante Inferior, assim como seu conceito dual, o coproduto, equivale a encontrar o Menor Limitante Superior.
Aqui temos um detalhe de implementação que vale ressaltar. Dependendo do algoritmo que nos gerar os caminhos, ele pode, e provavelmente não irá, agrupar pontos euclidianamente próximos como um mesmo ponto. Pode ser necessário aplicarmos produtos fibrados que identifiquem pontos com uma distância menor que um certo valor, e então agrupe os caminhos e "cole-os"de maneira que os pontos se tornem um mesmo ponto.
Essa é uma aplicação bastante interessante, a qual podemos dar várias interpretações. Talvez a mais interessante seja de, a partir de um momento t da execução do movimento pelo mapa de nossos agentes, podemos descobrir se há em um futuro próximo ou distante um momento em que eles se direcionem a uma mesma posição no espaço, o que pode representar uma possível colisão.
Resumindo, o que esta técnica nos permite é analisar interseções nos caminhos entre os agentes, o que pode representar um ponto de partida ou de chegada em comum, possíveis colisões, áreas do ambiente de atuação conjunta, etc. Vale lembrar que isso vale para quantos agentes for de nosso interesse, dependendo de como os caminhos tiverem sido agrupados.

6. Soma Amalgamada e Gramática de Grafos

Gramáticas de Grafos é uma das construções mais poderosas e úteis decorrentes do estudo da Teoria das Categorias. Elas são definidas a partir da aplicação da soma amalgamada de uma regra pré-definida com uma seta que instancia a regra no grafo onde ela operará. Tal construção funciona exatamente como Gramáticas em Linguagens Formais, e as aplicações são confluentes.
Os tipos de regras que podemos criar para modificar caminhos são das mais diversas, e só a descrição no parágrafo anterior já é suficiente para podermos implementar e aplicar a ideia. Basicamente, o que criaremos são regras que geram padrões que podem ser aplicados em situações específicas.

6.1. Alterações de densidade do caminho

A maneira mais simples de alterarmos um caminho seria com a simplificação ou a complexificação em relação à densidade de nodos. Aqui, nós teremos uma regra que apaga ou adiciona um estado no caminho, de preferência sem alterar sua forma geral. Isto pode servir, dependendo da implementação, para modificar a velocidade do agente em um certo trecho do caminho, caso os nodos representem estados num tempo pré-determinado.
No caso de inserirmos um nodo, temos em mente a ideia de interpolação, já que adicionaremos um nodo que certamente não modificará a maneira como o caminho se porta. Aumentar a densidade de um caminho pode representar um aumento de cautela de um robô que se aventura num ambiente hostil, no caso, por exemplo, de ele realizar uma leitura dos sensores a cada novo estado da iteração do movimento.

6.2. Desvio de colisão

Um jeito bem interessante de utilizar padrões de movimento é para criar um tipo de movimento para os agentes que precisam desviar de um estado que os farão colidir com outro agente. Aqui, nós implementamos regras que equivalem a uma convenção para o tratamento destes casos, como numa estrada brasileira, onde a convenção é a de ultrapassar sempre à esquerda
Alguns algoritmos, que fazem o que se chama de *multiple-query planning*, tratam dinamicamente modificações no ambiente, isto é, tratam colisões com obstáculos mudam de posição ao longo do tempo. Utilizando uma regra de gramática de grafos, temos solução simples e eficiente que fará não ser necessária a intervenção de um algoritmo de multiple-query, para casos simples de colisão, ou mesmo evitem a reaplicação do algoritmo inteiro, caso ele não seja do tipo *single-query*.

6.3. Tipos de movimento para terrenos especiais

Pode ser interessante fazer um robô se mexer de um jeito especial, algo que Path Planners não levam em consideração. Por exemplo, um robô que precisa se movimentar em uma área lamacenta não pode ficar muito tempo parado em uma mesma região, caso contrário afundará; neste caso, nós poderíamos fazê-lo tomar posições ligeiramente afastadas de seu caminho original, de forma que pise em locais diferentes do terreno e distribua sua pressão total sobre o solo. Poderíamos aplicar padrões de movimento também no caso do movimento automatizado de um personagem em jogo, por exemplo, de uma dançarina, que realiza rodopios enquanto se move de um ponto a outro de um palco.(figura)

7. Conclusão

O objetivo da Teoria das Categorias é o de estudar e generalizar os conceitos comuns a todos os tipos de sistemas especificáveis formalmente. Isso nos dá ferramentas para fazermos literalmente o que quisermos com nossos sistemas, inclusive relacionar e estudar conjuntamente sistemas que a princípio em nada se assemelhavam. As operações definidas pela Teoria são bem definidas e têm se mostrado de importante valia computacional, por ter mecanismos de fácil implementação e notável performance geral.
Temos ainda uma vantagem a longo prazo no fato de modelarmos matematicamente nossas aplicações e soluções, pois, a partir de uma formalização, temos muito mais controle na análise do sistema como um todo, além de tornar mais fácil modificações futuras, pois todos conceitos estão bem discretizados e descritos, e as relações entre eles bem definidas e por isso mais expressivas.
Essas foram algumas reflexões que podemos fazer sobre o trabalho aqui exposto. Os argumentos são auto-contidos, pois a partir do momento que nos utilizamos das ferramentas da formalização das Categorias, e percebemos como seus conceitos são minimalísticos e poderosos e nos fazem enxergar mais facilmente as possibilidades que emergem, concluímos então que o próprio trabalho de aplicá-las em um contexto específico nos trará os mesmos benefícios no futuro, quando revisarmos e reutilizarmos conceitos desenvolvidos para a fundamentação de outros ainda mais complexos e poderosos.
Todas ideias aqui expostas e comentadas, no jargão da Engenharia de Software, seguem

o princípio Aberto/Fechado ([Meyer 1988]): são fechadas em si mesmas por serem consistentes e bem fundamentadas, então não necessitam ser modificadas em sua essência, mas elas são abertas pois são altamente genéricas e têm alta capacidade para extensão e desenvolvimento mais profundo.

8. Referências

Referências

Eilenberg, S. & Mac Lane, S. (1945). General theory of natural equivalences.

Goguen, J. A. (1991). A categorical manifesto.

Healy, M. (2000). Category theory applied to neural modeling and graphical representations.

Kainen, P. C. (2005). Category theory and living systems.

Kuffner, J.J. & Jr. LaValle, S. (2000). Rrt-connect: An efficient approach to single-query path planning.

Longo, A. A. . G. (1991). Categories, types and structures.

Menezes, Paulo Blauth & Haeusler, E. H. (2006). *Teoria das Categorias para Ciência da Computação*. Editora Sagra Luzzatto, 2ª edition.

Meyer, B. (1988). Object-oriented software construction.

VANBREMEERSCH, A. E. . J.-P. (1999). Memory evolutive systems.