CONTRIBUIÇÕES DAS RP-NETS AO SOAR

Ricardo Capitanio Martins da Silva

martins@dca.fee.unicamp.br

Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação
IA718 Introdução à Ciência Cognitiva

RESUMO

Este artigo faz uma breve apresentação das RP-Nets e a arquitetura cognitiva SOAR. È feito um paralelo entre as duas teorias e ao final é mostrado como as RP-Nets podem contribuir removendo limitação do SOAR.

PALAVRAS-CHAVE: SOAR, RP-Nets, arquiteturas cognitivas, cognição, simulação, recursos.

1 INTRODUÇÃO

A RP-NET (Rede de Processamento de Recursos) [2] é um modelo matématico-formal desenvolvido no Departamento de Computação e automação da Faculdade de Engenharia Elétrica e de Computação para simular sistemas dinâmicos a eventos discretos. Esse modelo é o amadurecimento de outras formulações anteriores. Inicialmente surgiu como "Rede de Objetos", na primeira tentativa de formalização por Gudwin em [9], evoluindo para "Redes de Agentes" [2] [8], passando a Redes Semiônicas [5][7] e por fim RP-Net [2].

SOAR é uma teoria computacional de cognição humana na forma de uma arquitetura cognitiva geral. É uma tentativa de unificar uma série de fenomenos cognitivos com um conjunto único de mecanismos para o tratamento uma série de problemas metodológicos e teóricos comuns as teorias c cognitivas. SOAR representa um dos maiores trabalhos de Allen Newell, um dos fundadores da ciencia cognitiva moderna e um dos pioneiros no desenvolvimento dessas classes de teoria cognitiva.

Nesse trabalho será mostrado uma primeira especulação de como as RP-Nets podem remover o conhecido gargalo do SOAR que é o disparo de uma única regra por ciclo de decisão. Primeiramente será apresentado as características do SOAR, após os conceitos básicos ligados as RP-Nets e em seguida a comparação entre as duas teorias e como as RP-Nets podem contribuir.

2 SOAR

SOAR (State, Operator and Result) é uma arquitetura cognitiva em desenvolvimento pela Universidade de Michigan [1]. Essa arquitetura possui um sistema de símbolos físicos que realiza busca em espaços de problemas através de uma estrutura de controle automática de dois níveis.
No SOAR, o conhecimento acerca de um determinado problema é descrito através de metas, estados e operadores a a escolha de aplicação de um operador em um determinado estado se dá pelo princípio da racionalidade. O funcionamento da arquitetura independe do domínio, bastando ao criador do modelo decompor o problema de maneira correta (metas, estados, operadores). Os estados contém toda a informação acerca da situação correte, incluindo a percepção e descrição de metas correntes e espaços de problemas. Os operadores utilizam o conteúdo do conhecimento para determinar sua relevancia diante do estado e meta correntes e realizam os passos no espaço do problema. Utilizando estados e operadores é possível modelar cada comportamento – seja de ação ou pensamento sobre a ação – como uma função de qual conhecimento se precisa produzir.

2.1 ESTRUTURA DE MEMÓRIA

Para servir de arquitetura cognitiva SOAR diferencia a memória em dois tipos: uma contendo conhecimento geral e outra para aplicação específica de conhecimento. Conhecimento que existe independentemente da situação corrente é armazenado na arquitetura em uma memória de logo prazo (LTM – Long Term Memory). SOAR diferencia três tipos de LTMs: procedural, semântica e episódica. Conhecimento procedural armazena como e quando fazer as coisas (como dirigir um carro, como resolver um problema de álgebra, como ler uma receita e fazer um bolo). Conhecimento semântico consiste em fatos sobre o mundo: coisas que acreditamos ser verdade de maneira

geral – coisas que você “sabe”, como moto tem duas rodas, em um jogo de futebol há 11 jogadores de cada lado. Conhecimento episódico se trata das coisas que você “lembra” – situações específicas vivenciadas: como a vez em que se caiu da motocicleta e machucou o joelho. LTM não está disponível diretamente mas pode ser utilizada para realizar buscas para se encontrar o que é relevente para a situação corrente.

O conhecimento relevante a situação corrente fica armazenado na memória de trabalho (WM – working memory). É um conhecimento específico a determinanda situação e contem o que é verdadeiro naquele dado tempo. WM pode também conter conhecimentos/memórias gerais que serão úteis durante a tomada de decisão acerca da situação atual. No SOAR, WM é representada como um conjunto de funcionalidades e valores que juntos formam o estado atual (e seus substados), que podem incluir representação da meta atual, espaço do problema e operadores.

Assim, podemos dizer que a LTM contem o que pode ser

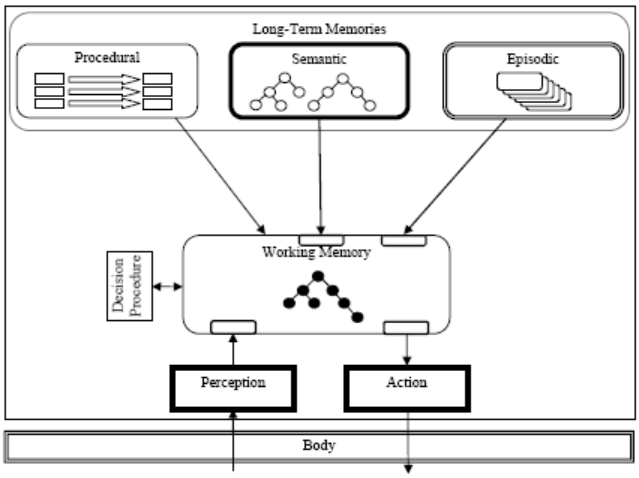

Figura 1. Estrutura de Memória no SOAR. Adaptado de [1].

relevante para várias situações e que podém ser explicitamente recuperado enquanto que a WM contem o que é relevante para o modelo de uma situação particular corrente. Um ponto chave é que o conhecimento da WM pode ser utilizado para recuperar conhecimentos na LTM. O conhecimento é armazenado em regras se-então escritas em uma linguagem English-like. Essas reguras representam associações entre o conjunto de condições expresso em termos de características e valores especificados pela parte “se” e um conjunto de ações pela parte “then”. Primeiramente a porção “se” de cada regra testa se uma percepção ou elemento do estado (e da sua substrutura de características e valores, que podem incluir metas e estado do problema) ou operador. Caso haja o casamento entre a parte “if” da regra e a WM, é dito que a regra foi “ativada”. Isso causa que a parte “então” seja disparada através do envio de uma mensagem ao sistema motor, ou sugerindo alterações nas metas, estado e operador, uma vez que as metas mapeiam mudanças nesses objetos.

Podem existir dependencias entra as reguras. Entratanto SOAR não reconhece a existência delas: simplesmente realiza o casamento das partes “if” e executa as partes “então”. Pois todos os elementos das regras são expressos em termos de percepções, ações, estados e opereadores eles são processados pela arquitetura de modo geral totalmente independente do domínio do problema.

Além disso é importante notar que para as tarefas em que o modelo tem conhecimento procedural suficiente, não é necessário requerer o acesso aos conhecimentos semantico e episodico. Esses conhecimentos são importantes para o comportamento não-especialista, quando o conhecimento procedural é imcompleto.

2.2 CICLO DE DECISÃO

O ciclo de decisão é um componente de processamento da arquitetura que gera o comportamento fora do conteudo residente na LTM e WM. O propósito do ciclo de decisão é selecionar o próximo operador a ser aplicado.

Na arquitetura SOAR o mecanismo do ciclo de decisão é dividido em cinco fases: input, elaboração, decisão, aplicação, e output. Na fase de input elementos da WM são criados para refletiro mudanças na percepção (veja fig 1). Durante a fase de elaboração, os elementos da WM são deparados com as partes “if” das regras na LTM. Na elaboração o conteúdo. Todas as regras casadas na memória procedural, disparadas em paralelo, resultam em alterações nas características e valores do estado, e, em sugestões ou preferencias para a seleção do operador corrente. Como resultado das mudnças na WM, mais regras podem ser disparadas. A fase de elaboração continua com regras disparando paralelamente até que nenhuma regra seja disparada. As alterações na WM durante a elaboração são, como o nome sugere apenas elaborações. São apenas conclusões ou sugestões e não ações no mundo ou alterações das estruturas internas que devem persistir além da situação corrente. As alterações nos estados são realizados pelos operadores.

Após todo o conhecimento estiver disponível (constatado pela ausência de regras disparando na fase de elaboração), então as preferencias adicionadas a WM são avaliadas independente do dominio. O vocabulário inclui preferencias simbólicas como um operador é melhor que o outro, mas também inclui preferencias numerocas como um mvalor esperado para um operador é X. Após a seleção o operador deve ser aplicado para produzir o estado de transição que completa um movimento no espaço do problema. A aplicação do operador ocorre durenta a fase de aplicação. No SOAR um único operador pode ser selecionado para um dadao estado em um certo tempo. Multiplas ações podem ser empacotadas juntas em um único operador para que elas seja executadas de forma unitária. A execução do operador pode ser acompanhada por regras disparando em paralelo (e um sistema motor paralelo). A restrição real é que o

conjunto de atividades paralelas devem ser representadas como um único operador que pode ser selecionadno e aplicado como unitário. Assim, SOAR suporta uma forma limitada de paralelismo, em que as atividads em paralelo são preparadas a priori em um operador, e não suporta um paralelismo arbitrário.

2.3 APRENDIZADO

Desde os primórdios do desenvolvimento do SOAR apenas um método de aprendizado existiu chamado chunking. Recentemente outros métodos de aprendizado foram adicionados: aprendizado por reforço, semântico e episódico.

2.3.1 Chunking

Quando um impasse aparece isso significa que o sistema não tem regras disponíveis na LTM que levam a um movimento único no espaço do problema. Assim, um impasse é a maneira da arquiteirua sinalizar a falta de conhecimento. Essa falta de conhecimento é uma oportunidade de aprendizado.

Em resposta ao impasse um subestado para busca de conhecimento é criado pela arquitertura. O conteúdo do domínio é expandido nesse susbestado envolvendo operadores de recall e avaliação, o que prove o contexto que pode trazer conhecimento relevante da LTM. Como o resultado do processamento do substado, esse novo conhecimento deve levar a escolha correta do operador e resolver assim o impasse. Assim, é tida a oportunidade de aprender uma nova regra que esse conhecimento trouxe a tona sob as condições que levaram ao impasse. Na verdade, a arquitetura automaticamente gera essa nova regra sempre que um impasse é gerado. Para diferenciar regras que s”ao aprendidads das originalmente criadas pelo modelo, é dado um nome especial: chunk, e o processo de aprendizado de chunking. Para criar um chunk a arquiteetura leva em consideração caad parte do estado orignal que existia no ambiente pré-impasse., determinando o que foi “usado” pela análise de todas as chamadas a LTM (ativação de regras, chamadas as memórias episódica e semântica) as quais foram o caminha para se gerar o resultado. Assim, chunking é essencialmente um mecanismo de aprendizado compositional, e dedutivo.

2.3.2 Aprendizado por reforço

Uma fonte de conhecimento vindo do meio externo é o feedback do ambiente ou o que chamamos recompensa. A recompensa pode vir do “corpo” no qual o modelo está corporificado, ou pode ser gerado internamento quando uma meta é alcançada. A recompensa pode ser positiva ou negativ a, como prazer ou dor como recompensa de uma tarefa executada com sucesso ou punição por uma falaha. Recompensa pode ter impactor imediato no comportamento por informar ao modelo que ele deve evitar a situação atual (em caso de recompensa negativa) ou tentar mantê-lo (em caso de recompensa positiva). Além do impacto instantâneo recompensa pode ser utilizada para impactar em comportamento de longo prazo, através de aprendizado.

Baseado em experiência, o aprendizado por reforço ajudata as predições das futuras recompensas, as quais s”ao utilizadas para selecionar ações que maximizam as recompensas futuras. SOAR já representa conhecimento para seleção de regras que geram preferencias para selecionar operadores, sendo natural aprender regras que geram preferencias baseadas na expectativa futura de recompensas.

Tanto chunking como aprendizado por reforço criam novas regras, mas último cria somente operadores de seleção de regras e se basea em regularidades estatísticas em recompensas ao invés de raciocínio interno.

2.3.3 Memória episódica

Outra fonte de conhecimento que está disponível é a própria experiência. A memória de experieência é chamado de memória episódica.

Um conjunto de episódios podem melhorar o compoartamento atraves de outro tipos de apreneizado, como chuncking ou aprendizado por reforço, fazneo com que seja possivel reprisar uma experiencia quando tiver tempo para reflexão deliberada.

Um episódio consistem em um subconjunto de elementos da WM uqe existe em um tempo de recordação. SOAR seleciona esses elementos da WM que foram usados recentemente. Para recuperar um pesidoio uma pista é crianda na WM através de regras. O epsiodio que melhor casar com a pista é encontrado e recriado na WM. Ele é então rotulado como um episodio para não haver confusão entre o que está na memória e o que está se recuperando na memória.

2.3.4 Memória Semântica

A última forma de fonte de conhecimento é a co-ocorrencia de estruturas na WM. Essas co-ocorrencias são mais gerais que episodios, as qusis representam apenas ocurrencias singulares. Essas são estruutras delcarativas, também chamadas de conhecimento semantico, que são que se “sabe” (em contrate com oque se “lembra” em uma memória episódica). A maneira que uma memória semântica é recuperada é realizada da mesma maneira que uma memória episódica.

3 RP-NETS

Uma RP-NET (Resource Processing Networks) ou Rede de Procesamento de Recursos é um modelo formal-computacional de sistemas a eventos discretos com capacidade de aprendizagem e adaptação que destina-se a modelar sistemas que envolvem o processamento de

recursos de qualquer natureza, transformando e/ou consumindo esses recursos de acordo com a dinâmica do sistema. Essa rede é um trabalho desenvolvido na Faculdade de Engenharia Elétrica e de Computação – UNICAMP [2][3]. O processamento dos recursos pode se dar de maneira paralela e concorrente, sendo possível ocorrer o processamento em vários agentes concorrentemente. Os recursos processados na RP-NET são independentes do domínio do problema e podem representar os mais variados conhecimentos, idéias, objetos, signos, e qualquer artefato comumente utilizado na modelagem de sistemas inteligentes. As definições formais das RP-Nets podem ser encontradas em [3].

Uma RP-Net é composta por recursos situados em lugares conectados por arcos, o que forma um tipo especial de grafo direcionado.

Qualquer conceito concreto ou abstrato pode ser representado na forma de um recurso, obedecendo as seguintes características:

i) cada recurso é único e identificado pelo seu nome que pode ser realocado no caso de destruição do recurso. Um recurso nunca mudará de nome durante a existência.

ii) um recurso pode possuir função de transformação. Os recurso que possuem essa funcionalidade são chamados de recursos ativos, senão são denominados recursos passivos.

iii) um recurso ativo é capaz de consumir e/ou gerar outros recursos.

Na modelagem, recursos ativos são os que realizam o processamento de outros recursos (inclusive a si próprio). Recursos passivos são entidades meramente manipuladas como peças, matérias-primas, componentes, dinhiero, texto, planilhas, tabelas, etc.

Os recursos ativos podem ser classificados em dois tipos:

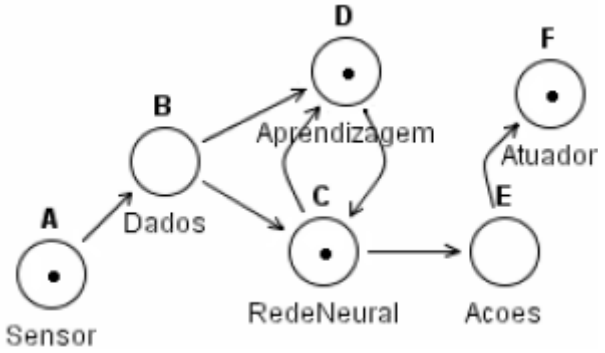

Figura 2. Exemplo RP-NET. Adaptado de [3].

mecânicos e inteligentes. Os recursos ativos mecânicos são aqueles processam outros recursos, mas não fazem avaliação dos mesmos. Por exemplo um recurso que monta uma máquina A, e que utiliza as peças B e C. Nesse caso, para o recurso ativo, não há distinção entre peças da classe B, ou peças da classe C. Qualquer peça B e C serão utilizadas para a montagem da máquina. Diferente disso, o recurso ativo inteligente é capaz de distinguir características dos recursos sendo processados e realizar a escolha de qual é melhor, por exemplo ao realizar a seleção de trabalhadores para compor uma determinada equipe.

Assim, de maneira geral um recurso ativo é um recurso que possui a capacidade de selecionar outros recursos em um dado conjunto e de posse deles, extrair seu conteúdo, podendo destruir ou preservá-los, e realizar operações internas de forma a gerar novos recursos.

As ações que os recursos ativos podem executar são determinadas na definição das chamadas funções de transformação. Essas funções estão associadas a portas de entrada ou saída, pela quais são obtidos os recursos a serem processados. Essas funções através de operadores determinados são capazes de avaliar a importância dos elementos de entrada e realizar as alterações nos recursos. Para isso são definidos dois tipos de operadores: de avaliação e de transformação. Os operadores de avaliação indicam o interesse do recurso ativo por um dado recurso passivo presente em um conjunto de candidatos, por meio de uma função de utilidade (a qual dá um valor a ser “pago” pelo recurso ativo para o uso de determinado recurso passivo). Os operadores de transformação realizam as operações para construção/destruição/alteração de recursos passivos previamente escolhidos. Evidentemente muitas vezes recursos ativos podem escolher os mesmo recursos passivos que são limitados. Assim se faz necessário algum algoritmo para eliminação de conflitos. Na RP-Nets o algoritmo utilzado é o BMSA mostrado primeiramente em [4].

Tendo como base o grau de utilidade de um recurso passivo para os recursos ativos o BMSA é responsável por escolher as ações possíveis. O algorítimo pode ser sintetizado em:

1. Dentre todas as ações a ser executadas escolha a ação de maior utilidade para execução. Se existir empate entre os graus de utilidade escolha de forma aleatória. Retire a ação escolhida do conjunto de ações a serem executadas.

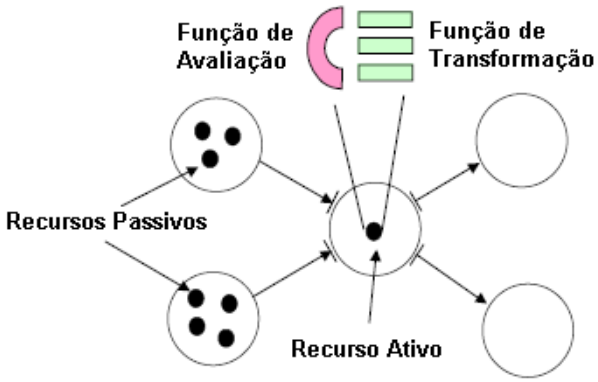

Figura 3. Recursos ativos e passivos. Adaptado de [5].

2. Avalie a possibilidade de execução das outras ações diante da escolha realizada em 1. Uma outra ação pode não

ser mais possível diante do acesso ao um mesmo recurso limitado. Ações podem coexistir caso acessem um recurso desde que o acesso seja compartilhado por ambas (não há consumo ou alteração do recurso).

3. Retorne ao passo 1 até que não existam mais ação no conjunto de soluções passíveis.

Como dito em [3] uma particularidade interessante desse algoritmo é a não necessidade de backtracking, isto é, o retorno a um estado prévio devido a problemas de conscistência da solução oferecida.

4 RP-NETS COMO UM SOAR DISTRIBUÍDO

A RP-Net pode ser utilizada como um framework para a criação de sistemas de computação flexível híbridos sendo possível de forma razoavelmente simples o uso técnicas de sistemas fuzzy, redes neurais, e algoritmos evolutivos, podendo inclusive compô-los com outras técnicas s de inteligência artificial [6]. Gudwin também mostra em [6] que as RP-Nets podem ser vistas em um plano abstrato como sistema de processamento e geração de conhecimento. De posse das ferramentas de sistemas inteligentes, devidamente modeladas em RP-Nets, as caracterícas das variadas memórias, tipos de aprendizado o processamento de informação podem também ser modelados na forma de RP-Nets.

Em [1] temos que um gargalo cognitivo do SOAR é a imposição de seu ciclo de decisão de limitar a seleção de um único operador por ciclo. Isso restringe o quanto de trabalho cognitivo pode ser realizado em uma única vez. Nesse ponto as RP-Nets podem oferecer contribuições ao trabalho desenvolvido em Michigan.

Continuando o paralelo entre o SOAR e RP-Nets, podemos imaginar os recursos ativos como os elementos que tem a capacidade de alterar o modelo ou estado da rede por meio de suas funções de transformação as quais realizam consumo/produção/alteração de recursos do sistema, assim como os operadores no SOAR. Porém no caso das RP-Nets, fazendo o uso do algoritmo BMSA várias funções de transformação podem disparar durante um ciclo de simulação, ou seja, não há a limitação de um único “operador” Assim, as RP-Nets são capazes de remover a limitação do SOAR de um único operador sendo disparado por ciclo de decisão.

A desvantagem dessa abordagem seria deixar para o modelador criar os seus mecanismos de aprendizagem e memória já presentes na arquitetura SOAR. Entretanto, pode-se desenvolver frameworks sobre as RP-Nets para facilitar o seu uso e mitigar esse problema.

5 CONCLUSÃO

Este artigo apresentou as teorias SOAR e RP-Nets. Foi mostrado em teoria como as RP-Nets podem suprir o problema do SOAR de ser limitado a um operador disparado por ciclo de decisão, através de um SOAR distribuído Entranto, para trabalhos futuros são necessários testes nas duas arquiteturas, uma vez que ambas oferecem ambientes computacionais de simulação.

REFERÊNCIAS BIBLIOGRÁFICAS

[1]Lehman, J.F., Laird, J. Rosembloom P. (2006). A Gentle Introduction to SOAR, An Arquitecture for Human Cognition: 2006 Update.

[2]Gomes, S. R. A., (2000). Contribuições ao estudo de redes de agentes. *Dissertação de Mestrado*, DCA/FEEC/UNICAMP.

[3]Tatai, V. K., (2003). Técnicas de Sistemas Inteligentes Aplicadas ao Desenvolvimento de Jogos de Computador. *Dissertação de Mestrado*, DCA/FEEC/UNICAMP

[4]Guerreiro, J. A. S., Gomes A. S. e Gudwin, R. R. (1999). A Computational tool to model intelligent systems. *Anais do IV Simpósio Brasileiro de Automação Inteligente – SBAI'99*: 227-232.

[5]Gudwin, R. R., (2002). Semiotic Synthesis and Semionic Networks. S.E.E.D Journal (Semiotics, Evolution, Energy, and Development), v. 2 (2), pp. 55-83.

[6]Gudwin, R. R. e Gomide, F.A.C. (1997). A Computational Semiotics Approach for Soft Computing. *IEEE International Conference on Systems, Man and Cybernetics SMC'97 -12-15/Outubro.*, vol 4: 3981-3986.

[7]Gudwin, R. R., (2002). A Abordagem Semiônica em Sistemas a Eventos Discretos

[8]Guerrero, J. A. S., (1999). Rede de Agentes: Uma ferramenta para o Projeto de Sistemas Inteligentes. *Dissertação de Mestrado*, DCA/FEEC/UNICAMP.

[9]Gudwin, R. R., (1996). Contribuições ao Estudo Matemático de Sistemas Inteligentes. *Tese de Doutorado* , DCA/FEEC/UNICAMP.